Úvod do teoretické informatiky

Zdeněk Sawa

Katedra informatiky, FEI,
Vysoká škola báňská – Technická univerzita Ostrava
17. listopadu 2172/15, Ostrava-Poruba 708 00
Česká republika

16. února 2025

Přednášející

Jméno: doc. Ing. Zdeněk Sawa, Ph.D.
E-mail: zdenek.sawa@vsb.cz
Místnost: EA413
Web: https://www.cs.vsb.cz/sawa/uti

Na těchto stránkách najdete:

- Informace o předmětu
- Učební texty
- Slidy z přednášek
- Zadání příkladů na cvičení
- Aktuální informace
- Odkaz na stránku s animacemi

Požadavky

- **Zápočet** (30 bodů):
  - Zápočtová písemka (24 bodů) — bude se psát na cvičení
    - Minimum pro získání zápočtu je 12 bodů.
    - Možnost opravy za 20 bodů.
  - Aktivita na cvičení (6 bodů)
    - Minimum pro získání zápočtu jsou 3 body.
- **Zkouška** (70 bodů)
  - Písemná zkouška skládající se ze dvou částí po 35 bodech, přičemž z každé části je nutné získat nejméně 12 bodů.
  - Celkově je třeba získat minimálně 30 bodů.

Teoretická informatika

Teoretická informatika — vědní obor na pomezí mezi informatikou a matematikou

- zkoumání obecných otázek týkajících se algoritmů a výpočtů
- zkoumání různých formalismů pro popis algoritmů
- zkoumání různých prostředků pro popis syntaxe a sémantiky formálních jazyků (zejména s důrazem na programovací jazyky)
- matematický přístup k analýze a řešení problémů (dokazování obecně platných matematických tvrzení týkajících se algoritmů)

Teoretická informatika

Příklady některých typických otázek studovaných v teoretické informatice:

- Je možné daný problém řešit pomocí nějakého algoritmu?
- Pokud je možné daný problém řešit pomocí algoritmu, jaká je výpočetní složitost tohoto algoritmu?
- Existuje pro daný problém nějaký efektivní algoritmus, který ho řeší?
- Jak se přesvědčit o tom, že daný algoritmus je skutečně korektním řešením daného problému?
- Jaké instrukce musí umět vykonat stroj, který by mohl provádět daný algoritmus?

Algoritmy a problémy

Algoritmus — mechanický postup, jak něco spočítat (může být vykonáván počítačem)

Algoritmy slouží k řešení různých **problémů**.

Příklad algoritmického problému:

Vstup: Přirozená čísla x a y.

Výstup: Přirozené číslo z takové, že $z = x + y$.

Algoritmy a problémy

Algoritmus — mechanický postup, jak něco spočítat (může být vykonáván počítačem)

Algoritmy slouží k řešení různých **problémů**.

Příklad algoritmického problému:

Vstup: Přirozená čísla x a y.

Výstup: Přirozené číslo z takové, že $z = x + y$.

Konkrétní vstup nějakého problému se nazývá **instance** problému.

Příklad: Instancí výše uvedeného problému je například dvojice čísel 728 a 34.

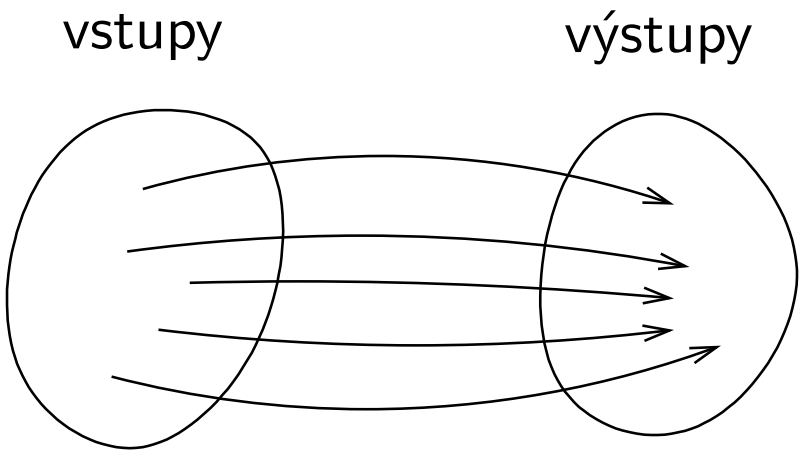
Výstupem pro tuto instanci je číslo 762.

Problémy

Problém

V zadání **problému** musí být určeno:

- co je množinou možných vstupů
- co je množinou možných výstupů
- jaký je vztah mezi vstupy a výstupy



Příklady problémů

Problém „Třídění“

Vstup: Sekvence prvků $a_1, a_2, \ldots, a_n$.

Výstup: Prvky sekvence $a_1, a_2, \ldots, a_n$ seřazené od nejmenšího po největší.

Příklad:

- Vstup: $8, 13, 3, 10, 1, 4$
- Výstup: $1, 3, 4, 8, 10, 13$

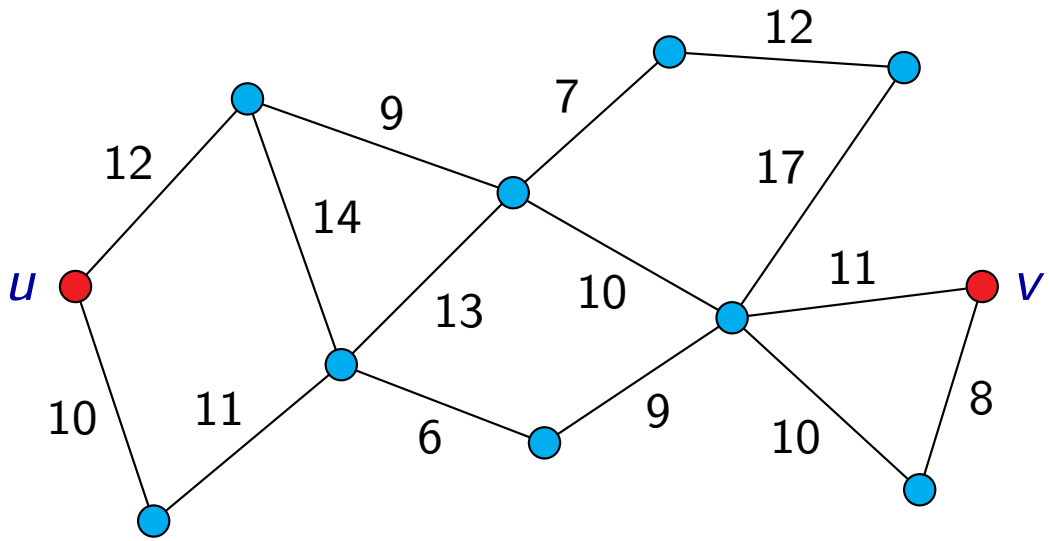
Příklad algoritmického problému

Problém „Hledání nejkratší cesty v (neorientovaném) grafu“

Vstup: Neorientovaný graf $G = (V, E)$ s ohodnocením hran a dvojice vrcholů $u, v \in V$.

Výstup: Nejkratší cesta z vrcholu u do vrcholu v. (Nebo informace, že žádná taková cesta neexistuje.)

Příklad:



Algoritmy a problémy

Algoritmus **řeší** daný problém pokud:

- Se pro každý vstup po konečném počtu kroků zastaví.
- Pro každý vstup vydá správný výstup.

Korektnost algoritmu — ověření toho, že daný algoritmus skutečně řeší daný problém

Výpočetní složitost algoritmu:

- **časová složitost** — jak závisí doba výpočtu na velikosti vstupu
- **paměťová** (nebo též **prostorová**) **složitost** — jak závisí množství použité paměti na velikosti vstupu

Poznámka: Pro jeden problém může existovat celá řada algoritmů, které jej korektně řeší.

Další příklady problémů

Problém „Prvočíselnost“

Vstup: Přirozené číslo n.

Výstup: ANO pokud je n prvočíslo, NE v opačném případě.

Poznámka: Přirozené číslo n je **prvočíslo**, pokud je větší než 1 a je dělitelné beze zbytku pouze čísly 1 a n.

Prvních několik prvočísel: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, ...

Rozhodovací problémy

Problémům, kde množina výstupů je $\{\text{Ano}, \text{Ne}\}$ se říká **rozhodovací problémy**.

Rozhodovací problémy jsou většinou specifikovány tak, že místo popisu toho, co je výstupem, je uvedena otázka.

Příklad:

Problém „Prvočíselnost“

Vstup: Přirozené číslo n.

Otázka: Je n prvočíslo?

Optimalizační problémy

Problémům, kde je pro daný vstup určena nějaká množina **přípustných řešení** a kde je úkolem mezi těmito přípustnými řešeními vybrat takové, které je v nějakém ohledu minimální nebo maximální (případně zjistit, že žádné přípustné řešení neexistuje), se říká **optimalizační problémy**.

Příklad:

Problém „Hledání nejkratší cesty v (neorientovaném) grafu“

Vstup: Neorientovaný graf $G = (V, E)$ s ohodnocením hran, a dvojice vrcholů $u, v \in V$.

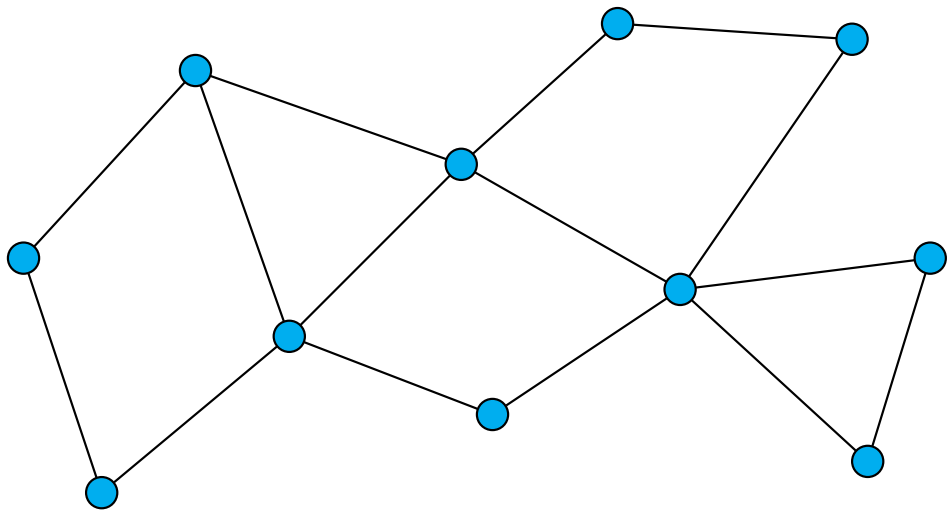
Výstup: Nejkratší cesta z vrcholu u do vrcholu v.

Optimalizační problémy

Problém „Barvení grafu“

Vstup: Neorientovaný graf G.

Výstup: Minimální počet barev, kterými je možné obarvit vrcholy grafu G tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu, a konkrétní příklad obarvení vrcholů používající tento minimální počet barev.

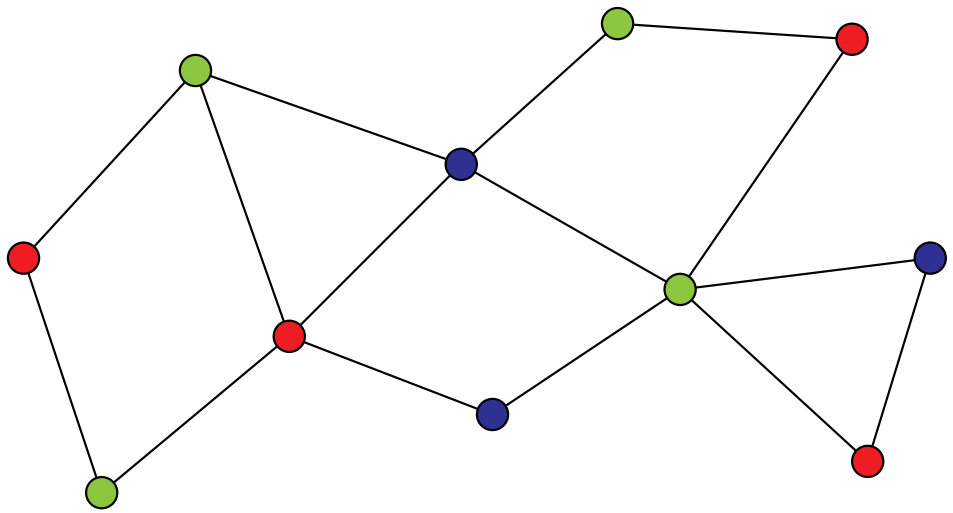


Optimalizační problémy

Problém „Barvení grafu“

Vstup: Neorientovaný graf G.

Výstup: Minimální počet barev, kterými je možné obarvit vrcholy grafu G tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu, a konkrétní příklad obarvení vrcholů používající tento minimální počet barev.



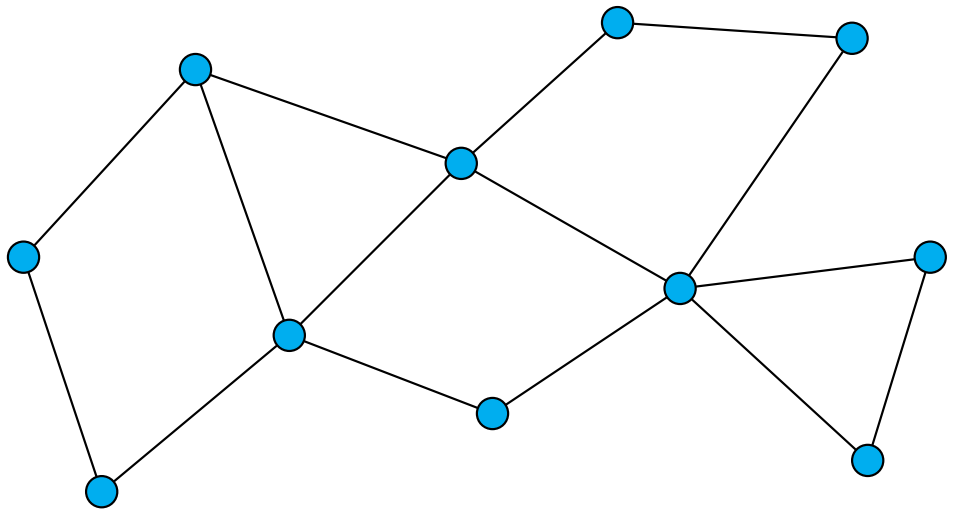
Barvy: 3

Optimalizační problémy přeformulované jako rozhodovací

Problém „Barvení grafu k barvami“

Vstup: Neorientovaný graf G a přirozené číslo k.

Otázka: Je možné obarvit vrcholy grafu G k barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?



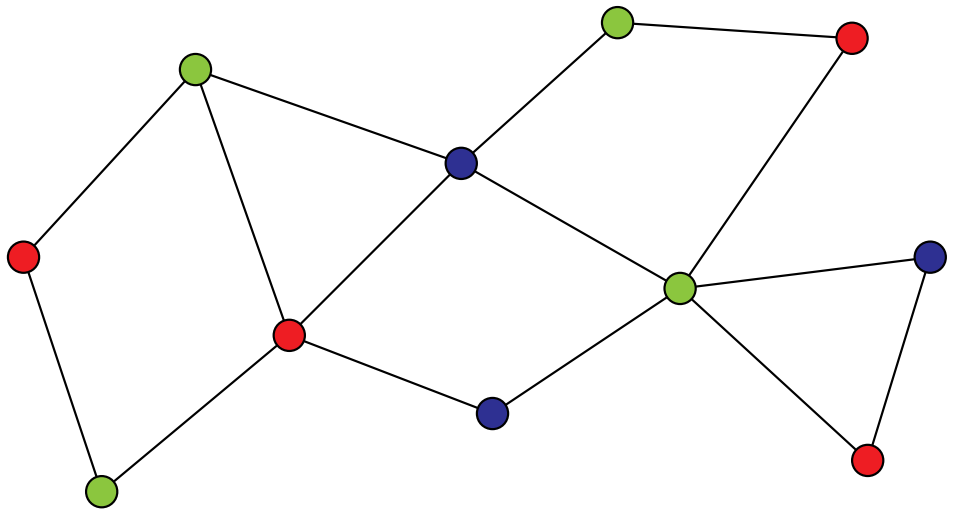
$k = 3$

Optimalizační problémy přeformulované jako rozhodovací

Problém „Barvení grafu k barvami“

Vstup: Neorientovaný graf G a přirozené číslo k.

Otázka: Je možné obarvit vrcholy grafu G k barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?



$k = 3$

Algoritmicky řešitelné problémy

Předpokládejme, že máme dán nějaký problém P.

Jestliže existuje nějaký algoritmus, který řeší problém P, pak říkáme, že problém P je **algoritmicky řešitelný**.

Jestliže P je rozhodovací problém a jestliže existuje nějaký algoritmus, který problém P řeší, pak říkáme, že problém P je **(algoritmicky) rozhodnutelný**.

Když chceme ukázat, že problém P je algoritmicky řešitelný, stačí ukázat nějaký algoritmus, který ho řeší (a případně ukázat, že daný algoritmus problém P skutečně řeší).

Algoritmicky neřešitelné problémy

Problém, který není algoritmicky řešitelný, je **algoritmicky neřešitelný**.

Rozhodovací problém, který není rozhodnutelný, je **nerozhodnutelný**.

Kupodivu existuje řada algoritmických problémů (přesně definovaných), o kterých je dokázáno, že nejsou algoritmicky řešitelné.

Teorie vyčíslitelnosti — oblast teoretické informatiky, která se zabývá zkoumáním toho, které problémy jsou a které nejsou algoritmicky řešitelné.

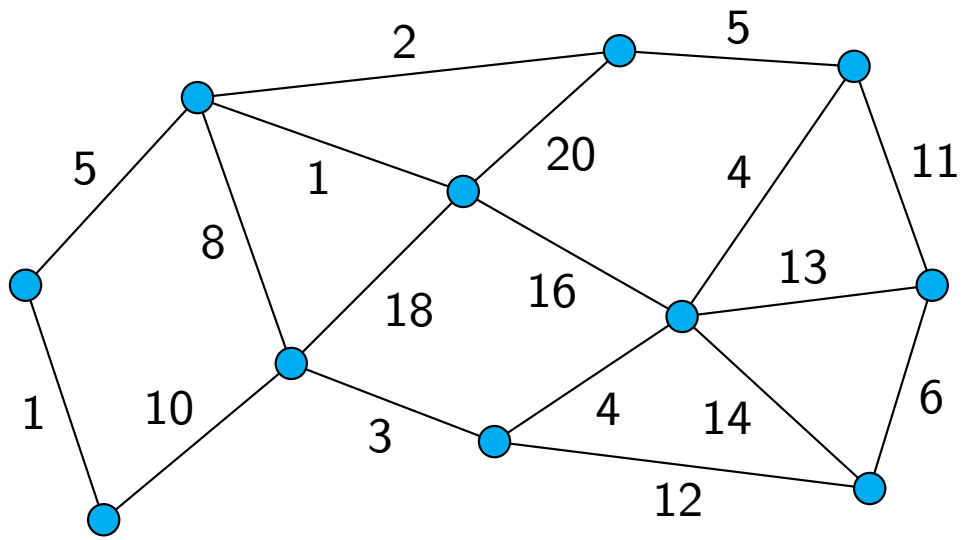
Teorie složitosti

Řada problémů je algoritmicky řešitelných, ale neexistují (nebo nejsou známy) efektivní algoritmy, které by je řešily:

TSP - Problém obchodního cestujícího

Vstup: Neorientovaný graf G s hranami ohodnocenými přirozenými čísly.

Výstup: Nejkratší uzavřená cesta, která projde všemi vrcholy a skončí v tom vrcholu, kde začíná.



Teoretická informatika

Některé další oblasti teoretické informatiky:

- teorie složitosti
- teorie formálních jazyků
- výpočetní modely
- paralelní a distribuované algoritmy
- ...

Teorie formálních jazyků

Oblast teoretické informatiky zabývající se otázkami týkajícími se **syntaxe**.

- **Jazyk** — množina slov
- **Slovo** — sekvence symbolů z určité abecedy
- **Abeceda** — množina **symbolů** (nebo též **znaků**)

Slova a jazyky se v informatice objevují na mnoha místech:

- Reprezentace vstupních a výstupních dat
- Reprezentace kódu programů
- Manipulace s řetězci znaků nebo se soubory
- ...

Teorie formálních jazyků – motivace

Příklady typů problémů, při jejichž řešení se využívá poznatků z teorie formálních jazyků:

- Tvorba překladačů:
  - lexikální analýza
  - syntaktická analýza
- Vyhledávání v textu:
  - hledání zadaného vzorku
  - hledání textu zadaného regulárním výrazem

Abeceda, slovo

- **Abeceda** — libovolná neprázdná konečná množina **symbolů** (**znaků**)

  Příklad: $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$

- **Slovo** — libovolná konečná posloupnost symbolů z dané abecedy

  Příklad: $\mathtt{cabcbba}$

  Množina všech slov nad abecedou Σ se označuje zápisem Σ^*.

  Pro proměnné, jejichž hodnoty jsou slova, budeme používat názvy w, u, v, x, y, z, apod., případně s indexy (např. w_1, w_2)

  Zápis $w = \mathtt{cabcbba}$ tedy znamená, že hodnotou proměnné w je slovo $\mathtt{cabcbba}$.

  Podobně zápis $w \in \Sigma^*$ znamená, že hodnotou proměnné w je nějaké slovo tvořené symboly z abecedy Σ.

Formální jazyky

Definice

(Formální) jazyk L v abecedě Σ je nějaká libovolná podmnožina množiny Σ^*, tj. $L \subseteq \Sigma^*$.

Příklad: Předpokládejme, že $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$:

- Jazyk $L_1 = \{\, \mathtt{aab}, \mathtt{bcca}, \mathtt{aaaaa} \,\}$
- Jazyk $L_2 = \{\, w \in \Sigma^* \mid \text{počet výskytů symbolů } \mathtt{b} \text{ ve slově } w \text{ je sudý} \,\}$

Formální jazyky

Příklad:

Abeceda Σ je množina všech ASCII znaků.

Příklad slova:

```
#include <stdio.h>

int main()
{
    printf("Hello, world!\n");
    return 0;
}
```

$$\texttt{\#include␣<stdio.h>}\hookleftarrow\hookleftarrow\texttt{int␣main()}\hookleftarrow\texttt{\{}\hookleftarrow\texttt{␣␣␣␣printf("He}\cdots$$

Formální jazyky

Prostředky používané pro popis formálních jazyků:

- automaty
- gramatiky
- regulární výrazy

Kódování vstupu a výstupu

U algoritmických problémů často předpokládáme, že vstupy i výstupy jsou kódovány jako slova v nějaké abecedě Σ.

Příklad: Například u problému „Třídění“ bychom mohli zvolit jako abecedu $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ,\}$.

Vstupem by pak mohlo být například slovo

$$826,13,3901,128,562$$

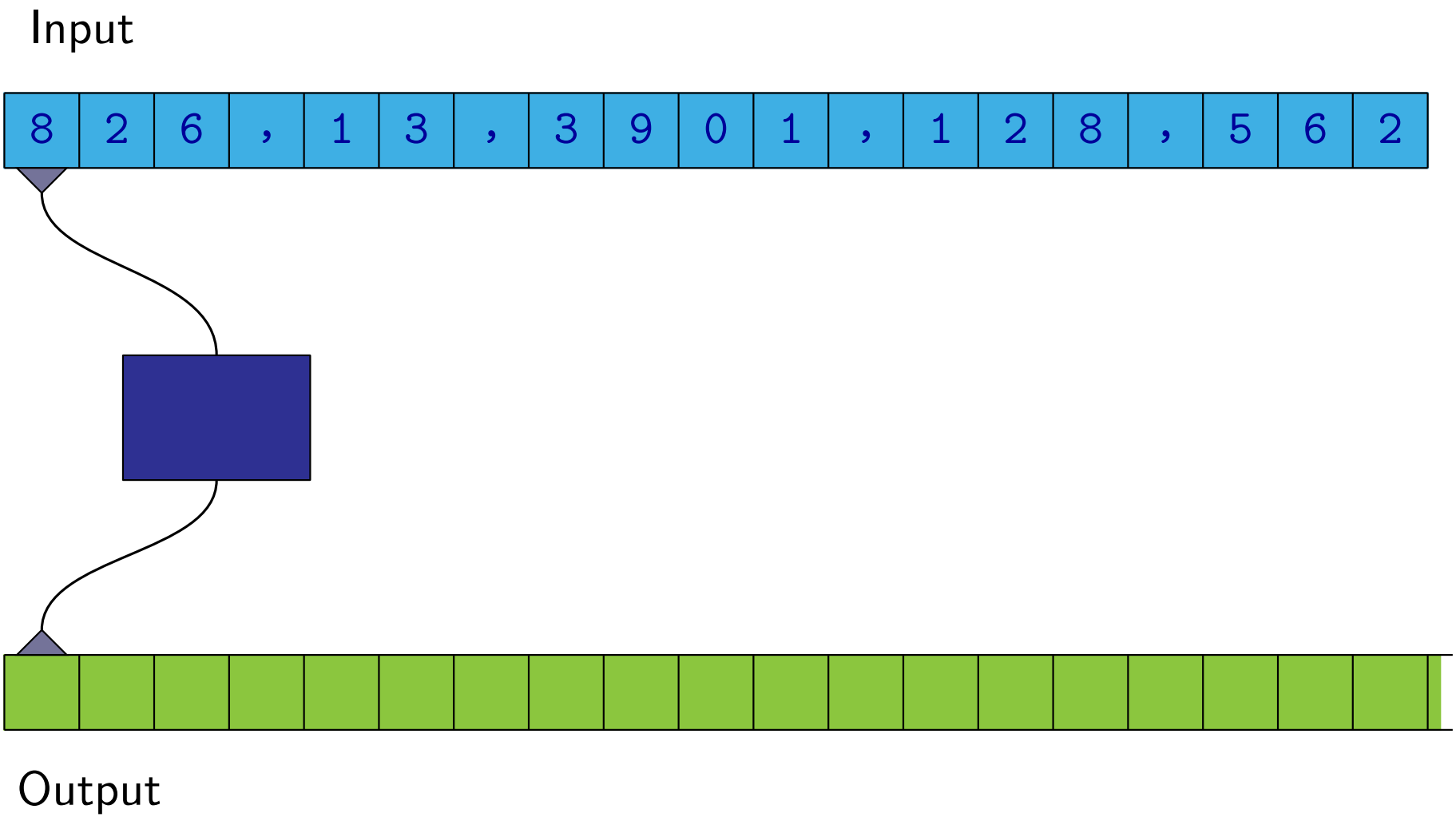
a výstupem slovo

$$13,128,562,826,3901$$

Poznámka: Ne každé slovo ze Σ^* musí reprezentovat nějaký vstup. Kódování bychom ale měli zvolit tak, abychom byli schopni snadno poznat ta slova, která nějaký vstup reprezentují.

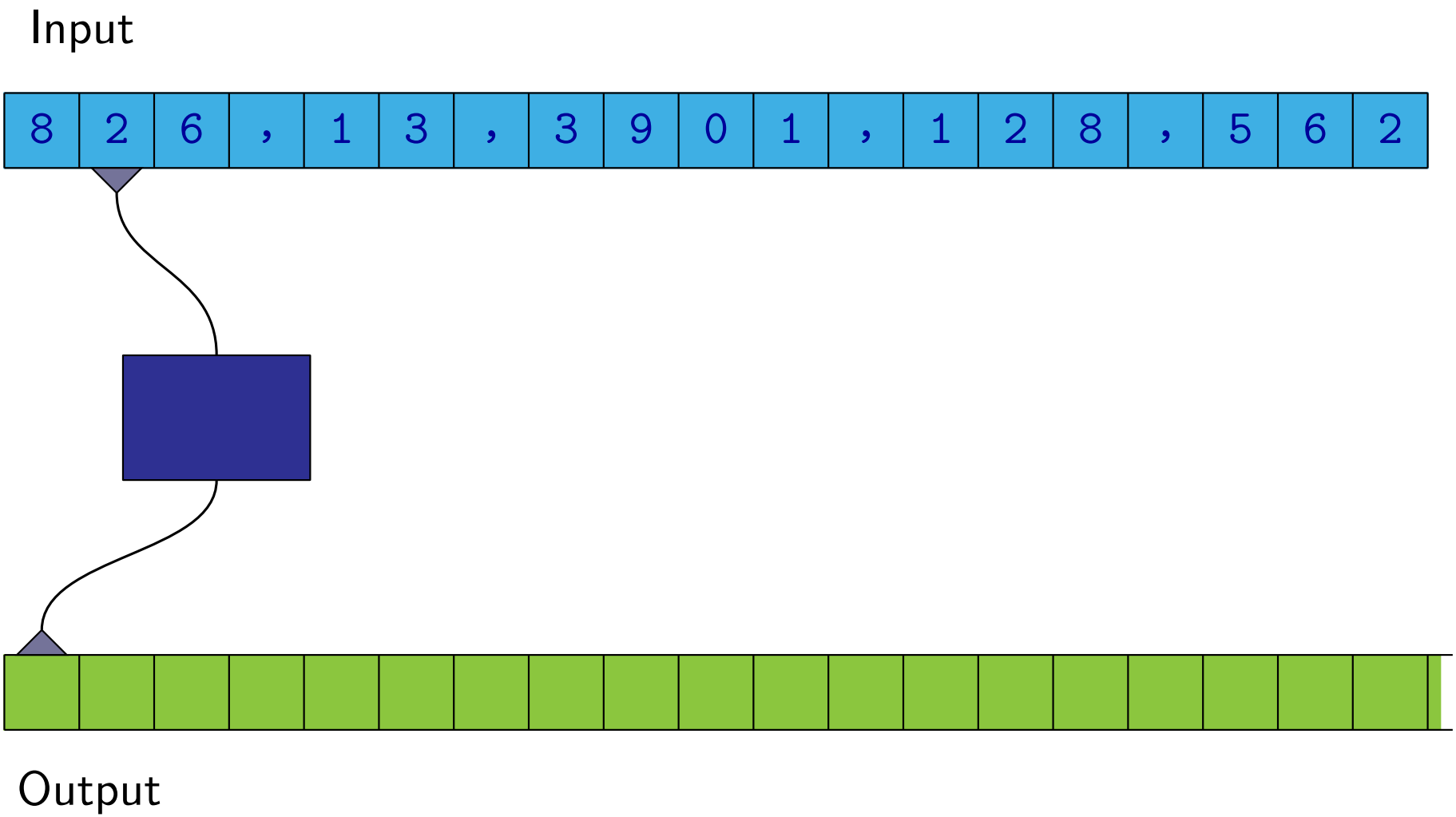
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



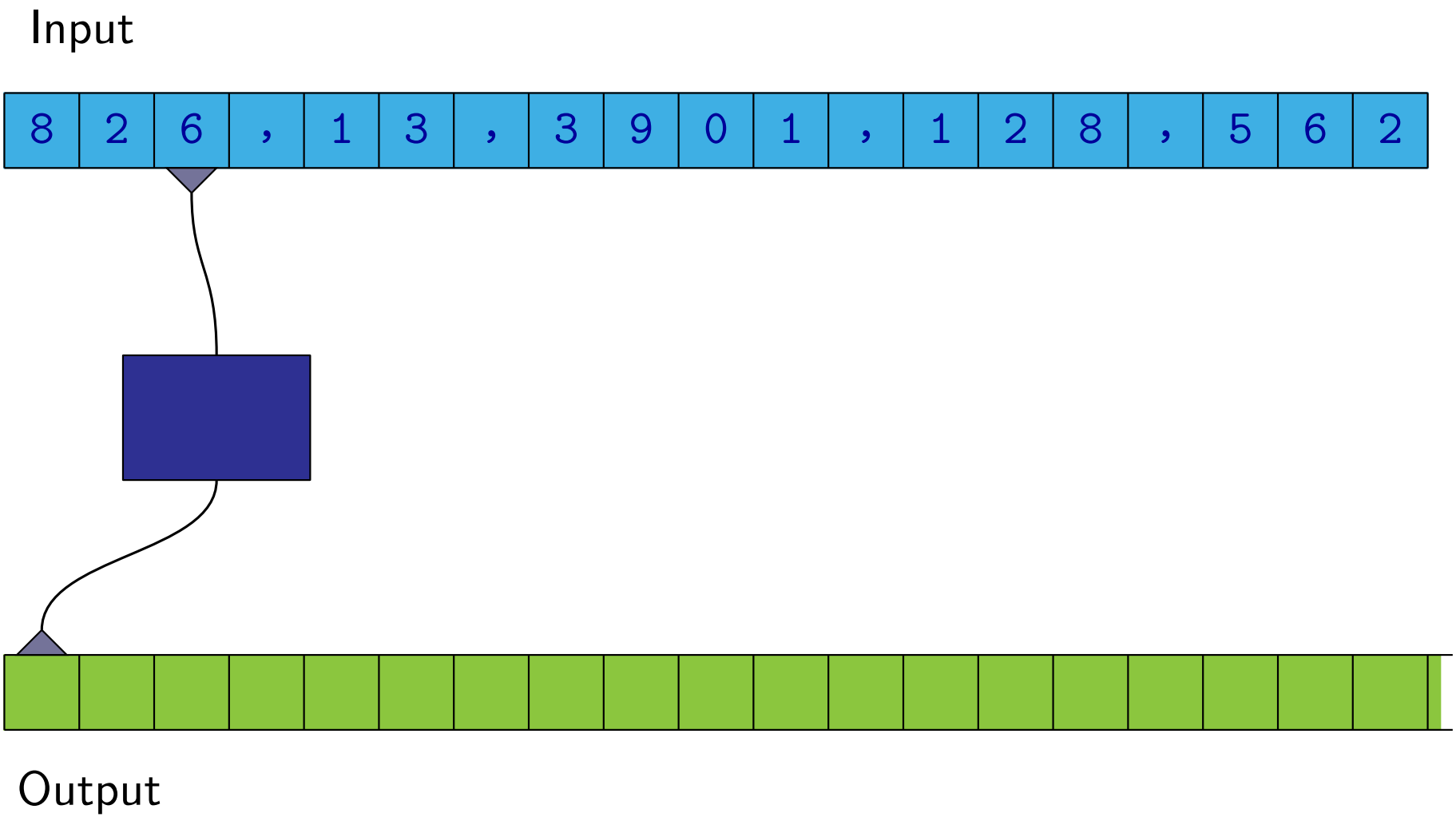
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



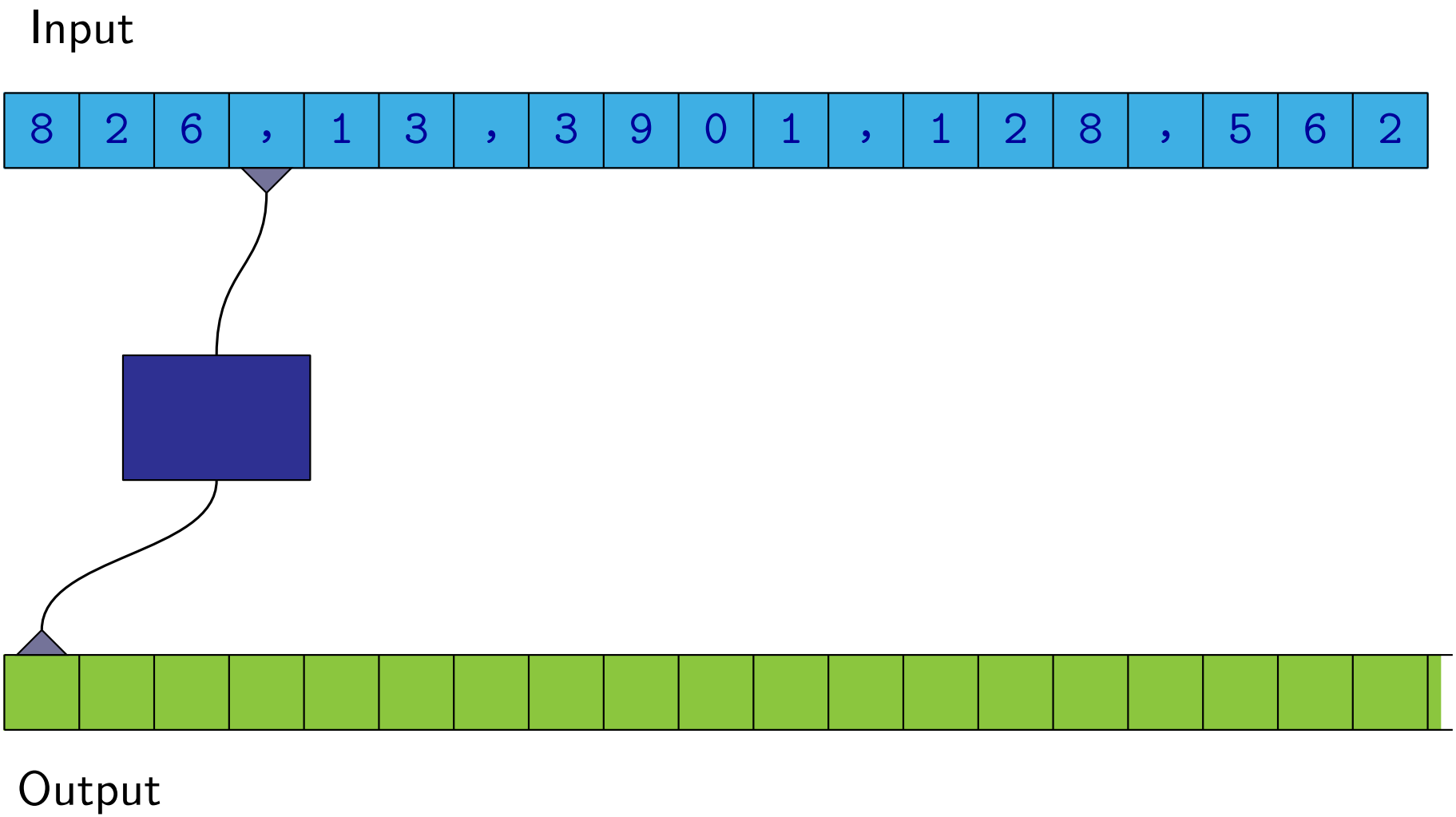
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



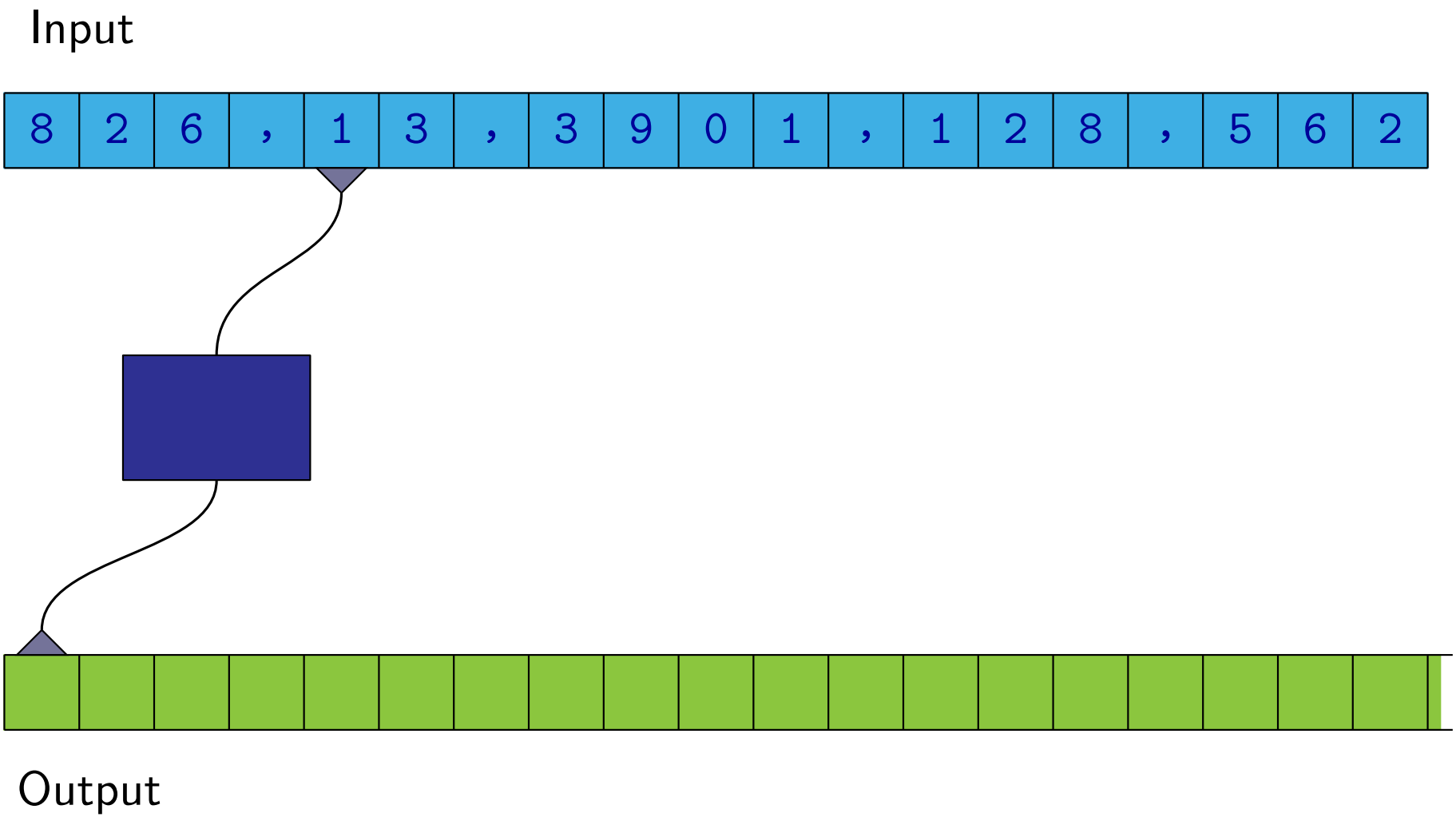
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



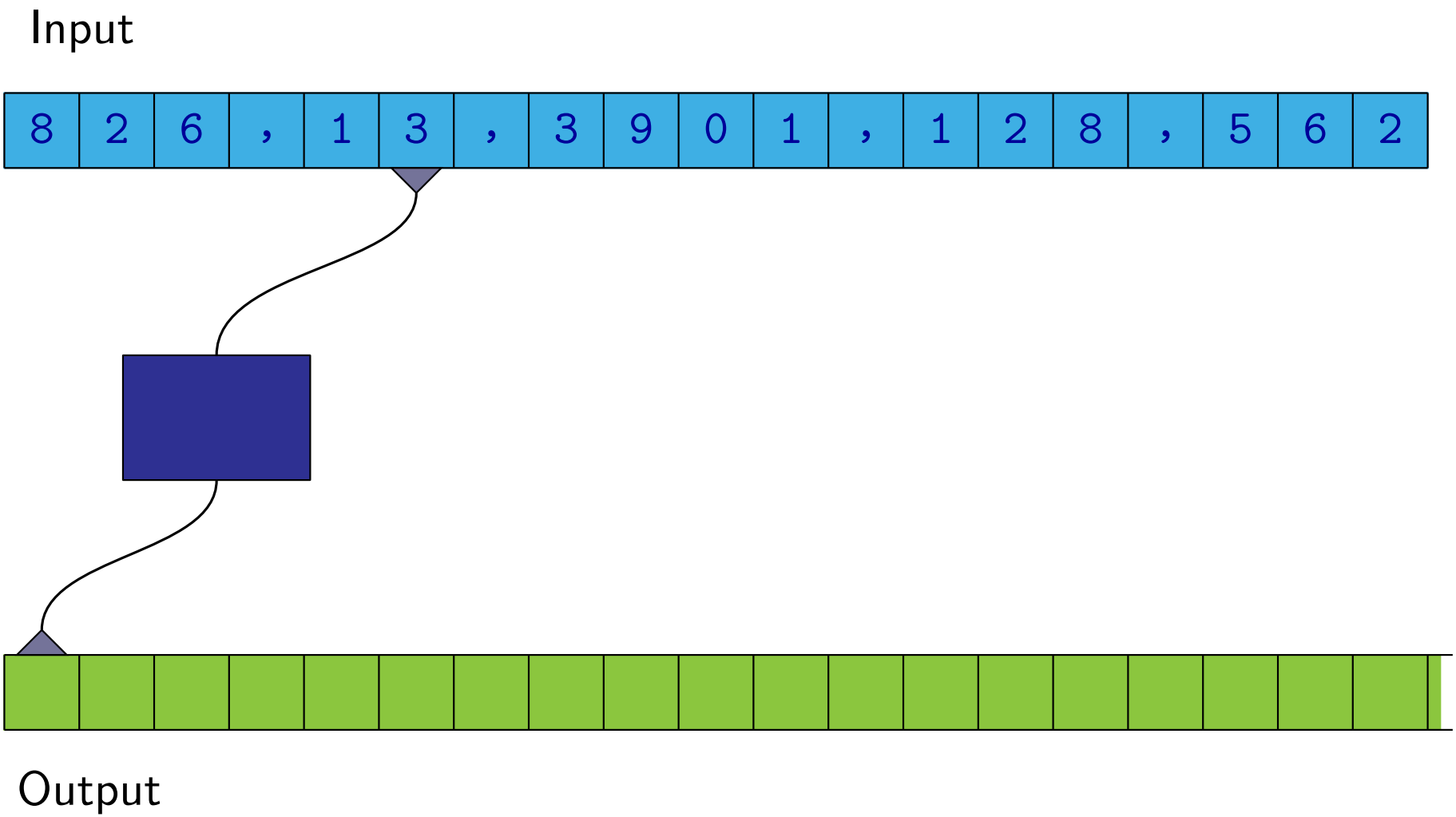
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



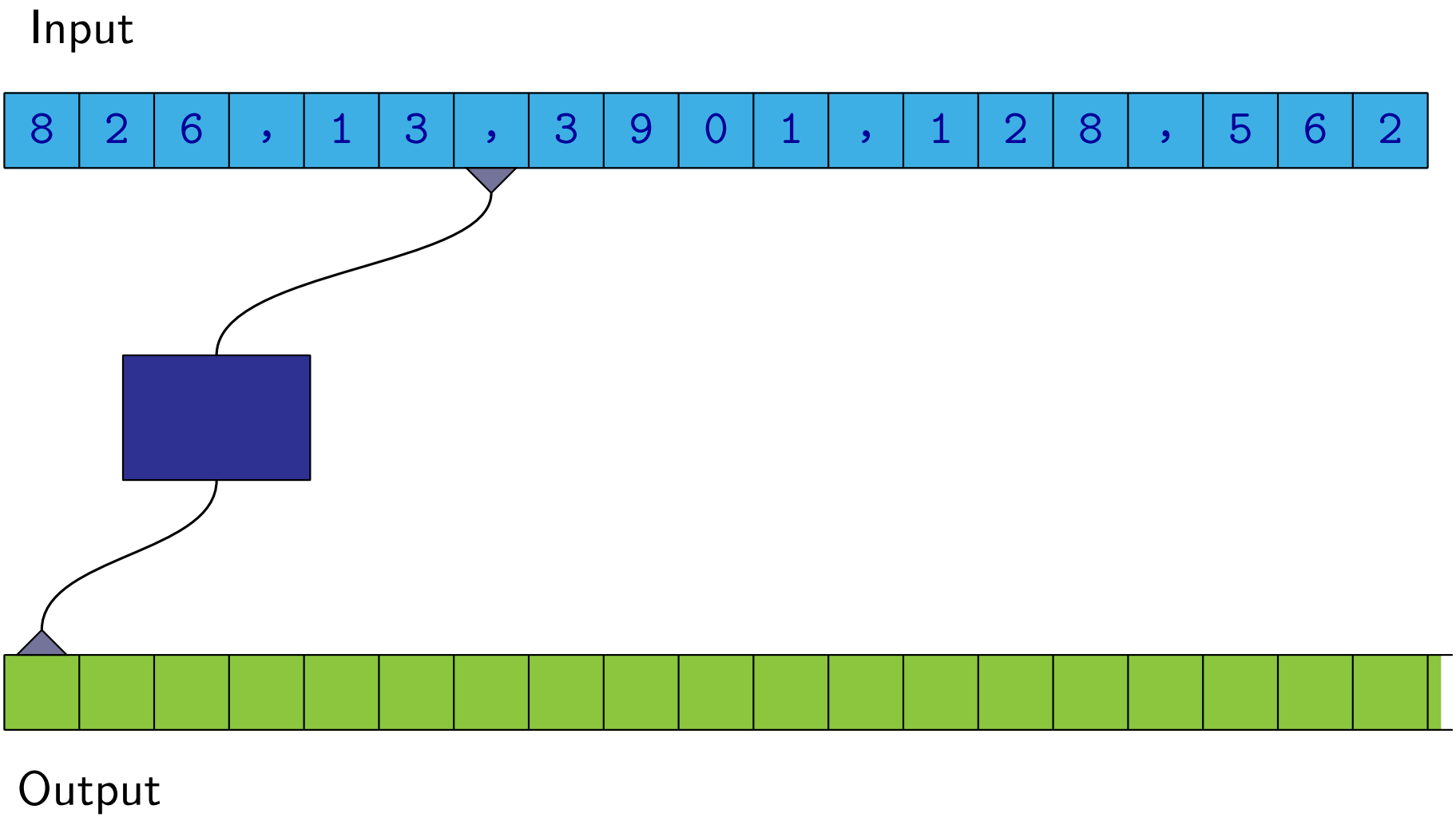
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



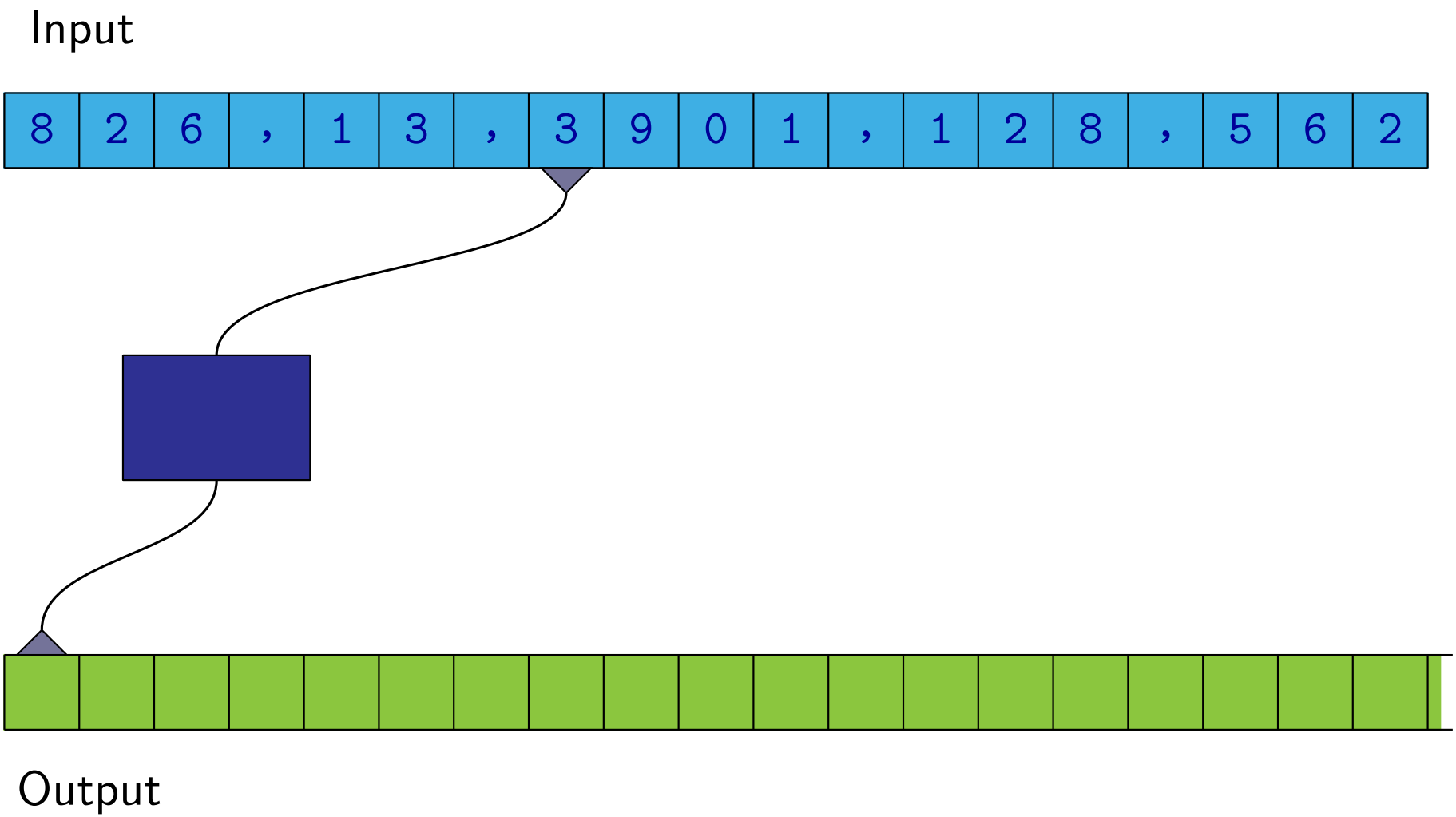
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



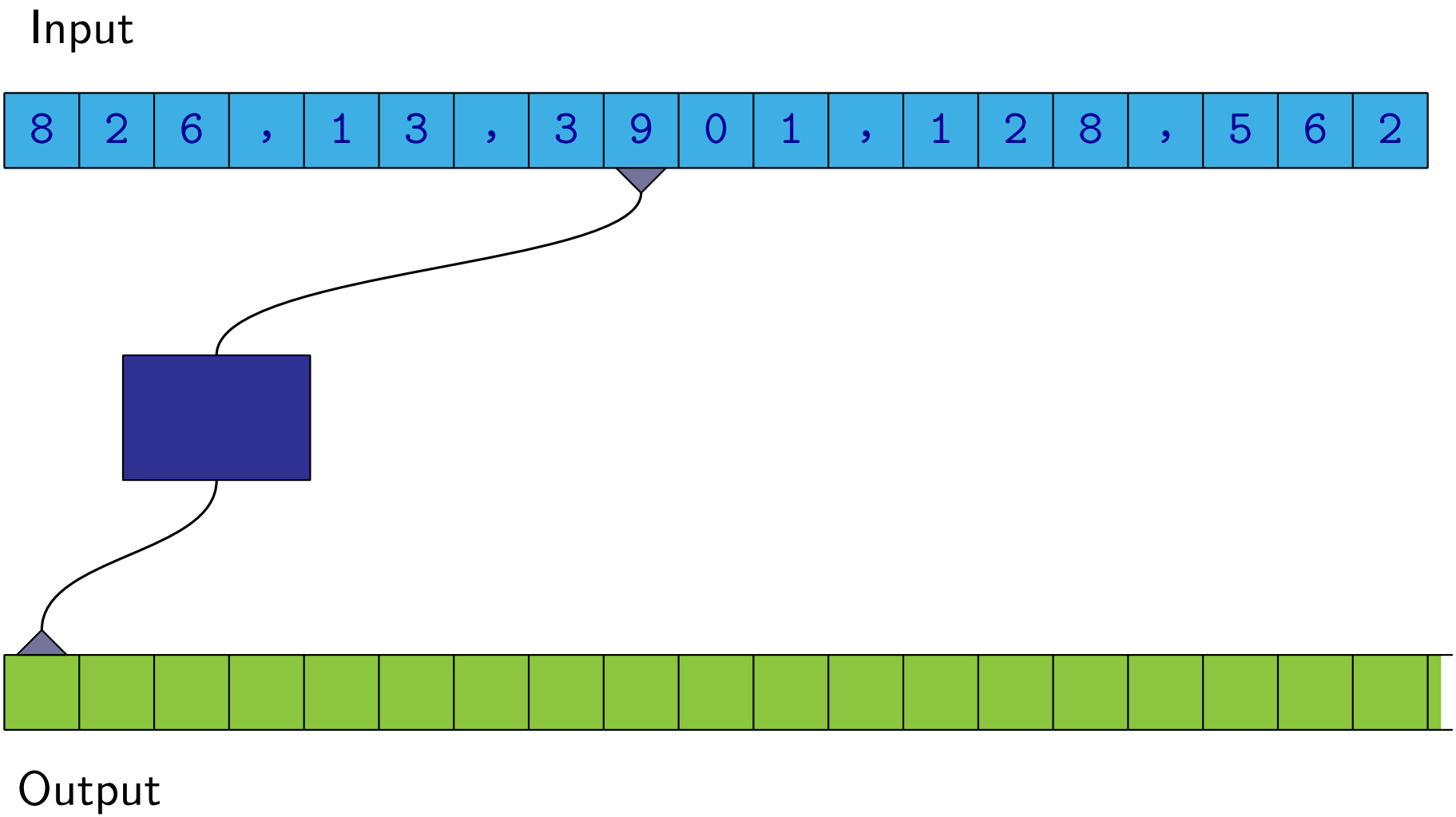
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



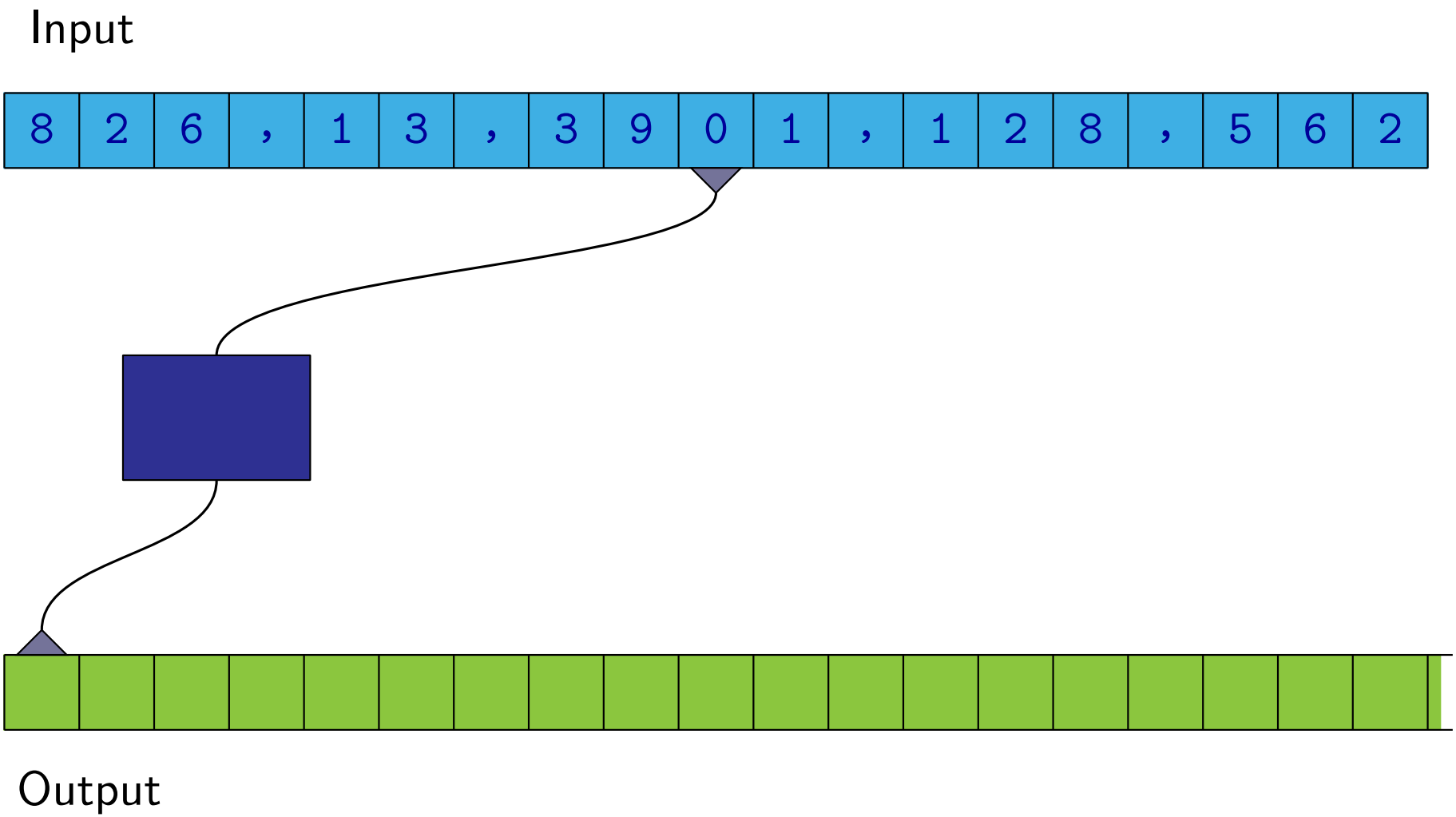
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



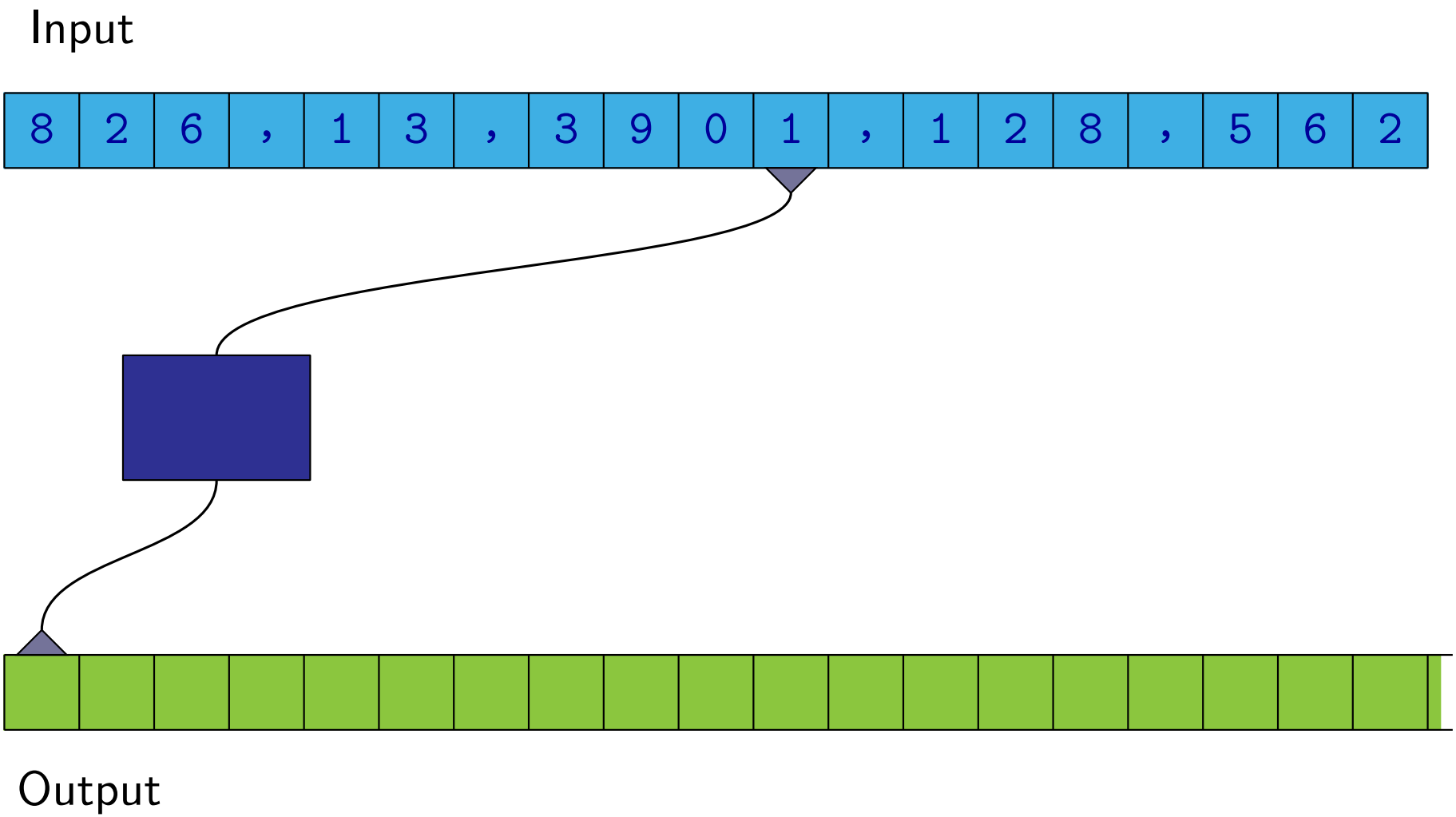
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



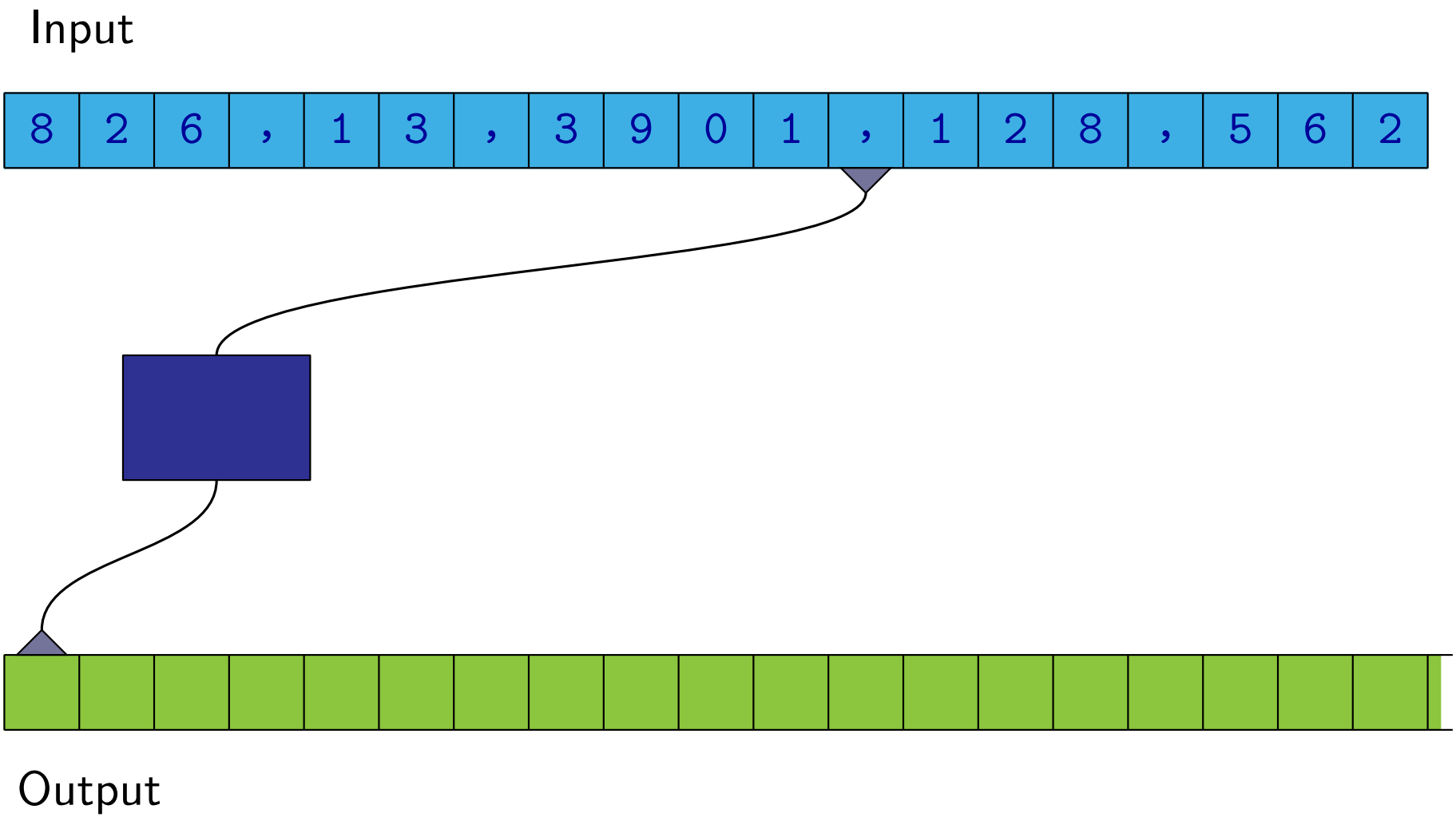
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



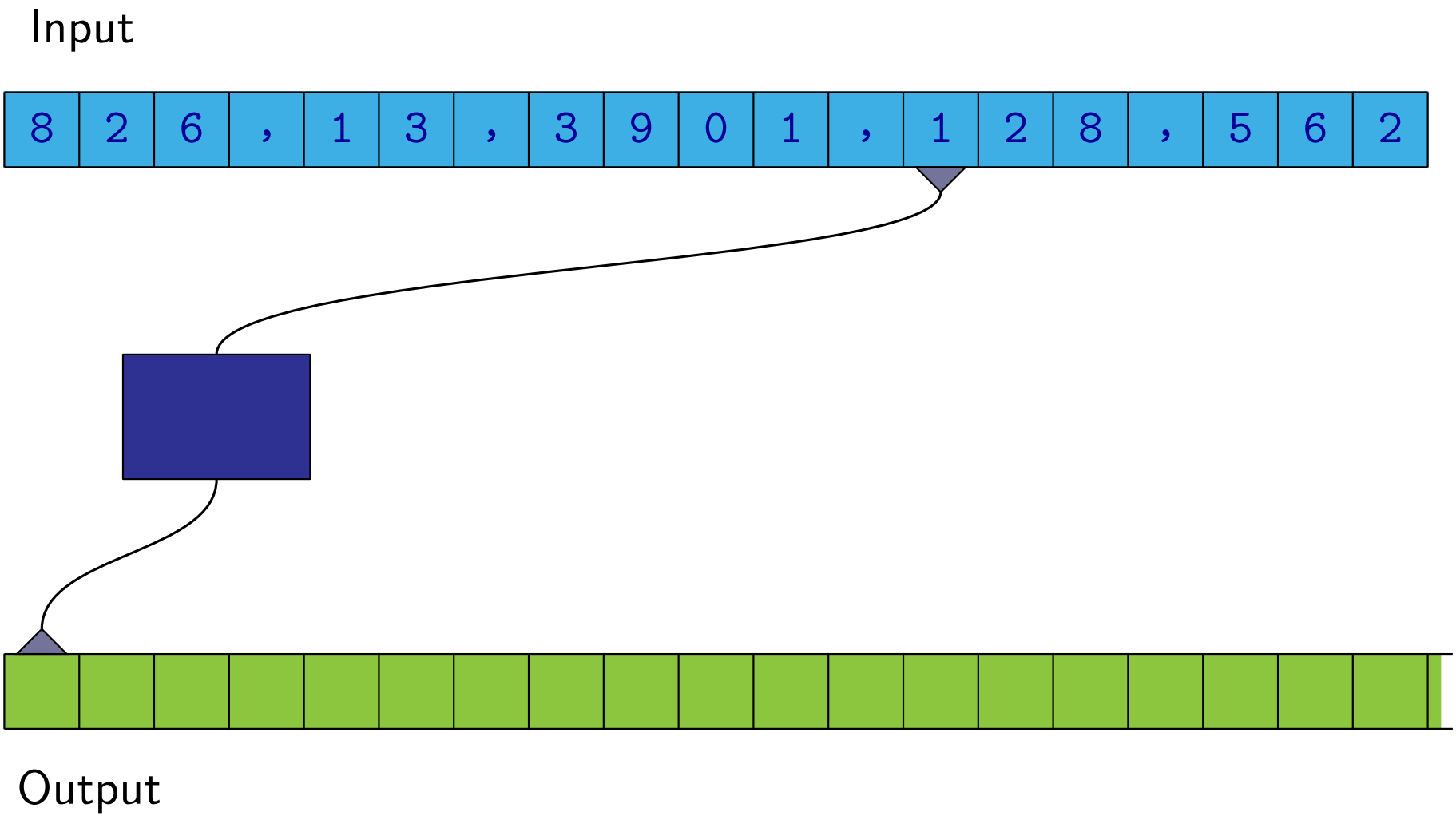
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



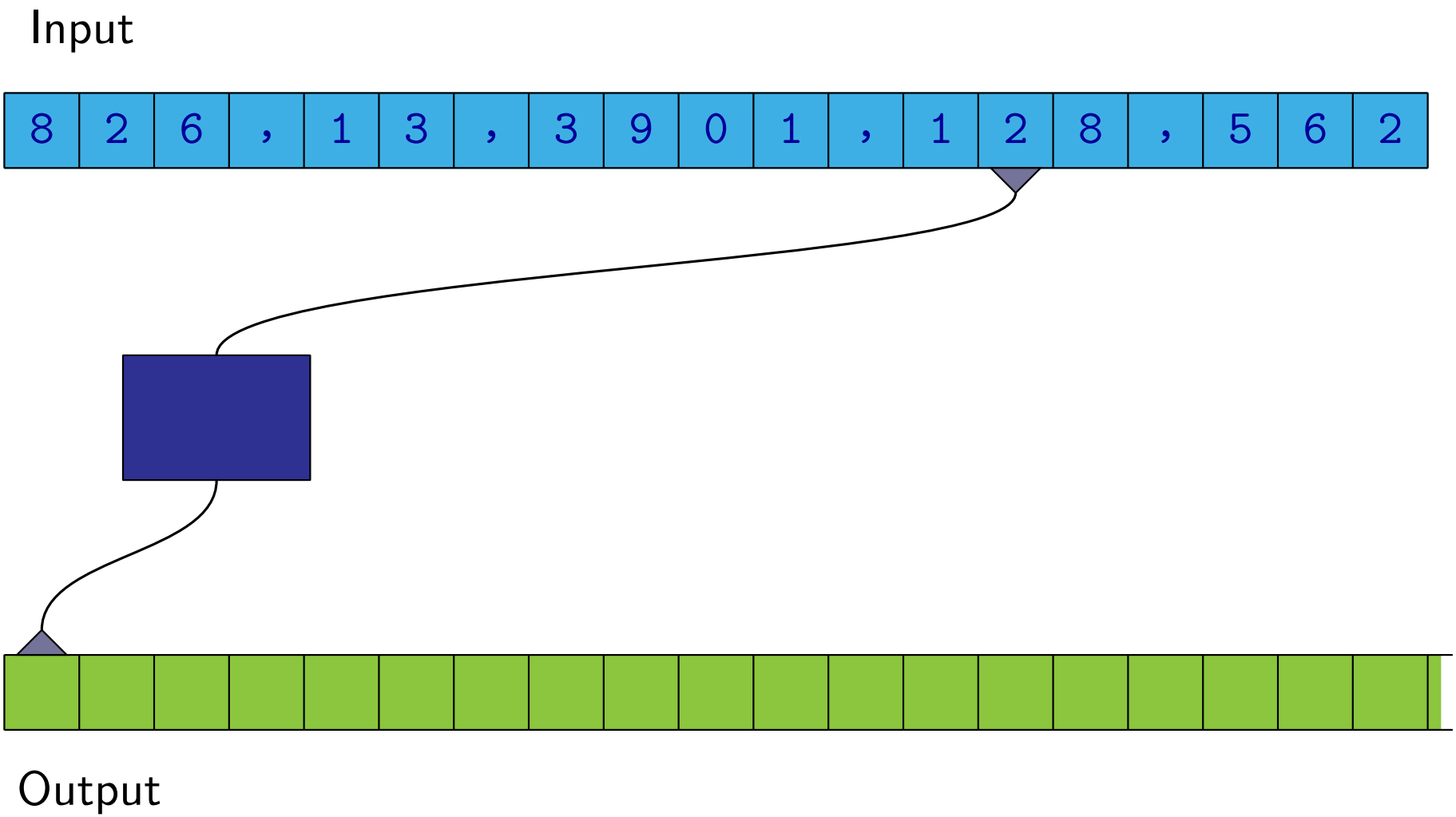
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



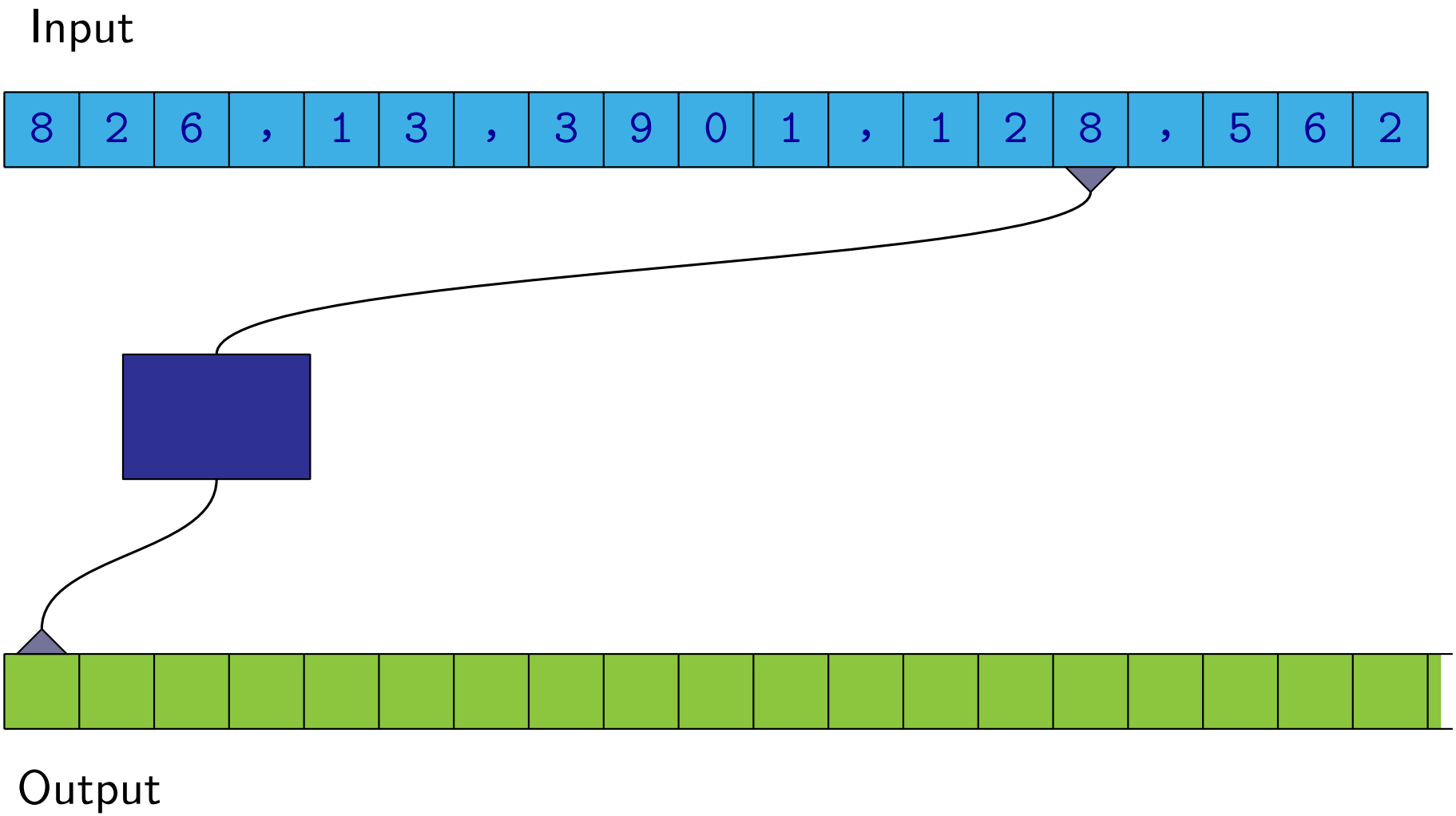
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



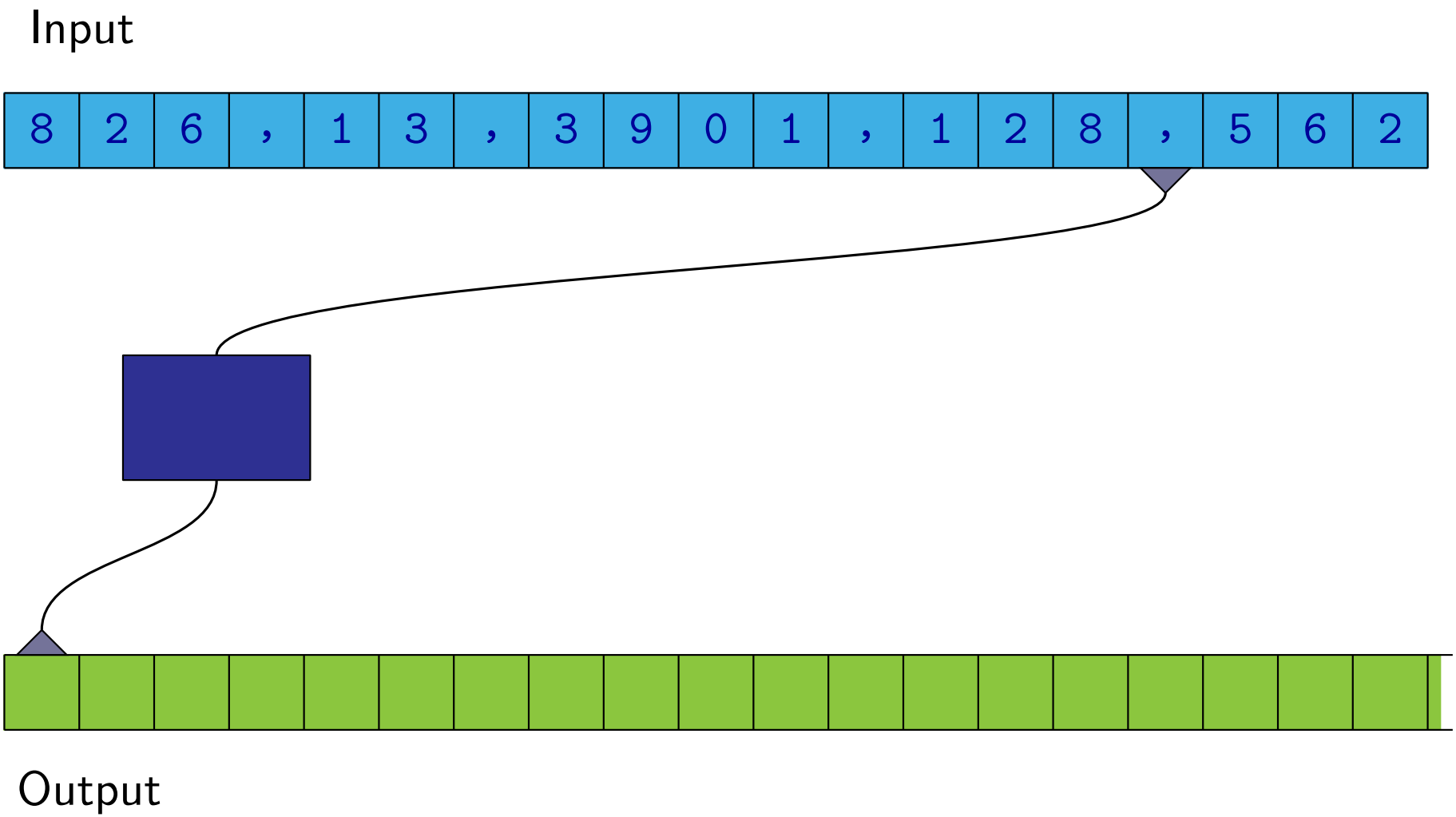
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



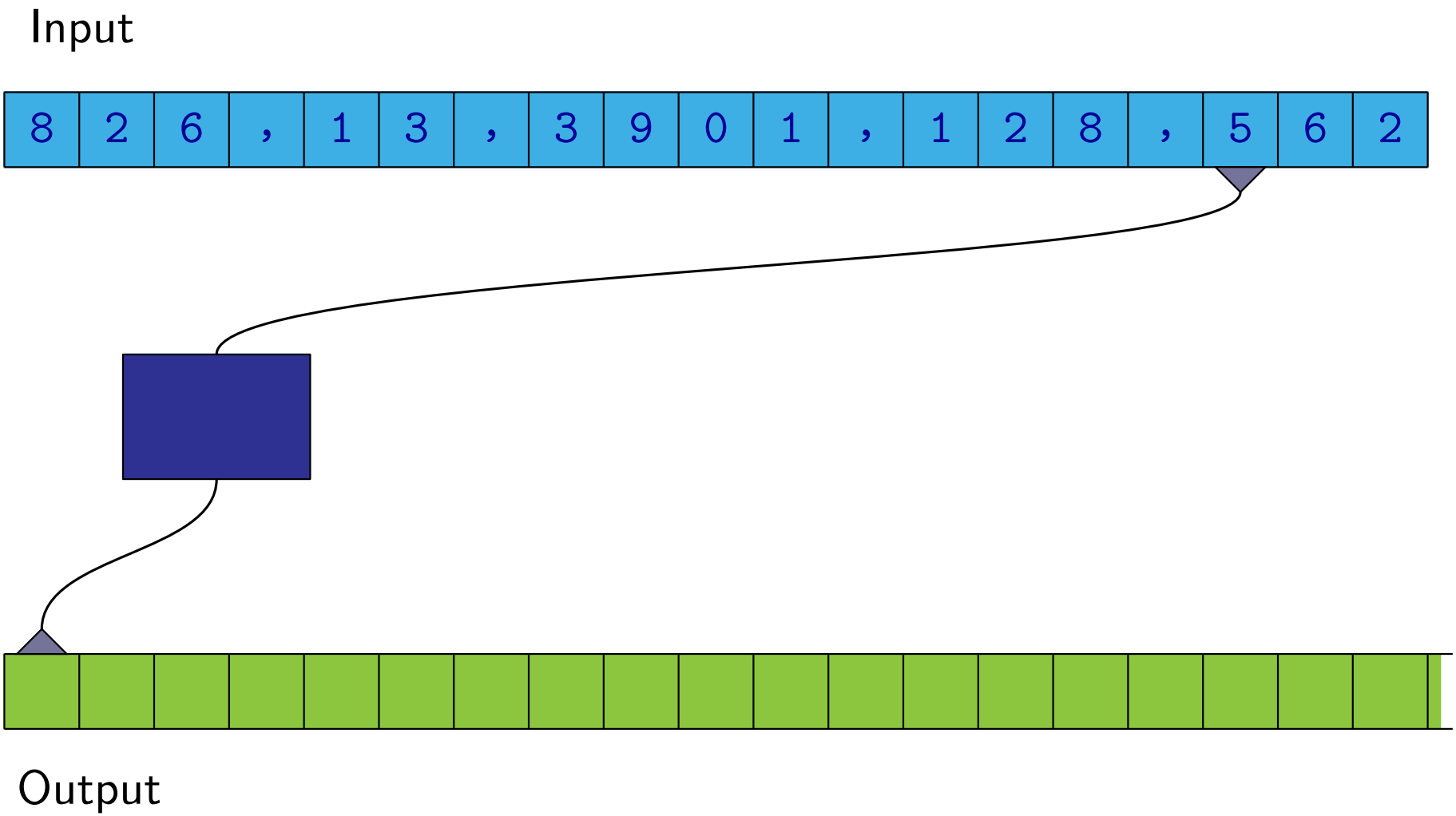
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



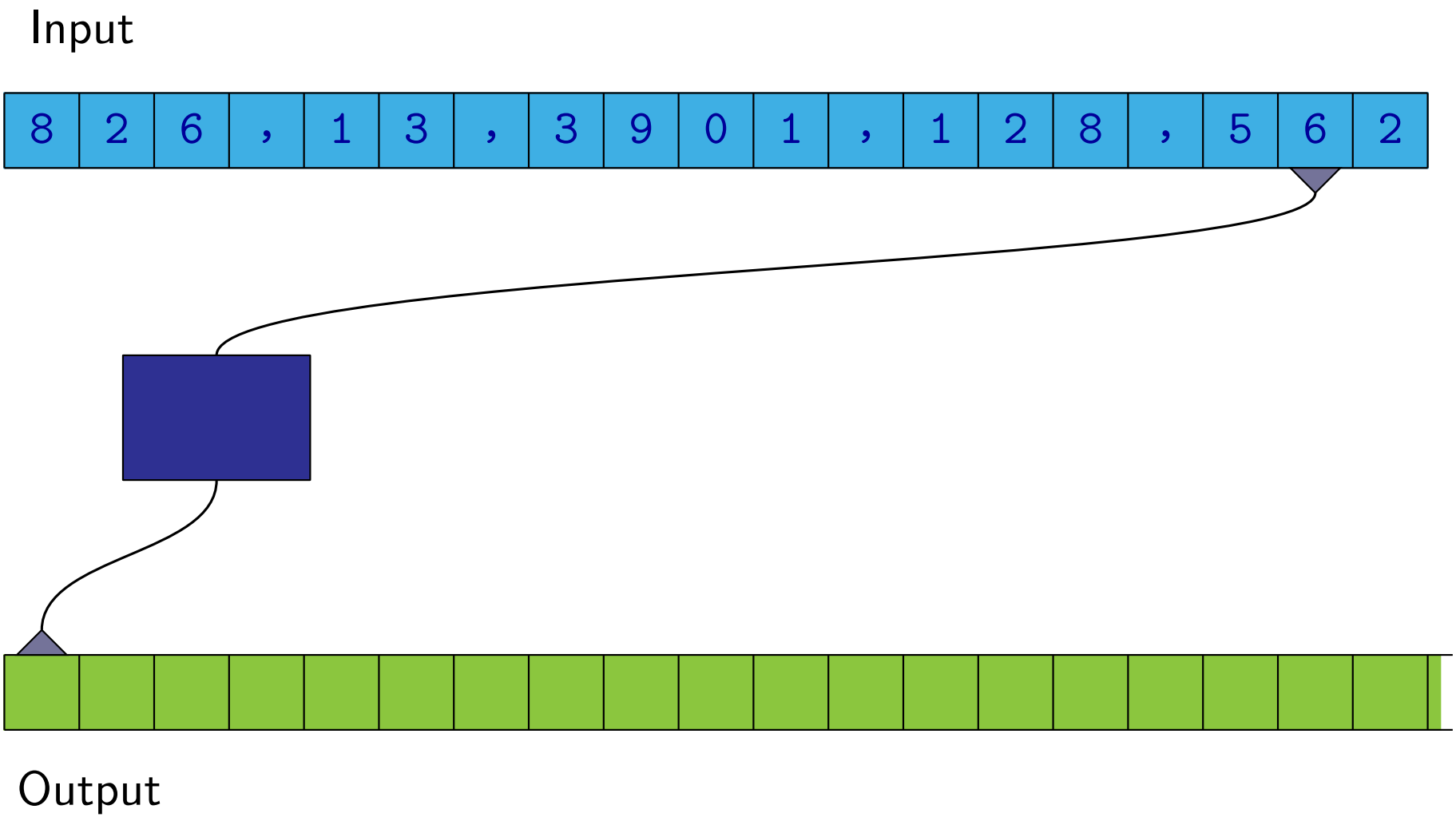
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



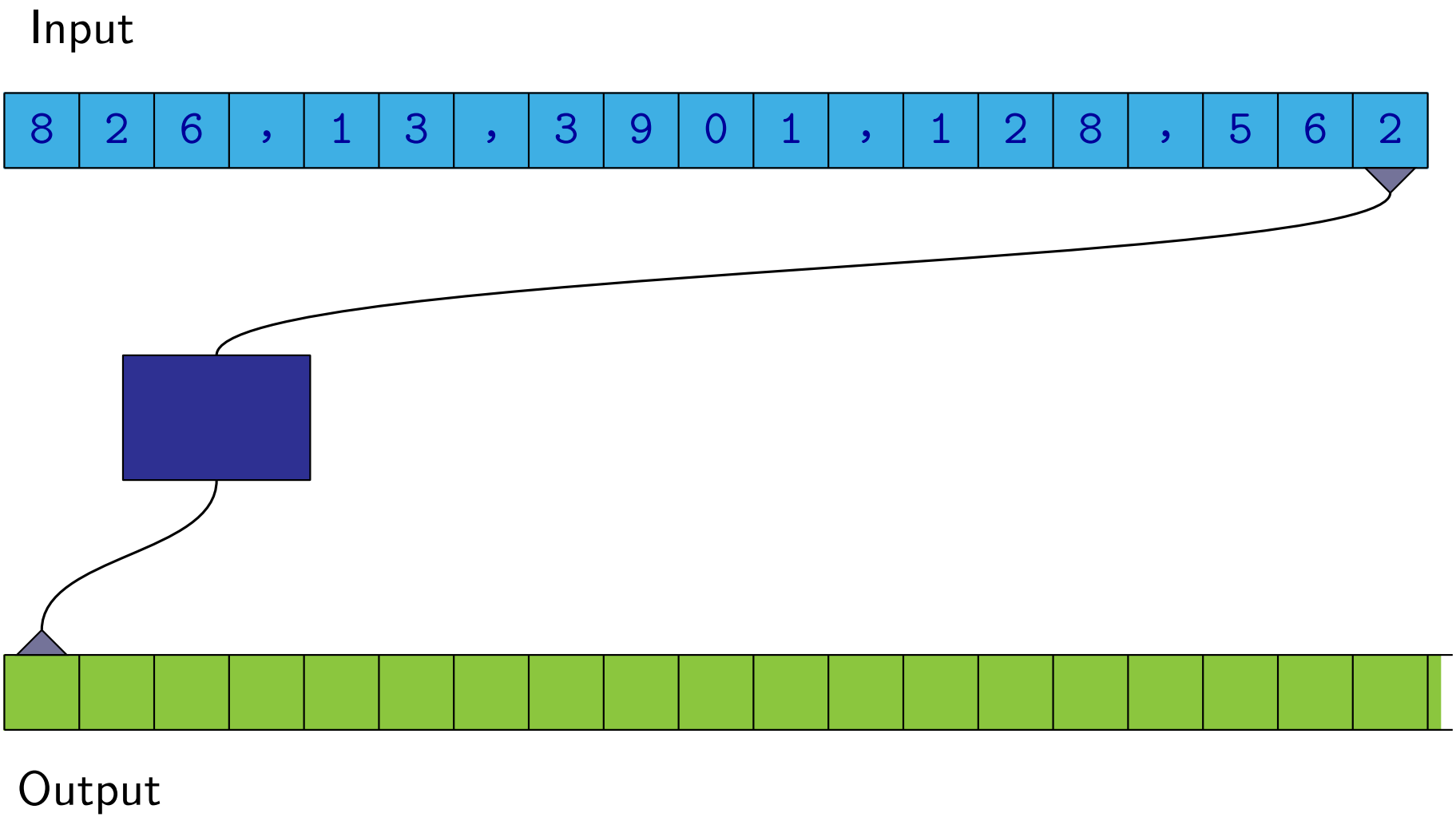
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



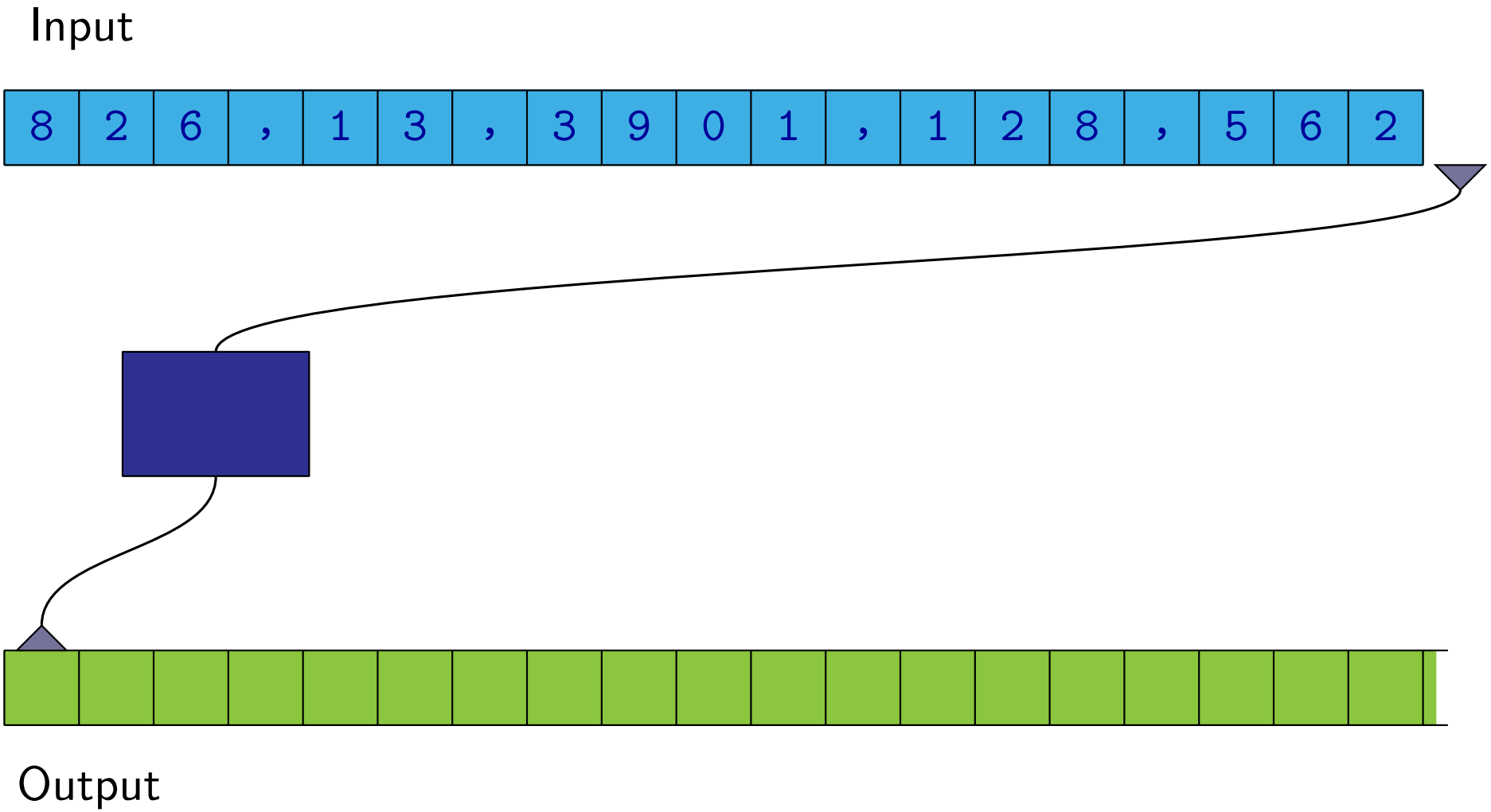
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



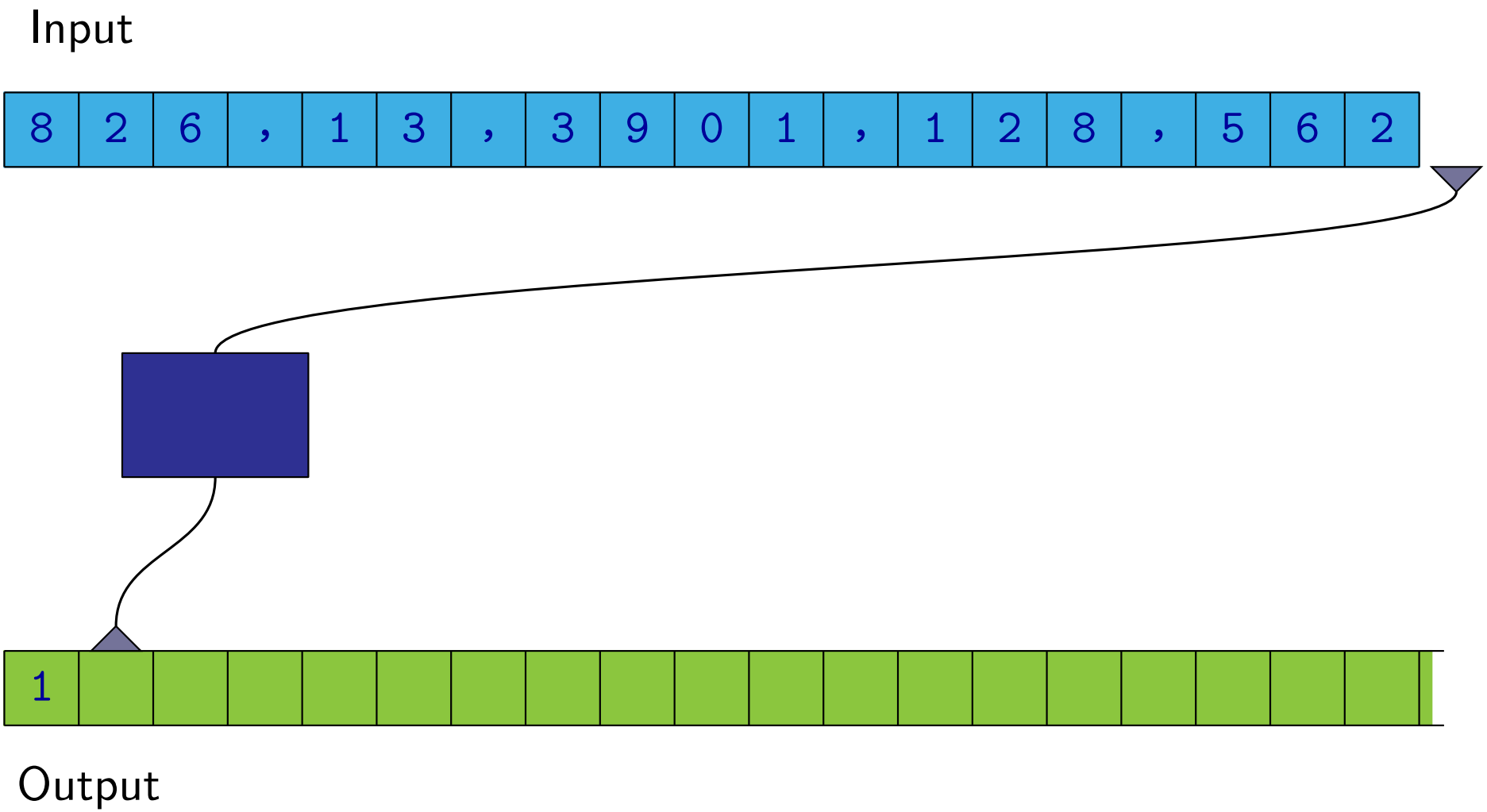
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



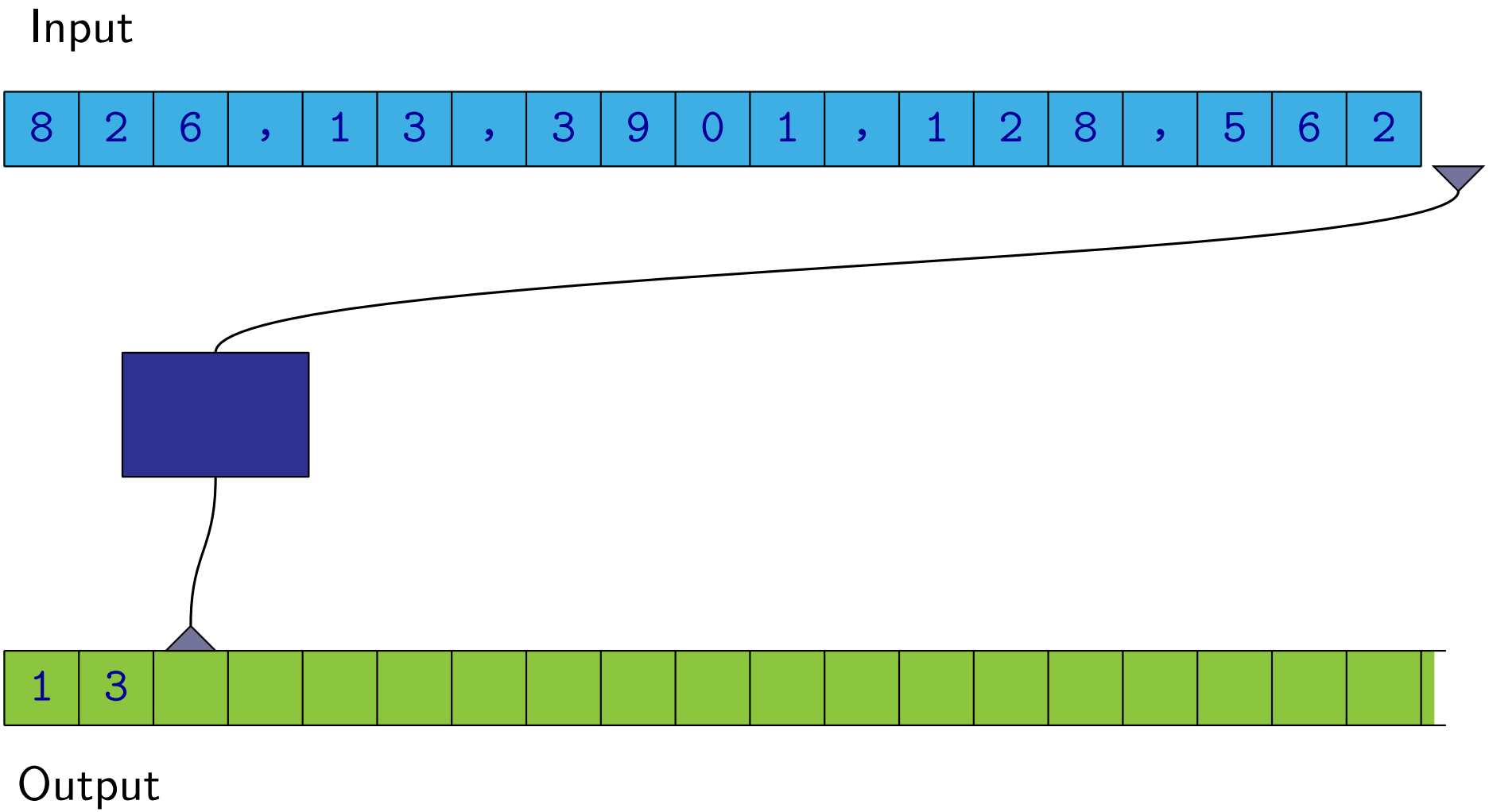
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



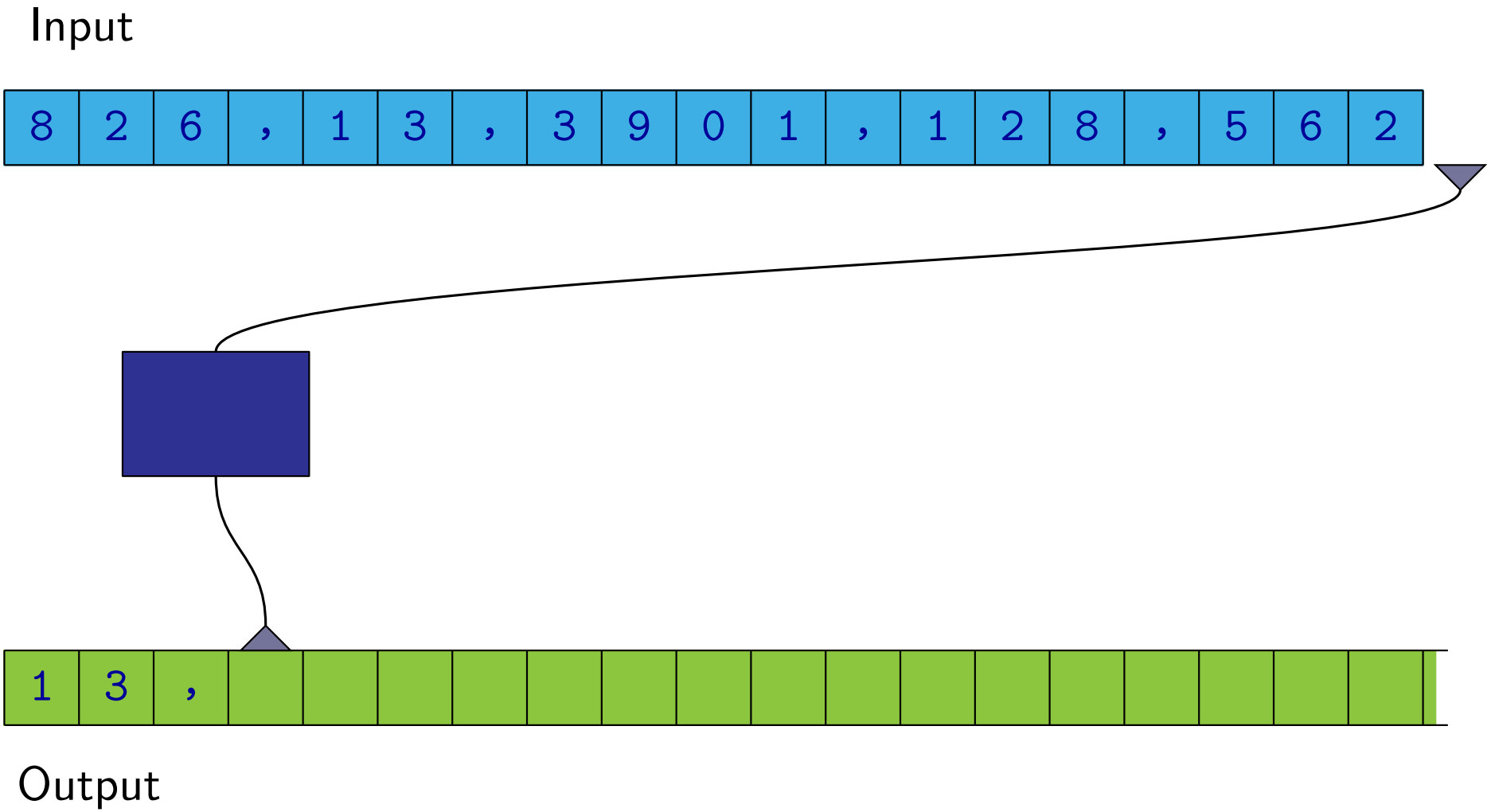
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



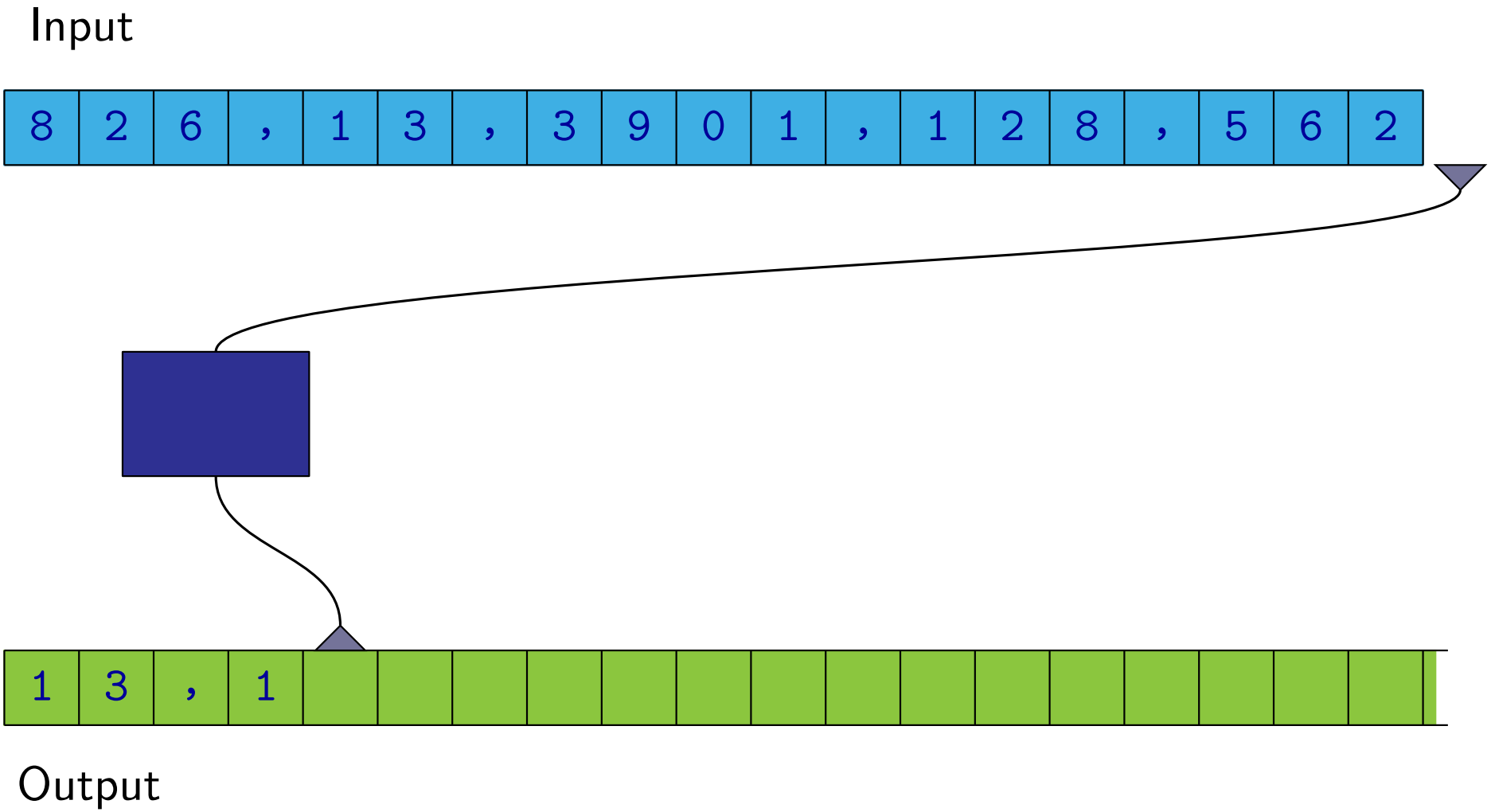
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



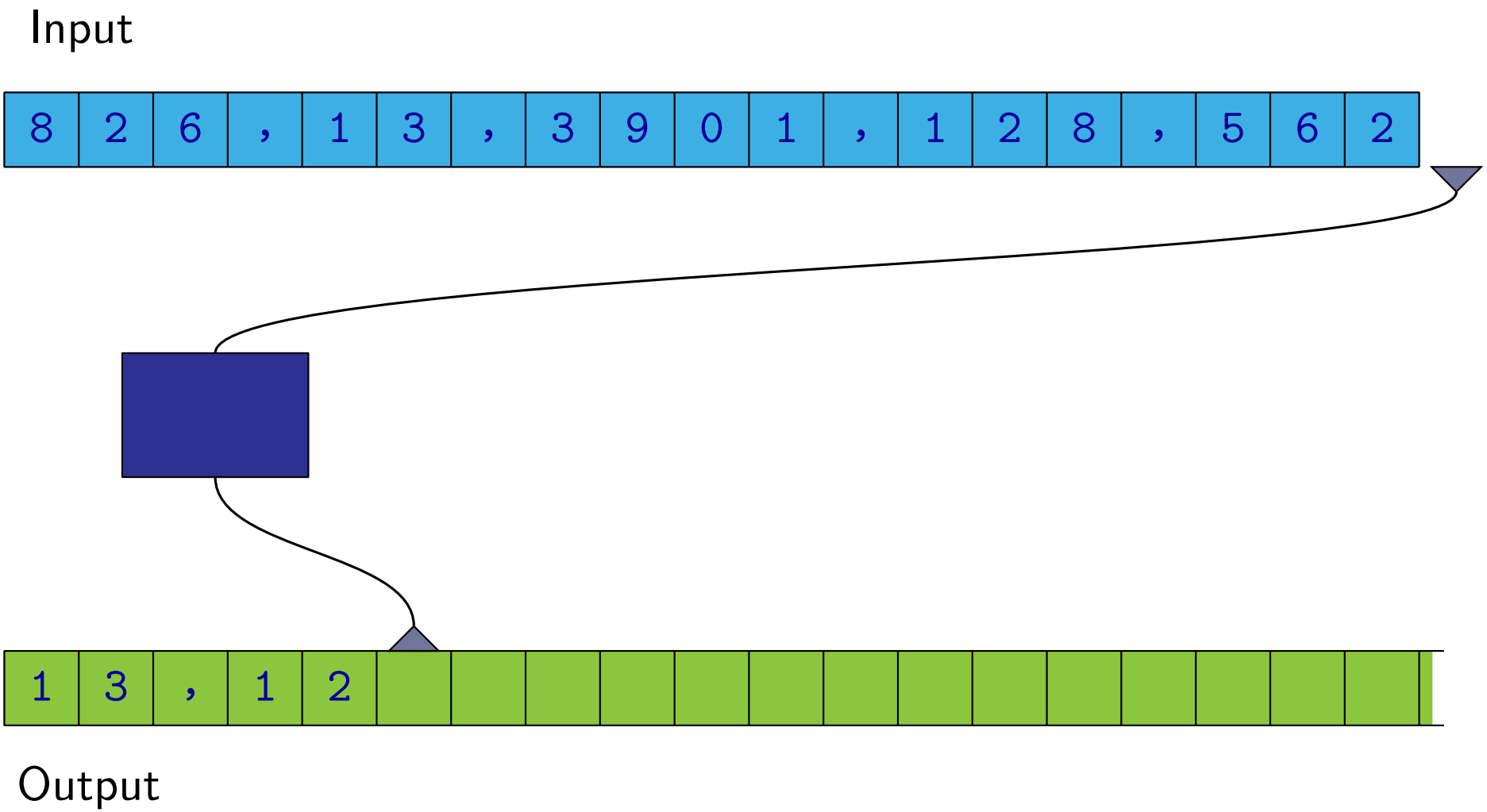
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



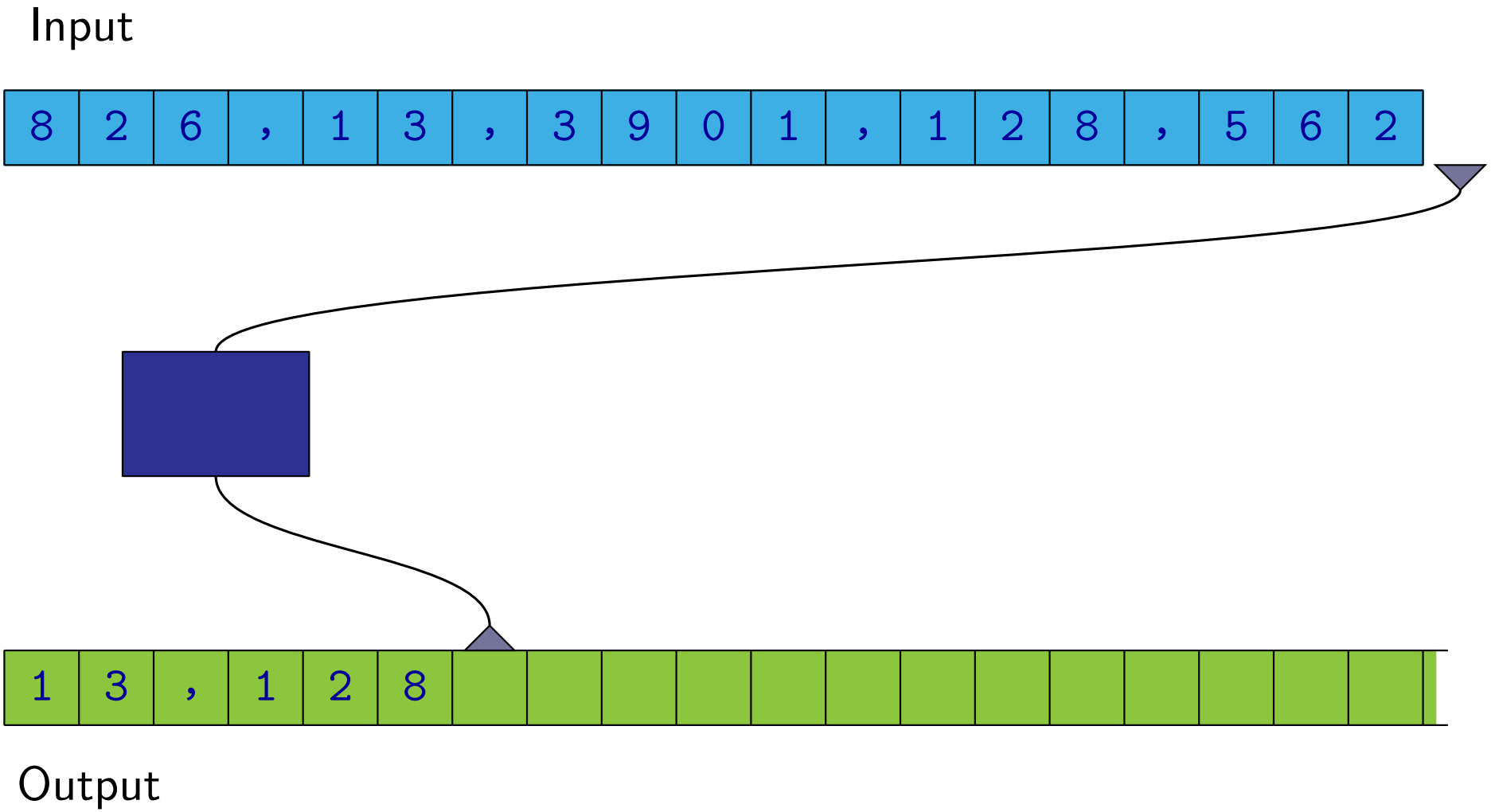
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



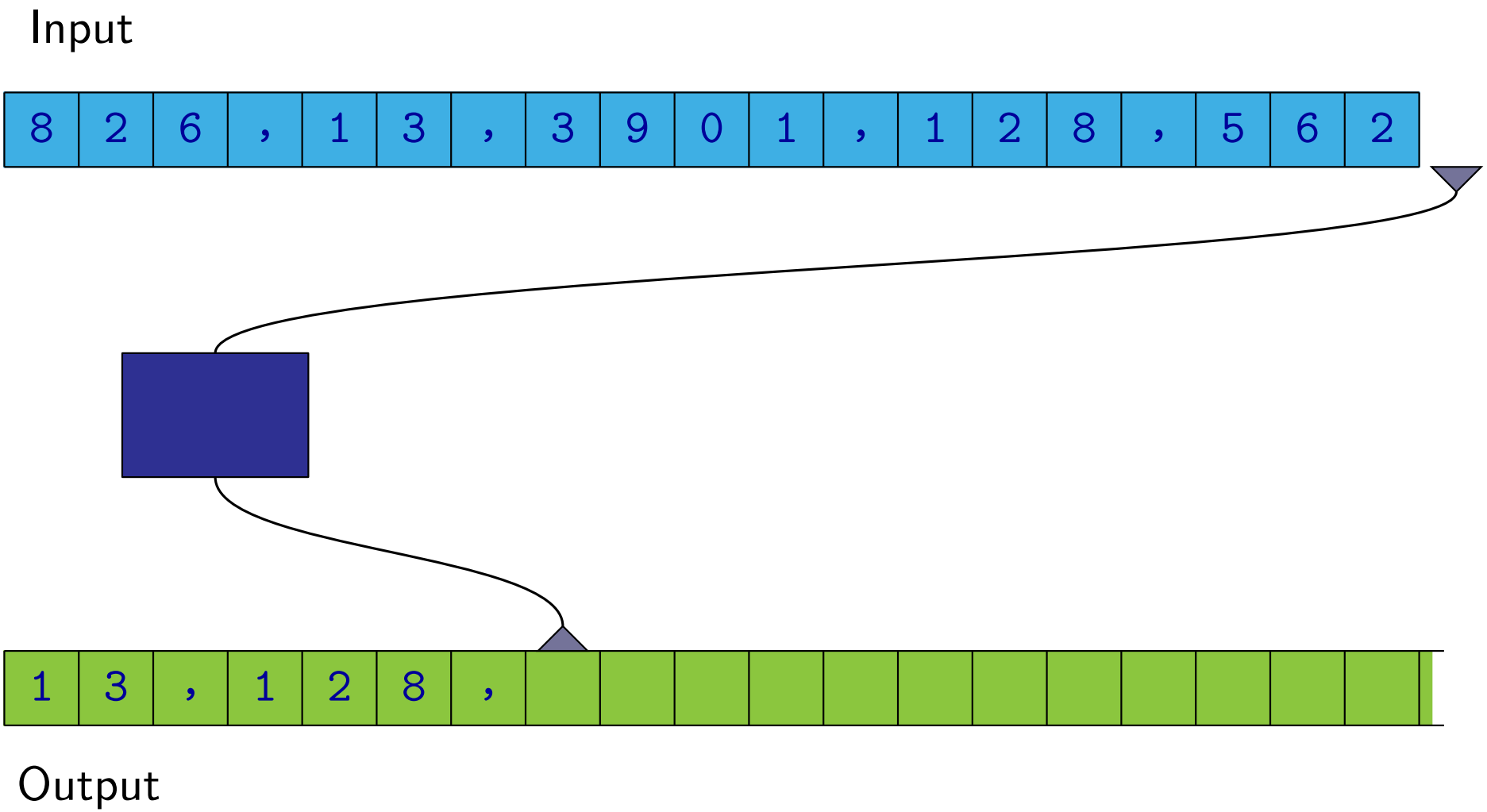
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



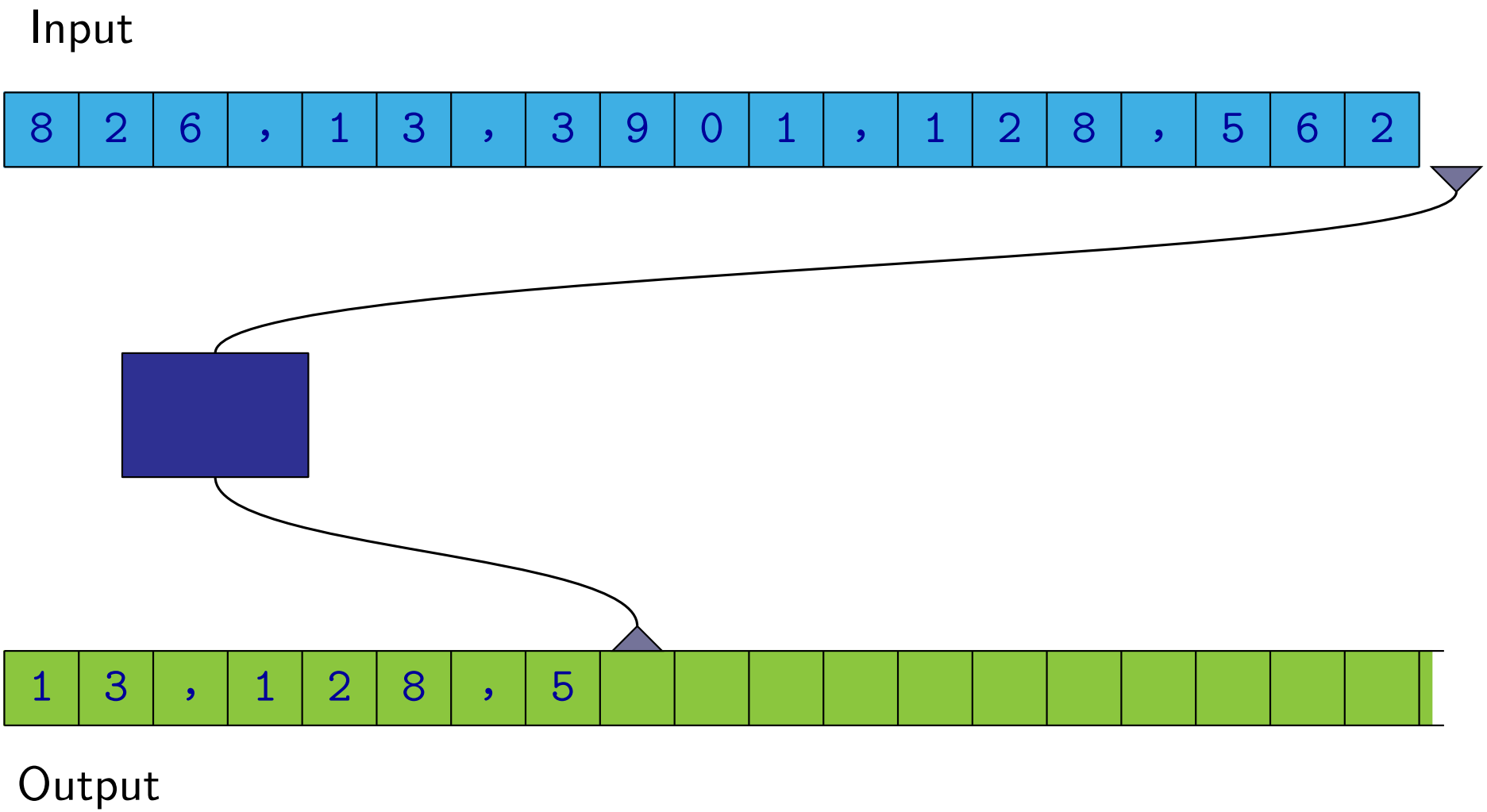
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



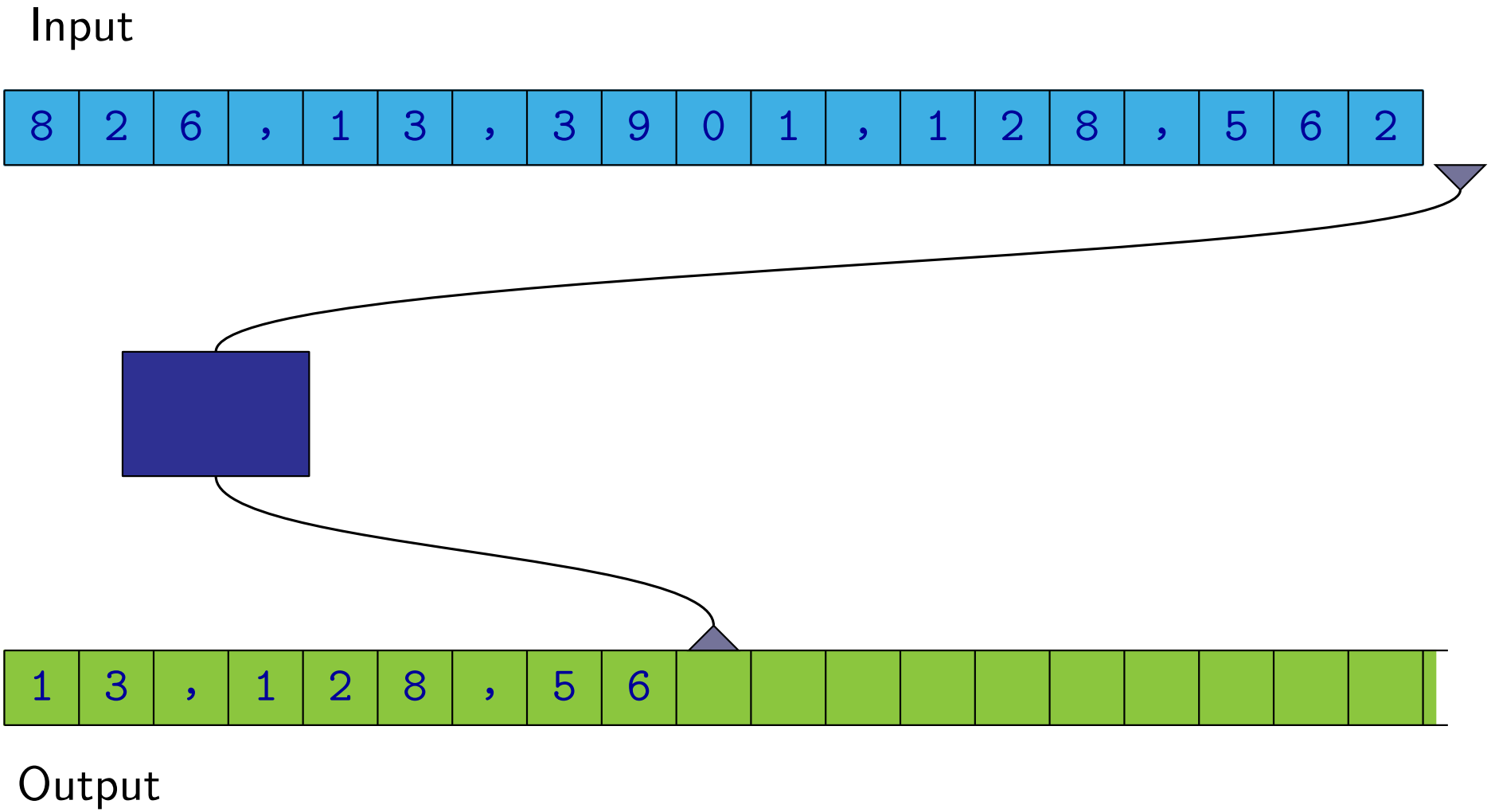
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



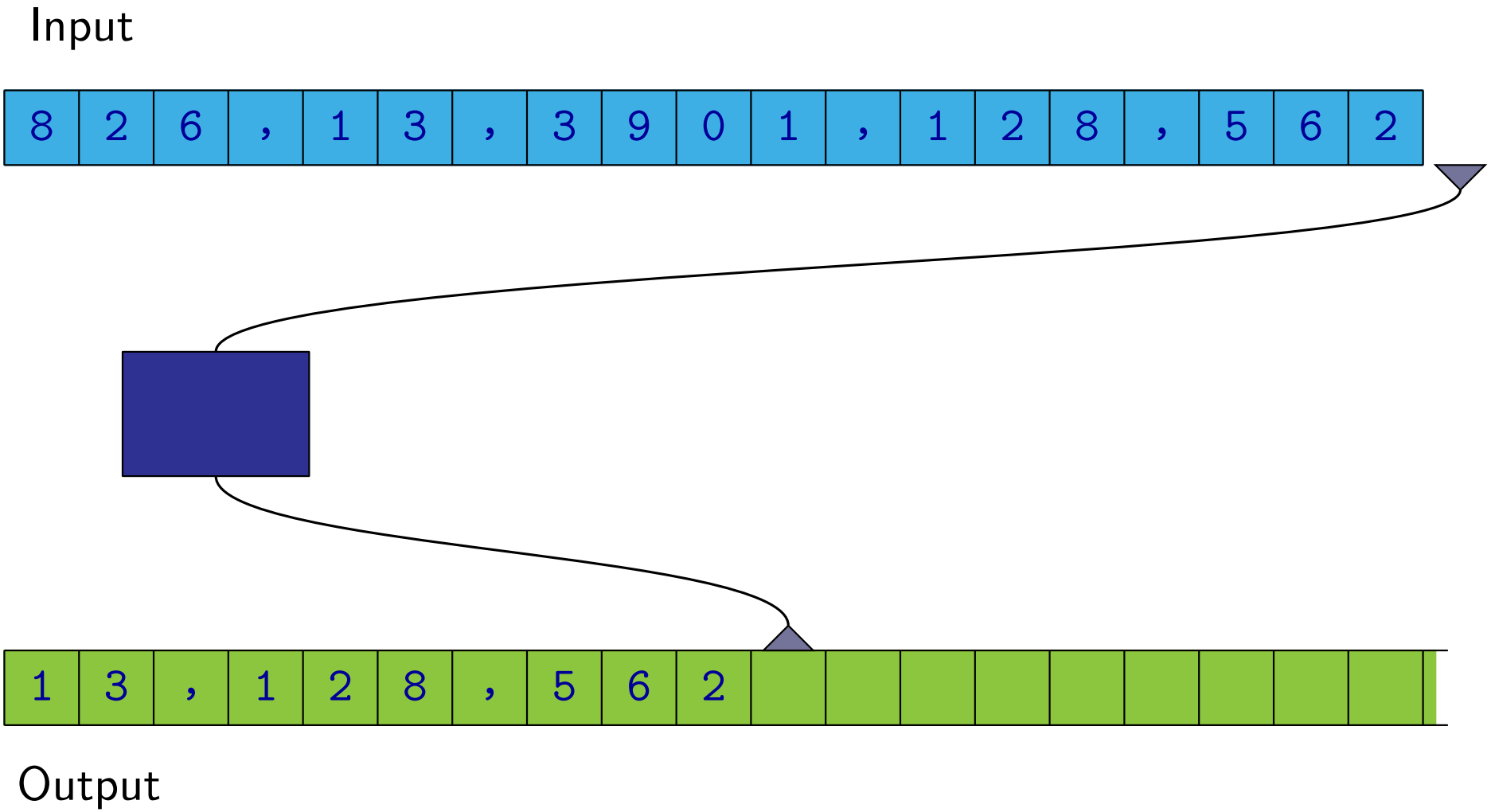
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



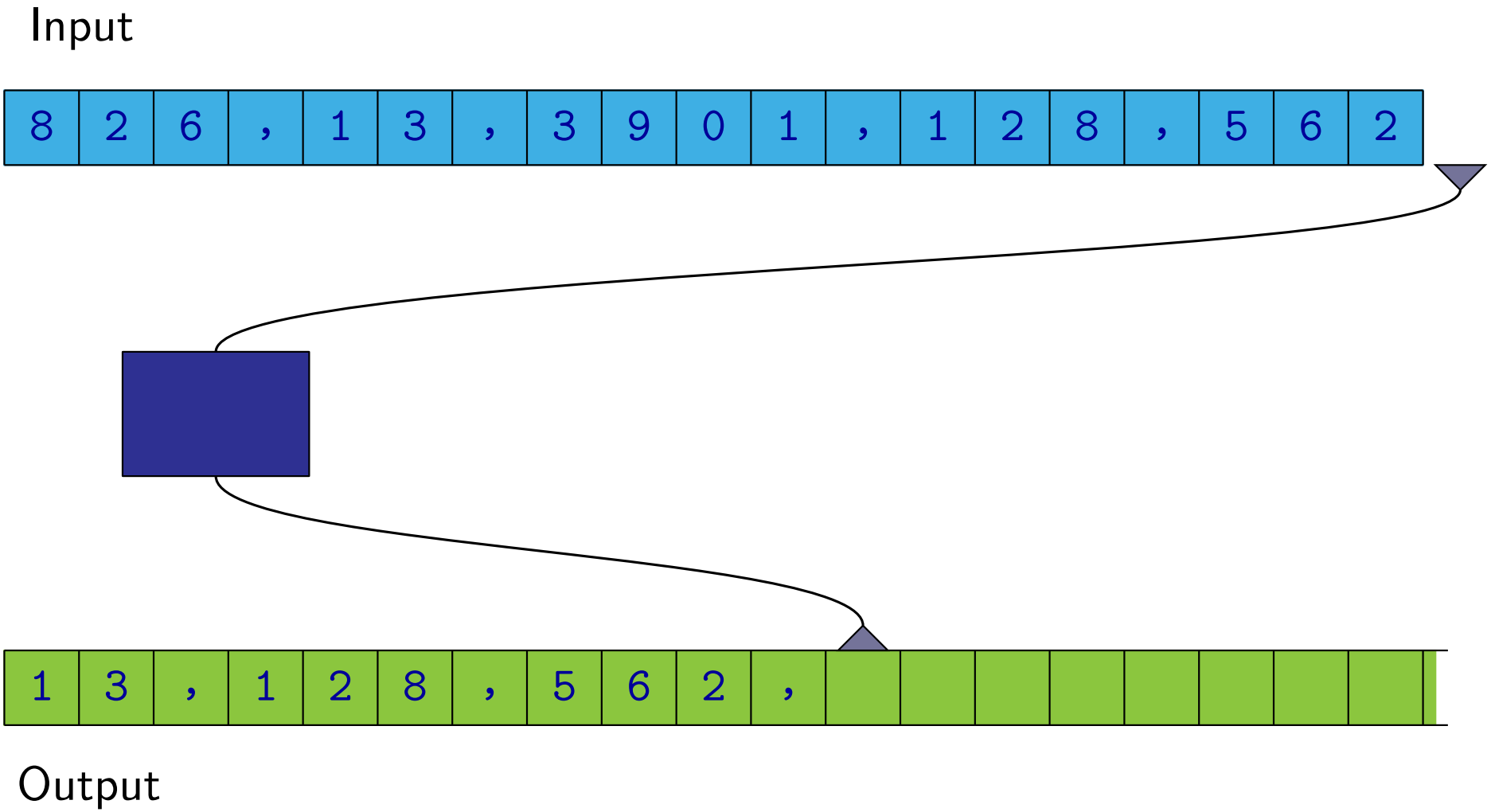
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



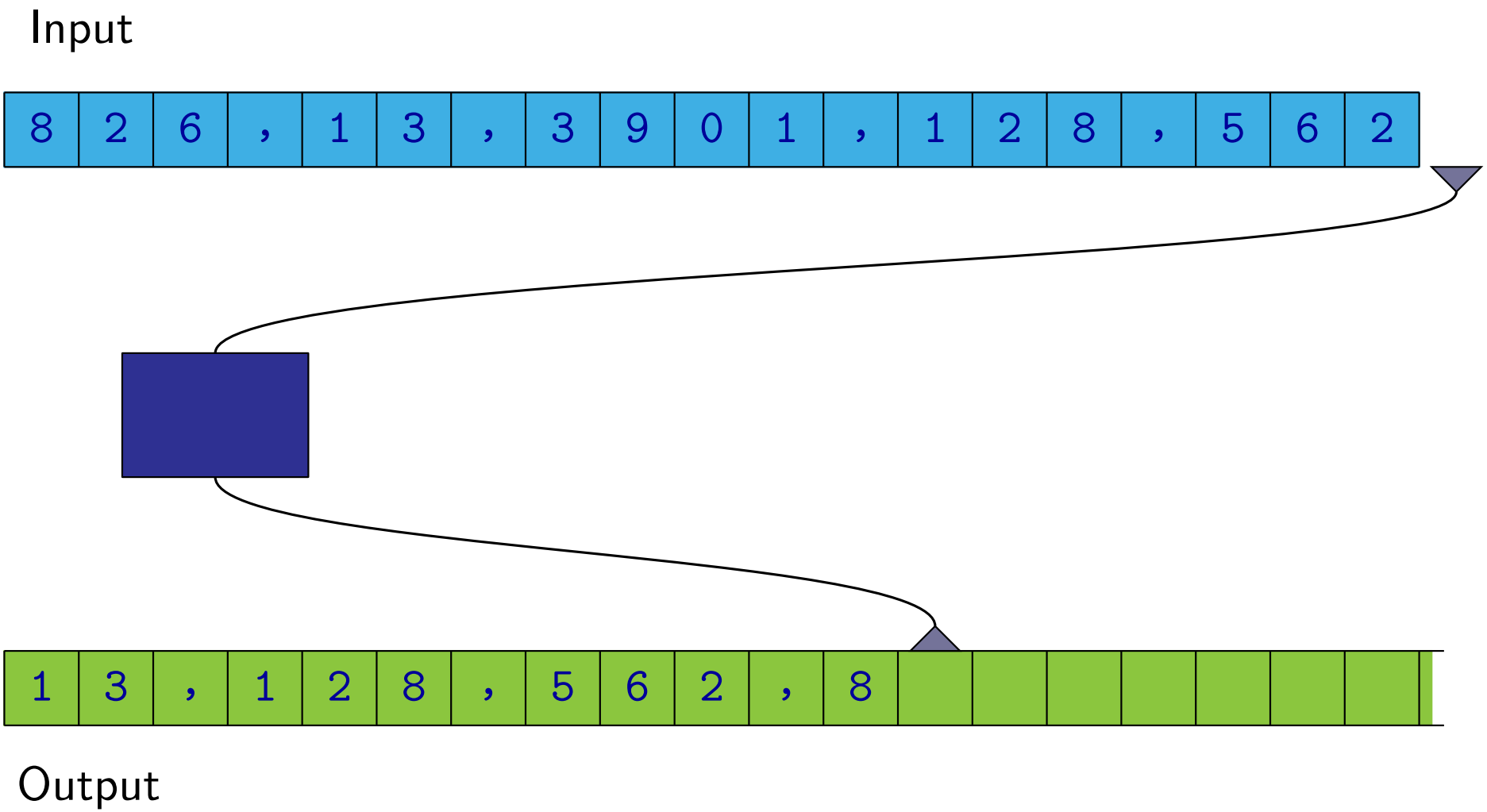
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



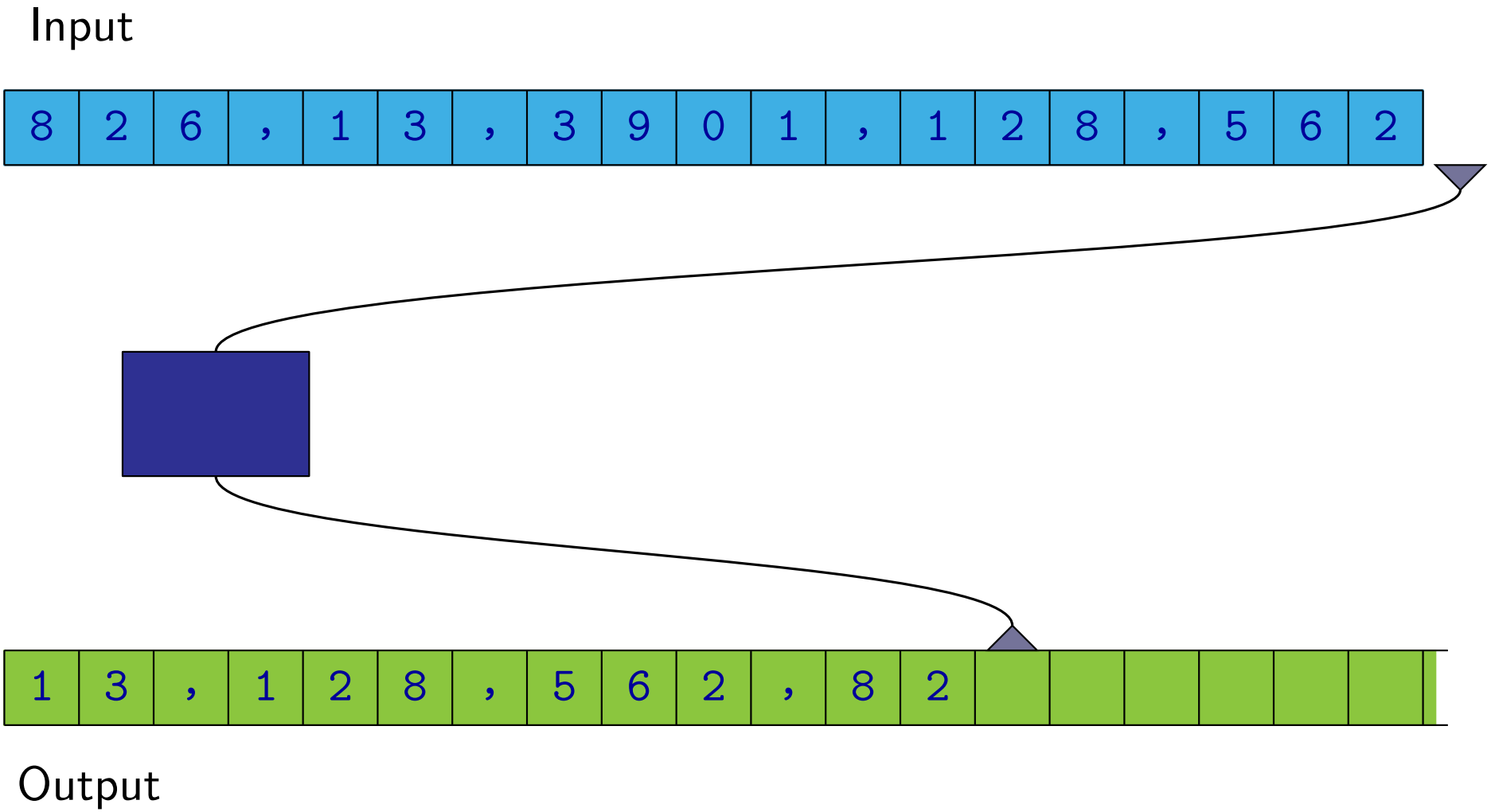
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



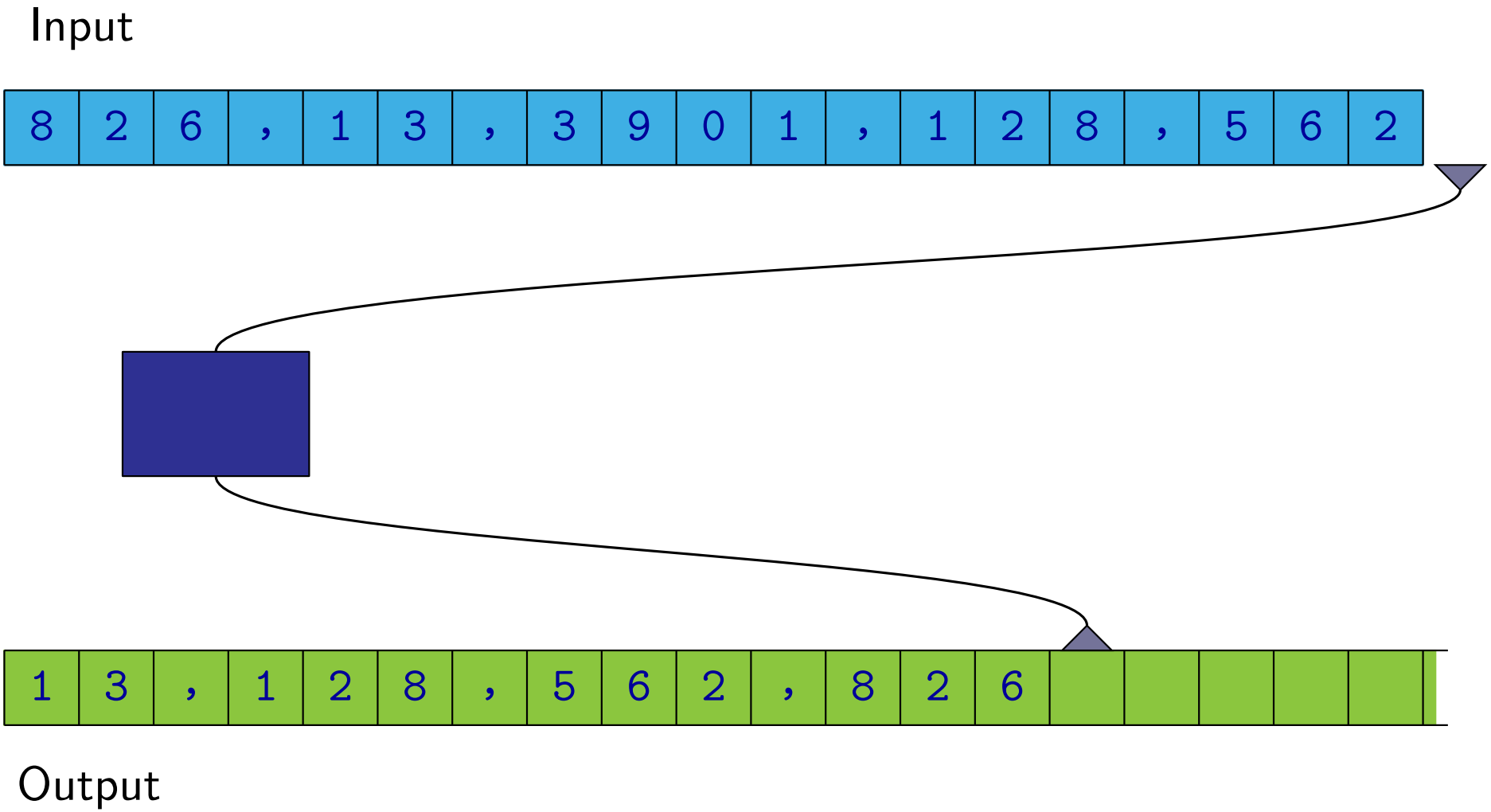
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



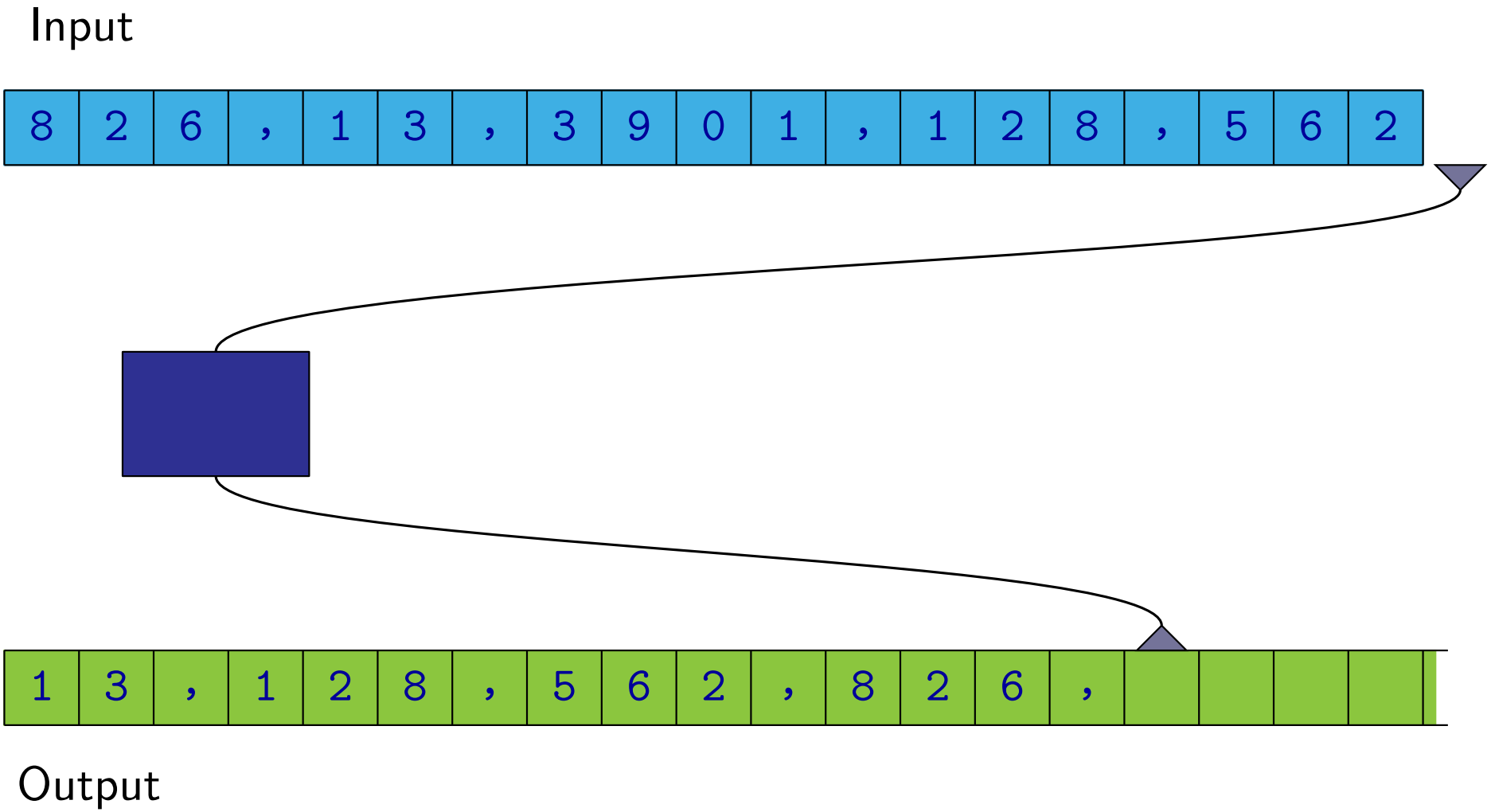
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



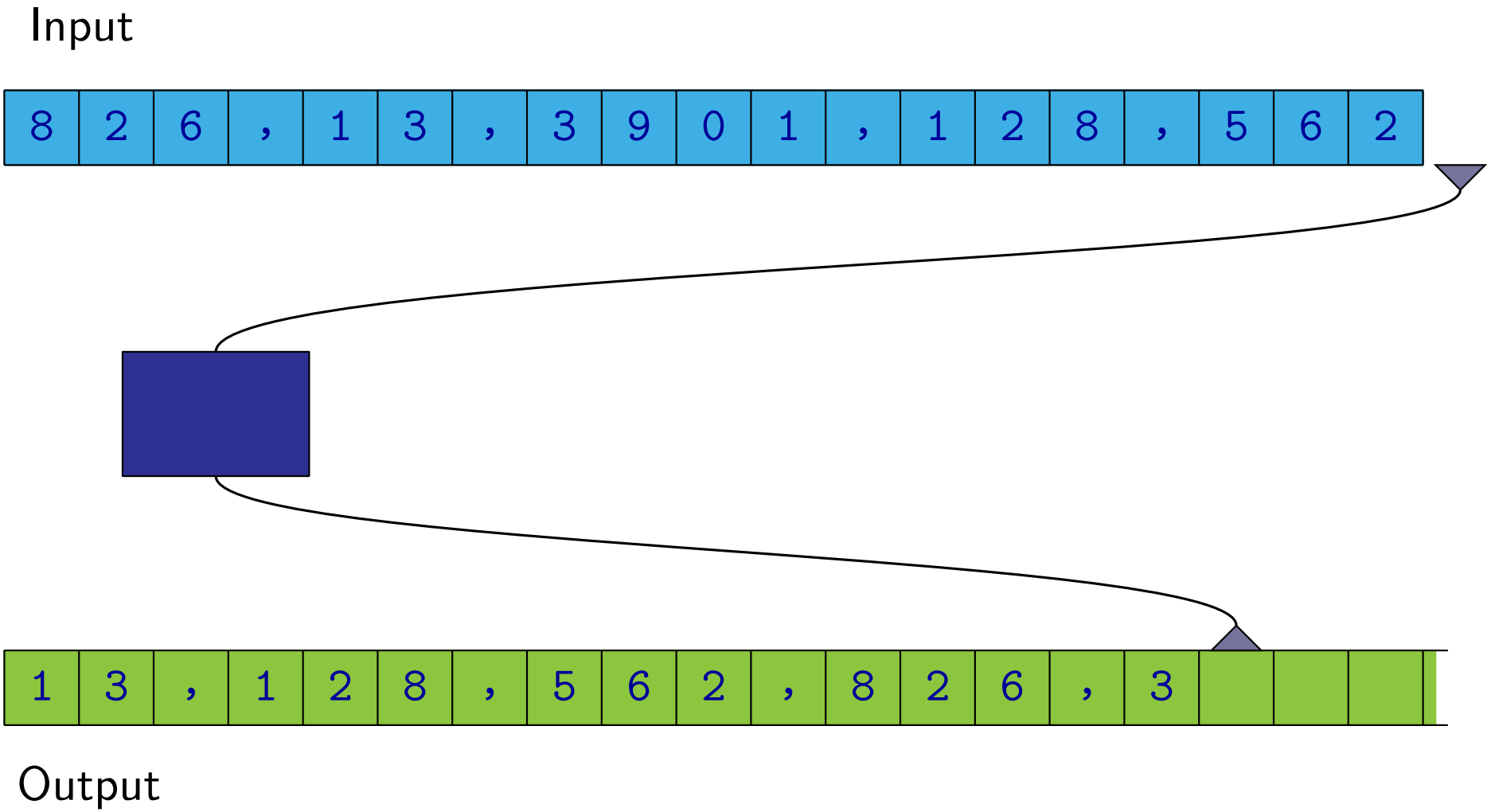
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



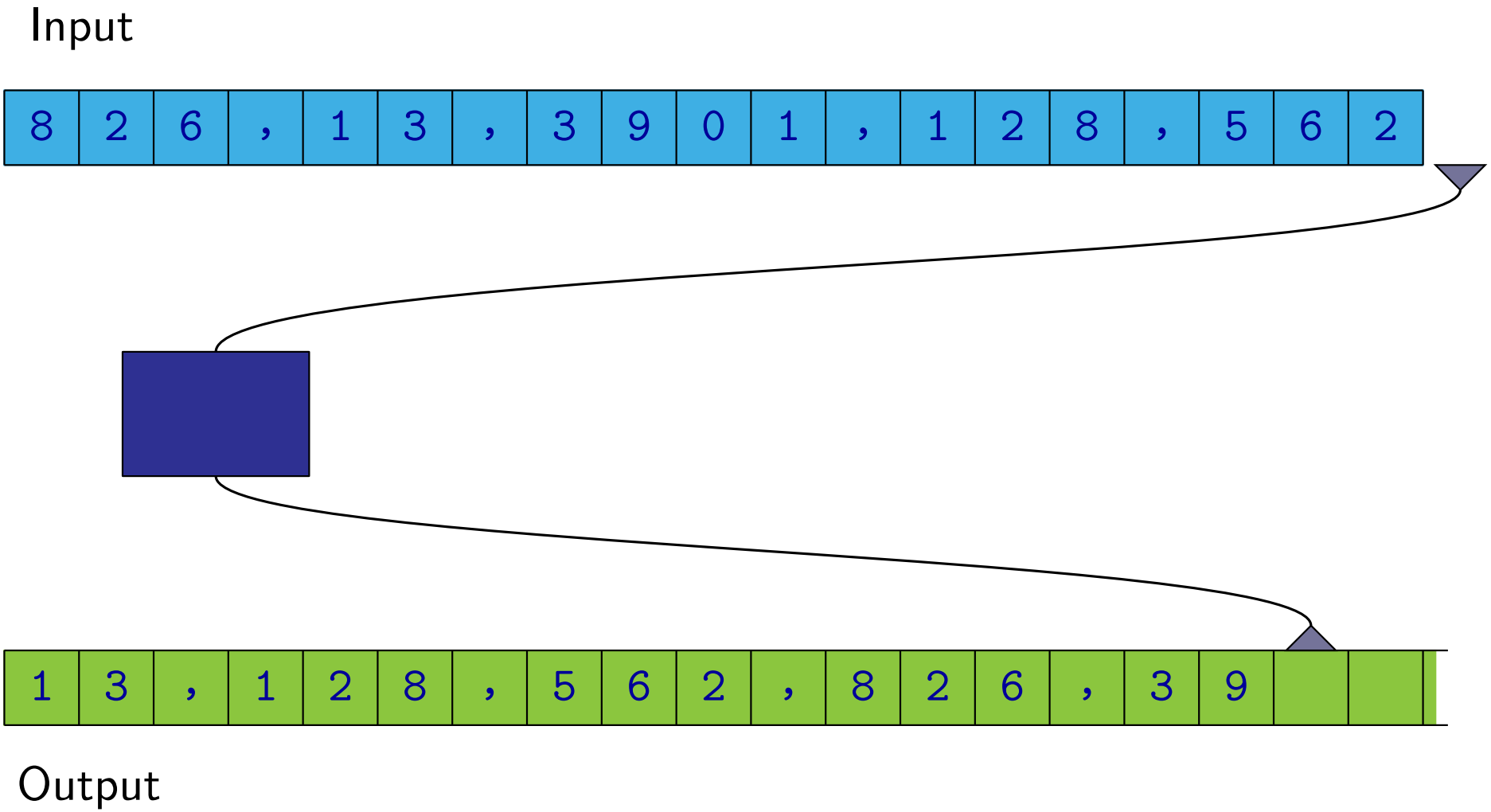
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



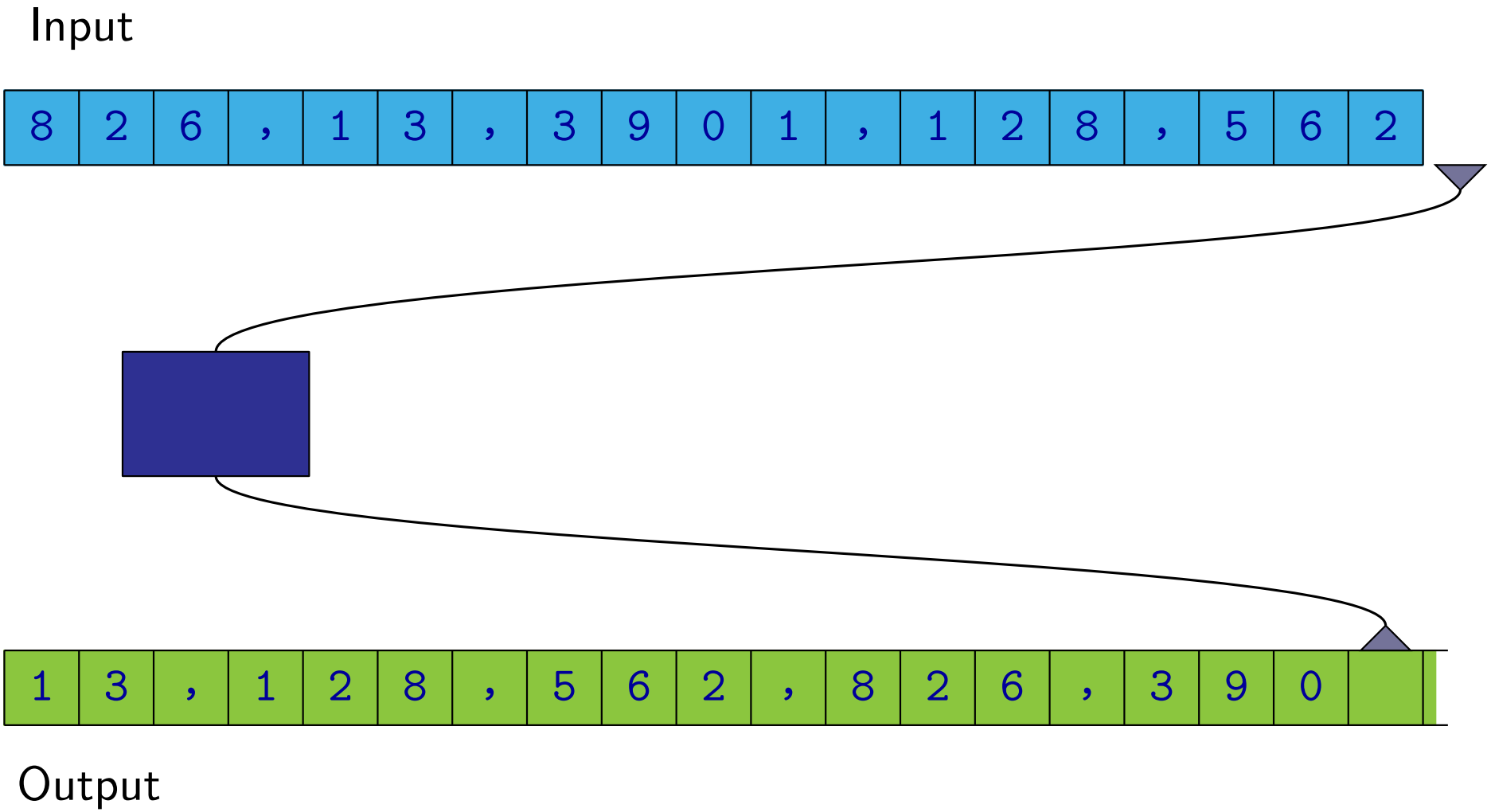
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



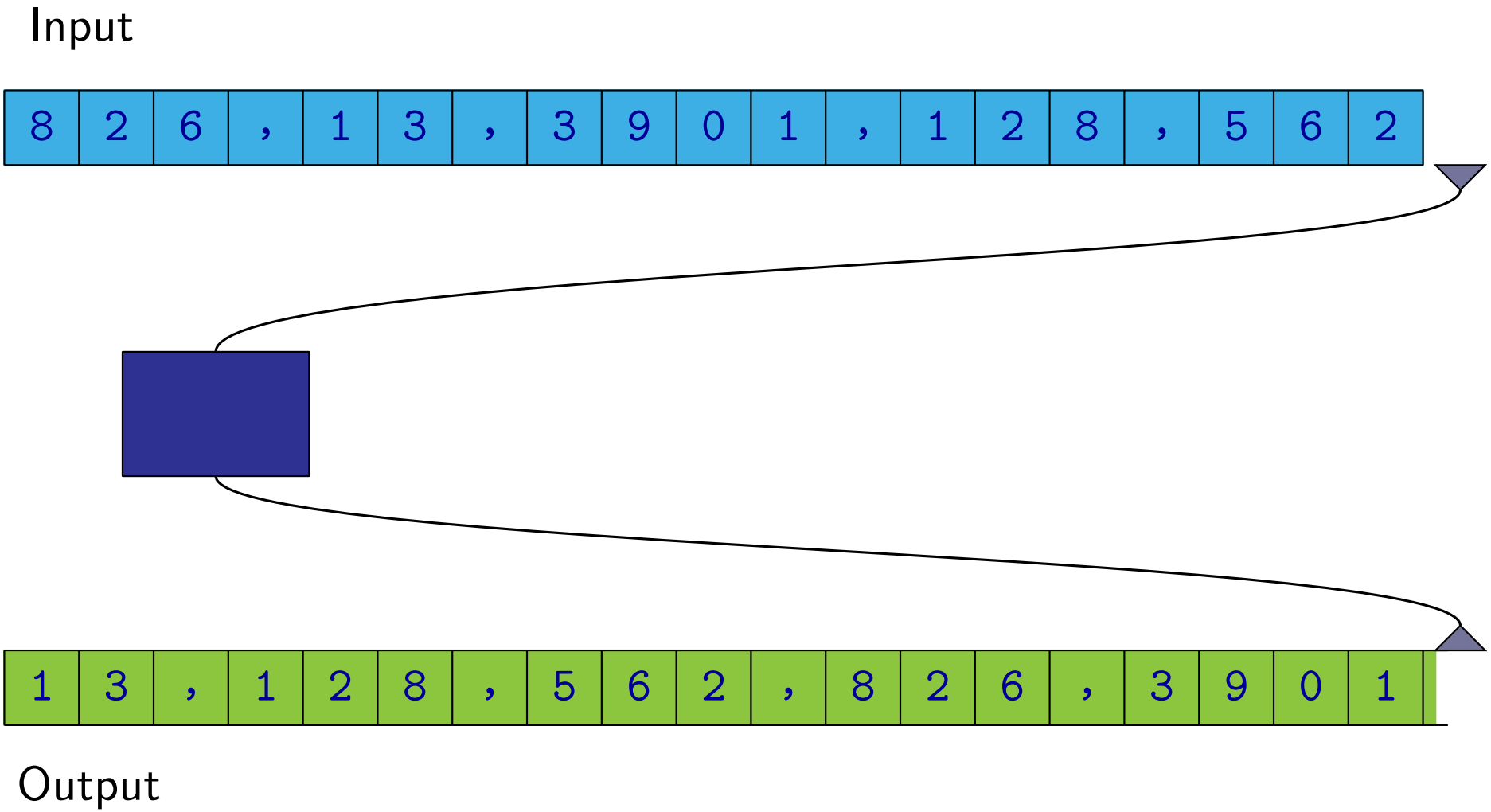
Činnost algoritmu

Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



Činnost algoritmu

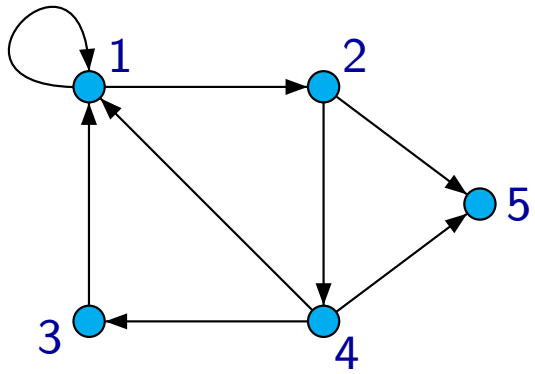
Předpokládáme, že algoritmus je vykonáván nějakým druhem stroje.



Kódování vstupu a výstupu

Příklad: Pokud je vstupem nějakého problému například graf, můžeme ho reprezentovat jako seznam vrcholů a hran:

Například následující graf



můžeme reprezentovat jako slovo

$$(1,2,3,4,5),((1,2),(2,4),(4,3),(3,1),(1,1),(2,5),(4,5),(4,1))$$

v abecedě $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ,, (,)\}$.

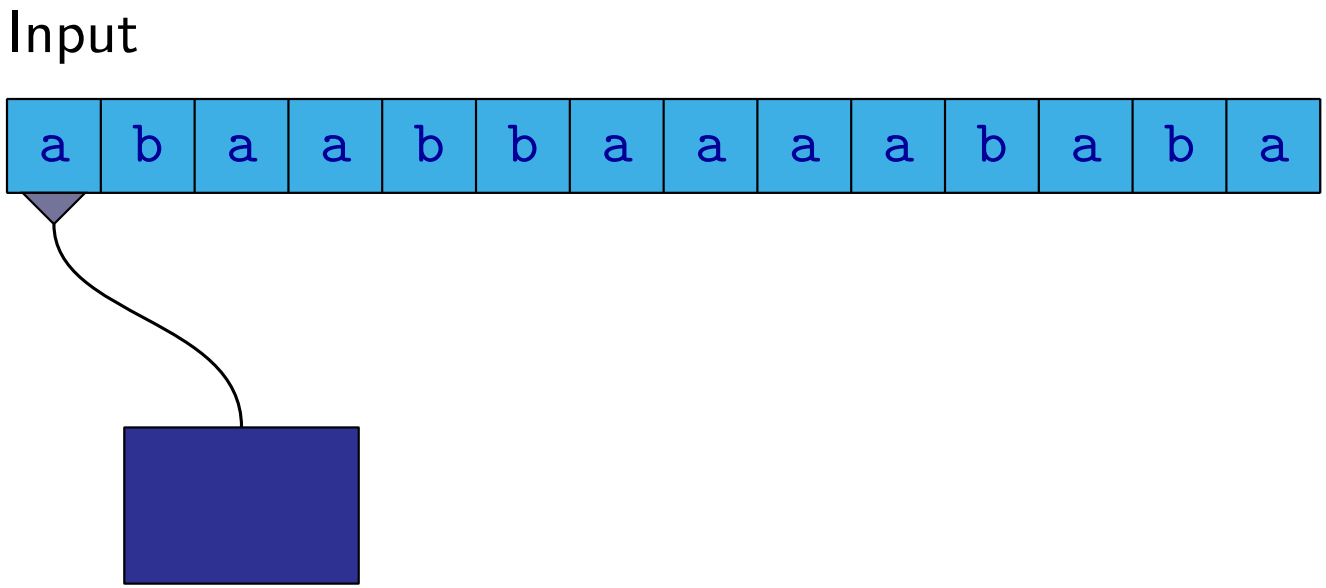
Činnost algoritmu řešícího rozhodovací problém

V případě algoritmu, který řeší nějaký **rozhodovací** problém stačí, když algoritmus vydá jako výstup vždy jen ANO nebo NE.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?



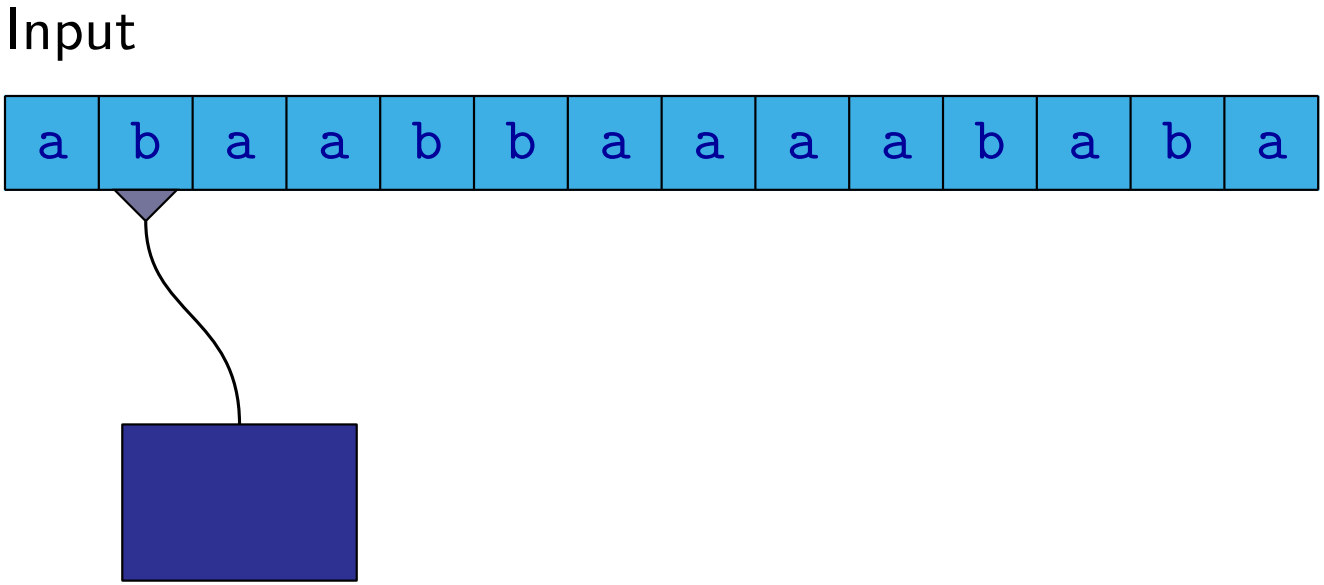
Činnost algoritmu řešícího rozhodovací problém

V případě algoritmu, který řeší nějaký **rozhodovací** problém stačí, když algoritmus vydá jako výstup vždy jen ANO nebo NE.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?



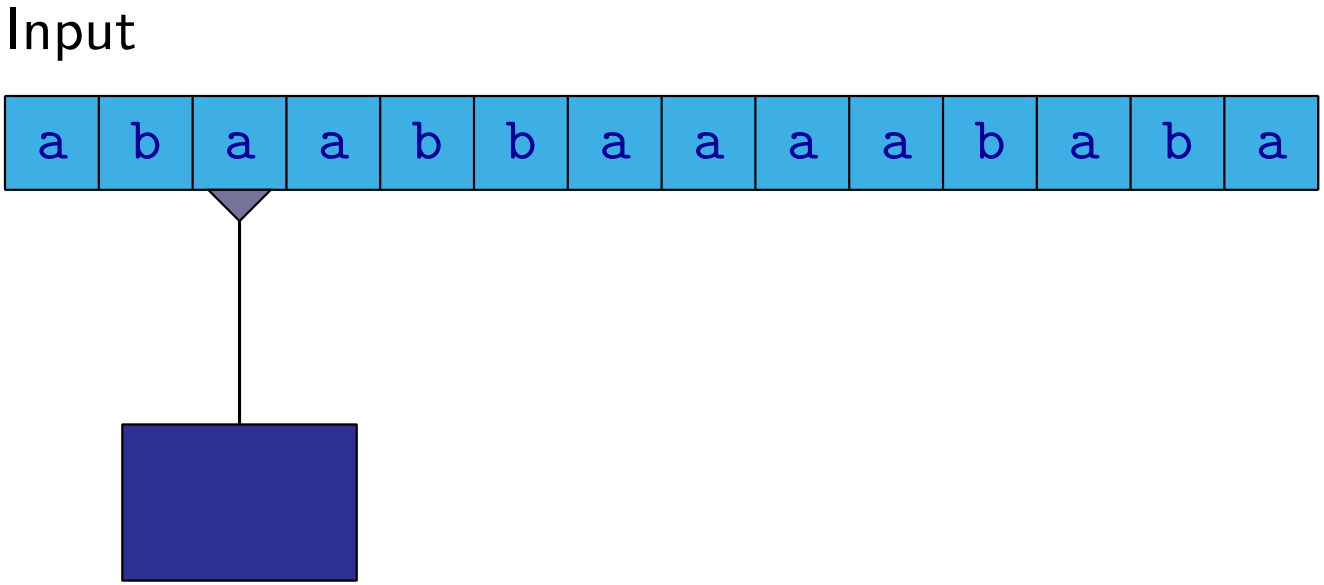
Činnost algoritmu řešícího rozhodovací problém

V případě algoritmu, který řeší nějaký **rozhodovací** problém stačí, když algoritmus vydá jako výstup vždy jen ANO nebo NE.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?



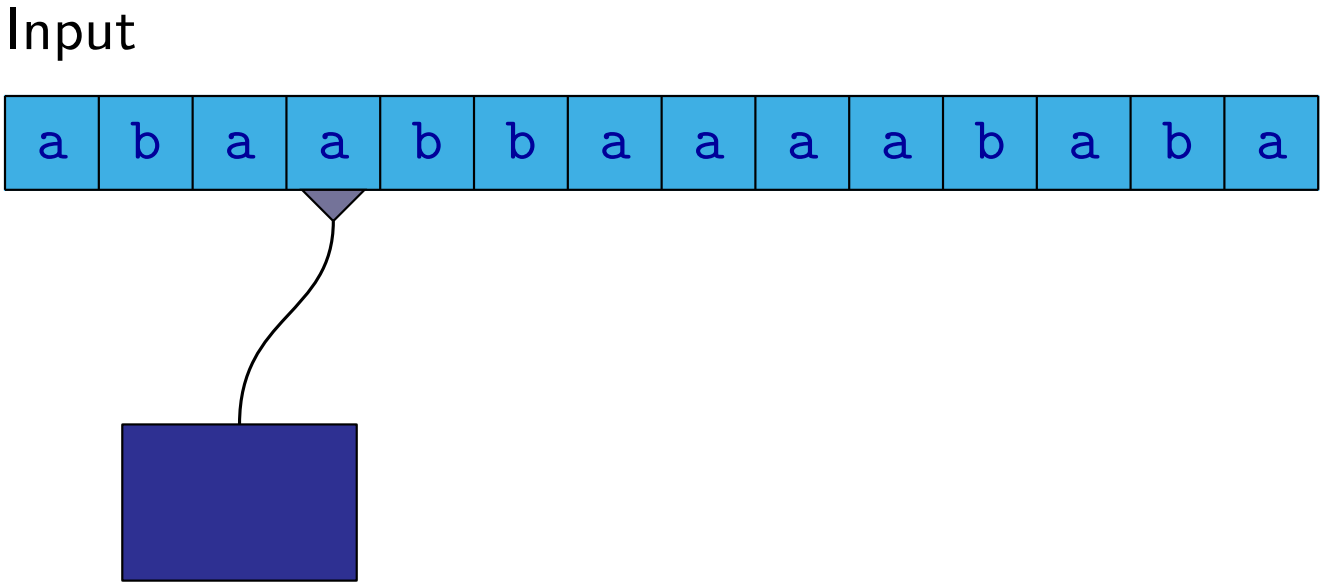
Činnost algoritmu řešícího rozhodovací problém

V případě algoritmu, který řeší nějaký **rozhodovací** problém stačí, když algoritmus vydá jako výstup vždy jen ANO nebo NE.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?



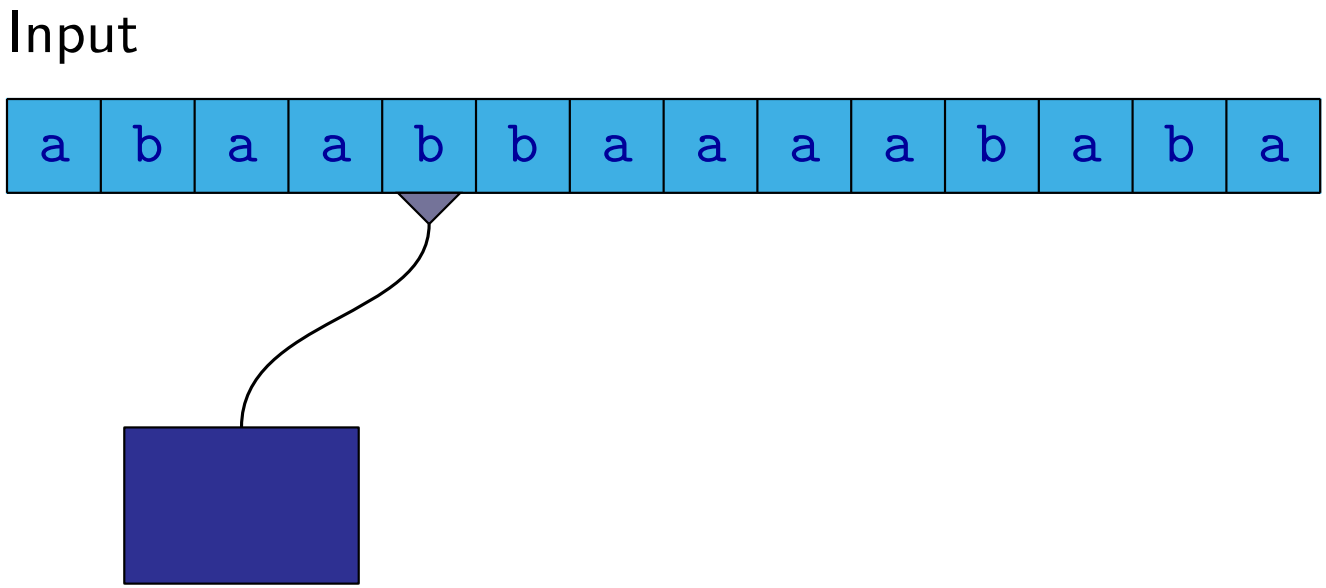
Činnost algoritmu řešícího rozhodovací problém

V případě algoritmu, který řeší nějaký **rozhodovací** problém stačí, když algoritmus vydá jako výstup vždy jen ANO nebo NE.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?



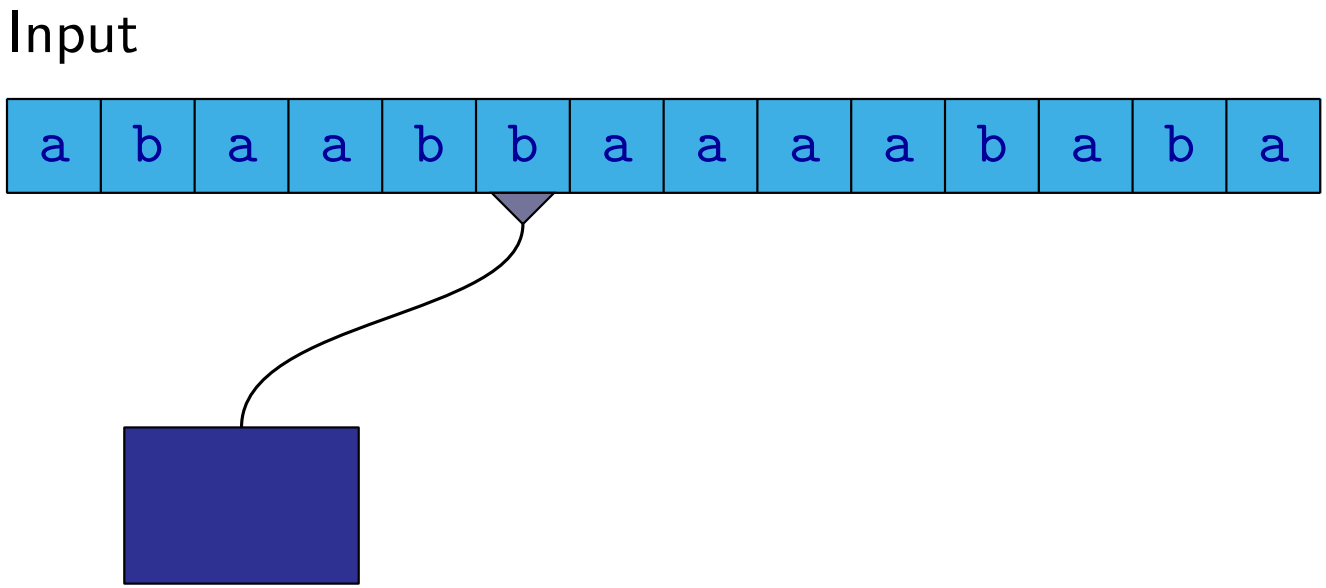
Činnost algoritmu řešícího rozhodovací problém

V případě algoritmu, který řeší nějaký **rozhodovací** problém stačí, když algoritmus vydá jako výstup vždy jen ANO nebo NE.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?



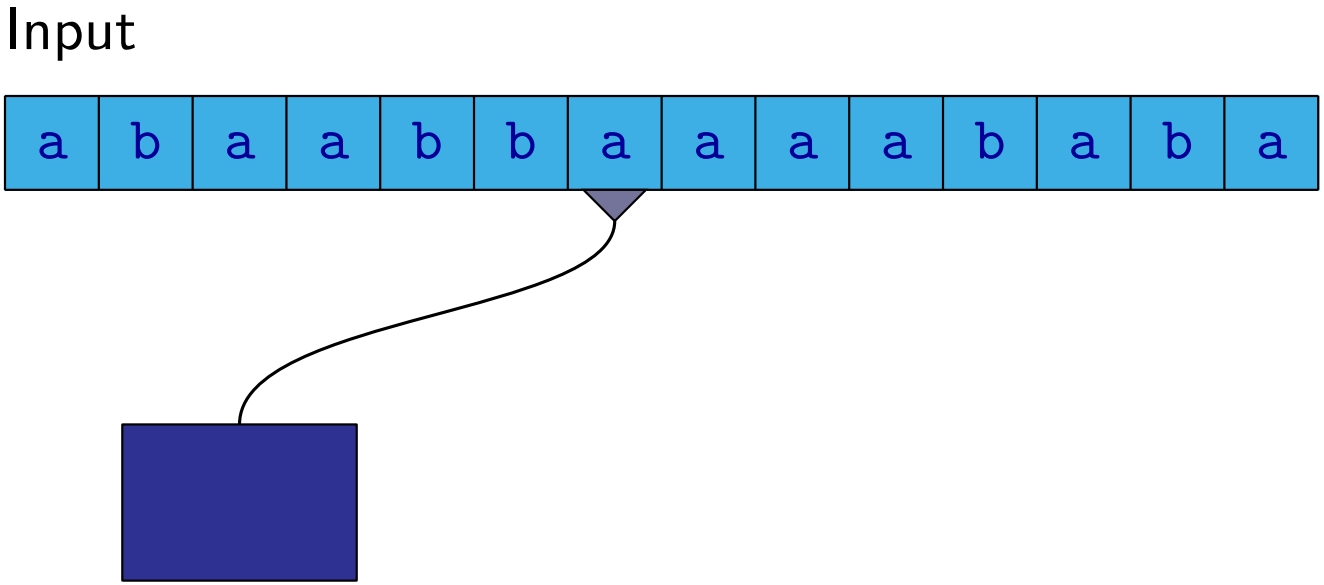
Činnost algoritmu řešícího rozhodovací problém

V případě algoritmu, který řeší nějaký **rozhodovací** problém stačí, když algoritmus vydá jako výstup vždy jen ANO nebo NE.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?



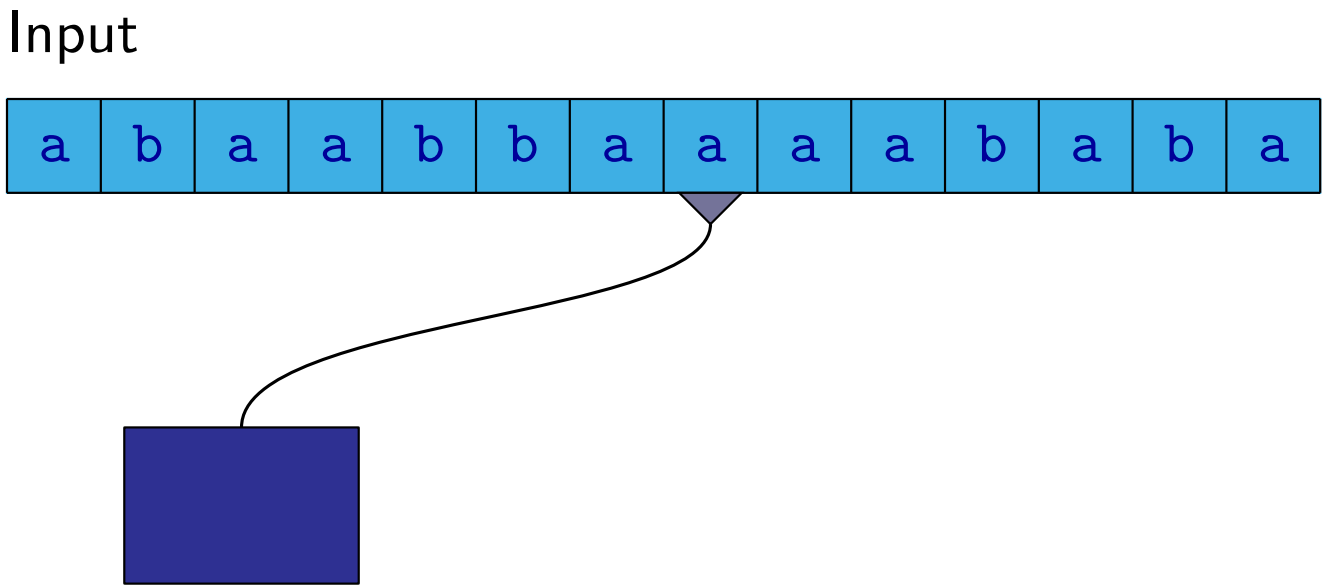
Činnost algoritmu řešícího rozhodovací problém

V případě algoritmu, který řeší nějaký **rozhodovací** problém stačí, když algoritmus vydá jako výstup vždy jen ANO nebo NE.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?



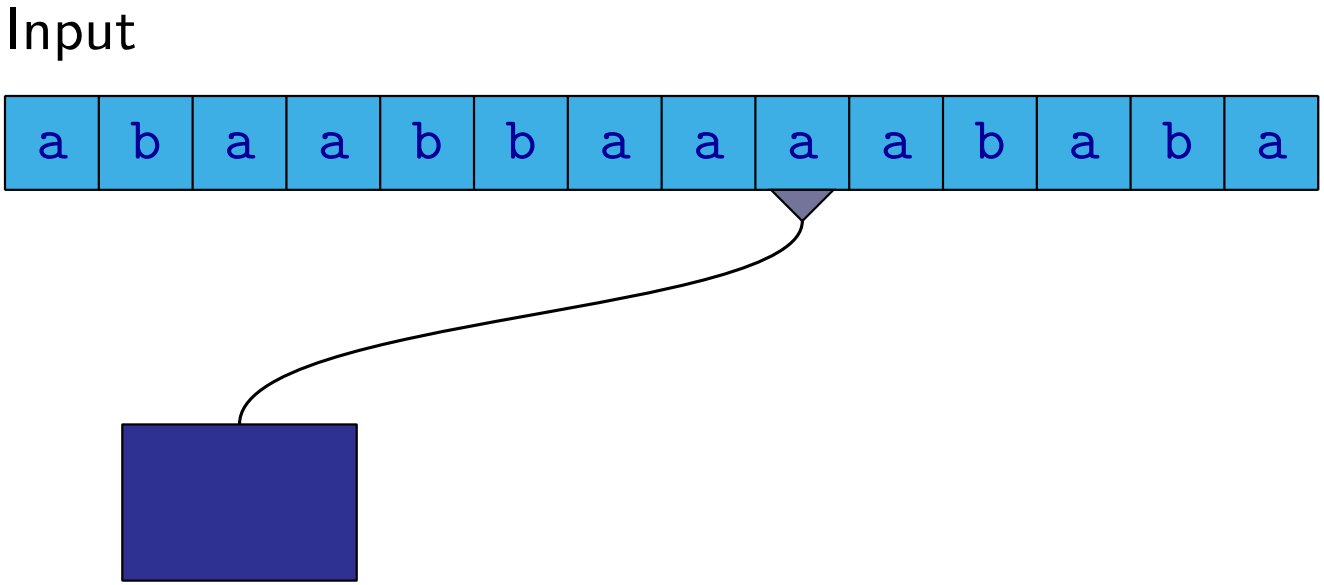
Činnost algoritmu řešícího rozhodovací problém

V případě algoritmu, který řeší nějaký **rozhodovací** problém stačí, když algoritmus vydá jako výstup vždy jen ANO nebo NE.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?



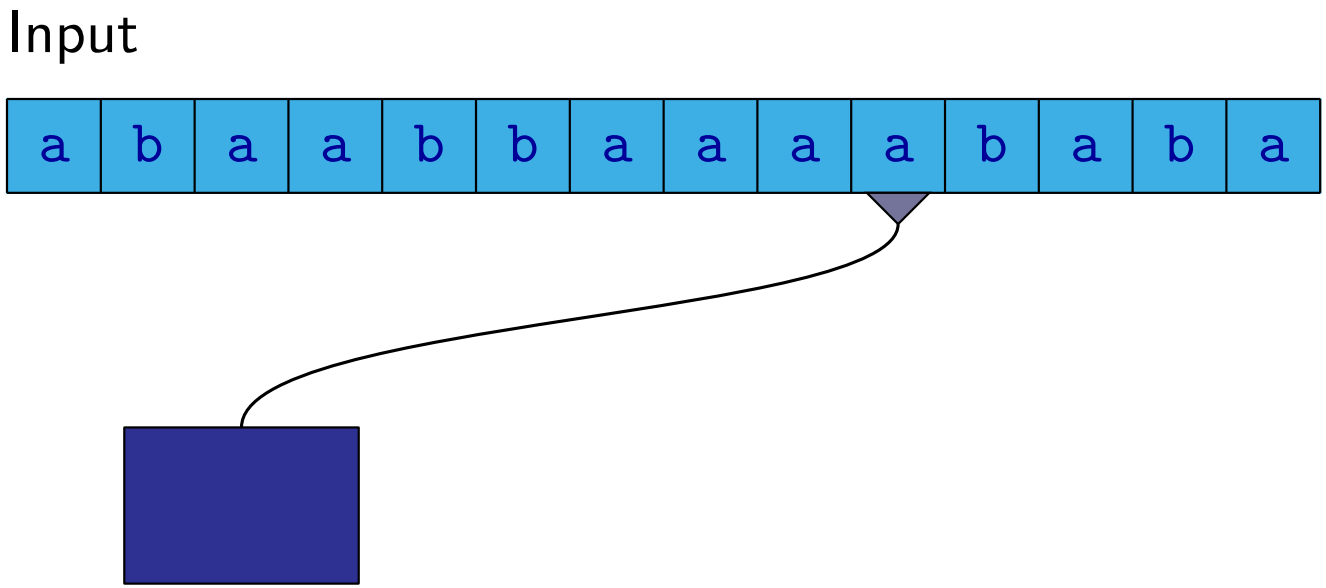
Činnost algoritmu řešícího rozhodovací problém

V případě algoritmu, který řeší nějaký **rozhodovací** problém stačí, když algoritmus vydá jako výstup vždy jen ANO nebo NE.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?



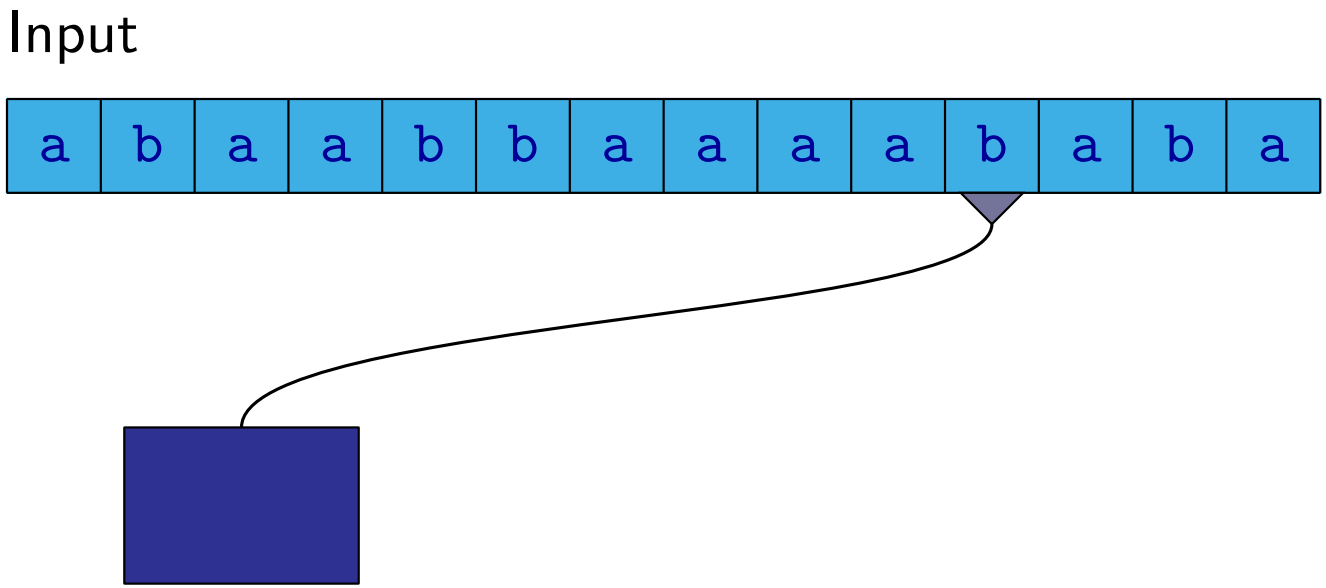
Činnost algoritmu řešícího rozhodovací problém

V případě algoritmu, který řeší nějaký **rozhodovací** problém stačí, když algoritmus vydá jako výstup vždy jen ANO nebo NE.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?



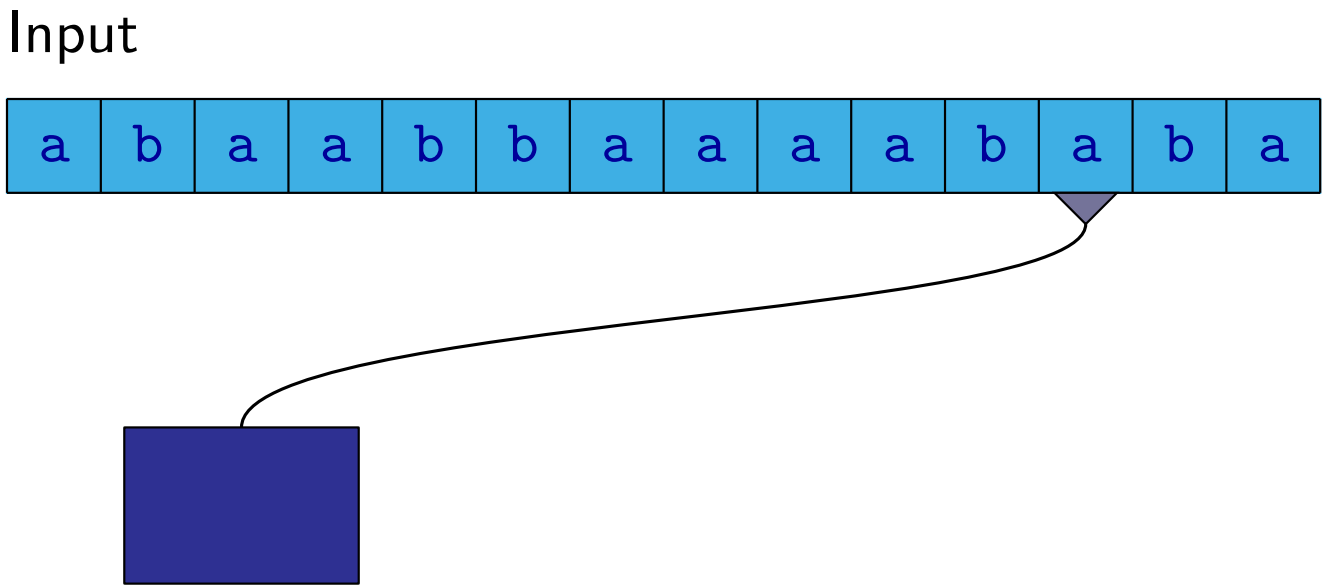
Činnost algoritmu řešícího rozhodovací problém

V případě algoritmu, který řeší nějaký **rozhodovací** problém stačí, když algoritmus vydá jako výstup vždy jen ANO nebo NE.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?



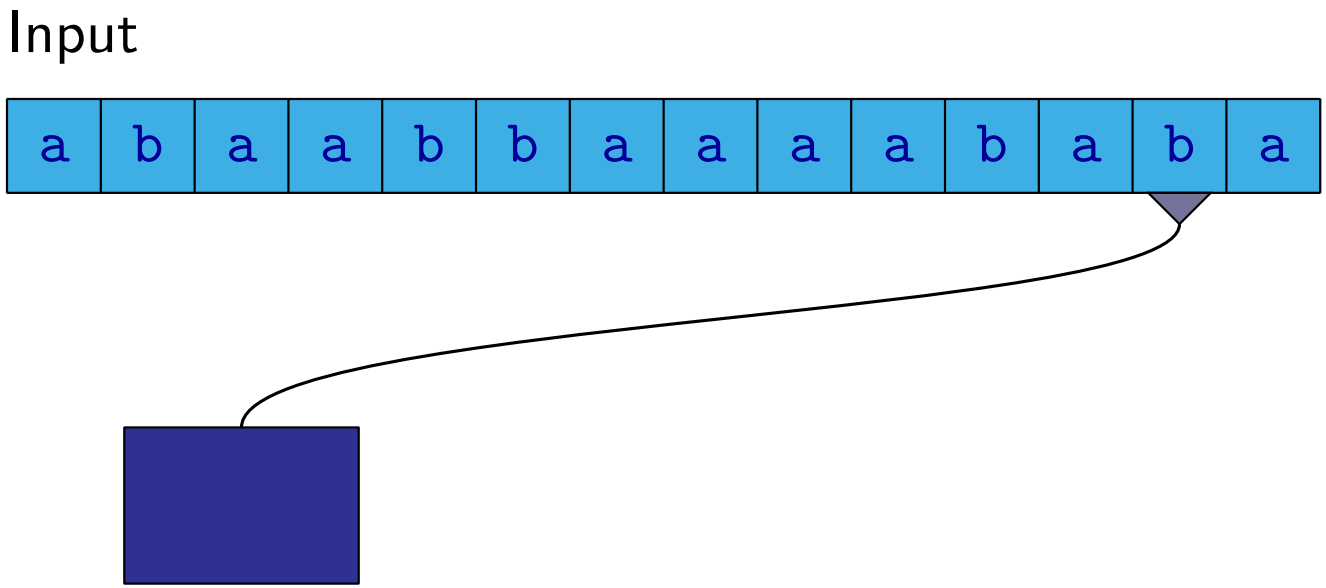
Činnost algoritmu řešícího rozhodovací problém

V případě algoritmu, který řeší nějaký **rozhodovací** problém stačí, když algoritmus vydá jako výstup vždy jen ANO nebo NE.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?



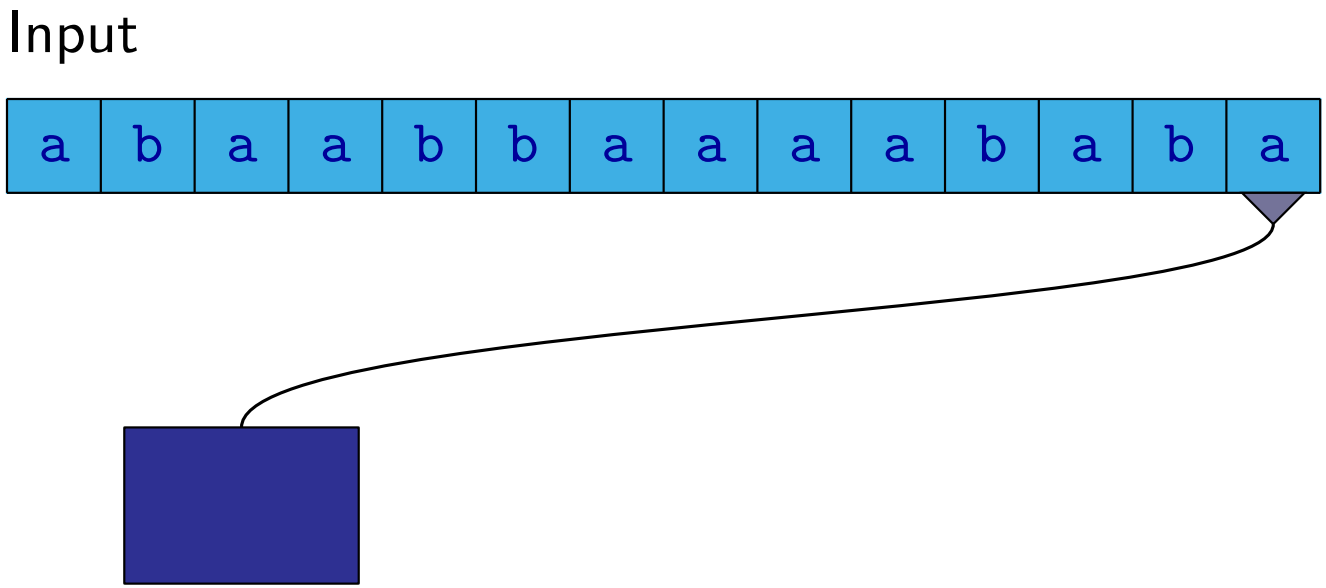
Činnost algoritmu řešícího rozhodovací problém

V případě algoritmu, který řeší nějaký **rozhodovací** problém stačí, když algoritmus vydá jako výstup vždy jen ANO nebo NE.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?



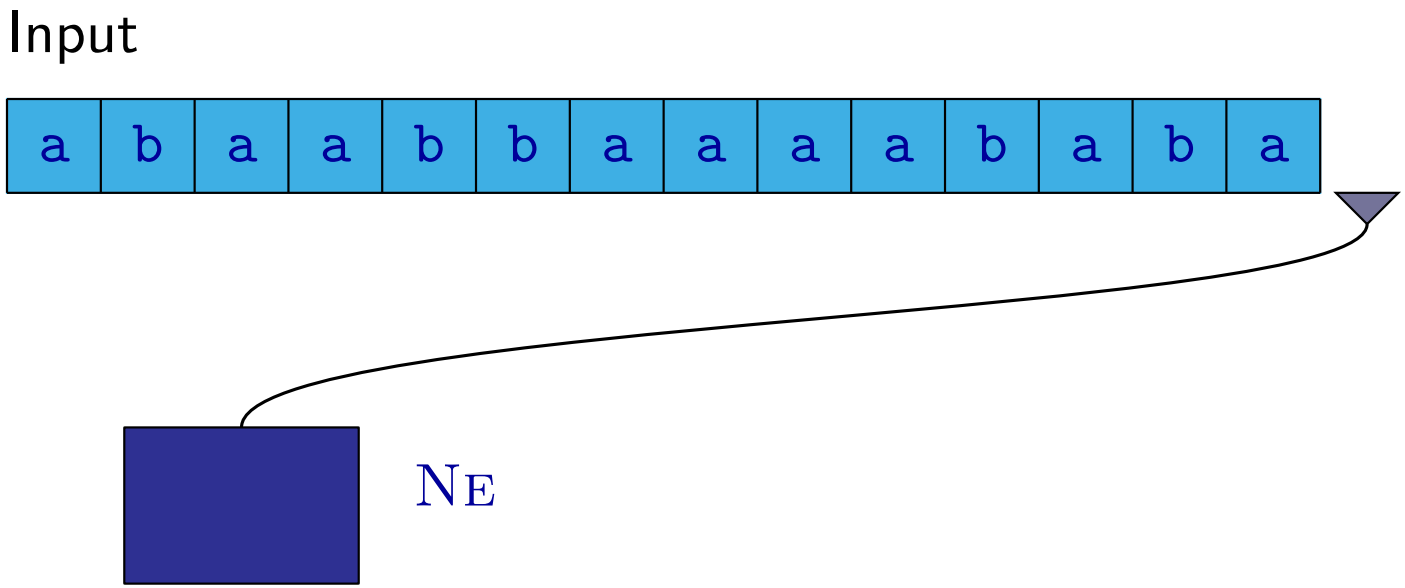
Činnost algoritmu řešícího rozhodovací problém

V případě algoritmu, který řeší nějaký **rozhodovací** problém stačí, když algoritmus vydá jako výstup vždy jen ANO nebo NE.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?



Vztah mezi rozpoznáváním formálních jazyků a rozhodovacími problémy

Mezi rozpoznáváním slov z daného jazyka a rozhodovacími problémy je úzký vztah:

- Každému jazyku L nad nějakou abecedou Σ odpovídá následující rozhodovací problém:

Vstup: Slovo w nad abecedou Σ.

Otázka: Patří slovo w do jazyka L?

- Ke každému rozhodovacímu problému P, jehož vstupy jsou kódovány jako slova nad abecedou Σ, existuje jemu odpovídající jazyk:

Jazyk L obsahující právě ta slova w nad abecedou Σ, pro která je odpověď na příslušnou otázku specifikovanou v zadání problému P "ANO".

Vztah mezi rozpoznáváním formálních jazyků a rozhodovacími problémy

Příklad: Na následující rozhodovací problém se můžeme dívat jako na níže uvedený jazyk L a naopak.

Problém

Vstup: Slovo w nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Otázka: Obsahuje slovo w sudý počet výskytů symbolů $\mathtt{b}$?

Jazyk $L = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid \text{slovo } w \text{ obsahuje sudý počet výskytů symbolů } \mathtt{b} \}$

Výpočetní modely

Můžeme uvažovat různé druhy strojů, které mohou provádět nějaký algoritmus.

Tyto různé druhy strojů se mohou lišit v mnoha ohledech:

- jaké instrukce jsou schopny provádět
- jaký druh dat jsou schopny ukládat do své paměti a jak je tato paměť organizována
- ...

Různé druhy takovýchto strojů se označují jako různé **výpočetní modely**.

V případě velmi jednoduchých druhů strojů se běžně tyto stroje v teorii formálních jazyků označují jako **automaty**.

V tomto předmětu se seznámíme s několika druhy takovýchto automatů.

Výpočetní modely

Pro různé druhy výpočetních modelů můžeme zkoumat například:

- jaké algoritmické problémy jsou schopny řešit či jaké jazyky jsou schopny rozpoznávat.
- jak efektivně jsou schopny realizovat různé algoritmy
- jakým způsobem může určitý druh stroje simulovat činnost jiného druhu stroje
- jak při takové simulaci narůstá počet instrukcí provedených daným strojem
- ...

Formální jazyky

Abeceda a slovo

Definice

Abeceda je libovolná neprázdná konečná množina **symbolů** (**znaků**).

Poznámka: Abeceda se často označuje řeckým písmenem Σ (velké sigma).

Definice

Slovo v dané abecedě je libovolná konečná posloupnost symbolů z této abecedy.

Příklad 1:

$\Sigma = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{G}, \mathtt{H}, \mathtt{I}, \mathtt{J}, \mathtt{K}, \mathtt{L}, \mathtt{M}, \mathtt{N}, \mathtt{O}, \mathtt{P}, \mathtt{Q}, \mathtt{R}, \mathtt{S}, \mathtt{T}, \mathtt{U}, \mathtt{V}, \mathtt{W}, \mathtt{X}, \mathtt{Y}, \mathtt{Z}\}$

Slova v abecedě Σ: `AHOJ` `XYZZY` `COMPUTER`

Abeceda a slovo

Příklad 2:

$\Sigma_2 = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{G}, \mathtt{H}, \mathtt{I}, \mathtt{J}, \mathtt{K}, \mathtt{L}, \mathtt{M}, \mathtt{N}, \mathtt{O}, \mathtt{P}, \mathtt{Q}, \mathtt{R}, \mathtt{S}, \mathtt{T}, \mathtt{U}, \mathtt{V}, \mathtt{W}, \mathtt{X}, \mathtt{Y}, \mathtt{Z}, \sqcup\}$

Slovo v abecedě Σ_2: $\mathtt{HELLO}\sqcup\mathtt{WORLD}$

Příklad 3:

$\Sigma_3 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Slova v abecedě Σ_3: 0, 31415926536, 65536

Příklad 4:

Slova v abecedě $\Sigma_4 = \{0, 1\}$: 011010001, 111, 1010101010101010

Příklad 5:

Slova v abecedě $\Sigma_5 = \{\mathtt{a}, \mathtt{b}\}$: $\mathtt{aababb}$, $\mathtt{abbabbba}$, $\mathtt{aaab}$

Jazyk

Množina všech slov tvořených symboly z abecedy Σ se označuje Σ^*.

Definice

(Formální) jazyk L v abecedě Σ je nějaká libovolná podmnožina množiny Σ^*, tj. $L \subseteq \Sigma^*$.

Příklad 1: Množina $\{00, 01001, 1101\}$ je jazyk v abecedě $\{0, 1\}$.

Příklad 2: Množina všech syntakticky správných programů v jazyce C je jazyk v abecedě tvořené množinou všech ASCII znaků.

Příklad 3: Množina všech textů obsahujících sekvenci znaků `ahoj` je jazyk v abecedě tvořené množinou všech ASCII znaků.

Některé základní pojmy

Délka slova je počet znaků ve slově.

Například délka slova $\texttt{abaab}$ je 5.

Délku slova w označujeme $|w|$.

Pokud tedy např. $w = \texttt{abaab}$, pak $|w| = 5$.

Počet výskytů znaku a ve slově w označujeme $|w|_a$.

Příklad: Pokud $w = \texttt{cabcbba}$, pak $|w| = 7$, $|w|_{\texttt{a}} = 2$, $|w|_{\texttt{b}} = 3$, $|w|_{\texttt{c}} = 2$, $|w|_{\texttt{d}} = 0$.

Prázdné slovo je slovo délky 0, tj. slovo neobsahující žádné znaky.

Prázdné slovo se označuje řeckým písmenem ε (epsilon).

$$|\varepsilon| = 0$$

Zřetězení slov

Se slovy je možné provádět operaci **zřetězení**:

Například zřetězením slov `cabc` a `bba` vznikne slovo `cabcbba`.

Operace zřetězení se označuje symbolem $\cdot$ (podobně jako násobení). Tento symbol je možné vypouštět.

Pokud $u, v \in \Sigma^*$, pak zřetězení slov u a v tedy zapisujeme buď jako $u \cdot v$ nebo jen jako uv.

Příklad: Pokud $u = \texttt{cabc}$ a $v = \texttt{bba}$, pak

$$u \cdot v = \texttt{cabcbba}$$

Poznámka: Z formálního hlediska je zřetězení slov nad abecedou Σ funkcí typu

$$\Sigma^* \times \Sigma^* \to \Sigma^*$$

Zřetězení slov

Zřetězení je **asociativní**, tj. pro libovolná tři slova u, v a w platí

$$(u \cdot v) \cdot w = u \cdot (v \cdot w)$$

Díky tomu můžeme při zápisu více zřetězení vypouštět závorky a psát například $w_1 \cdot w_2 \cdot w_3 \cdot w_4 \cdot w_5$ místo $(w_1 \cdot (w_2 \cdot w_3)) \cdot (w_4 \cdot w_5)$.

Slovo ε je pro operaci zřetězení neutrálním prvkem, pro libovolné slovo w tedy platí:

$$\varepsilon \cdot w = w \cdot \varepsilon = w$$

Poznámka: Je zjevné, že pokud daná abeceda obsahuje alespoň dva různé symboly, tak operace zřetězení není komutativní, např.

$$\mathtt{a} \cdot \mathtt{b} \neq \mathtt{b} \cdot \mathtt{a}$$

Mocnina slova

Pro libovolné slovo $w \in \Sigma^*$ a libovolné $k \in \mathbb{N}$ můžeme definovat slovo w^k jako slovo, které vznikne zřetězením k kopií slova w.

Příklad: Pro $w = \mathtt{abb}$ je $w^4 = \mathtt{abbabbabbabb}$.

Příklad: Zápis $\mathtt{a}^5\mathtt{b}^3\mathtt{a}^4$ označuje slovo $\mathtt{aaaaabbbaaaa}$.

Poněkud formálnější induktivní definice vypadá takto:

$$w^0 = \varepsilon, \qquad w^{k+1} = w^k \cdot w \quad \text{pro } k \in \mathbb{N}$$

To znamená

$$\begin{aligned} w^0 &= \varepsilon \\ w^1 &= w \\ w^2 &= w \cdot w \\ w^3 &= w \cdot w \cdot w \\ w^4 &= w \cdot w \cdot w \cdot w \\ w^5 &= w \cdot w \cdot w \cdot w \cdot w \\ &\dots \end{aligned}$$

Zrcadlový obraz slova

Zrcadlový obraz slova w je slovo w zapsané „pozpátku“.

Zrcadlový obraz slova w značíme w^R.

Příklad: $w = \texttt{abbab}$ $w^R = \texttt{babba}$

Pokud tedy $w = a_1 a_2 \cdots a_n$ (kde $a_i \in \Sigma$), pak $w^R = a_n a_{n-1} \cdots a_1$.

Formálně můžeme definovat w^R pomocí následující induktivně definované funkce $rev : \Sigma^* \to \Sigma^*$ jako hodnotu $rev(w)$.

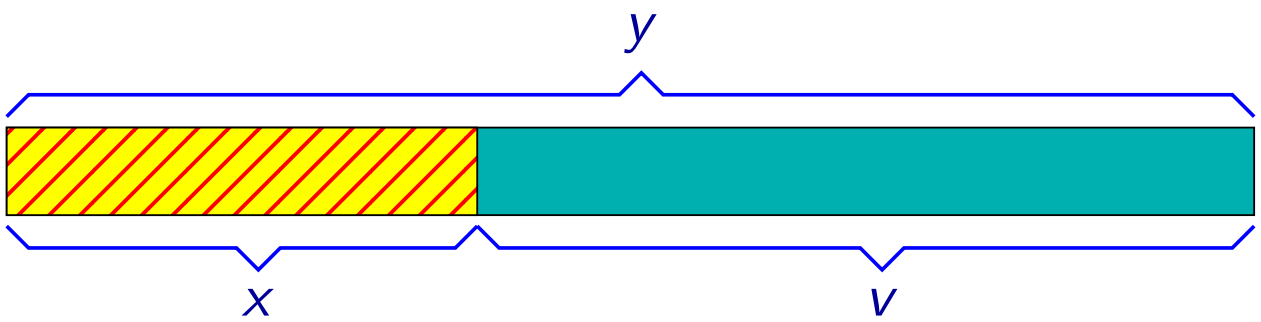
Funkce rev je definována následovně:

- $rev(\varepsilon) = \varepsilon$
- pro $a \in \Sigma$ a $w \in \Sigma^*$ je $rev(a \cdot w) = rev(w) \cdot a$

Prefix slova

Definice

Slovo x je **prefixem** slova y, jestliže existuje slovo v takové, že $y = xv$.

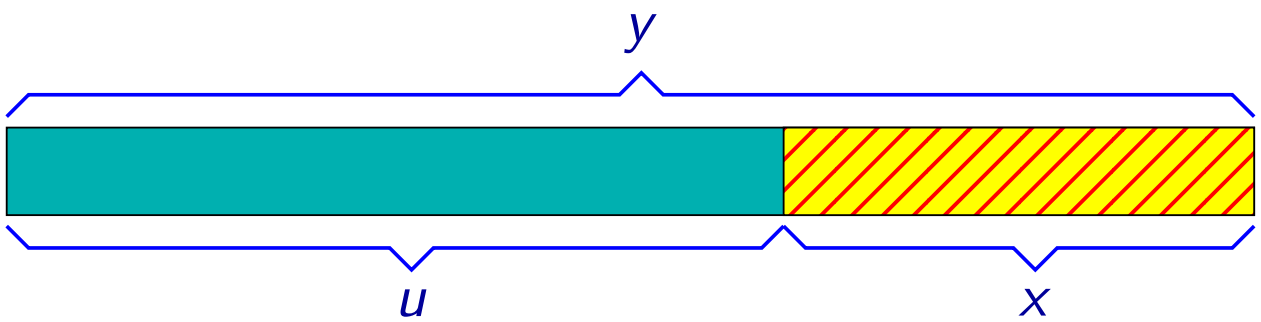


Příklad: Prefixy slova `abaab` jsou ε, `a`, `ab`, `aba`, `abaa`, `abaab`.

Sufix slova

Definice

Slovo x je **sufixem** slova y, jestliže existuje slovo u takové, že $y = ux$.

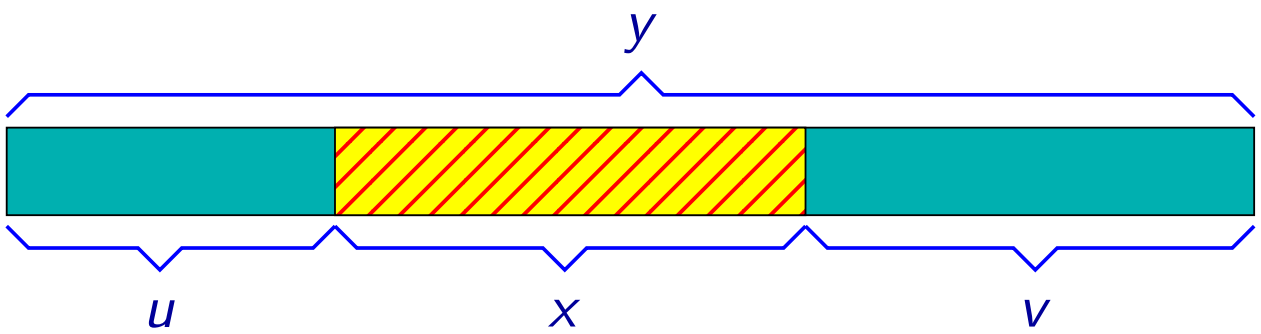


Příklad: Sufixy slova `abaab` jsou ε, `b`, `ab`, `aab`, `baab`, `abaab`.

Podslovo

Definice

Slovo x je **podslovem** slova y, jestliže existují slova u a v taková, že $y = uxv$.

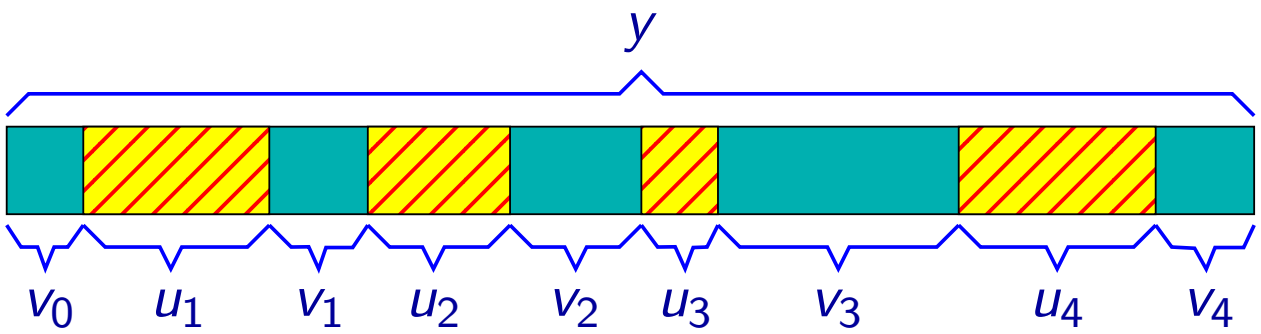


Příklad: Podslova slova abaab jsou ε, a, b, ab, ba, aa, aba, baa, aab, abaa, baab, abaab.

Podsekvence

Definice

Slovo x je **podsekvencí** slova y, jestliže existuje číslo n a slova $u_1, u_2, \ldots, u_n$ a $v_0, v_1, \ldots, v_n$ taková, že $x = u_1 u_2 \cdots u_n$ a $y = v_0 u_1 v_1 u_2 v_2 \cdots u_n v_n$.



Příklad: Slovo `cbab` je podsekvencí slova `acabccabbaa`.

Uspořádání na slovech

Předpokládejme určité (lineární) uspořádání $<$ symbolů abecedy Σ, tj. pokud $\Sigma = \{a_1, a_2, \ldots, a_n\}$, tak platí

$$a_1 < a_2 < \ldots < a_n \,.$$

Příklad: $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, přičemž $\mathtt{a} < \mathtt{b} < \mathtt{c}$.

Na množině Σ^* můžeme definovat následující (lineární) uspořádání $<_L$: $x <_L y$ právě tehdy, když:

- $|x| < |y|$, nebo
- $|x| = |y|$ a existují slova $u, v, w \in \Sigma^*$ a symboly $a, b \in \Sigma$ takové, že platí

$$x = uav \qquad y = ubw \qquad a < b$$

Neformálně můžeme říct v uspořádání $<_L$ řadíme slova podle délky a v rámci stejné délky lexikograficky (podle abecedy).

Uspořádání na slovech

Všechna slova nad abecedou Σ můžeme pomocí uspořádání $<_L$ seřadit do posloupnosti

$$w_0, w_1, w_2, \ldots$$

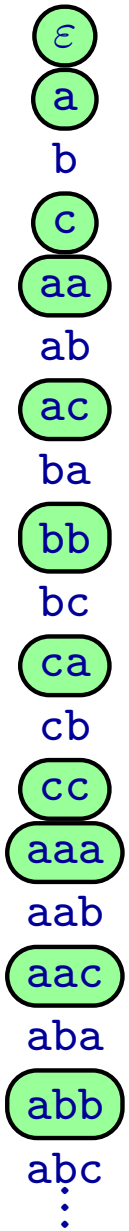
ve které se každé slovo $w \in \Sigma^*$ vyskytuje právě jednou a kde pro libovolná $i, j \in \mathbb{N}$ platí, že $w_i <_L w_j$ právě tehdy, když $i < j$.

Příklad: Pro abecedu $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ (kde $\mathtt{a} < \mathtt{b} < \mathtt{c}$) bude začátek posloupnosti vypadat následovně:

$$\varepsilon, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{aa}, \mathtt{ab}, \mathtt{ac}, \mathtt{ba}, \mathtt{bb}, \mathtt{bc}, \mathtt{ca}, \mathtt{cb}, \mathtt{cc}, \mathtt{aaa}, \mathtt{aab}, \mathtt{aac}, \mathtt{aba}, \mathtt{abb}, \mathtt{abc}, \ldots$$

Pokud budeme mluvit například o prvních deseti slovech jazyka $L \subseteq \Sigma^*$, máme tím na mysli deset slov, která patří do jazyka L a jsou mezi všemi slovy z jazyka L nejmenší vzhledem k uspořádání $<_L$.

Uspořádání na slovech

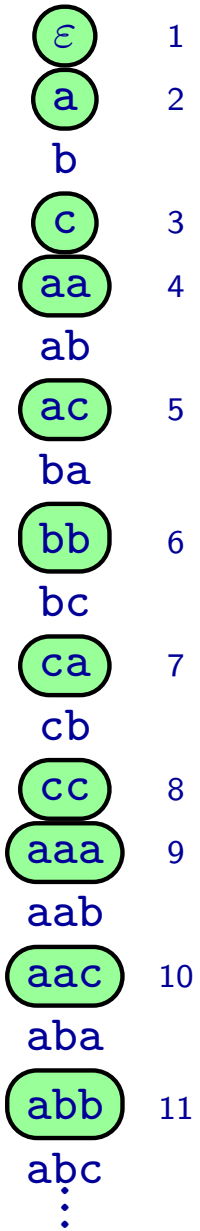


Příklad:

Jazyk

$L = \{ w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid |w|_{\mathtt{b}} \bmod 2 = 0 \}$

Uspořádání na slovech



Příklad:

Jazyk

$L = \{ w \in \{a, b, c\}^* \mid |w|_b \bmod 2 = 0 \}$

Operace na jazycích

Řekněme, že jsme nějaké jazyky již popsali. Z těchto jazyků můžeme vytvářet další nové jazyky pomocí nejrůznějších **operací na jazycích**.

Popis nějakého komplikovaného jazyka můžeme tedy „dekomponovat“ tím způsobem, že tento jazyk vyjádříme jako výsledek aplikování nějakých operací na nějaké jednodušší jazyky.

Příklady důležitých operací na jazycích:

- sjednocení
- průnik
- doplněk
- zřetězení
- iterace
- ...

Poznámka: Při operacích nad jazyky předpokládáme, že jazyky, se kterými operaci provádíme, používají tutéž abecedu Σ.

Množinové operace na jazycích

Vzhledem k tomu, že jazyky jsou množiny, můžeme s nimi provádět množinové operace:

Sjednocení – $L_1 \cup L_2$ je jazyk tvořený slovy, která patří buď do jazyka L_1 nebo do jazyka L_2 (nebo do obou).

Průnik – $L_1 \cap L_2$ je jazyk tvořený slovy, která patří současně do jazyka L_1 i do jazyka L_2.

Doplněk – $\overline{L_1}$ je jazyk tvořený těmi slovy ze Σ^*, která nepatří do L_1.

Rozdíl – $L_1 - L_2$ je jazyk tvořený slovy, která patří do L_1, ale nepatří do L_2.

Poznámka: Předpokládáme, že $L_1, L_2 \subseteq \Sigma^*$ pro nějakou danou abecedu Σ.

Množinové operace na jazycích

Formálně:

Sjednocení: $L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \vee w \in L_2\}$

Průnik: $L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \wedge w \in L_2\}$

Doplněk: $\overline{L_1} = \{w \in \Sigma^* \mid w \notin L_1\}$

Rozdíl: $L_1 - L_2 = \{w \in \Sigma^* \mid w \in L_1 \wedge w \notin L_2\}$

Množinové operace na jazycích

Příklad:

Uvažujme jazyky nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

- L_1 — množina všech slov obsahujících podslovo $\mathtt{baa}$
- L_2 — množina všech slov se sudým počtem výskytů symbolu $\mathtt{b}$

Pak

- $L_1 \cup L_2$ — množina všech slov obsahujících podslovo $\mathtt{baa}$ nebo sudý počet symbolů $\mathtt{b}$
- $L_1 \cap L_2$ — množina všech slov obsahujících podslovo $\mathtt{baa}$ a sudý počet symbolů $\mathtt{b}$
- $\overline{L_1}$ — množina všech slov, která neobsahují podslovo $\mathtt{baa}$
- $L_1 - L_2$ — množina všech slov, ve kterých se vyskytuje podslovo $\mathtt{baa}$, ale kde počet symbolů $\mathtt{b}$ není sudý

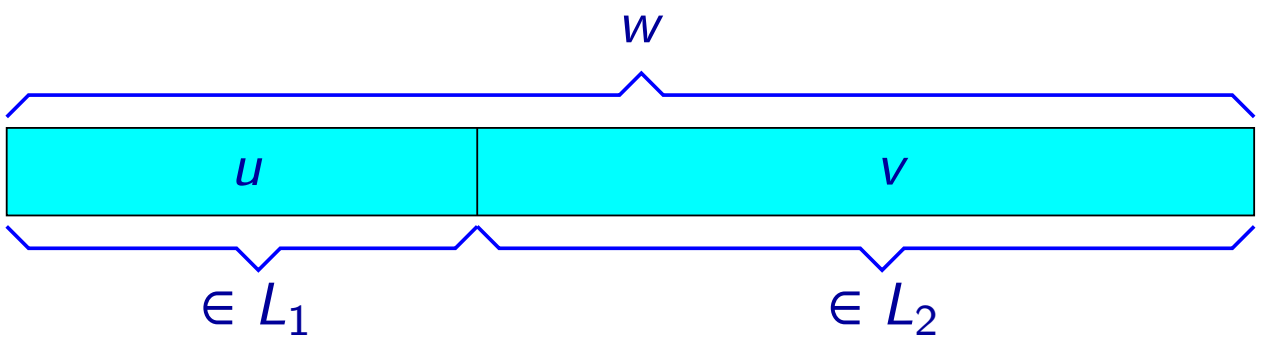
Zřetězení jazyků

Definice

Zřetězení jazyků L_1 a L_2, kde $L_1, L_2 \subseteq \Sigma^*$, je jazyk $L \subseteq \Sigma^*$ takový, že pro každé $w \in \Sigma^*$ platí

$$w \in L \iff (\exists u \in L_1)(\exists v \in L_2)(w = u \cdot v)$$

Zřetězení jazyků L_1 a L_2 označujeme zápisem $L_1 \cdot L_2$.



Zřetězení jazyků

Příklad:

$$\begin{aligned} L_1 &= \{\mathtt{abb}, \mathtt{ba}\} \\ L_2 &= \{\mathtt{a}, \mathtt{ab}, \mathtt{bbb}\} \end{aligned}$$

Jazyk $L_1 \cdot L_2$ obsahuje slova:

$$\mathtt{abba} \qquad \mathtt{abbab} \qquad \mathtt{abbbbb} \qquad \mathtt{baa} \qquad \mathtt{baab} \qquad \mathtt{babbb}$$

Poznámka: Všimněte si, že zřetězení jazyků je asociativní, tj. pro libovolné jazyky L_1, L_2, L_3 platí:

$$L_1 \cdot (L_2 \cdot L_3) = (L_1 \cdot L_2) \cdot L_3$$

Mocnina jazyka

Zápis L^k, kde $L \subseteq \Sigma^*$ a $k \in \mathbb{N}$, označuje zřetězení tvaru

$$L \cdot L \cdot \quad \cdots \quad \cdot L$$

kde se jazyk L vyskytuje k-krát, tj.

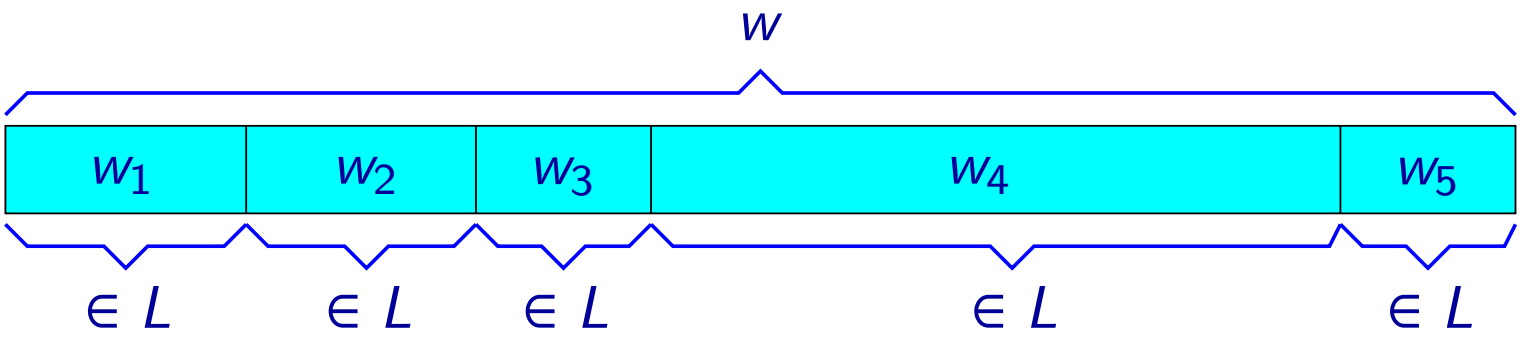
$$\begin{aligned} L^0 &= \{\varepsilon\} \\ L^1 &= L \\ L^2 &= L \cdot L \\ L^3 &= L \cdot L \cdot L \\ L^4 &= L \cdot L \cdot L \cdot L \\ L^5 &= L \cdot L \cdot L \cdot L \cdot L \\ &\ldots \end{aligned}$$

Příklad: Pokud $L = \{\mathtt{aa}, \mathtt{b}\}$, pak jazyk L^3 obsahuje následující slova:

$$\mathtt{aaaaaa} \quad \mathtt{aaaab} \quad \mathtt{aabaa} \quad \mathtt{aabb} \quad \mathtt{baaaa} \quad \mathtt{baab} \quad \mathtt{bbaa} \quad \mathtt{bbb}$$

Mocnina jazyka

Příklad: Slovo w patřící do jazyka L^5 vznikne zřetězením pěti slov z jazyka L:



Formálně můžeme **mocninu jazyka** L^k definovat pomocí následující induktivní definice:

$$L^0 = \{\varepsilon\}, \qquad L^{k+1} = L^k \cdot L \quad \text{pro } k \in \mathbb{N}$$

Iterace jazyka

Iterace jazyka L, označovaná zápisem L^*, je jazyk tvořený slovy vzniklými zřetězením libovolného počtu slov z jazyka L.

Tj., slovo w patří do L^* právě tehdy, když existuje posloupnost $w_1, w_2, \ldots, w_n$ slov z jazyka L taková, že

$$w = w_1 w_2 \cdots w_n .$$

Příklad: $L = \{\mathtt{aa}, \mathtt{b}\}$

$$L^* = \{\varepsilon, \mathtt{aa}, \mathtt{b}, \mathtt{aaaa}, \mathtt{aab}, \mathtt{baa}, \mathtt{bb}, \mathtt{aaaaaa}, \mathtt{aaaab}, \mathtt{aabaa}, \mathtt{aabb}, \ldots\}$$

Poznámka: Počet slov, která zřetězujeme, může být i 0, což znamená, že vždy platí $\varepsilon \in L^*$ (bez ohledu na to, zda $\varepsilon \in L$ nebo ne).

Iterace jazyka

Formálně můžeme definovat jazyk L^* jako sjednocení všech mocnin jazyka L. Tj. slovo w patří do jazyka L^* právě tehdy, když existuje $k \in \mathbb{N}$ takové, že $w \in L^k$:

Definice

Iterace jazyka L je jazyk

$$L^* = \bigcup_{k \geq 0} L^k$$

Poznámka:

$$\bigcup_{k \geq 0} L^k = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \cdots$$

Iterace jazyka

Zápis L^+ označuje jazyk tvořený právě těmi slovy, která vzniknou zřetězením nějakého nenulového počtu slov z jazyka L.

Platí tedy

$$L^+ = \bigcup_{k \geq 1} L^k$$

tj.

$$L^+ = L^1 \cup L^2 \cup L^3 \cup \cdots$$

Formálně můžeme jazyk L^+ definovat též následujícím způsobem:

$$L^+ = L \cdot L^*$$

Zrcadlový obraz jazyka

Zrcadlový obraz jazyka L je jazyk tvořený zrcadlovými obrazy všech slov z jazyka L.

Zrcadlový obraz jazyka L značíme L^R.

$$L^R = \{w^R \mid w \in L\}$$

Příklad: $L = \{\mathtt{ab}, \mathtt{baaba}, \mathtt{aaab}\}$

$L^R = \{\mathtt{ba}, \mathtt{abaab}, \mathtt{baaa}\}$

Některé vlastnosti operací na jazycích

$$
\begin{aligned}
L_1 \cup (L_2 \cup L_3) &= (L_1 \cup L_2) \cup L_3 \\
L_1 \cup L_2 &= L_2 \cup L_1 \\
L_1 \cup L_1 &= L_1 \\
L_1 \cup \varnothing &= L_1 \\
L_1 \cap (L_2 \cap L_3) &= (L_1 \cap L_2) \cap L_3 \\
L_1 \cap L_2 &= L_2 \cap L_1 \\
L_1 \cap L_1 &= L_1 \\
L_1 \cap \varnothing &= \varnothing \\
L_1 \cdot (L_2 \cdot L_3) &= (L_1 \cdot L_2) \cdot L_3 \\
L_1 \cdot \{\varepsilon\} &= L_1 \\
\{\varepsilon\} \cdot L_1 &= L_1 \\
L_1 \cdot \varnothing &= \varnothing \\
\varnothing \cdot L_1 &= \varnothing
\end{aligned}
$$

Některé vlastnosti operací na jazycích

$$
\begin{aligned}
L_1 \cdot (L_2 \cup L_3) &= (L_1 \cdot L_2) \cup (L_1 \cdot L_3) \\
(L_1 \cup L_2) \cdot L_3 &= (L_1 \cdot L_3) \cup (L_2 \cdot L_3) \\
(L_1^*)^* &= L_1^* \\
\varnothing^* &= \{\varepsilon\} \\
L_1^* &= \{\varepsilon\} \cup (L_1 \cdot L_1^*) \\
L_1^* &= \{\varepsilon\} \cup (L_1^* \cdot L_1) \\
(L_1 \cup L_2)^* &= L_1^* \cdot (L_2 \cdot L_1^*)^* \\
(L_1 \cdot L_2)^R &= L_2^R \cdot L_1^R
\end{aligned}
$$

Regulární výrazy

Regulární výrazy

Regulární výrazy popisující jazyky nad abecedou Σ:

- $\varnothing$, ε, a (kde $a \in \Sigma$) jsou regulární výrazy:

  $\varnothing$ ... označuje prázdný jazyk
  ε ... označuje jazyk $\{\varepsilon\}$
  a ... označuje jazyk $\{a\}$

- Jestliže α, β jsou regulární výrazy, pak i $(\alpha + \beta)$, $(\alpha \cdot \beta)$, (α^*) jsou regulární výrazy:

  $(\alpha + \beta)$ ... označuje sjednocení jazyků označených α a β
  $(\alpha \cdot \beta)$ ... označuje zřetězení jazyků označených α a β
  (α^*) ... označuje iteraci jazyka označeného α

- Neexistují žádné další regulární výrazy než ty definované podle předchozích dvou bodů.

Regulární výrazy

Příklad: abeceda $\Sigma = \{0, 1\}$

- Podle definice jsou 0 i 1 regulární výrazy.

Regulární výrazy

Příklad: abeceda $\Sigma = \{0, 1\}$

- Podle definice jsou 0 i 1 regulární výrazy.
- Protože 0 i 1 jsou regulární výrazy, je i $(0 + 1)$ regulární výraz.

Regulární výrazy

Příklad: abeceda $\Sigma = \{0, 1\}$

- Podle definice jsou 0 i 1 regulární výrazy.
- Protože 0 i 1 jsou regulární výrazy, je i $(0 + 1)$ regulární výraz.
- Protože 0 je regulární výraz, je i (0^*) regulární výraz.

Regulární výrazy

Příklad: abeceda $\Sigma = \{0, 1\}$

- Podle definice jsou 0 i 1 regulární výrazy.
- Protože 0 i 1 jsou regulární výrazy, je i $(0 + 1)$ regulární výraz.
- Protože 0 je regulární výraz, je i (0^*) regulární výraz.
- Protože $(0 + 1)$ i (0^*) jsou regulární výrazy, je i $((0 + 1) \cdot (0^*))$ regulární výraz.

Regulární výrazy

Příklad: abeceda $\Sigma = \{0, 1\}$

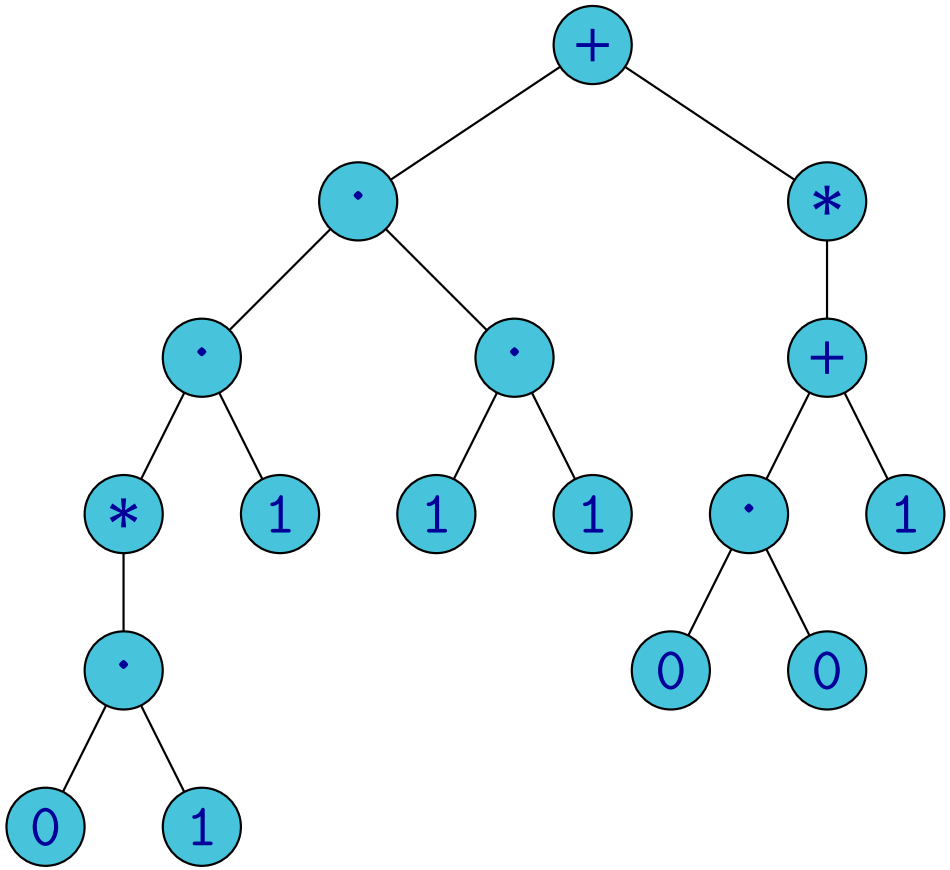
- Podle definice jsou 0 i 1 regulární výrazy.
- Protože 0 i 1 jsou regulární výrazy, je i $(0 + 1)$ regulární výraz.
- Protože 0 je regulární výraz, je i (0^*) regulární výraz.
- Protože $(0 + 1)$ i (0^*) jsou regulární výrazy, je i $((0 + 1) \cdot (0^*))$ regulární výraz.

Poznámka: Jestliže α je regulární výraz, zápisem $\mathcal{L}(\alpha)$ označujeme jazyk definovaný regulárním výrazem α.

$$\mathcal{L}(((0 + 1) \cdot (0^*))) = \{0,\ 1,\ 00,\ 10,\ 000,\ 100,\ 0000,\ 1000,\ 00000,\ \dots\}$$

Regulární výrazy

Strukturu regulárního výrazu si můžeme znázornit abstraktním syntaktickým stromem:



$$((((((0 \cdot 1)^*) \cdot 1) \cdot (1 \cdot 1)) + (((0 \cdot 0) + 1)^*))$$

Regulární výrazy

Formální definice sémantiky regulárních výrazů:

- $\mathcal{L}(\varnothing) = \varnothing$
- $\mathcal{L}(\varepsilon) = \{\varepsilon\}$
- $\mathcal{L}(a) = \{a\}$
- $\mathcal{L}(\alpha^*) = \mathcal{L}(\alpha)^*$
- $\mathcal{L}(\alpha \cdot \beta) = \mathcal{L}(\alpha) \cdot \mathcal{L}(\beta)$
- $\mathcal{L}(\alpha + \beta) = \mathcal{L}(\alpha) \cup \mathcal{L}(\beta)$

Regulární výrazy

Aby byl zápis regulárních výrazů přehlednější a stručnější, používáme následují pravidla:

- Vynecháváme vnější pár závorek.
- Vynecháváme závorky, které jsou zbytečné vzhledem k asociativitě operací sjednocení ($+$) a zřetězení ($\cdot$).
- Vynecháváme závorky, které jsou zbytečné vzhledem k prioritě operací (nejvyšší prioritu má iterace (*), menší zřetězení ($\cdot$) a nejmenší sjednocení ($+$)).
- Nepíšeme tečku pro zřetězení.

Příklad: Místo

$$(((((0 \cdot 1)^*) \cdot 1) \cdot (1 \cdot 1)) + (((0 \cdot 0) + 1)^*))$$

obvykle píšeme

$$(01)^*111 + (00 + 1)^*$$

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

$\mathtt{a}$ ... jazyk tvořený jediným slovem $\mathtt{a}$

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

$\mathtt{a}$ ... jazyk tvořený jediným slovem $\mathtt{a}$

$\mathtt{ab}$ ... jazyk tvořený jediným slovem $\mathtt{ab}$

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

$\mathtt{a}$ . . . jazyk tvořený jediným slovem $\mathtt{a}$

$\mathtt{ab}$ . . . jazyk tvořený jediným slovem $\mathtt{ab}$

$\mathtt{a} + \mathtt{b}$ . . . jazyk tvořený dvěma slovy $\mathtt{a}$ a $\mathtt{b}$

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

$\mathtt{a}$ . . . jazyk tvořený jediným slovem $\mathtt{a}$

$\mathtt{ab}$ . . . jazyk tvořený jediným slovem $\mathtt{ab}$

$\mathtt{a} + \mathtt{b}$ . . . jazyk tvořený dvěma slovy $\mathtt{a}$ a $\mathtt{b}$

$\mathtt{a}^*$ . . . jazyk tvořený slovy ε, $\mathtt{a}$, $\mathtt{aa}$, $\mathtt{aaa}$, . . .

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

$\mathtt{a}$ ... jazyk tvořený jediným slovem $\mathtt{a}$

$\mathtt{ab}$ ... jazyk tvořený jediným slovem $\mathtt{ab}$

$\mathtt{a} + \mathtt{b}$ ... jazyk tvořený dvěma slovy $\mathtt{a}$ a $\mathtt{b}$

$\mathtt{a}^*$ ... jazyk tvořený slovy ε, $\mathtt{a}$, $\mathtt{aa}$, $\mathtt{aaa}$, ...

$(\mathtt{ab})^*$ ... jazyk tvořený slovy ε, $\mathtt{ab}$, $\mathtt{abab}$, $\mathtt{ababab}$, ...

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

$\mathtt{a}$ ... jazyk tvořený jediným slovem $\mathtt{a}$

$\mathtt{ab}$ ... jazyk tvořený jediným slovem $\mathtt{ab}$

$\mathtt{a} + \mathtt{b}$ ... jazyk tvořený dvěma slovy $\mathtt{a}$ a $\mathtt{b}$

$\mathtt{a}^*$ ... jazyk tvořený slovy ε, $\mathtt{a}$, $\mathtt{aa}$, $\mathtt{aaa}$, ...

$(\mathtt{ab})^*$ ... jazyk tvořený slovy ε, $\mathtt{ab}$, $\mathtt{abab}$, $\mathtt{ababab}$, ...

$(\mathtt{a} + \mathtt{b})^*$ ... jazyk tvořený všemi slovy nad abecedou $\{\mathtt{a}, \mathtt{b}\}$

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

$\mathtt{a}$	... jazyk tvořený jediným slovem $\mathtt{a}$
$\mathtt{ab}$	... jazyk tvořený jediným slovem $\mathtt{ab}$
$\mathtt{a} + \mathtt{b}$	... jazyk tvořený dvěma slovy $\mathtt{a}$ a $\mathtt{b}$
$\mathtt{a}^*$	... jazyk tvořený slovy ε, $\mathtt{a}$, $\mathtt{aa}$, $\mathtt{aaa}$, ...
$(\mathtt{ab})^*$	... jazyk tvořený slovy ε, $\mathtt{ab}$, $\mathtt{abab}$, $\mathtt{ababab}$, ...
$(\mathtt{a} + \mathtt{b})^*$	... jazyk tvořený všemi slovy nad abecedou $\{\mathtt{a}, \mathtt{b}\}$
$(\mathtt{a} + \mathtt{b})^*\mathtt{aa}$	... jazyk tvořený všemi slovy končícími $\mathtt{aa}$

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

$\mathtt{a}$ ... jazyk tvořený jediným slovem $\mathtt{a}$

$\mathtt{ab}$ ... jazyk tvořený jediným slovem $\mathtt{ab}$

$\mathtt{a} + \mathtt{b}$ ... jazyk tvořený dvěma slovy $\mathtt{a}$ a $\mathtt{b}$

$\mathtt{a}^*$ ... jazyk tvořený slovy ε, $\mathtt{a}$, $\mathtt{aa}$, $\mathtt{aaa}$, ...

$(\mathtt{ab})^*$ ... jazyk tvořený slovy ε, $\mathtt{ab}$, $\mathtt{abab}$, $\mathtt{ababab}$, ...

$(\mathtt{a} + \mathtt{b})^*$ ... jazyk tvořený všemi slovy nad abecedou $\{\mathtt{a}, \mathtt{b}\}$

$(\mathtt{a} + \mathtt{b})^*\mathtt{aa}$ ... jazyk tvořený všemi slovy končícími $\mathtt{aa}$

$(\mathtt{ab})^*\mathtt{bbb}(\mathtt{ab})^*$ ... jazyk tvořený všemi slovy obsahujícími podslovo $\mathtt{bbb}$ předcházené i následované libovolným počtem slov $\mathtt{ab}$

Regulární výrazy

$(a + b)^*aa + (ab)^*bbb(ab)^*$ ... jazyk tvořený všemi slovy, která buď končí aa nebo obsahují podslovo bbb předcházené i následované libovolným počtem slov ab

Regulární výrazy

$(a + b)^*aa + (ab)^*bbb(ab)^*$ ... jazyk tvořený všemi slovy, která buď končí aa nebo obsahují podslovo bbb předcházené i následované libovolným počtem slov ab

$(a + b)^*b(a + b)^*$ ... jazyk tvořený všemi slovy obsahujícími alespoň jeden symbol b

Regulární výrazy

$(a + b)^*aa + (ab)^*bbb(ab)^*$ ... jazyk tvořený všemi slovy, která buď končí aa nebo obsahují podslovo bbb předcházené i následované libovolným počtem slov ab

$(a + b)^*b(a + b)^*$ ... jazyk tvořený všemi slovy obsahujícími alespoň jeden symbol b

$a^*(ba^*ba^*)^*$ ... jazyk tvořený všemi slovy obsahujícími sudý počet symbolů b

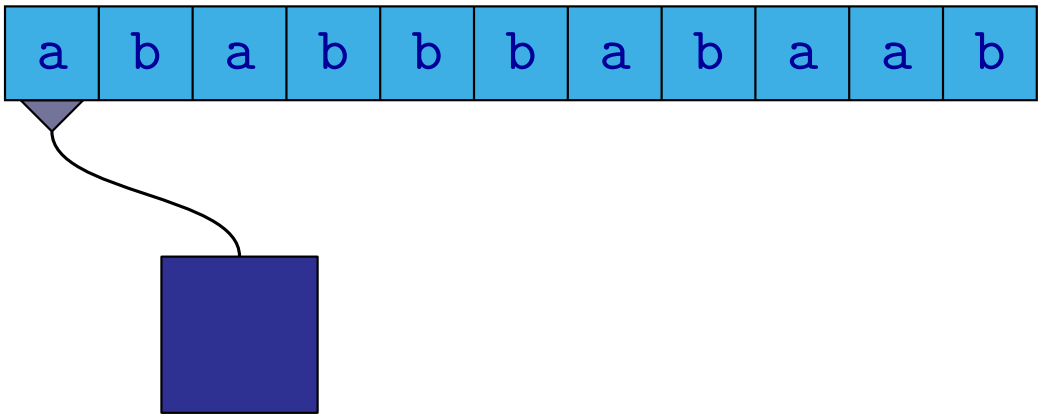
Konečné automaty

Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů $\mathtt{b}$.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

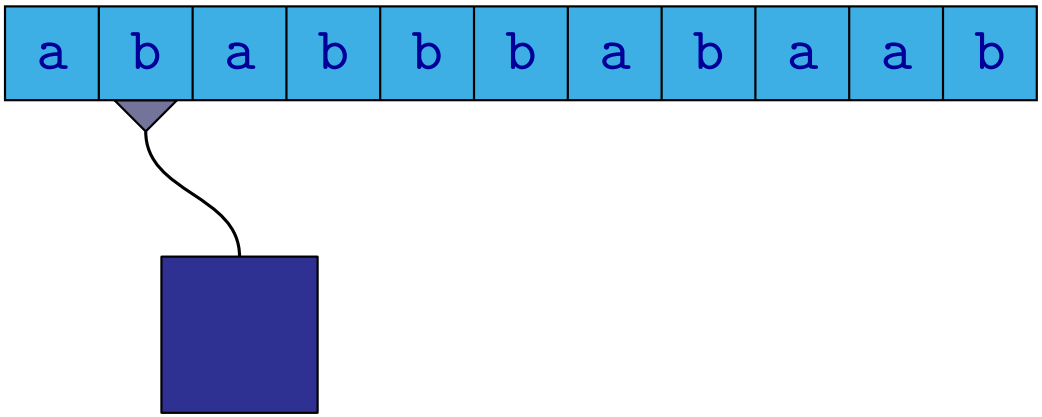


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů $\mathtt{b}$.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

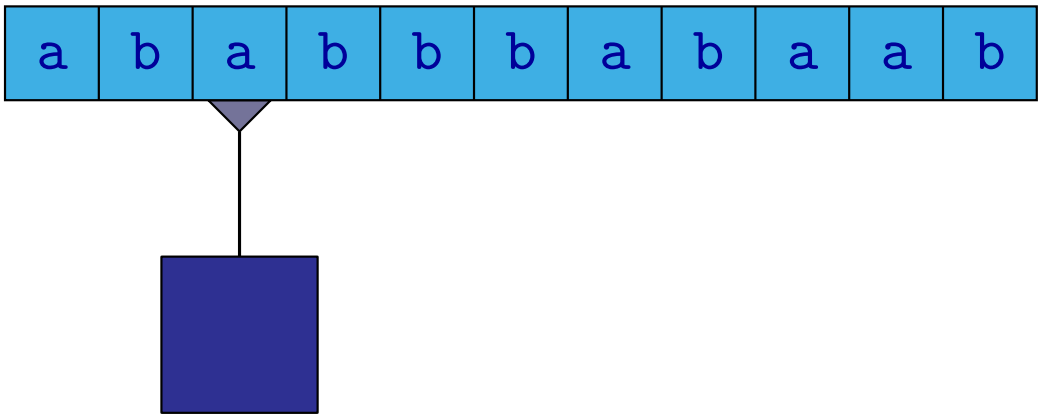


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů $\mathtt{b}$.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

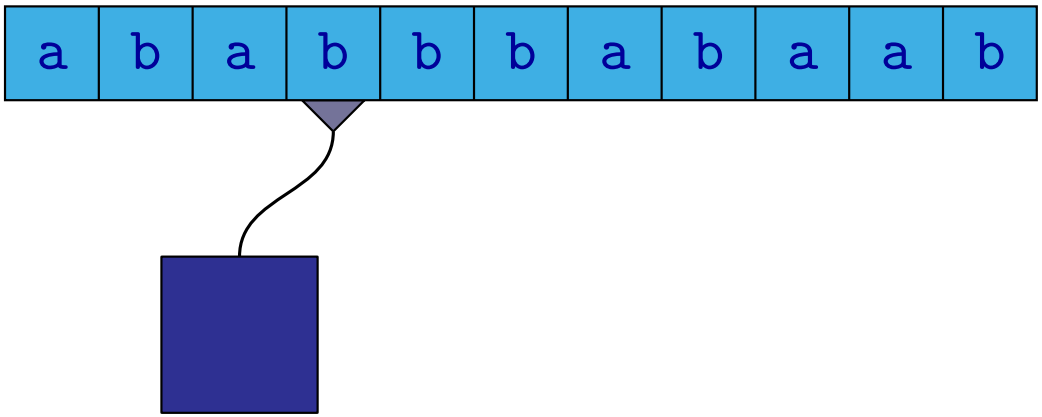


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů $\mathtt{b}$.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

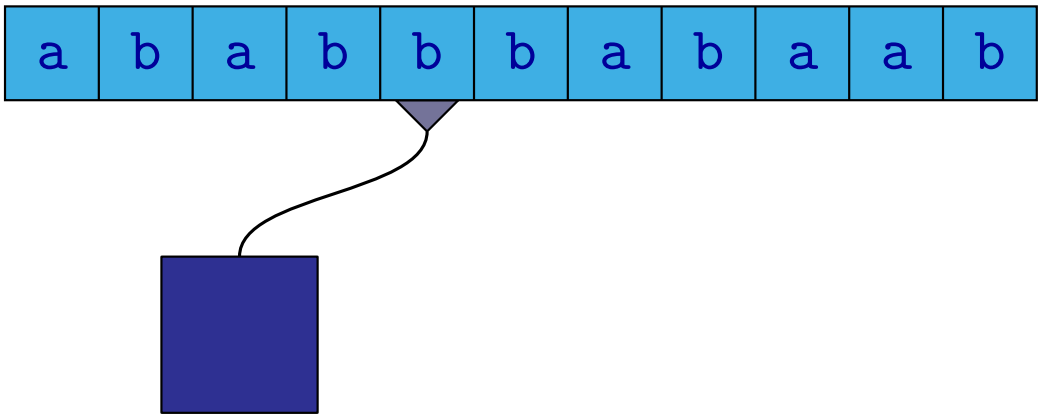


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů $\mathtt{b}$.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

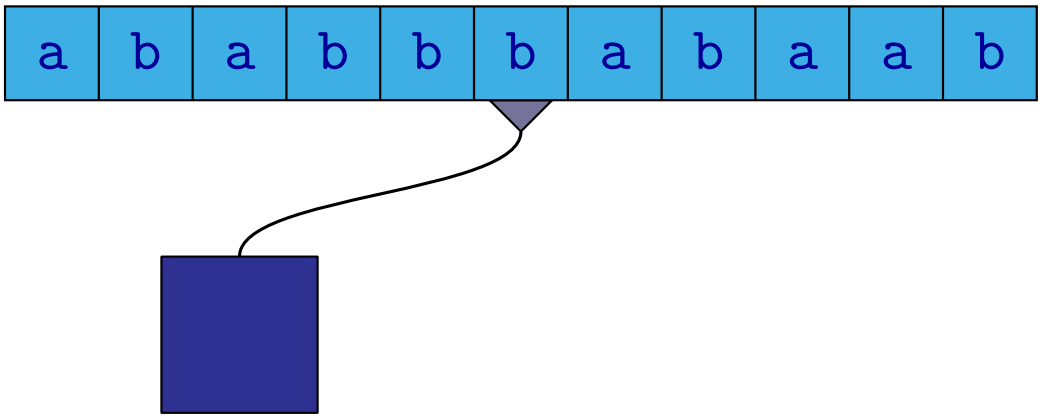


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů $\mathtt{b}$.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

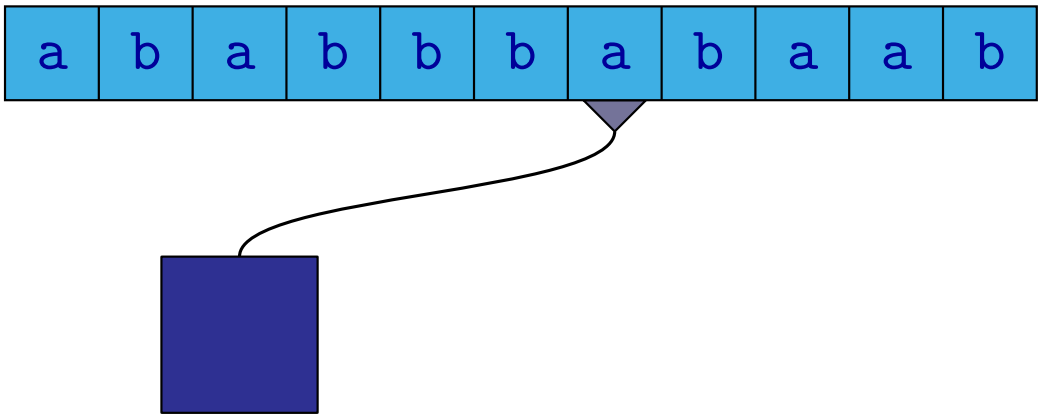


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů $\mathtt{b}$.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

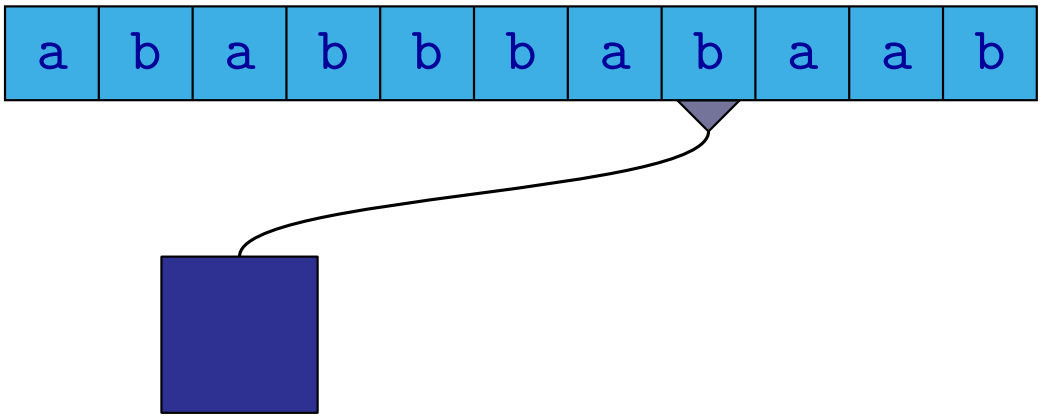


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů $\mathtt{b}$.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

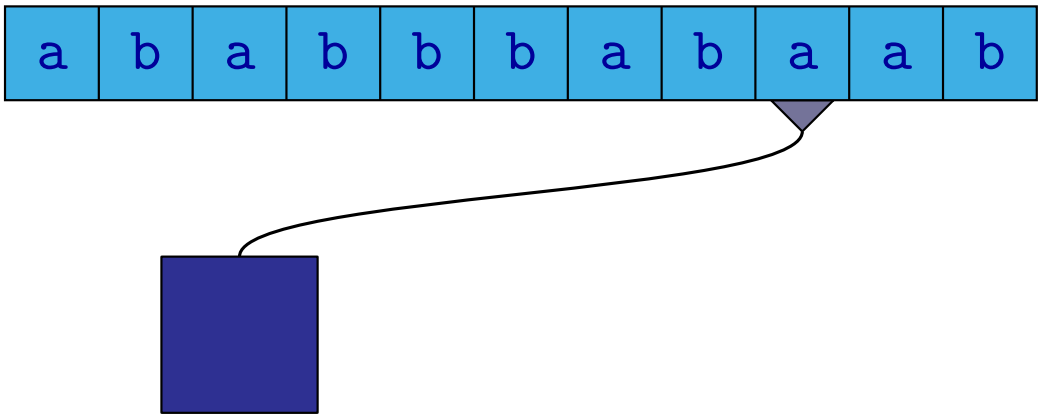


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů $\mathtt{b}$.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

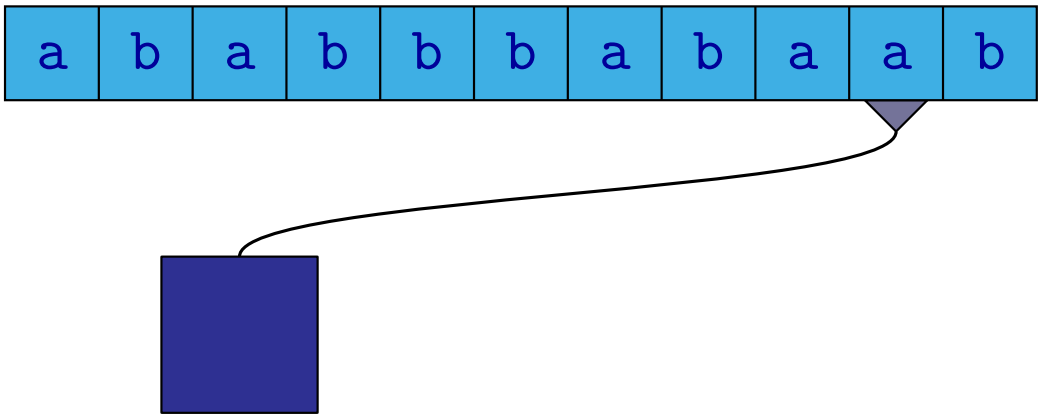


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů $\mathtt{b}$.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

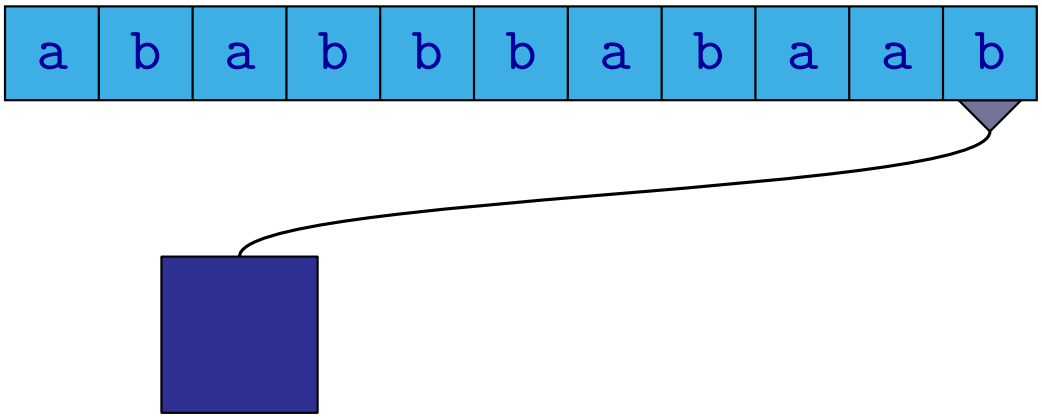


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů $\mathtt{b}$.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

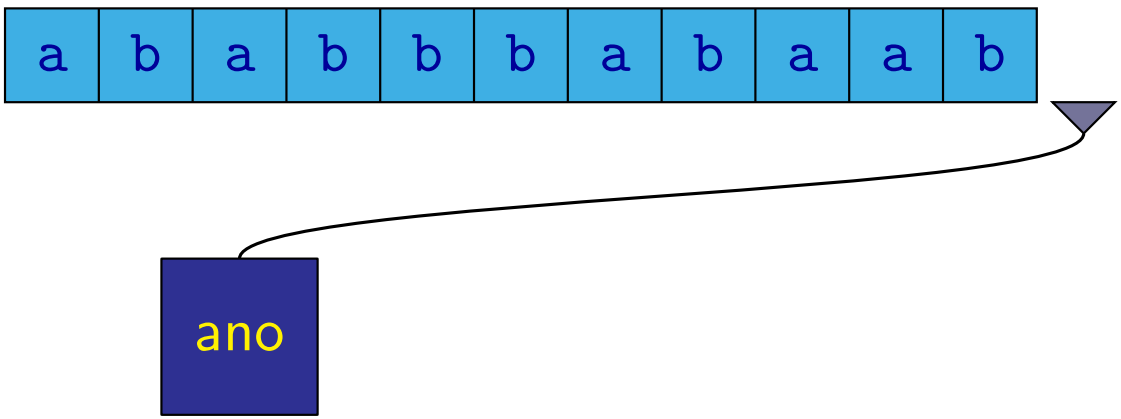


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{\mathtt{a}, \mathtt{b}\}$.

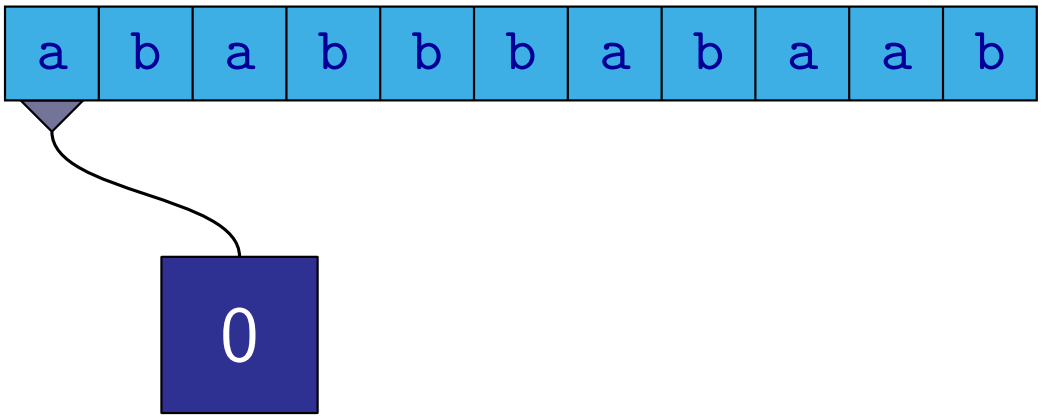
Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů $\mathtt{b}$.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.



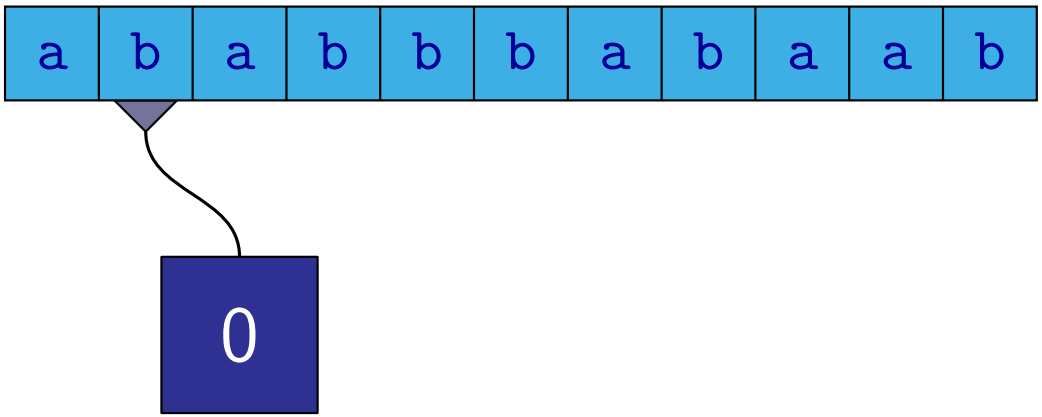
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů `b`.



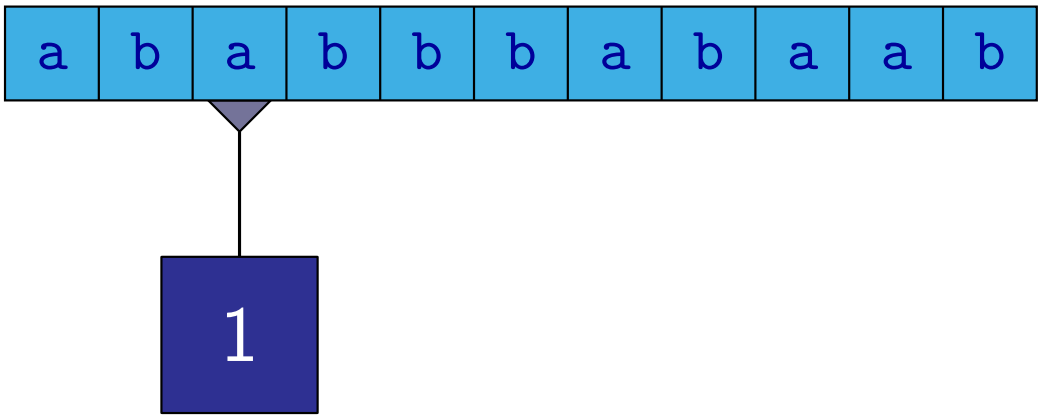
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů `b`.



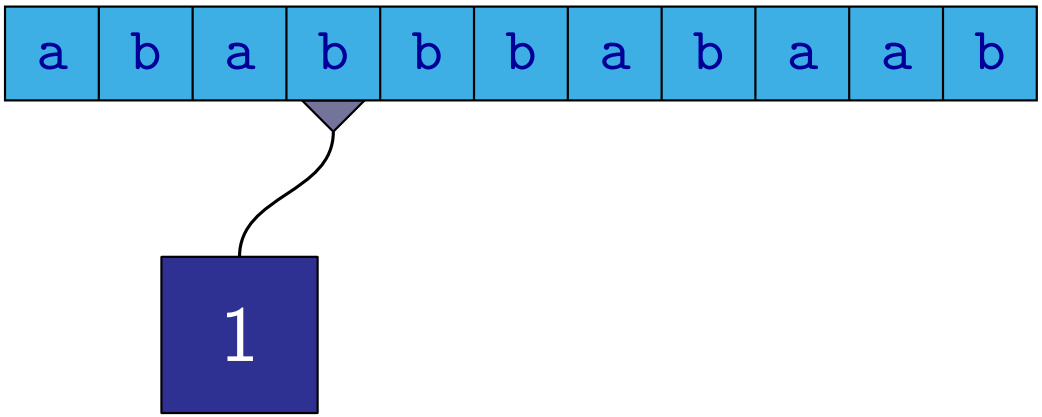
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů `b`.



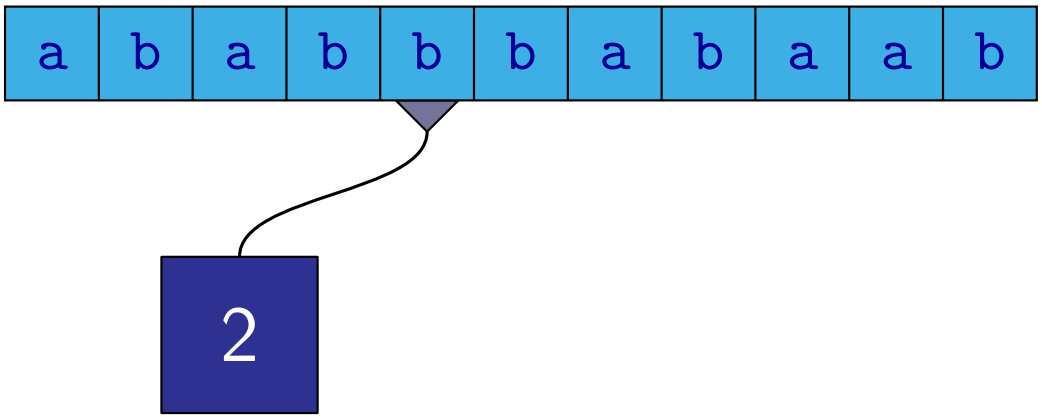
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů `b`.



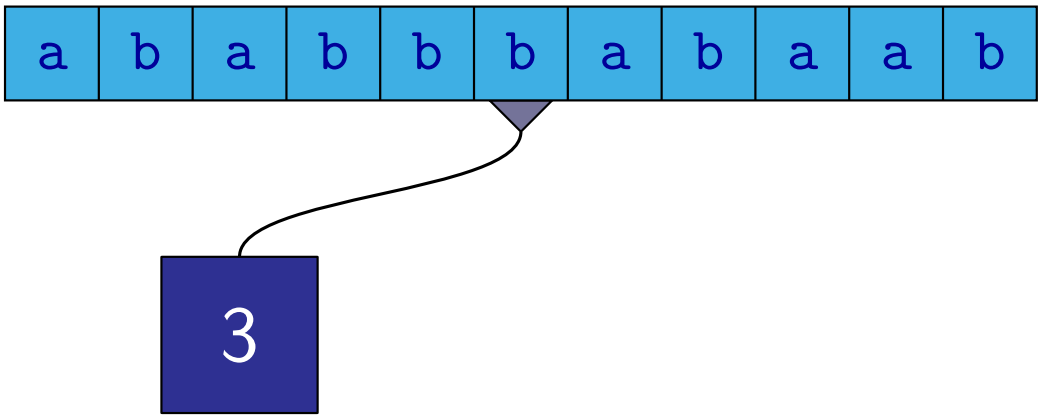
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů `b`.



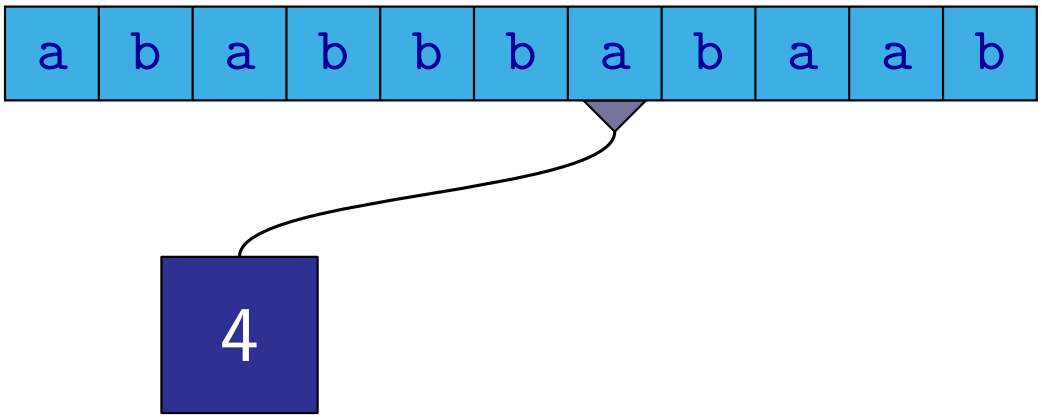
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů `b`.



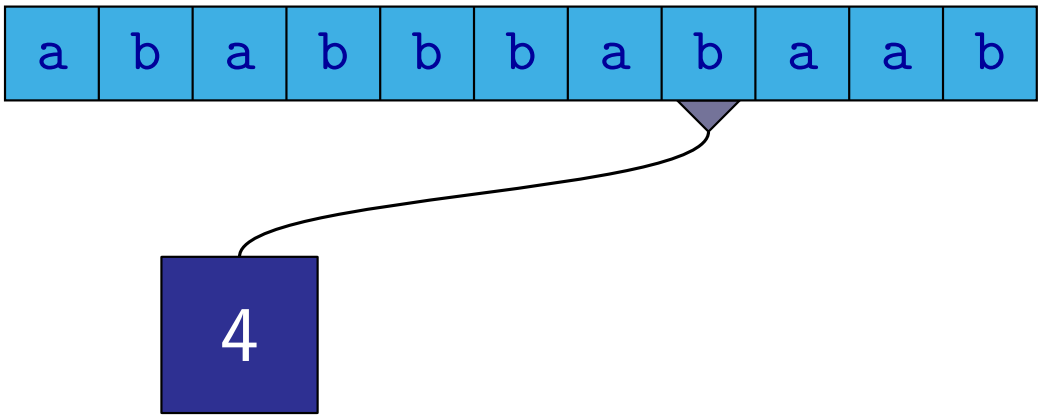
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů `b`.



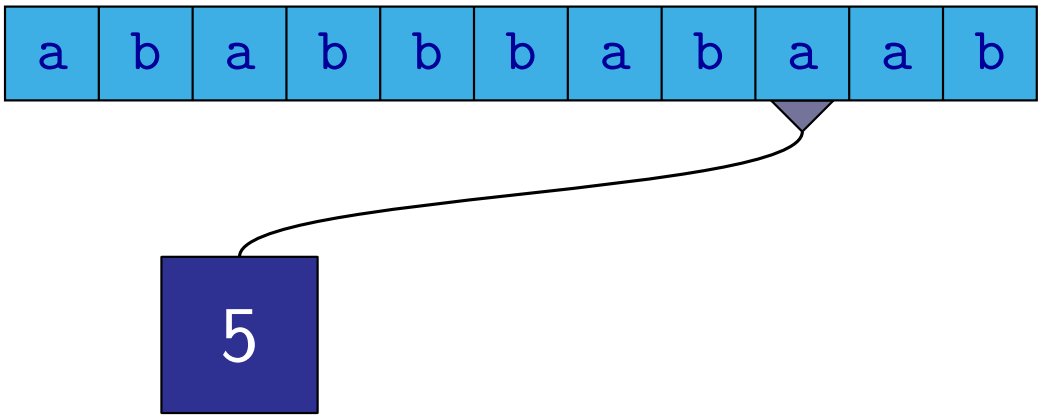
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů `b`.



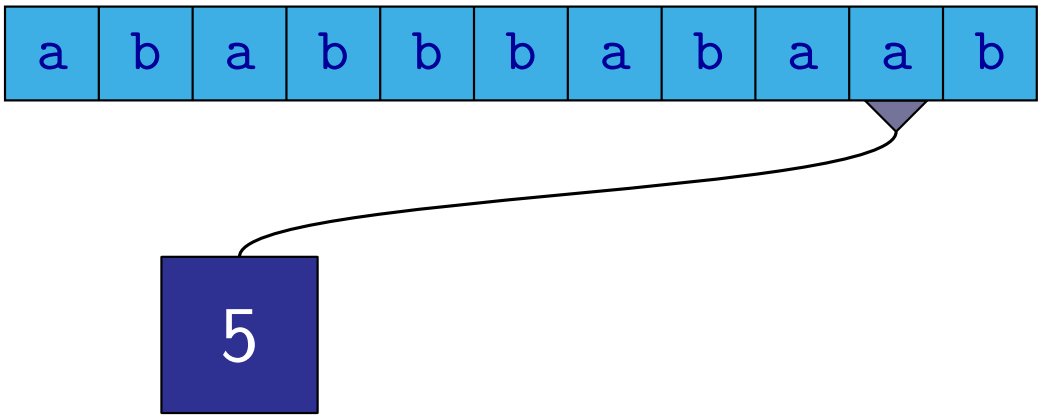
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů `b`.



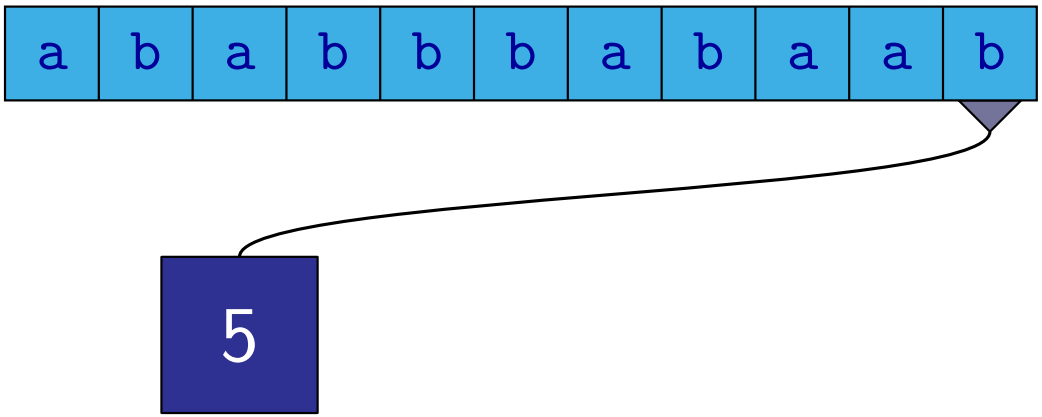
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů `b`.



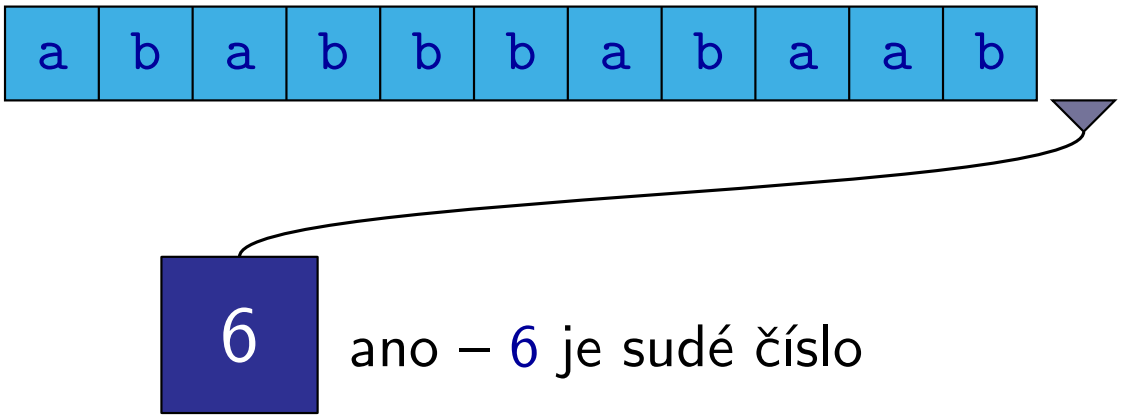
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů `b`.



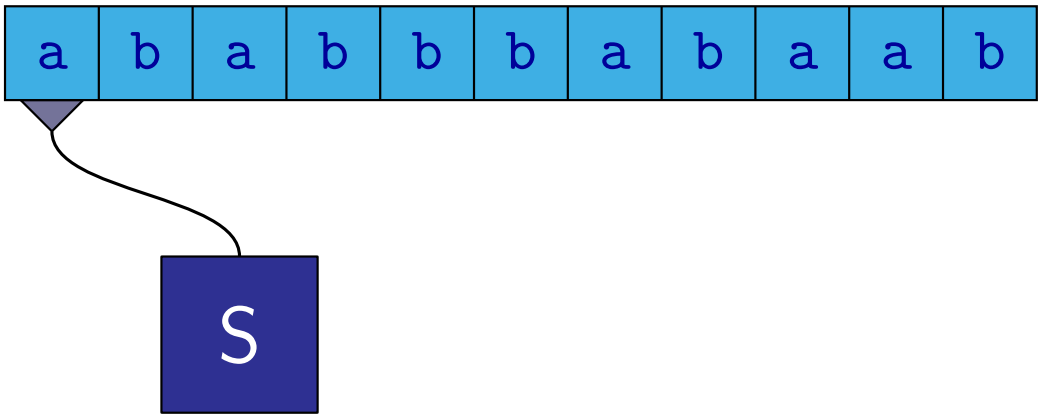
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů b.



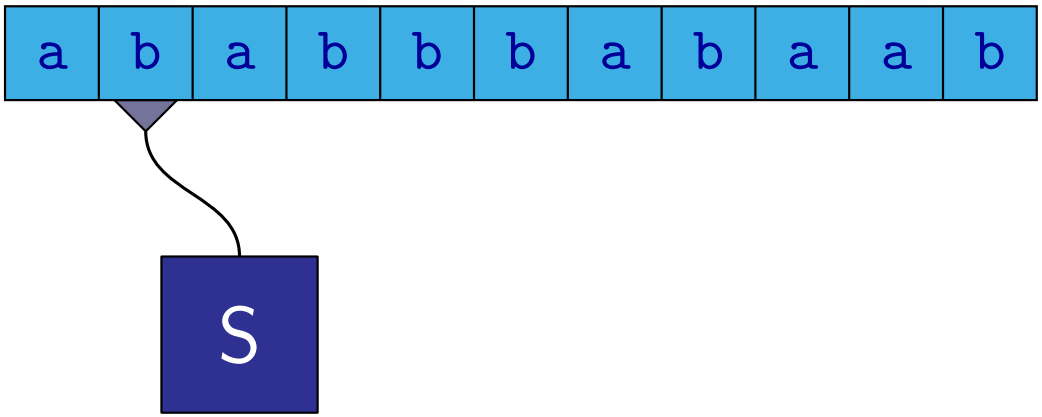
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů `b` je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



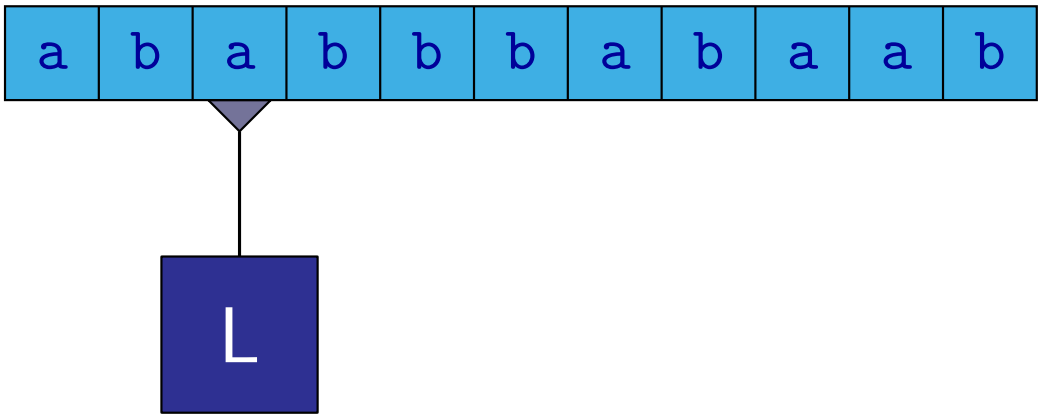
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů `b` je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



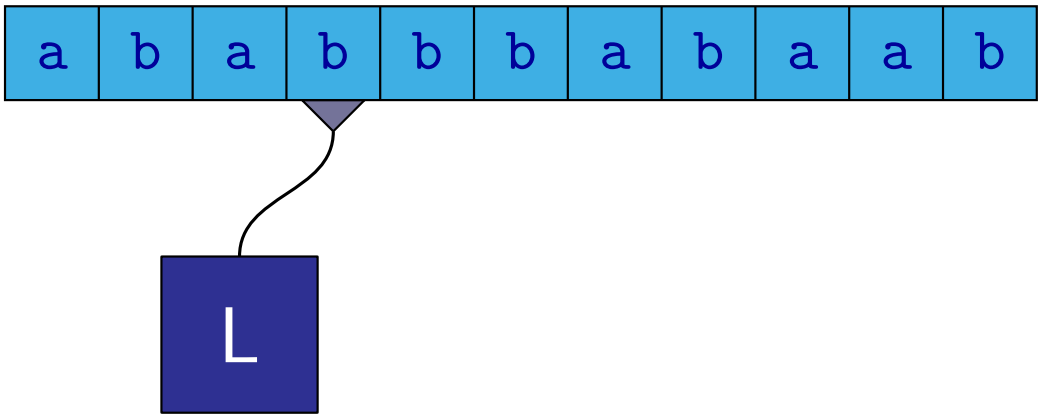
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů `b` je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



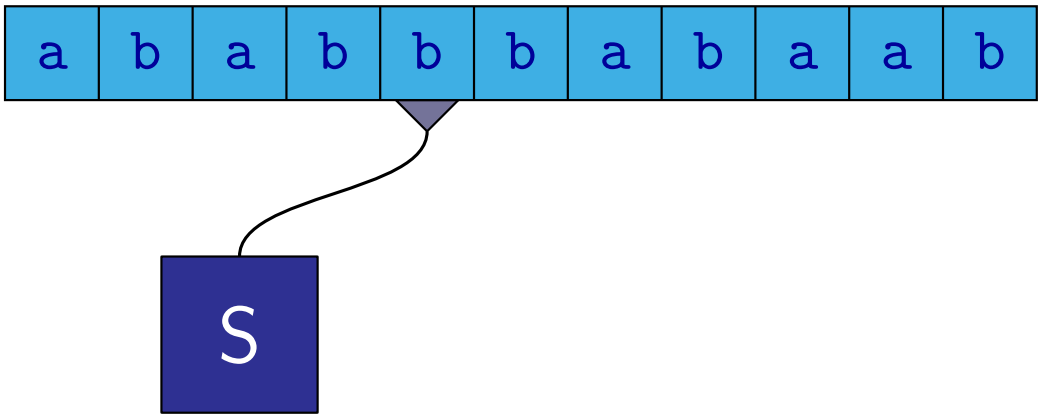
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů `b` je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



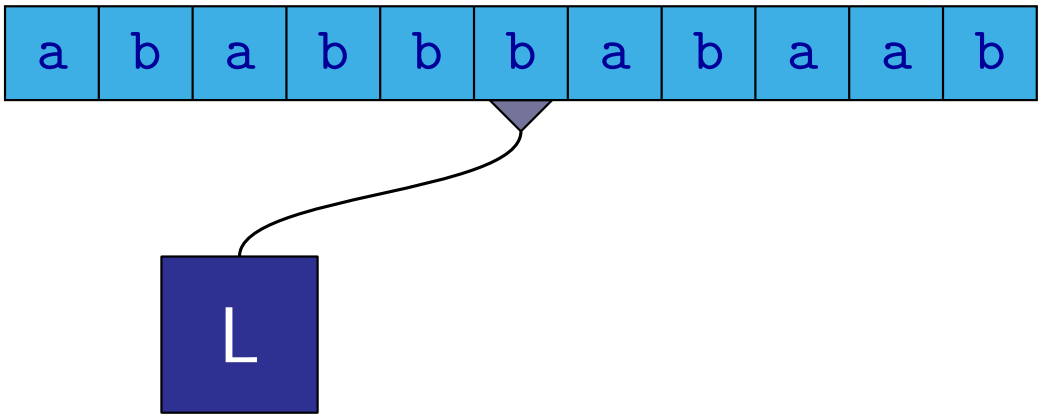
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů `b` je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



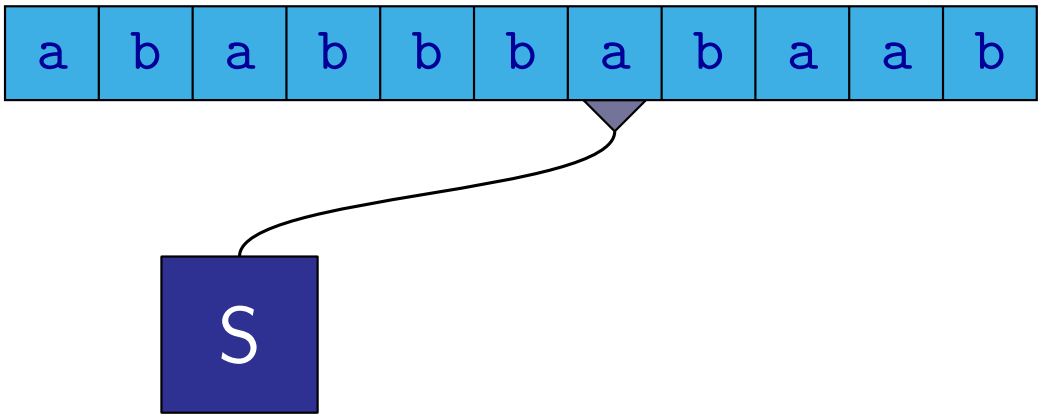
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů `b` je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



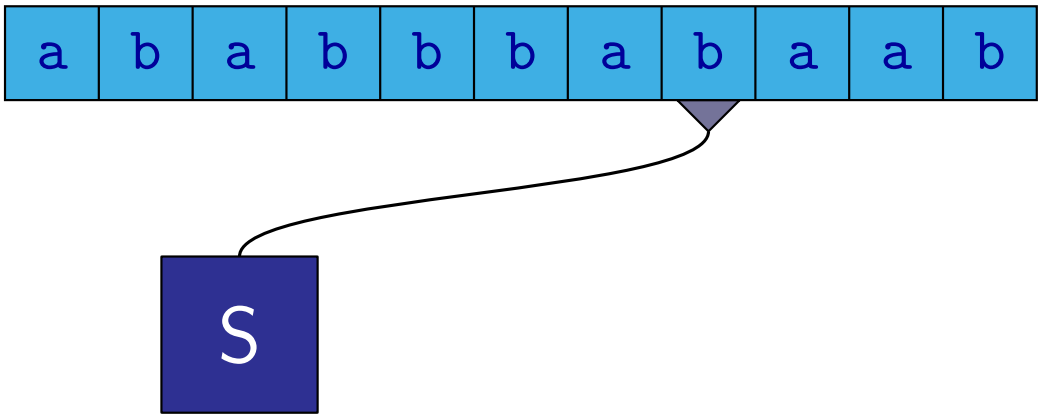
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů `b` je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



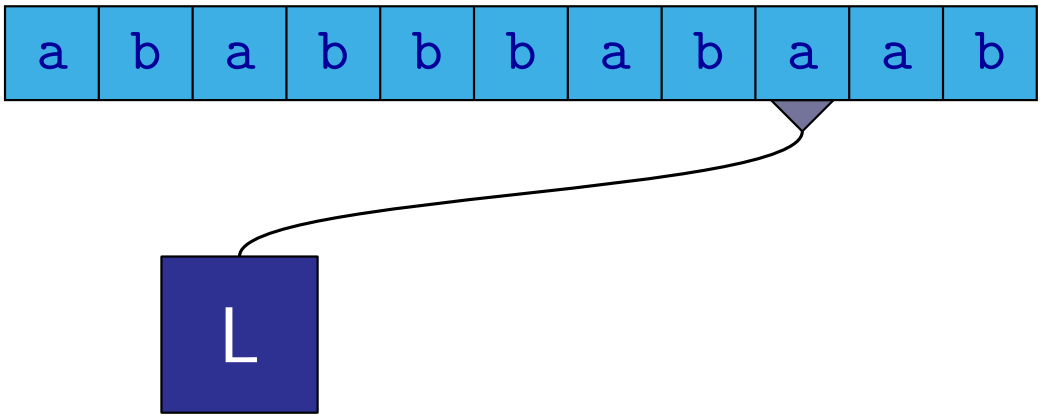
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů `b` je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



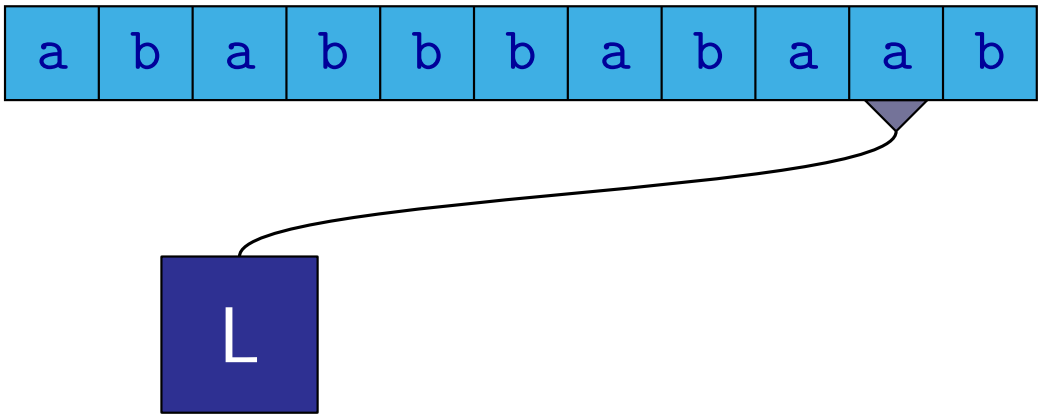
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů `b` je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



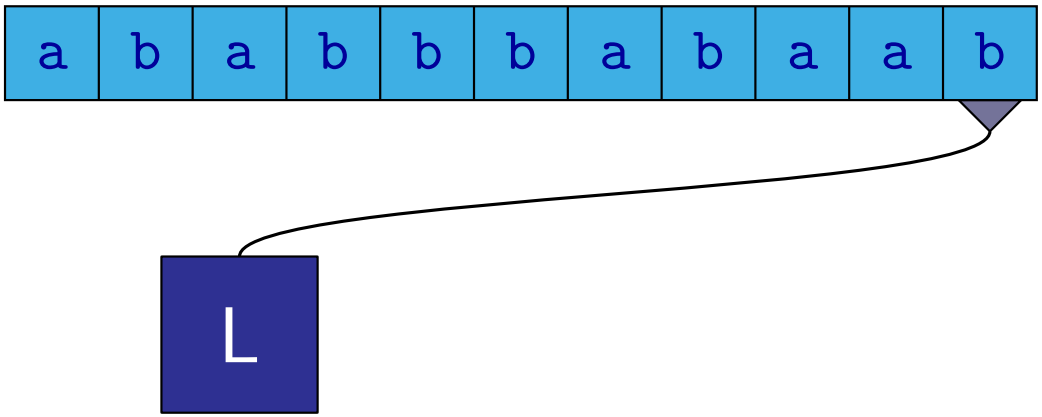
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů `b` je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



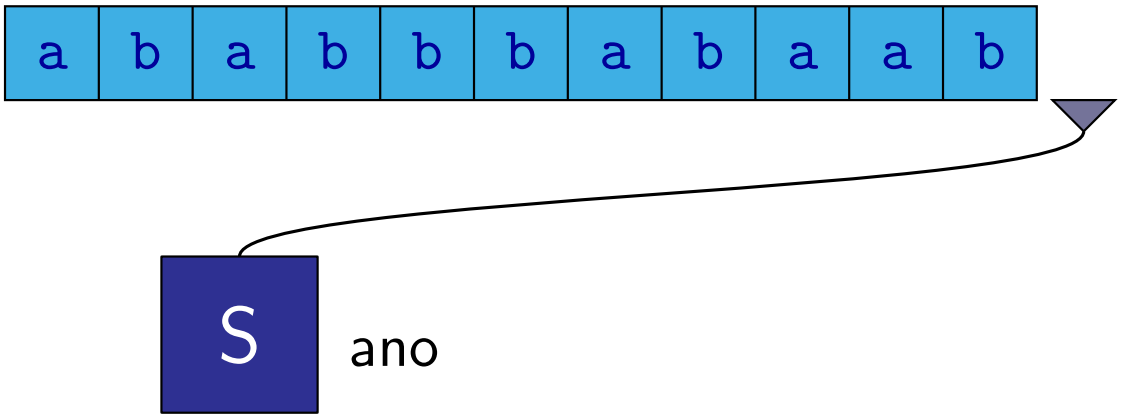
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů `b` je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů `b` je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



Rozpoznávání jazyka

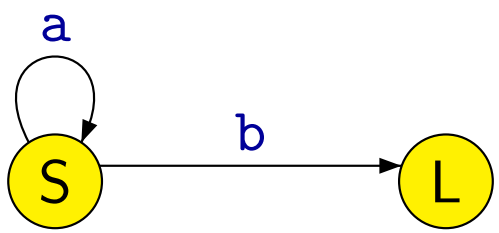
Chovaní tohoto zařízení můžeme popsat grafem:

Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:

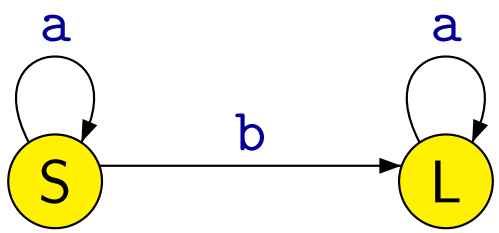
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



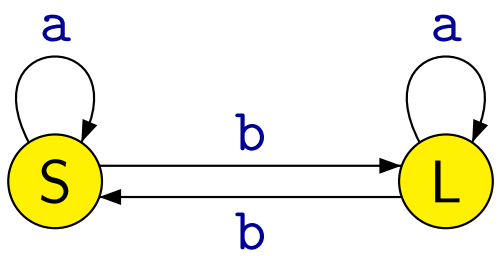
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



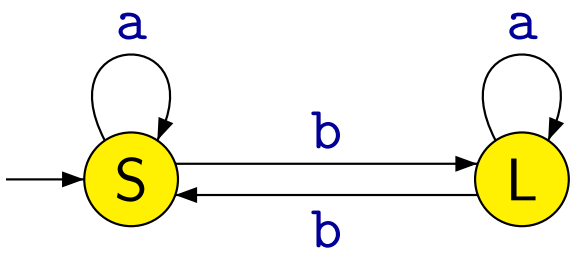
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



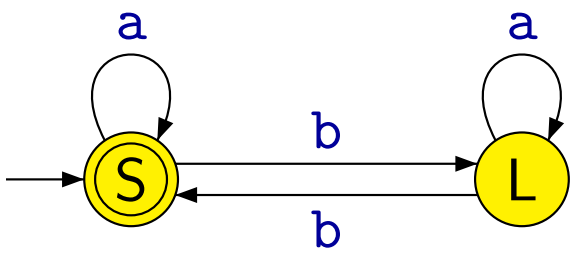
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



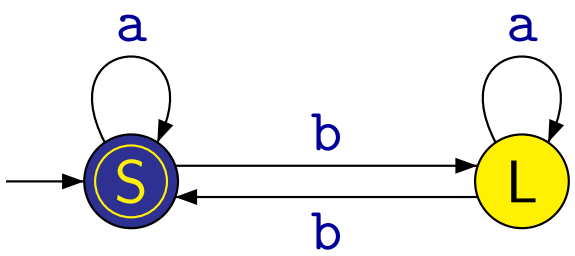
Rozpoznávání jazyka

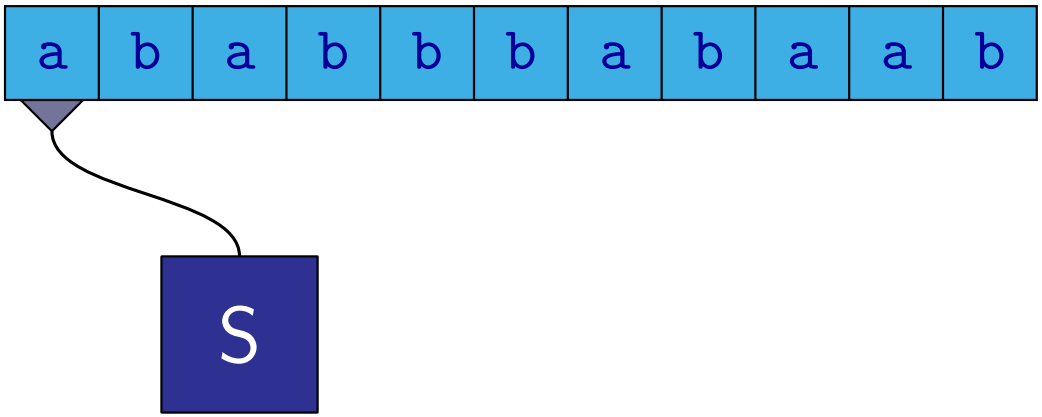
Chovaní tohoto zařízení můžeme popsat grafem:



Rozpoznávání jazyka

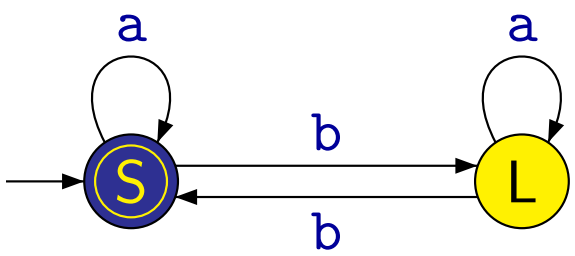
Chovaní tohoto zařízení můžeme popsat grafem:

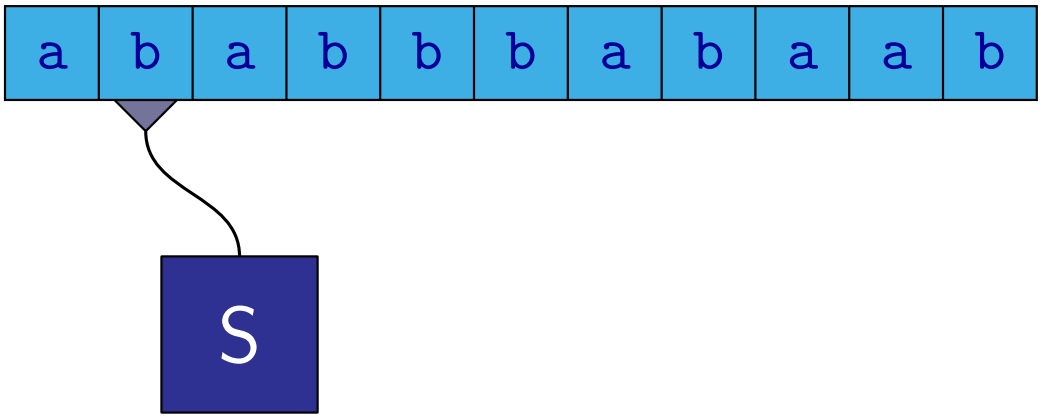




Rozpoznávání jazyka

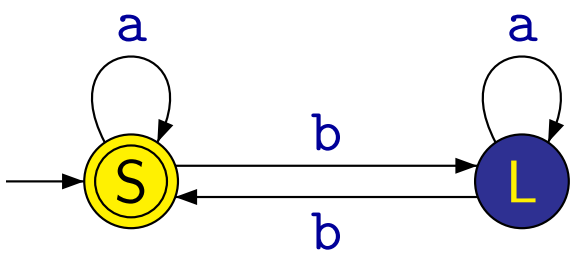
Chovaní tohoto zařízení můžeme popsat grafem:

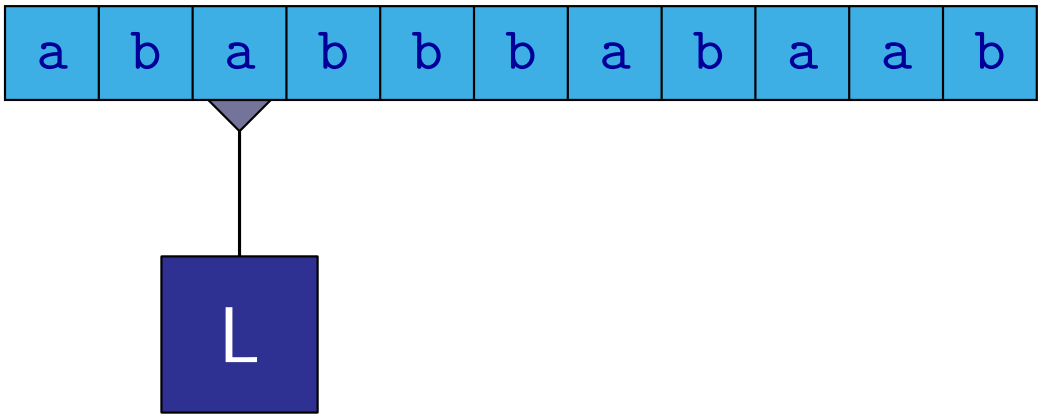




Rozpoznávání jazyka

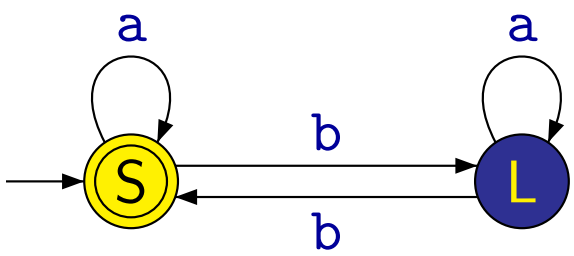
Chovaní tohoto zařízení můžeme popsat grafem:

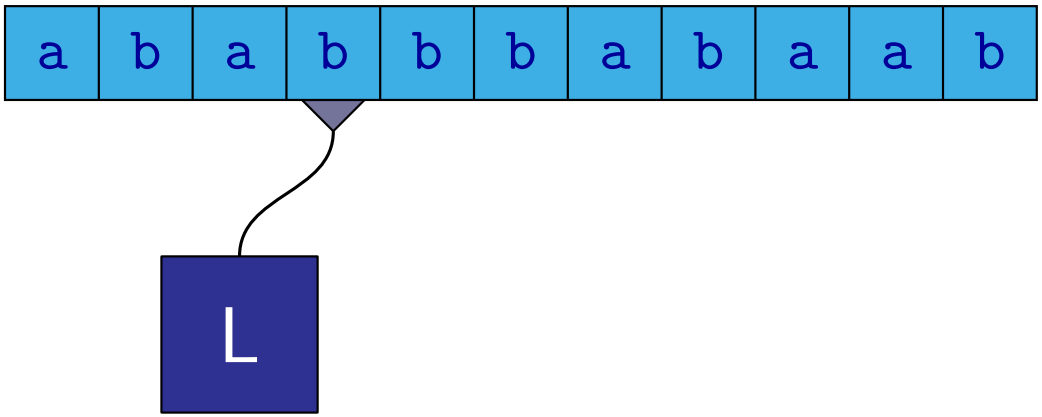




Rozpoznávání jazyka

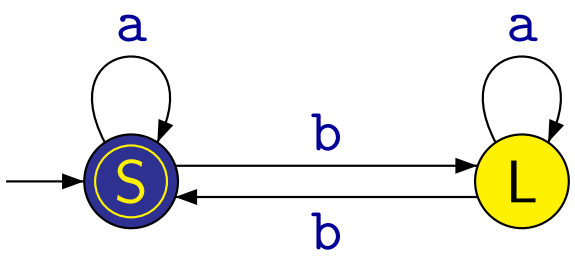
Chovaní tohoto zařízení můžeme popsat grafem:

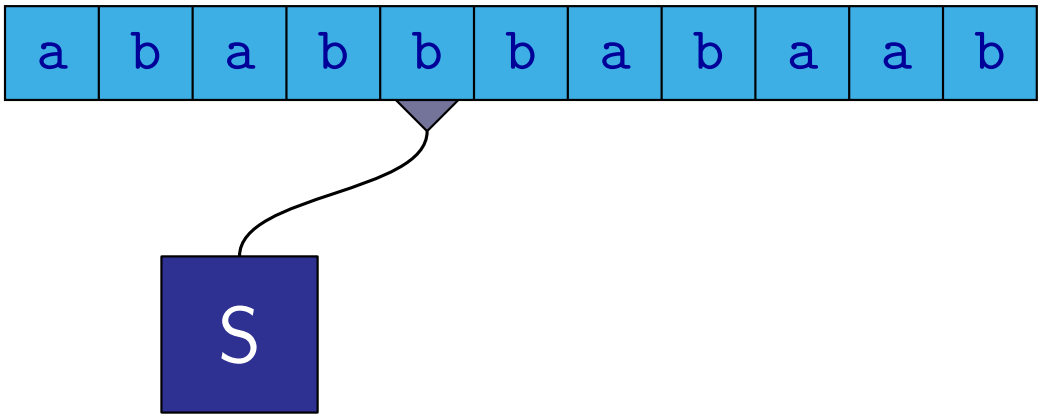




Rozpoznávání jazyka

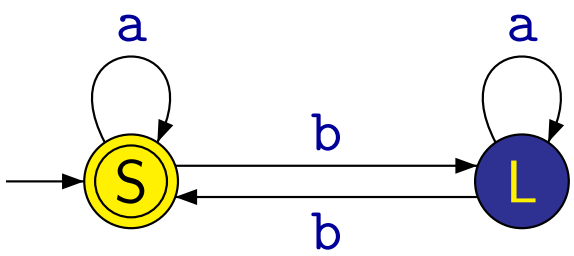
Chovaní tohoto zařízení můžeme popsat grafem:

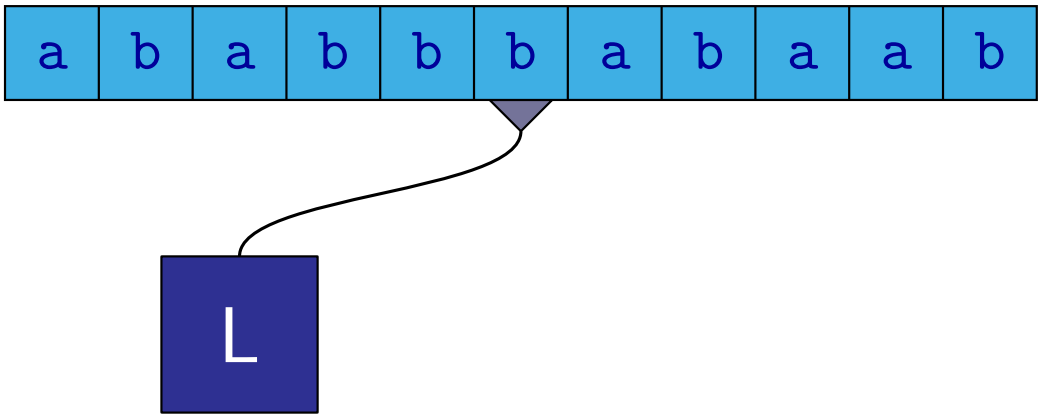




Rozpoznávání jazyka

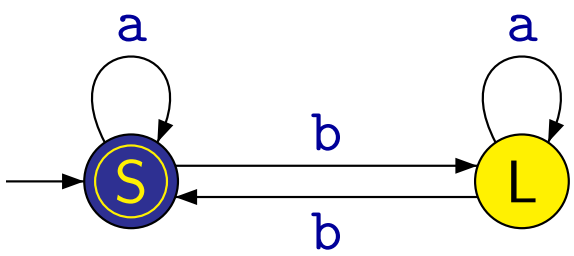
Chovaní tohoto zařízení můžeme popsat grafem:

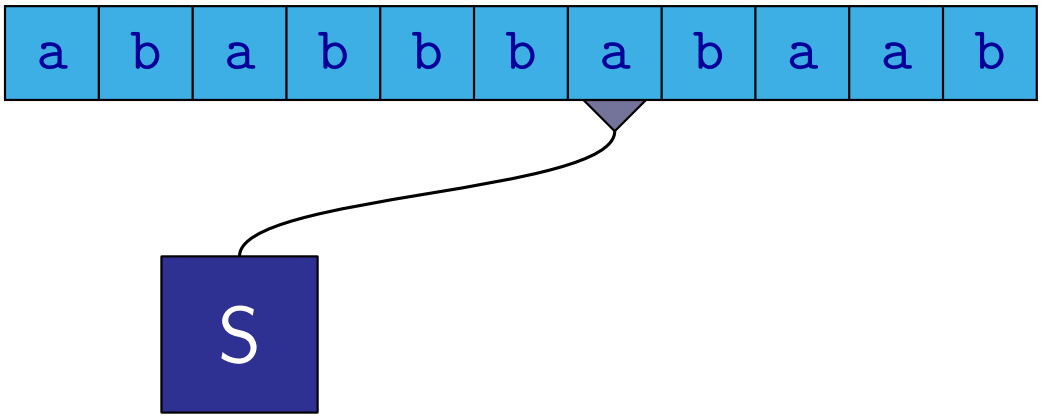




Rozpoznávání jazyka

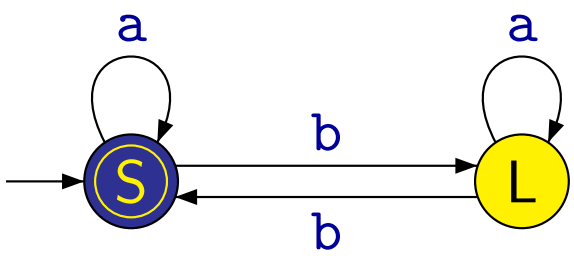
Chovaní tohoto zařízení můžeme popsat grafem:

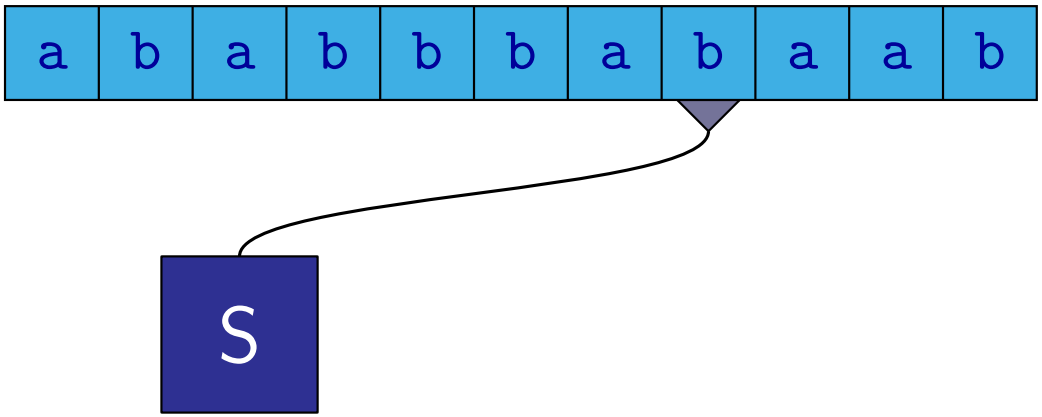




Rozpoznávání jazyka

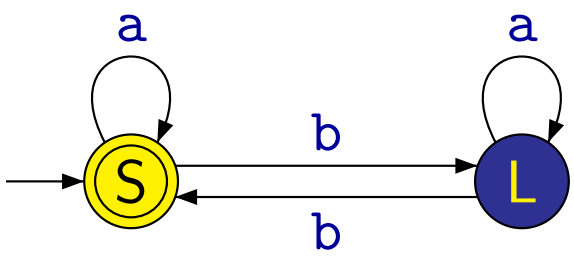
Chovaní tohoto zařízení můžeme popsat grafem:

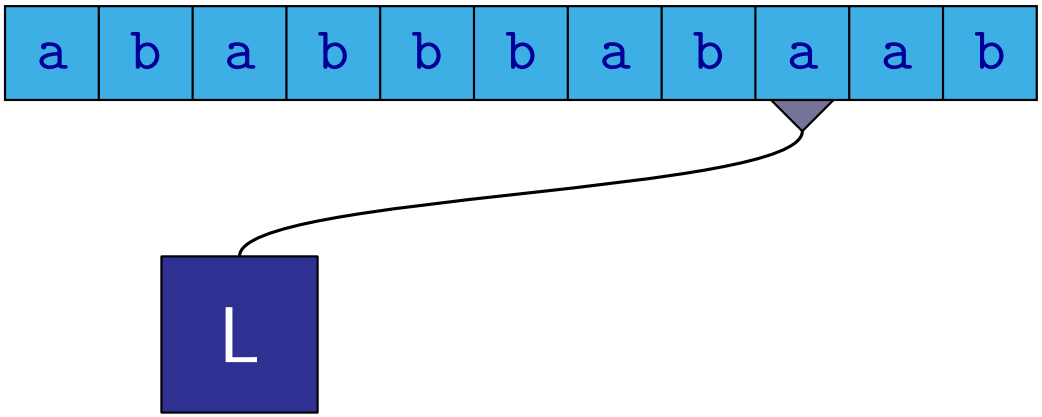




Rozpoznávání jazyka

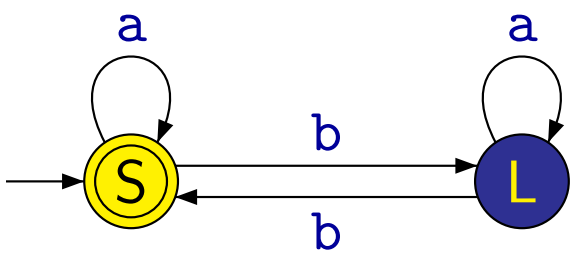
Chovaní tohoto zařízení můžeme popsat grafem:

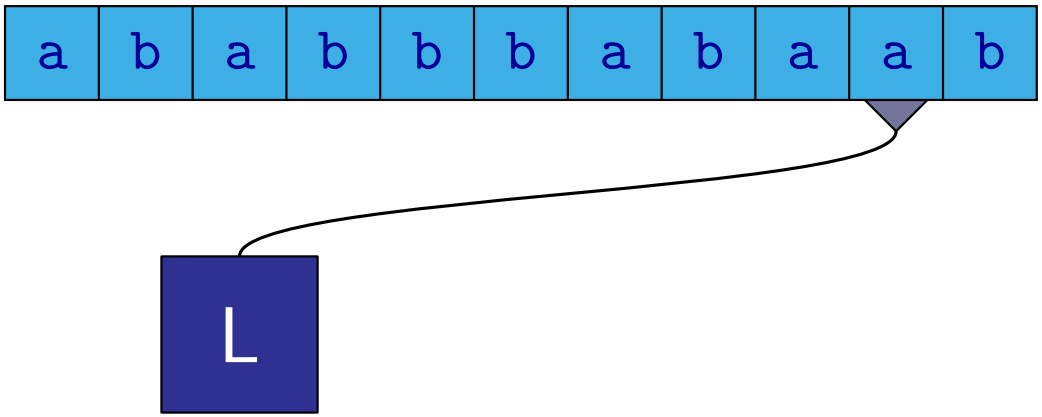




Rozpoznávání jazyka

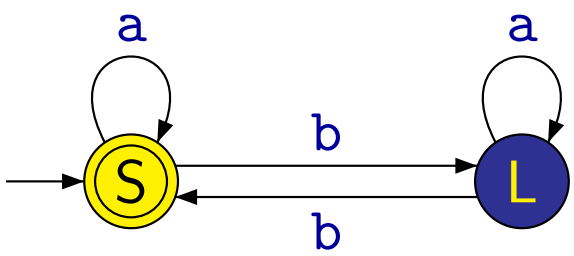
Chovaní tohoto zařízení můžeme popsat grafem:

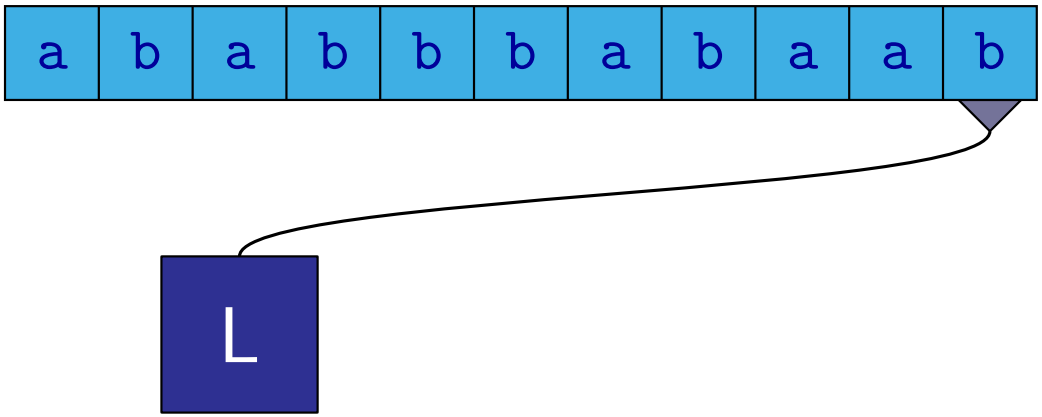




Rozpoznávání jazyka

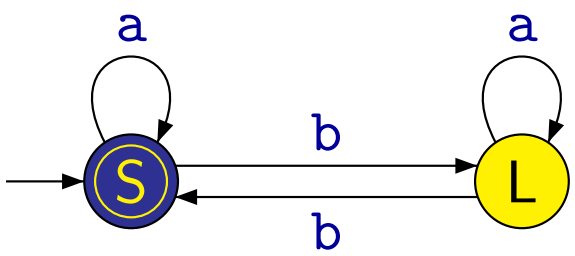
Chovaní tohoto zařízení můžeme popsat grafem:

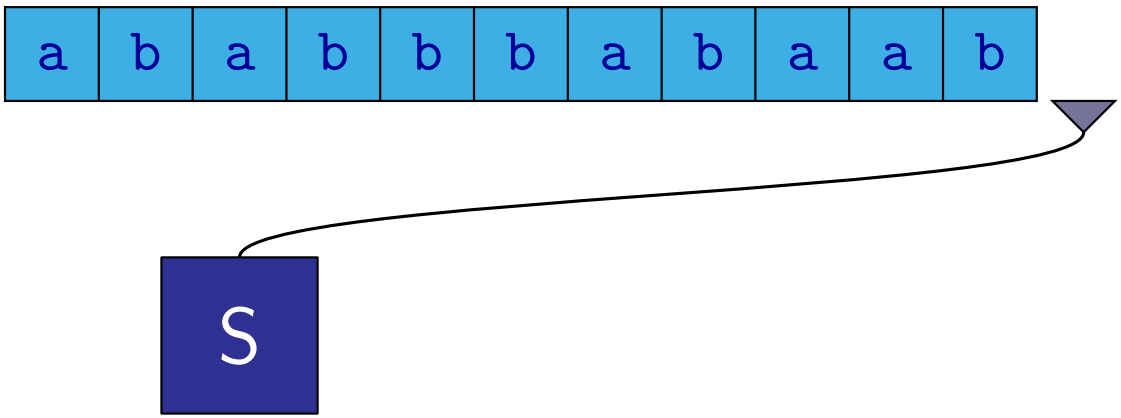




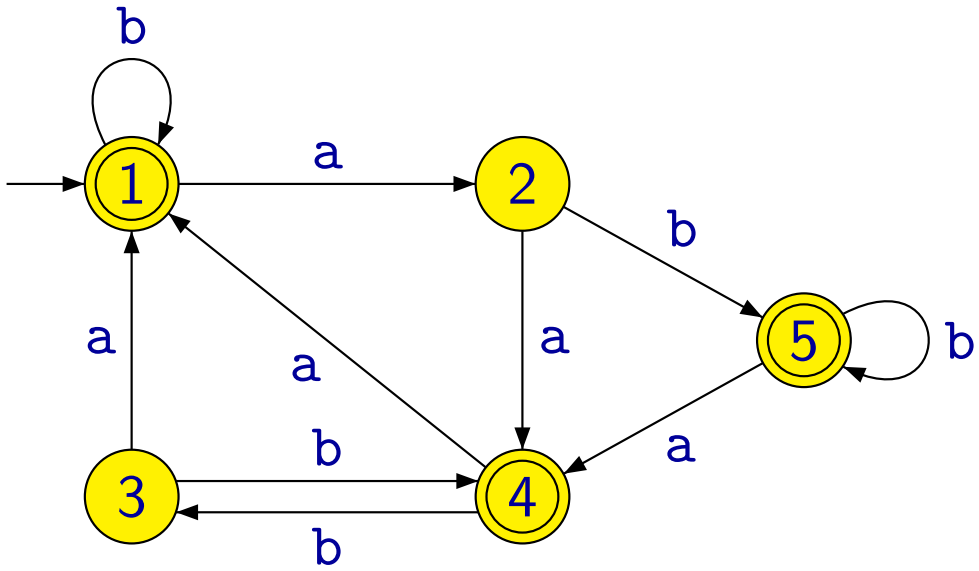
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:





Deterministický konečný automat



Deterministický konečný automat se skládá ze **stavů** a **přechodů**. Jeden ze stavů je označen jako **počáteční stav** a některé ze stavů jsou označeny jako **přijímající**.

Deterministický konečný automat

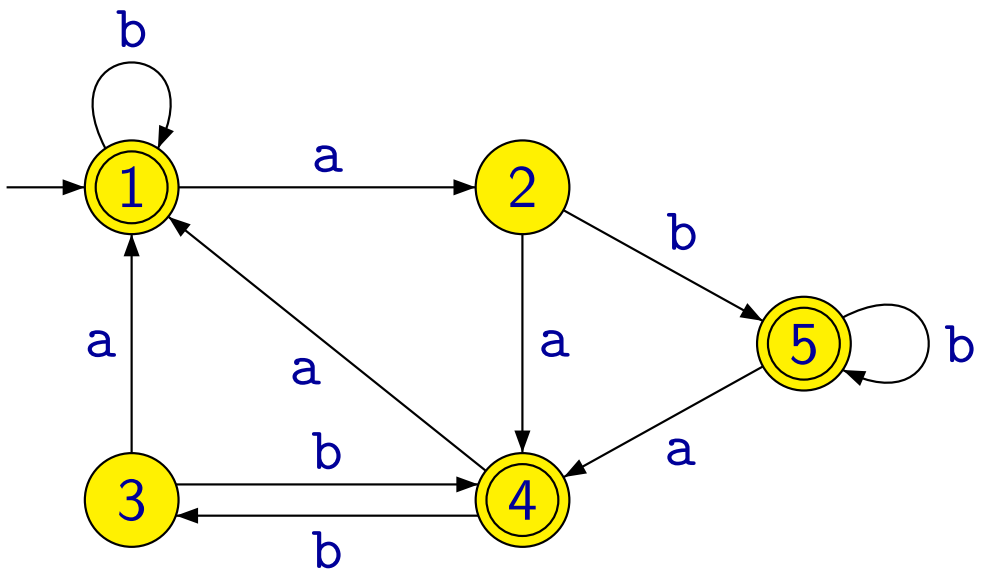
Formálně je **deterministický konečný automat** (**DKA**) definován jako pětice

$$(Q, \Sigma, \delta, q_0, F)$$

kde:

- Q je neprázdná konečná množina **stavů**
- Σ je **abeceda** (neprázdná konečná množina symbolů)
- $\delta : Q \times \Sigma \to Q$ je **přechodová funkce**
- $q_0 \in Q$ je **počáteční stav**
- $F \subseteq Q$ je množina **přijímajících stavů**

Deterministický konečný automat



- $Q = \{1, 2, 3, 4, 5\}$
- $\Sigma = \{\mathtt{a}, \mathtt{b}\}$
- $q_0 = 1$
- $F = \{1, 4, 5\}$

$$\delta(1, \mathtt{a}) = 2 \qquad \delta(1, \mathtt{b}) = 1$$
$$\delta(2, \mathtt{a}) = 4 \qquad \delta(2, \mathtt{b}) = 5$$
$$\delta(3, \mathtt{a}) = 1 \qquad \delta(3, \mathtt{b}) = 4$$
$$\delta(4, \mathtt{a}) = 1 \qquad \delta(4, \mathtt{b}) = 3$$
$$\delta(5, \mathtt{a}) = 4 \qquad \delta(5, \mathtt{b}) = 5$$

Deterministický konečný automat

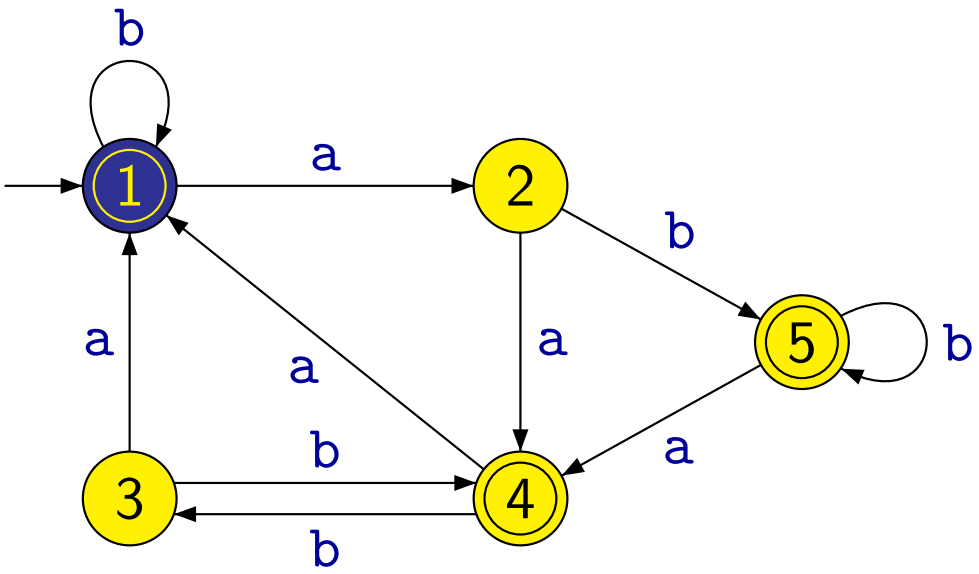
Místo zápisu

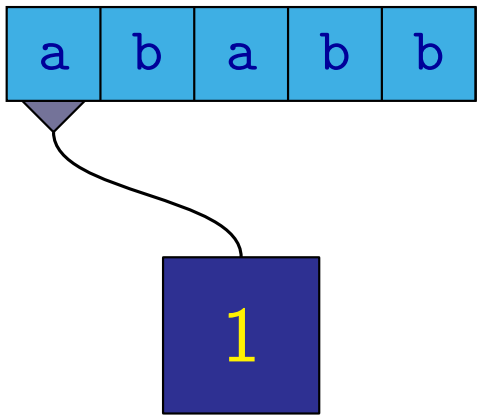
$$\begin{array}{ll} \delta(1,\mathtt{a}) = 2 & \delta(1,\mathtt{b}) = 1 \\ \delta(2,\mathtt{a}) = 4 & \delta(2,\mathtt{b}) = 5 \\ \delta(3,\mathtt{a}) = 1 & \delta(3,\mathtt{b}) = 4 \\ \delta(4,\mathtt{a}) = 1 & \delta(4,\mathtt{b}) = 3 \\ \delta(5,\mathtt{a}) = 4 & \delta(5,\mathtt{b}) = 5 \end{array}$$

budeme raději používat stručnější tabulku nebo grafické znázornění:

	δ	a	b
$\leftrightarrow$	1	2	1
	2	4	5
	3	1	4
$\leftarrow$	4	1	3
$\leftarrow$	5	4	5

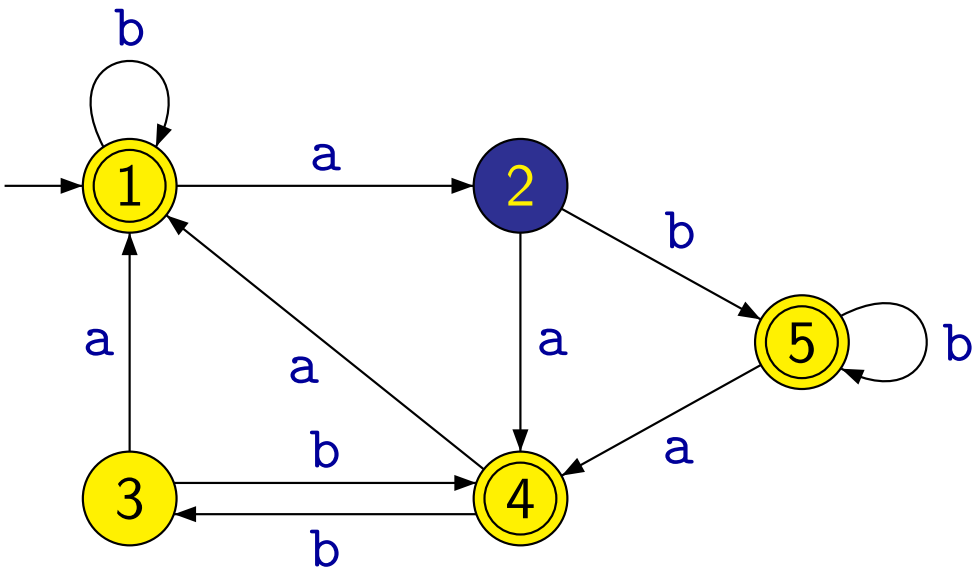
Deterministický konečný automat

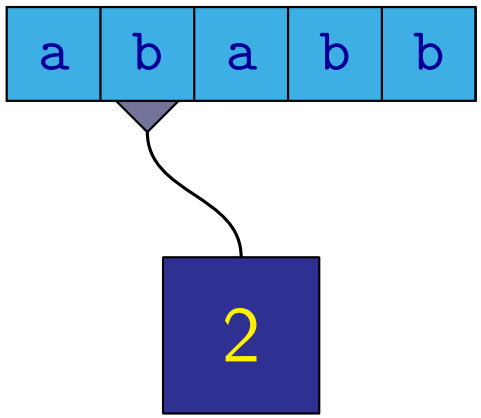




1

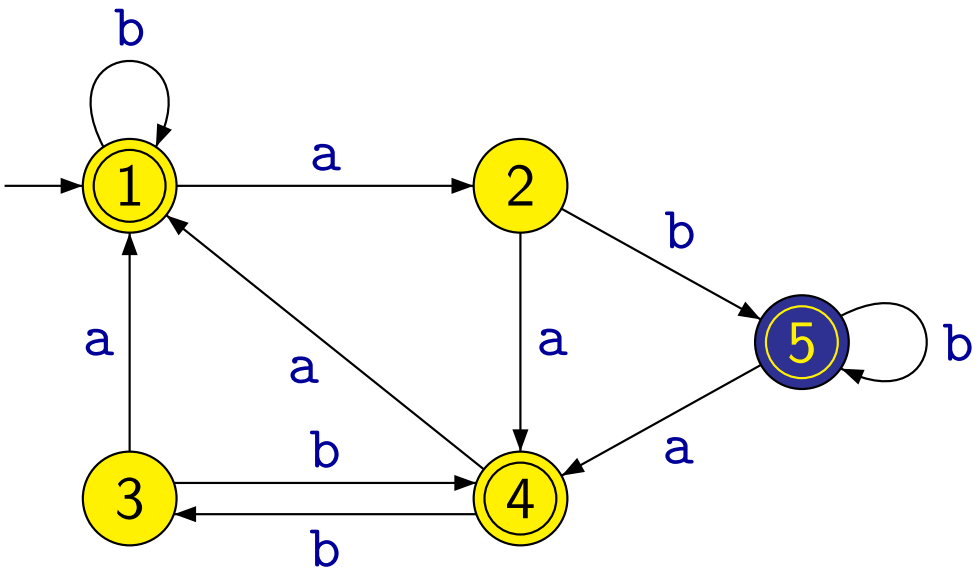
Deterministický konečný automat

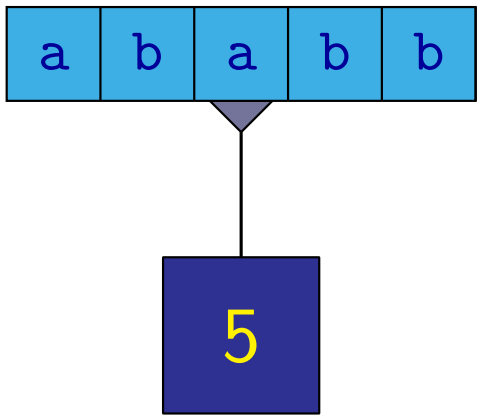




$1 \xrightarrow{a} 2$

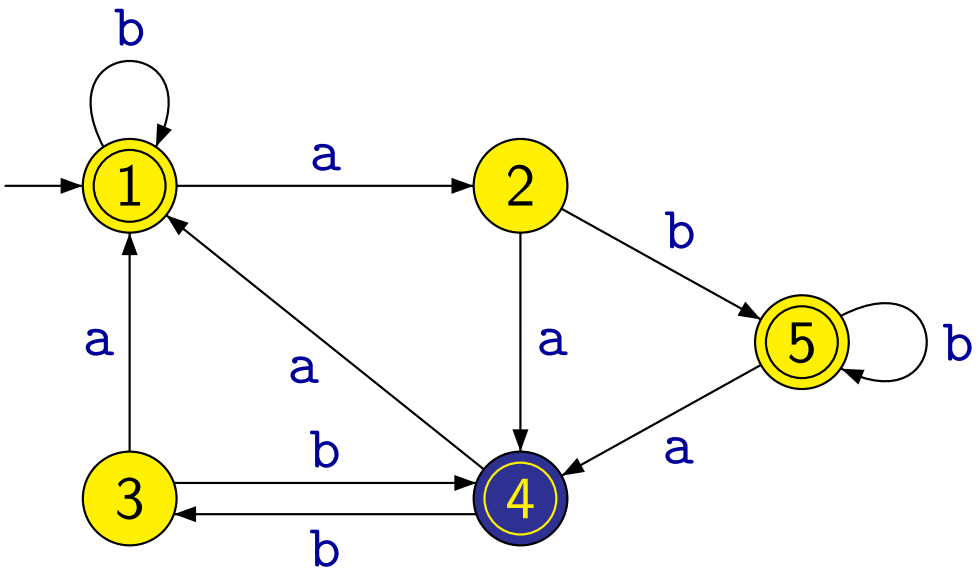
Deterministický konečný automat

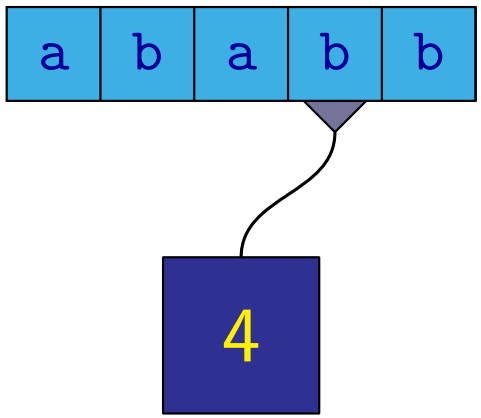




$1 \xrightarrow{a} 2 \xrightarrow{b} 5$

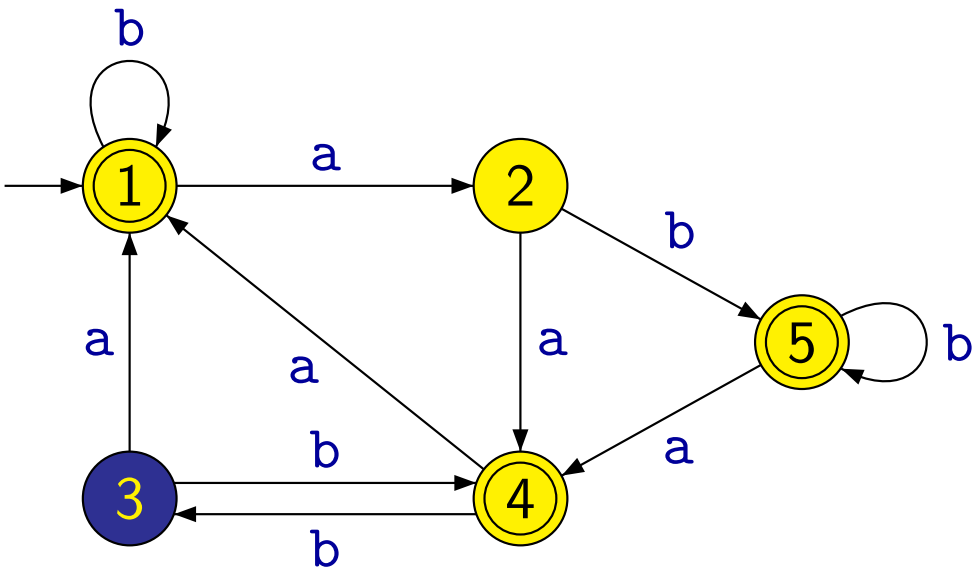
Deterministický konečný automat

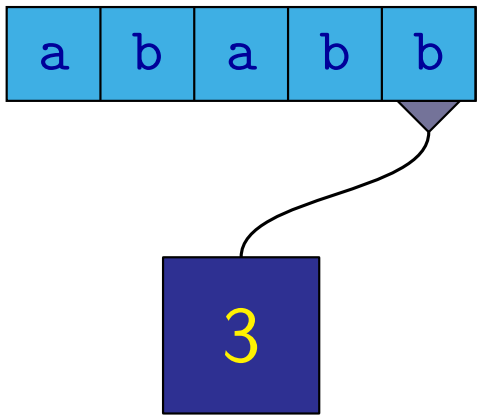




$$1 \xrightarrow{a} 2 \xrightarrow{b} 5 \xrightarrow{a} 4$$

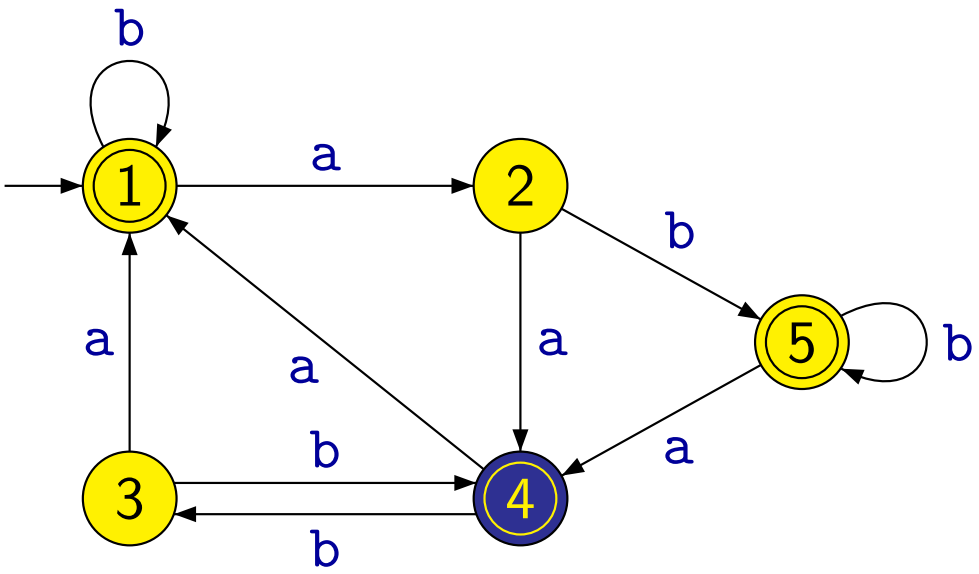
Deterministický konečný automat

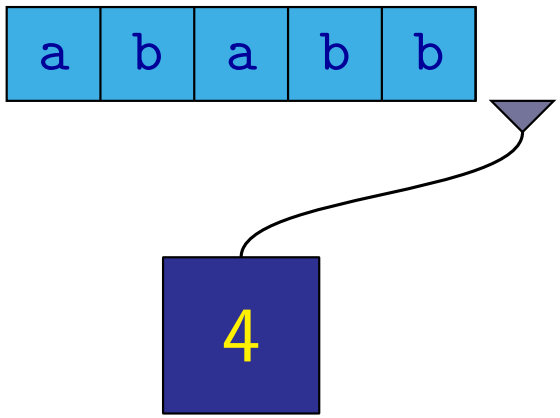




$$1 \xrightarrow{a} 2 \xrightarrow{b} 5 \xrightarrow{a} 4 \xrightarrow{b} 3$$

Deterministický konečný automat





$$1 \xrightarrow{a} 2 \xrightarrow{b} 5 \xrightarrow{a} 4 \xrightarrow{b} 3 \xrightarrow{b} 4$$

Deterministický konečný automat

Definice

Mějme DKA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$.

Zápisem $q \xrightarrow{w} q'$, kde $q, q' \in Q$ a $w \in \Sigma^*$, budeme označovat to, že pokud je automat ve stavu q, tak přečtením slova w přejde do stavu q'.

Poznámka: $\longrightarrow \subseteq Q \times \Sigma^* \times Q$ je ternární relace.

Místo $(q, w, q') \in \longrightarrow$ píšeme $q \xrightarrow{w} q'$.

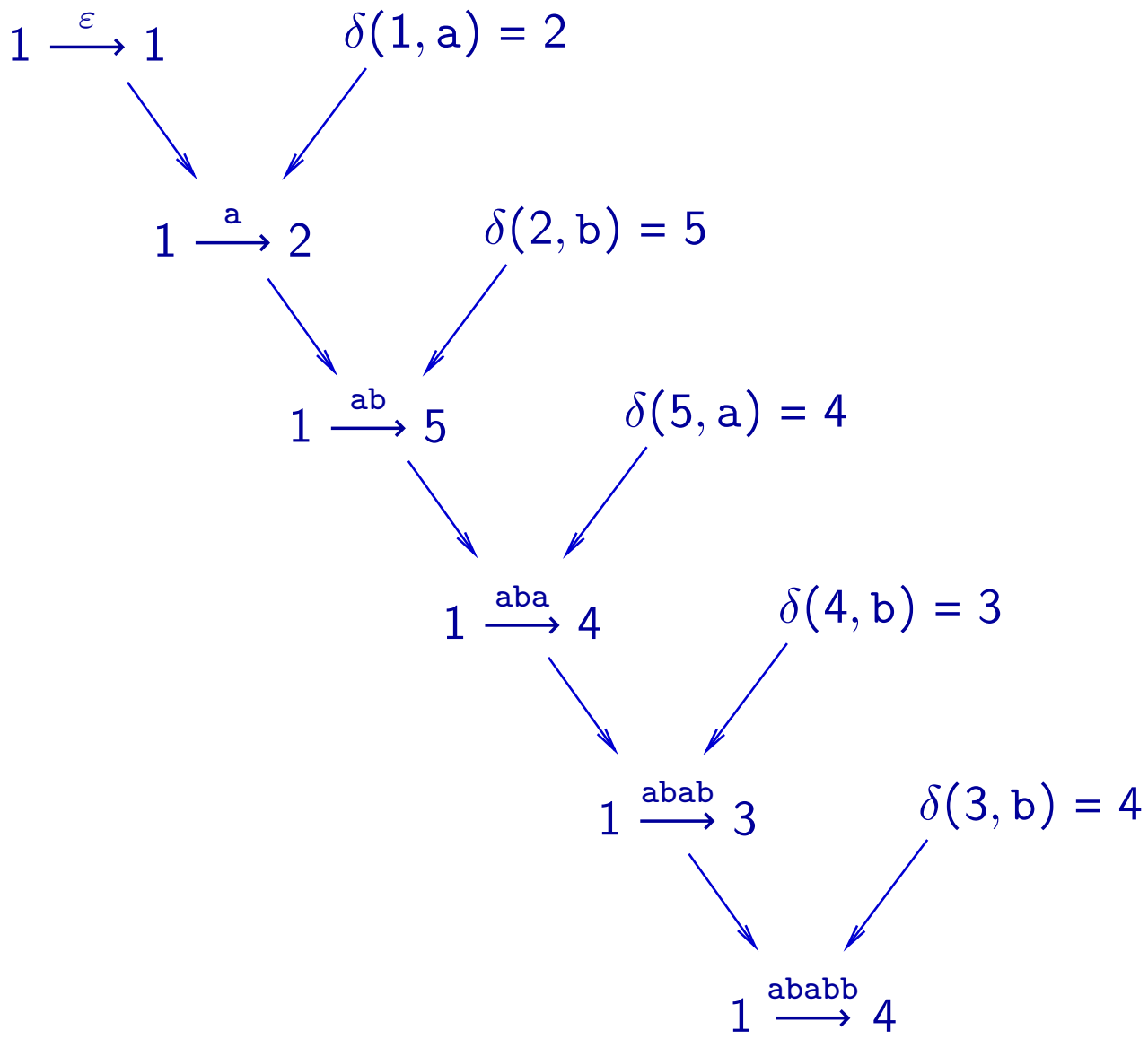
Pro DKA platí, že pro libovolný stav q a libovolné slovo w existuje právě jeden stav q' takový, že $q \xrightarrow{w} q'$.

Deterministický konečný automat

Relaci $\longrightarrow$ můžeme formálně definovat následující induktivní definicí:

- $q \xrightarrow{\varepsilon} q$ pro libovolné $q \in Q$
- Pro $w \in \Sigma^*$ a $a \in \Sigma$:
  $q \xrightarrow{wa} q'$ právě tehdy, když existuje $q'' \in Q$ takové, že
  $$q \xrightarrow{w} q'' \text{ a } \delta(q'', a) = q'$$

Deterministický konečný automat



	a	b
$\leftrightarrow$ 1	2	1
2	4	5
3	1	4
$\leftarrow$ 4	1	3
$\leftarrow$ 5	4	5

Deterministický konečný automat

Slovo $w \in \Sigma^*$ je **přijímáno** deterministickým konečným automatem $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ právě tehdy, když existuje stav $q \in F$ takový, že $q_0 \xrightarrow{w} q$.

Definice

Jazyk rozpoznávaný (přijímaný) daným deterministickým konečným automatem $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, označovaný $\mathcal{L}(\mathcal{A})$, je množina všech slov přijímaných tímto automatem, tj.

$$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \exists q \in F : q_0 \xrightarrow{w} q\}$$

Regulární jazyky

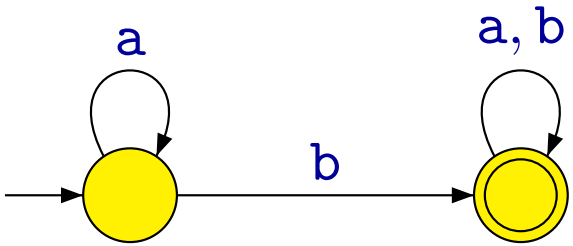
Definice

Jazyk L je **regulární** právě tehdy, když existuje nějaký deterministický konečný automat $\mathcal{A}$, který jej přijímá, tj. DKA $\mathcal{A}$ takový, že $\mathcal{L}(\mathcal{A}) = L$.

Příklady deterministických konečných automatů

Příklad: Automat rozpoznávající jazyk L nad abecedou $\{\mathtt{a}, \mathtt{b}\}$ tvořený slovy, která obsahují alespoň jeden výskyt symbolu $\mathtt{b}$, tj.

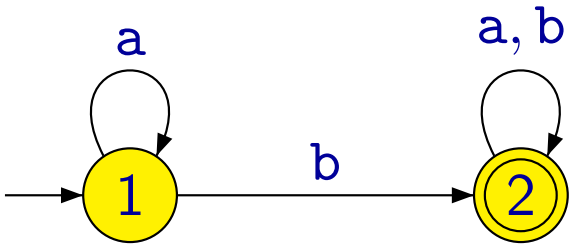
$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid |w|_{\mathtt{b}} \geq 1\}$$



Příklady deterministických konečných automatů

Příklad: Automat rozpoznávající jazyk L nad abecedou $\{\mathtt{a}, \mathtt{b}\}$ tvořený slovy, která obsahují alespoň jeden výskyt symbolu $\mathtt{b}$, tj.

$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid |w|_{\mathtt{b}} \geq 1\}$$

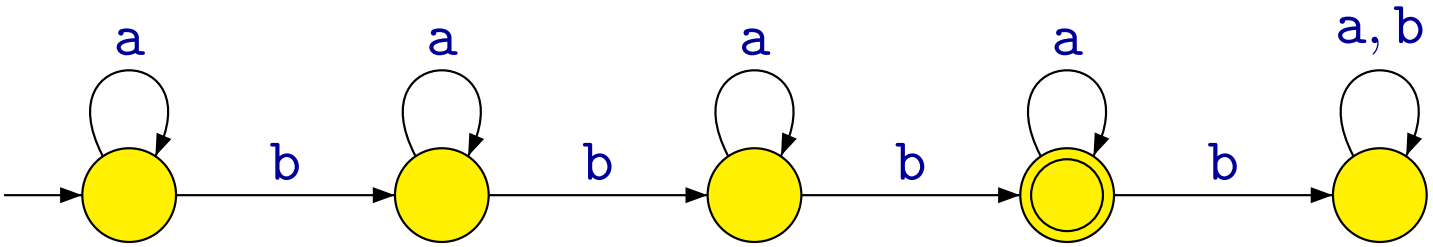


	a	b
$\rightarrow 1$	1	2
$\leftarrow 2$	2	2

Příklady deterministických konečných automatů

Příklad: Automat rozpoznávající jazyk L nad abecedou $\{\mathtt{a}, \mathtt{b}\}$ tvořený slovy, která obsahují právě tři výskyty symbolu $\mathtt{b}$, tj.

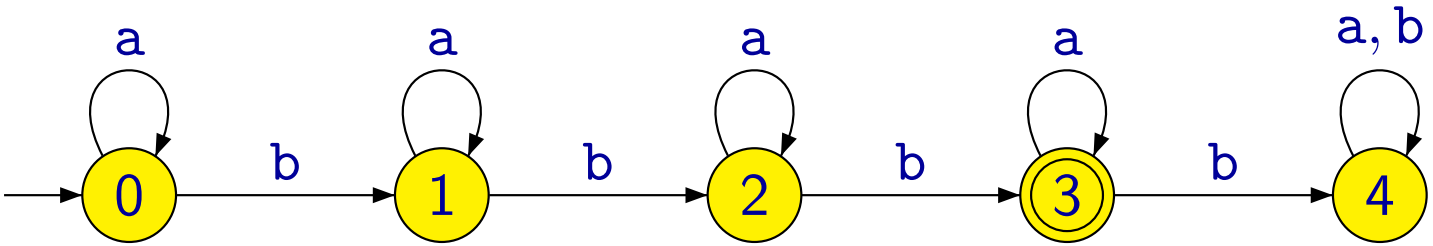
$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid |w|_{\mathtt{b}} = 3\}$$



Příklady deterministických konečných automatů

Příklad: Automat rozpoznávající jazyk L nad abecedou $\{\mathtt{a}, \mathtt{b}\}$ tvořený slovy, která obsahují právě tři výskyty symbolu $\mathtt{b}$, tj.

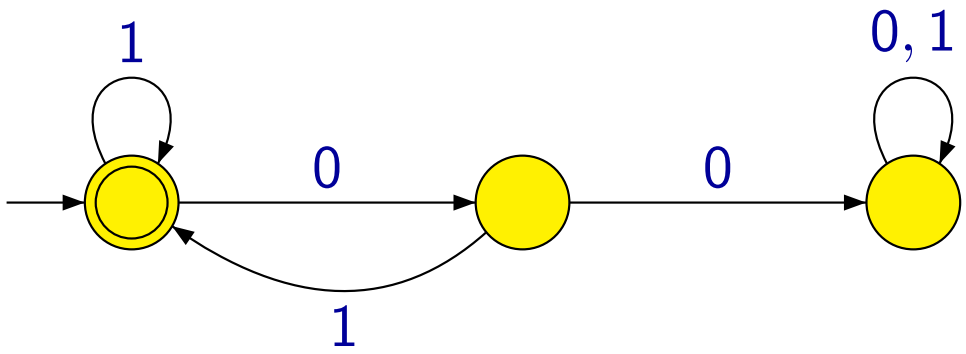
$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid |w|_{\mathtt{b}} = 3\}$$



	a	b
→ 0	0	1
1	1	2
2	2	3
← 3	3	4
4	4	4

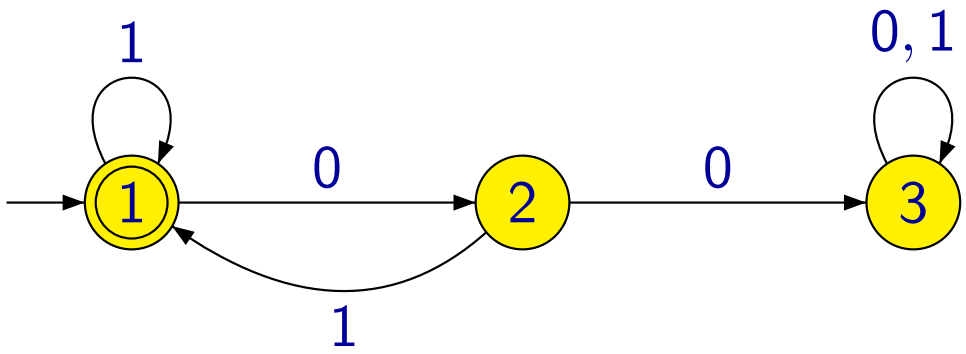
Příklady deterministických konečných automatů

Příklad: Automat rozpoznávající jazyk nad abecedou $\{0, 1\}$ tvořený slovy, kde každý výskyt symbolu 0 je bezprostředně následován symbolem 1.



Příklady deterministických konečných automatů

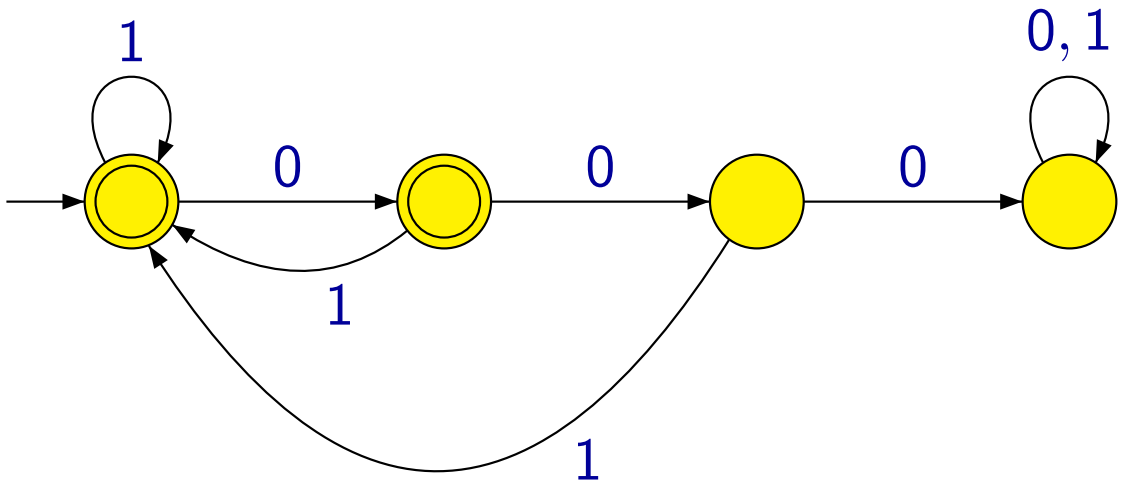
Příklad: Automat rozpoznávající jazyk nad abecedou $\{0, 1\}$ tvořený slovy, kde každý výskyt symbolu 0 je bezprostředně následován symbolem 1.



		0	1
$\leftrightarrow$	1	2	1
	2	3	1
	3	3	3

Příklady deterministických konečných automatů

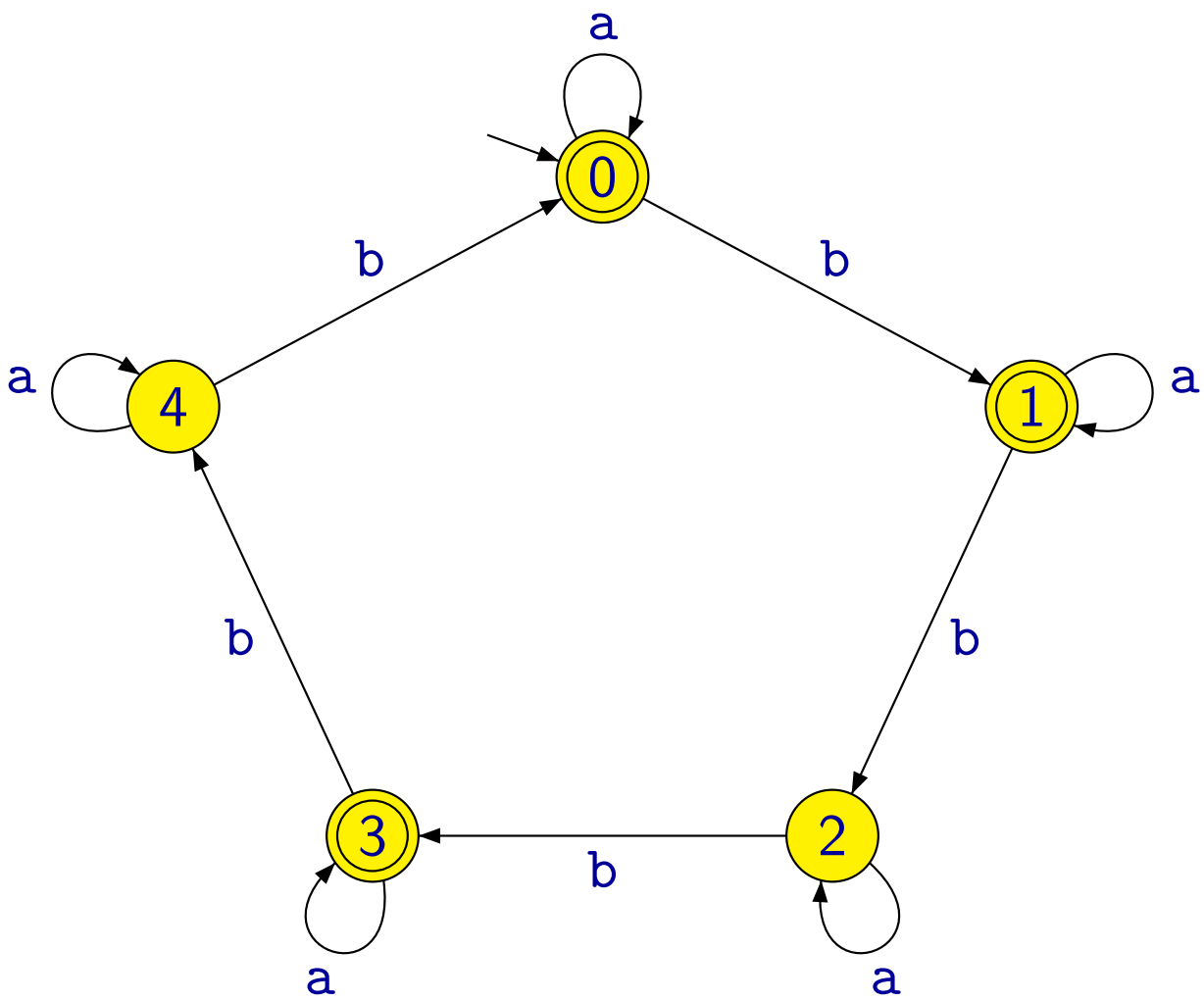
Příklad: Automat rozpoznávající jazyk nad abecedou $\{0, 1\}$ tvořený slovy, kde každá dvojice symbolů 0 je bezprostředně následována symbolem 1.



Příklady deterministických konečných automatů

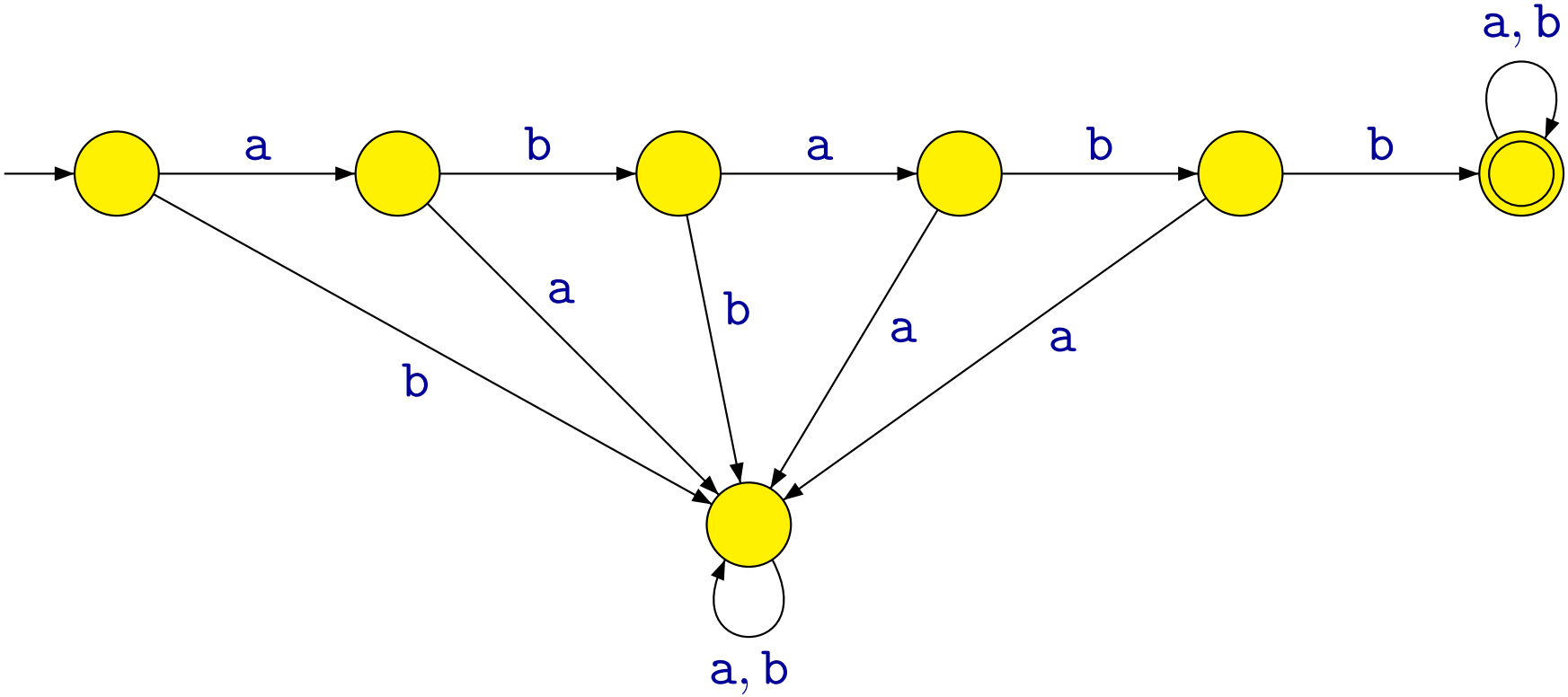
Příklad: Automat rozpoznávající jazyk

$$L = \{w \in \{a, b\}^* \mid (|w|_b \bmod 5) \in \{0, 1, 3\}\}$$



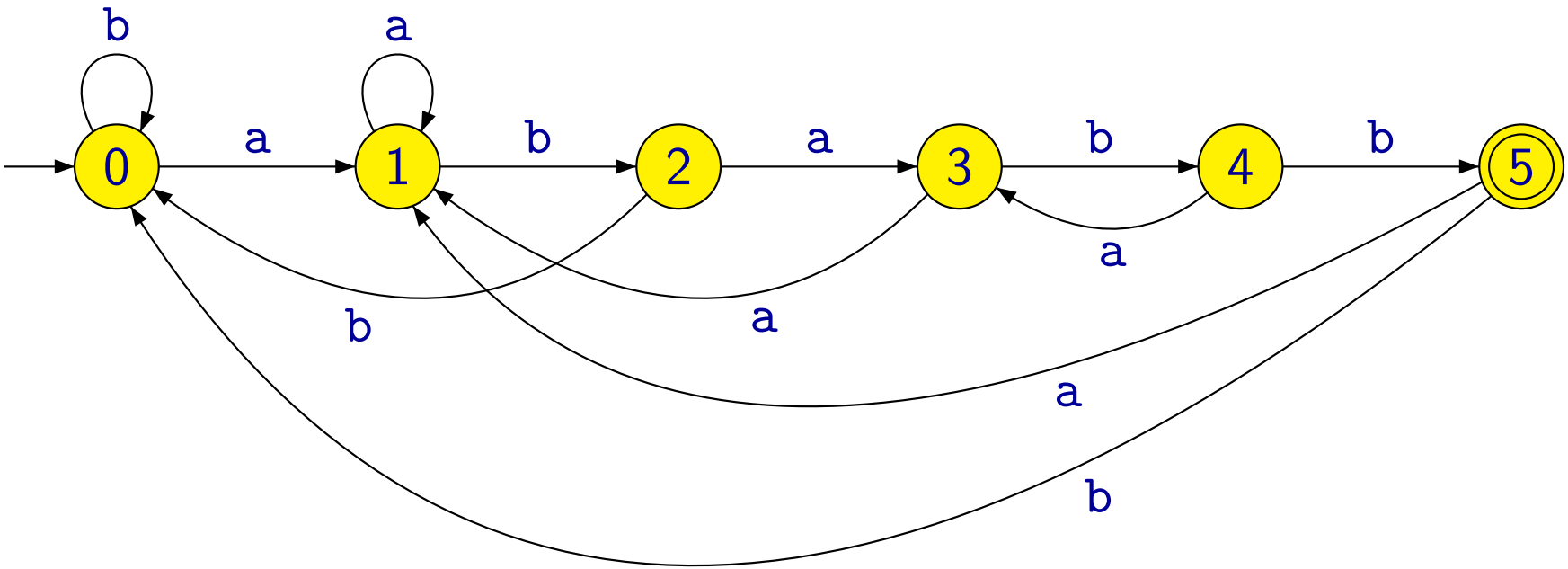
Příklady deterministických konečných automatů

Příklad: Automat rozpoznávající jazyk nad abecedou $\{a, b\}$ tvořený slovy, která začínají **prefixem** $ababb$.



Příklady deterministických konečných automatů

Příklad: Automat rozpoznávající jazyk nad abecedou $\{a, b\}$ tvořený slovy, která končí **sufixem** $abaabb$.



Příklady deterministických konečných automatů

Konstrukce tohoto automatu je založena na následující myšlence:

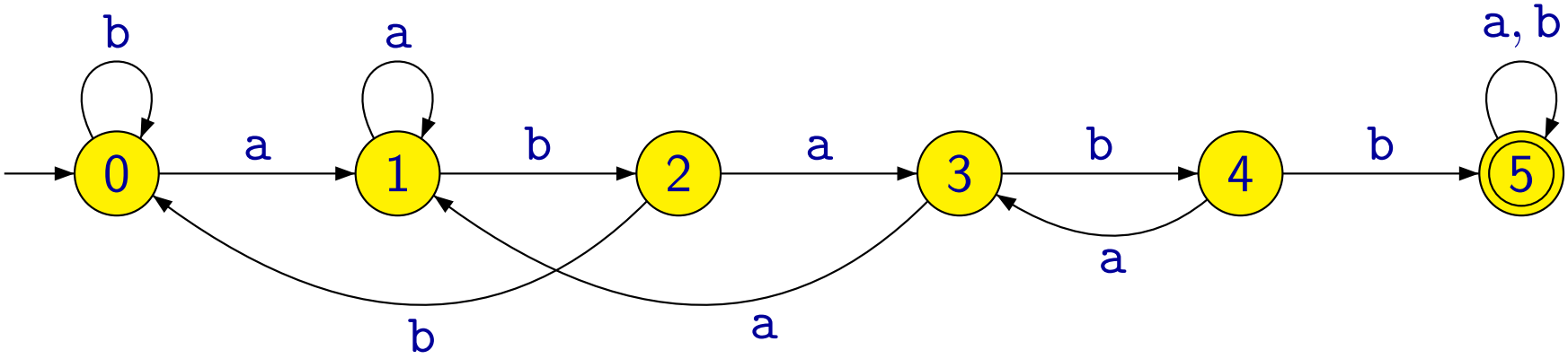
- Předpokládejme, že chceme vyhledávat slovo u délky n (tj. $|u| = n$). Stavy automatu jsou označeny čísly $0, 1, \ldots, n$.
- Stav s číslem i odpovídá situaci, kdy i je délka nejdelšího slova, které je zároveň:
  - prefixem hledaného vzorku u
  - sufixem té části vstupního slova, kterou automat zatím přečetl

Například pro slovo `ababb` stavy automatu odpovídají následujícím slovům:

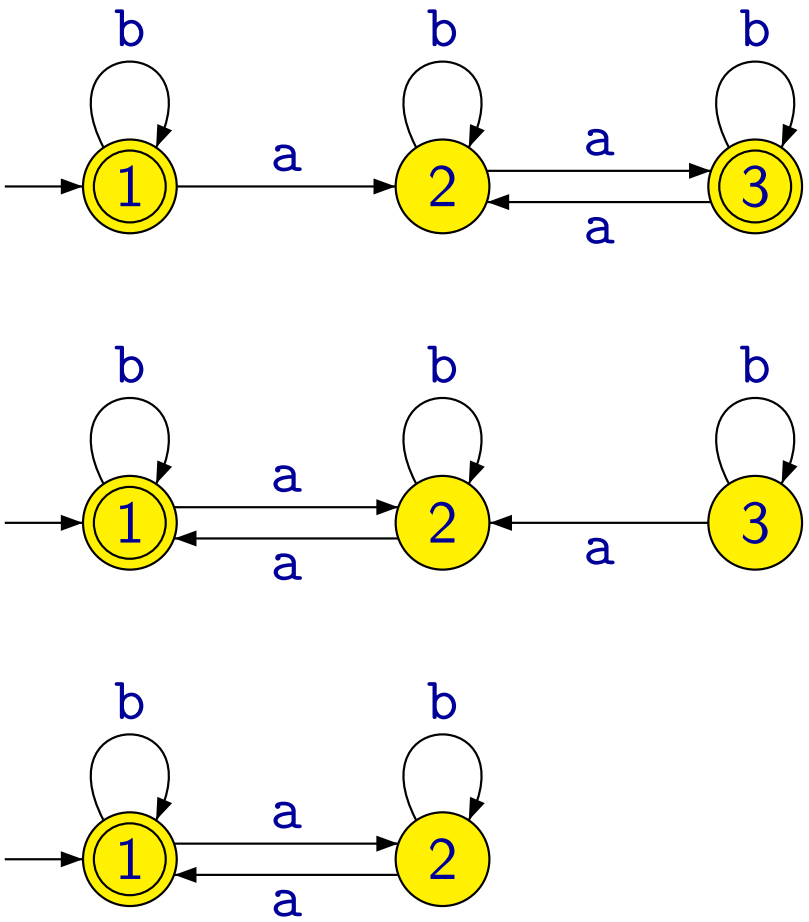
- Stav 0 ... ε
- Stav 1 ... `a`
- Stav 2 ... `ab`
- Stav 3 ... `aba`
- Stav 4 ... `abab`
- Stav 5 ... `ababb`

Příklady deterministických konečných automatů

Příklad: Automat rozpoznávající jazyk nad abecedou $\{a, b\}$ tvořený slovy, která obsahují **podslovo** $ababb$.



Ekvivalence automatů



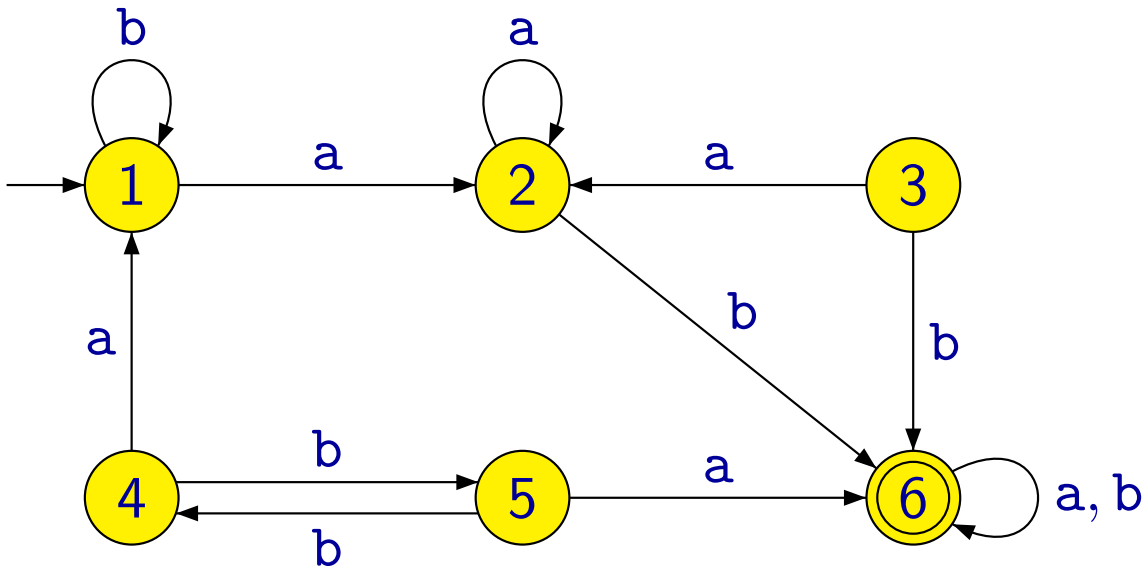
Všechny tři automaty přijímají jazyk všech slov se sudým počtem a.

Ekvivalence automatů

Definice

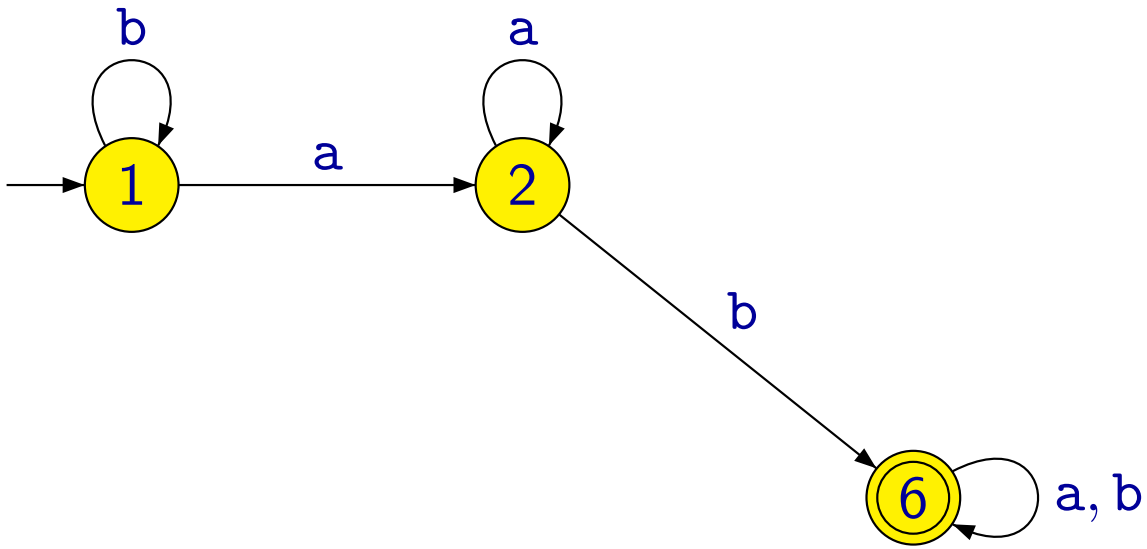
O konečných automatech $\mathcal{A}_1$, $\mathcal{A}_2$ řekneme, že jsou **ekvivalentní**, jestliže $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$.

Nedosažitelné stavy automatu



- Automat přijímá jazyk $L = \{w \in \{\mathtt{a},\mathtt{b}\}^* \mid w \text{ obsahuje podslovo } \mathtt{ab}\}$
- Pro žádnou posloupnost vstupních symbolů se automat nedostane do stavů 3, 4 nebo 5.

Nedosažitelné stavy automatu



- Automat přijímá jazyk $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ obsahuje podslovo } \mathtt{ab}\}$
- Pro žádnou posloupnost vstupních symbolů se automat nedostane do stavů 3, 4 nebo 5.
- Pokud tyto stavy odstraníme, pořád automat přijímá stejný jazyk L.

Nedosažitelné stavy automatu

Definice

Stav q konečného automatu $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ je **dosažitelný** pokud existuje nějaké slovo w takové, že $q_0 \xrightarrow{w} q$.

V opačném případě stav nazýváme **nedosažitelný**.

- Do nedosažitelných stavů nevede v grafu automatu žádná orientovaná cesta z počátečního stavu.
- Nedosažitelné stavy můžeme z automatu odstranit (spolu se všemi přechody vedoucími do nich a z nich). Jazyk přijímaný automatem se nezmění.

Automaty a operace na jazycích

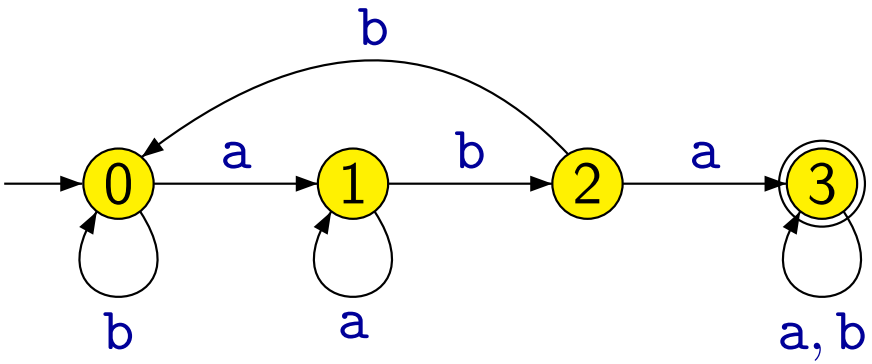
Při konstrukci automatů může být obtížné přímo zkonstruovat automat pro daný jazyk L.

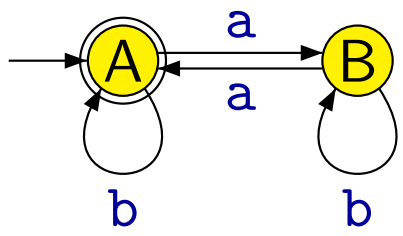
Pokud je možné jazyk L popsat jako výsledek nějakých jazykových operací (průnik, sjednocení, doplněk, zřetězení, iterace, ...) aplikovaných na nějaké jednodušší jazyky L_1 a L_2, může být výhodné postupovat modulárním způsobem:

- Nejprve zkonstruovat automaty pro jazyky L_1 a L_2.
- Poté použít některou z obecných konstrukcí, které umožňují k daným automatům rozpoznávajícím jazyky L_1 a L_2 algoritmicky zkonstruovat automat pro jazyk L, který je výsledkem aplikace dané jazykové operace na jazyky L_1 a L_2.

Automat pro průnik jazyků

Máme následující dva automaty:

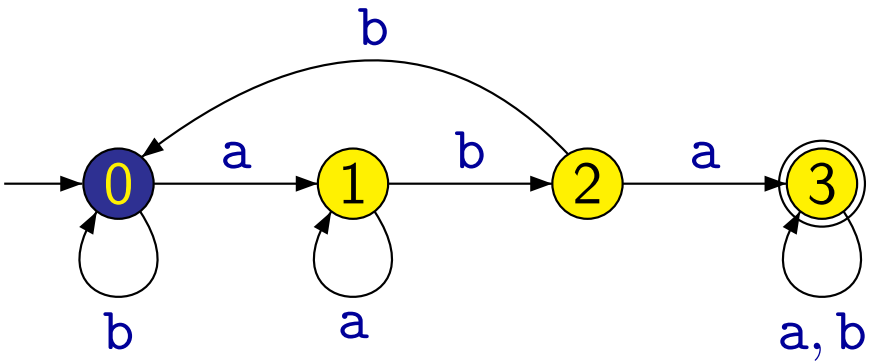


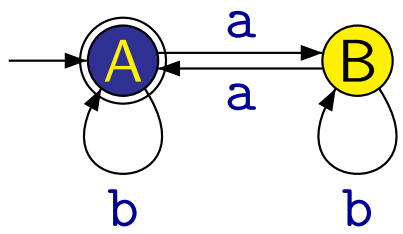


Příjmou oba slovo abbaaba?

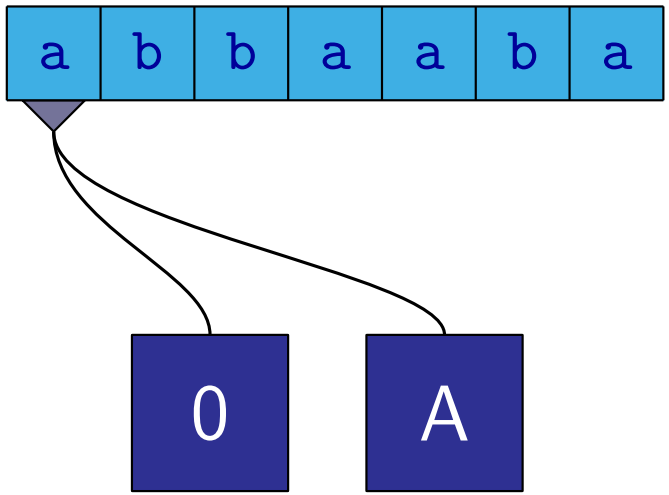
Automat pro průnik jazyků

Máme následující dva automaty:



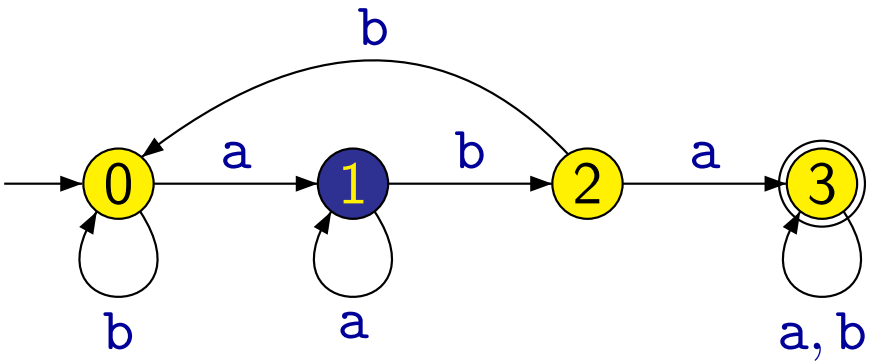


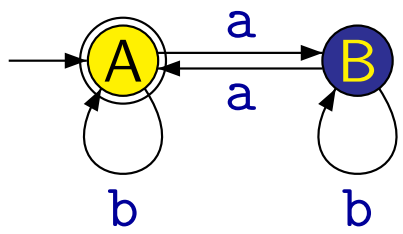
Příjmou oba slovo abbaaba?



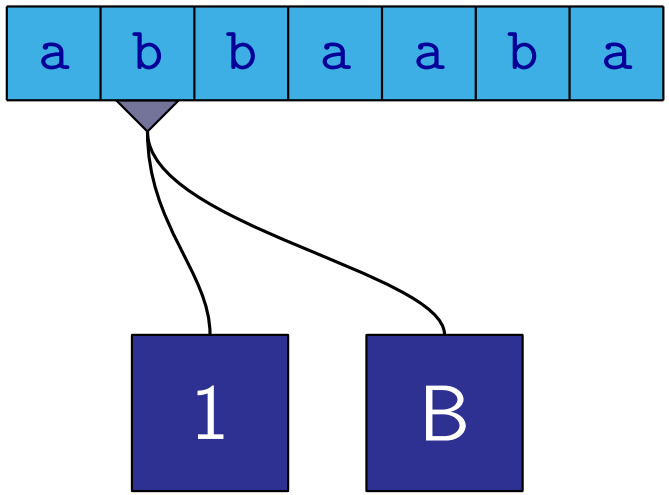
Automat pro průnik jazyků

Máme následující dva automaty:



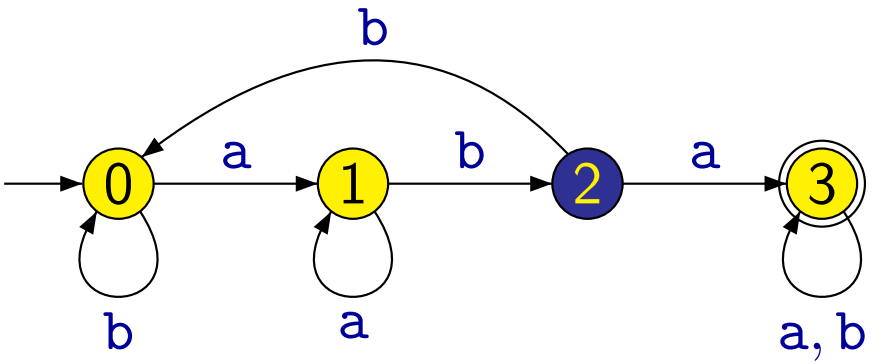


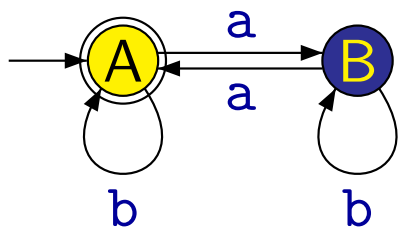
Příjmou oba slovo abbaaba?



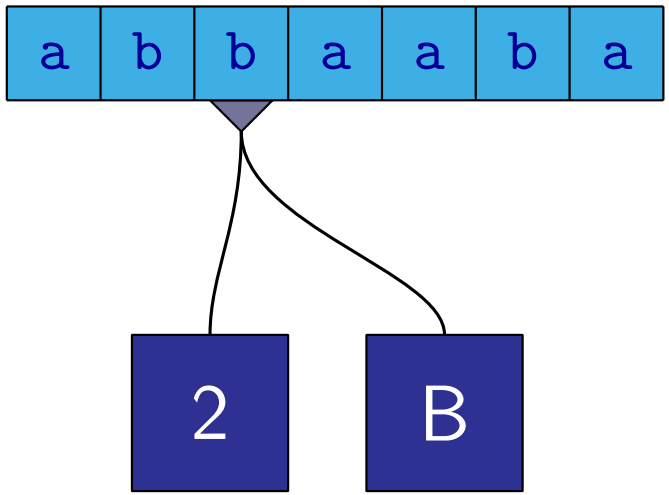
Automat pro průnik jazyků

Máme následující dva automaty:



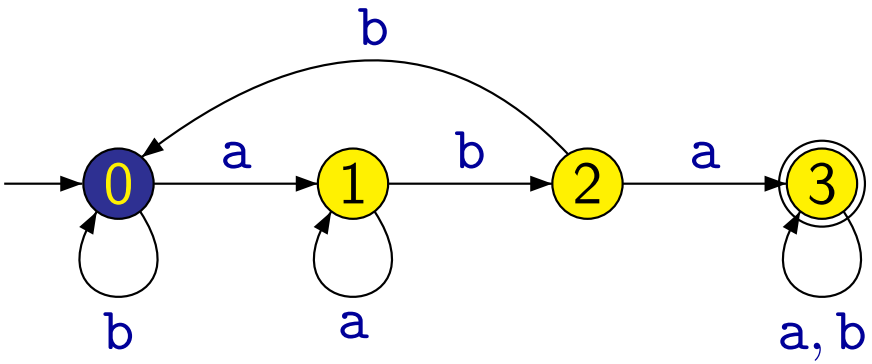


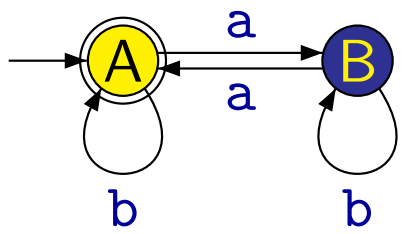
Příjmou oba slovo abbaaba?



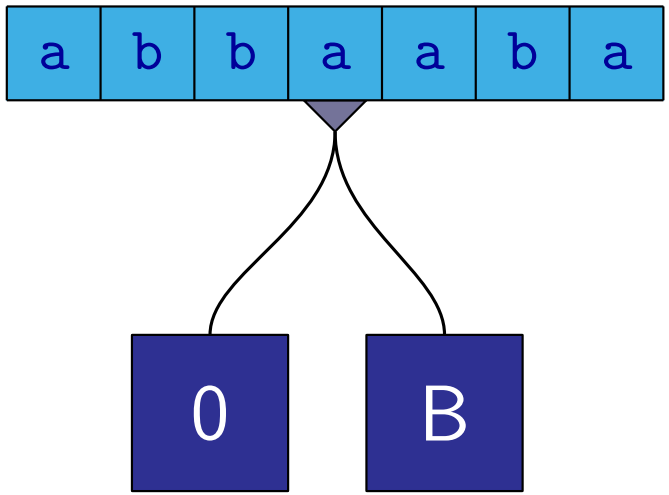
Automat pro průnik jazyků

Máme následující dva automaty:



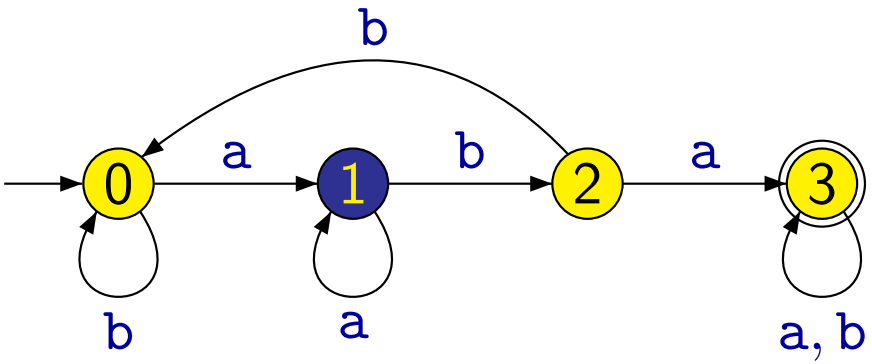


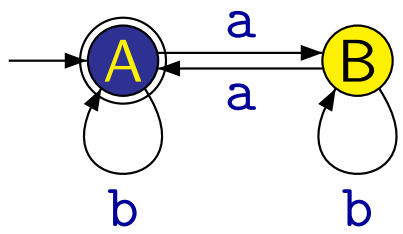
Příjmou oba slovo abbaaba?



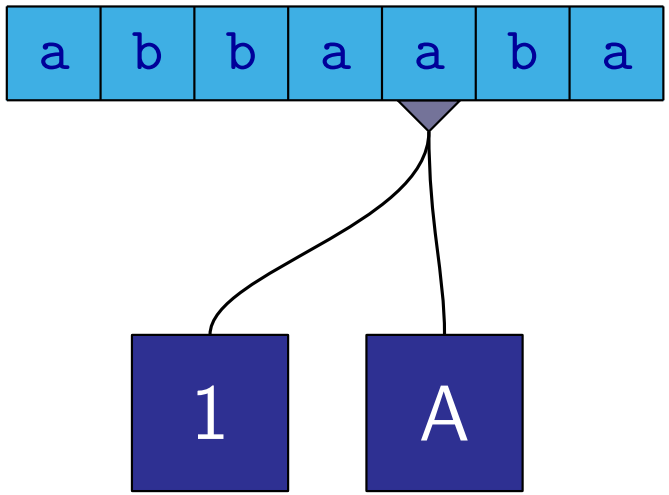
Automat pro průnik jazyků

Máme následující dva automaty:



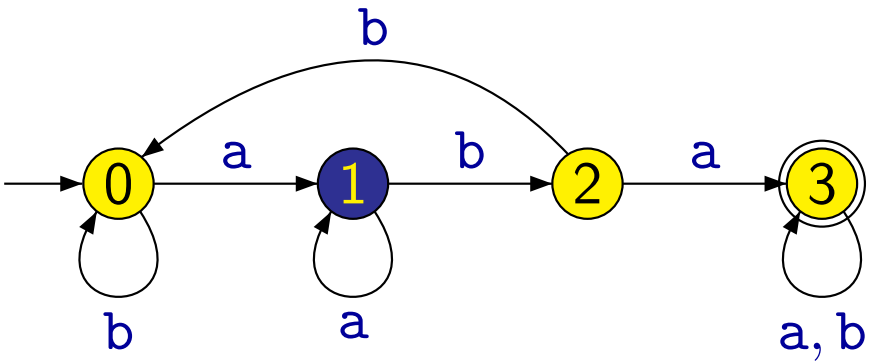


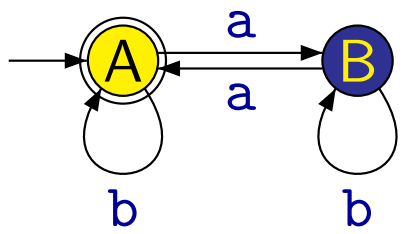
Příjmou oba slovo abbaaba?



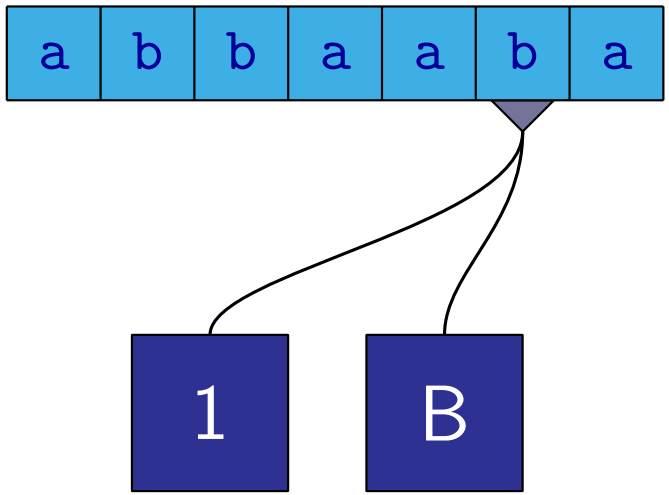
Automat pro průnik jazyků

Máme následující dva automaty:



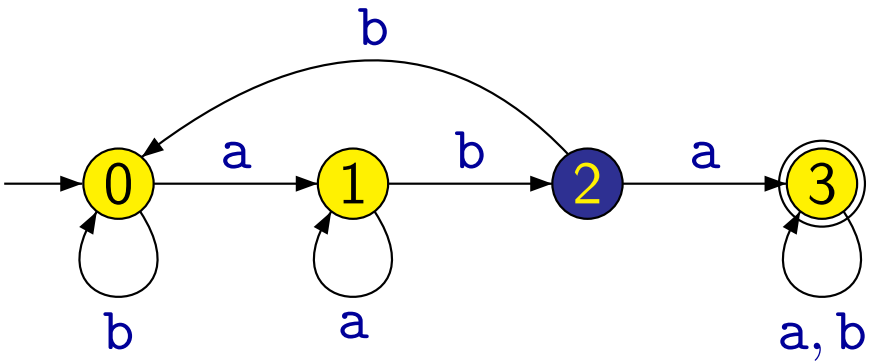


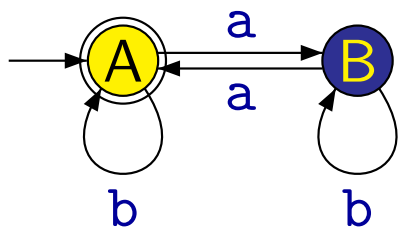
Příjmou oba slovo abbaaba?



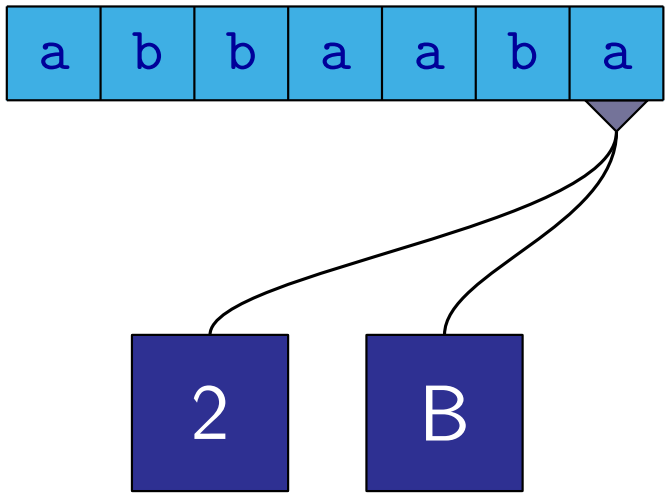
Automat pro průnik jazyků

Máme následující dva automaty:



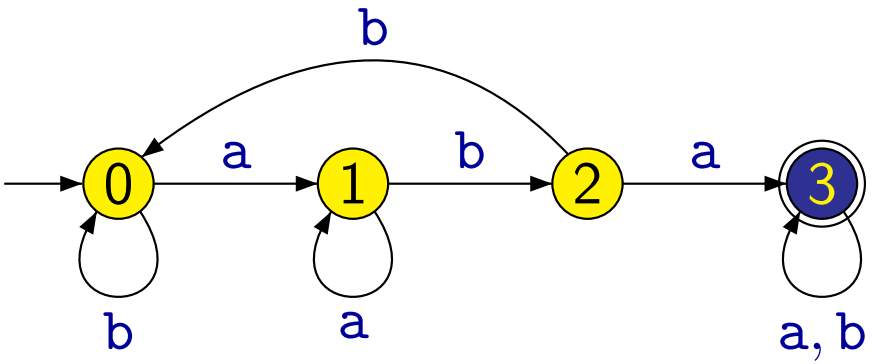


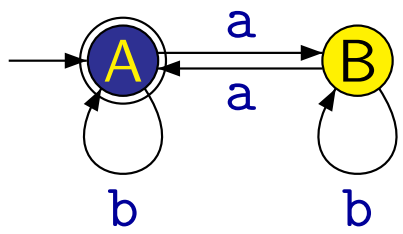
Příjmou oba slovo abbaaba?



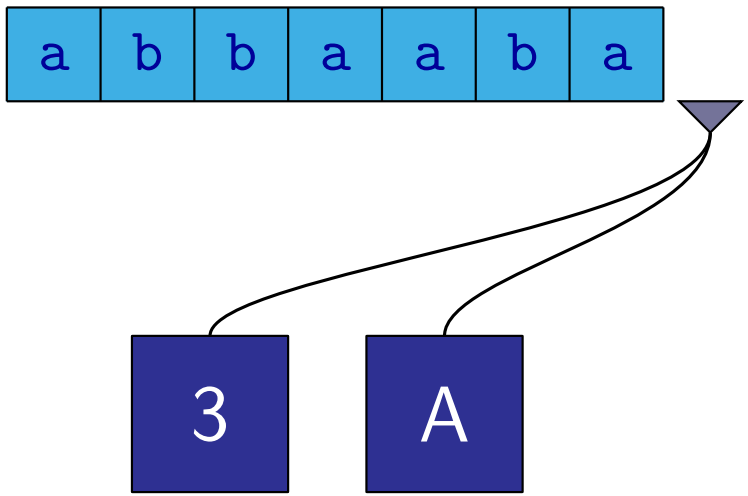
Automat pro průnik jazyků

Máme následující dva automaty:

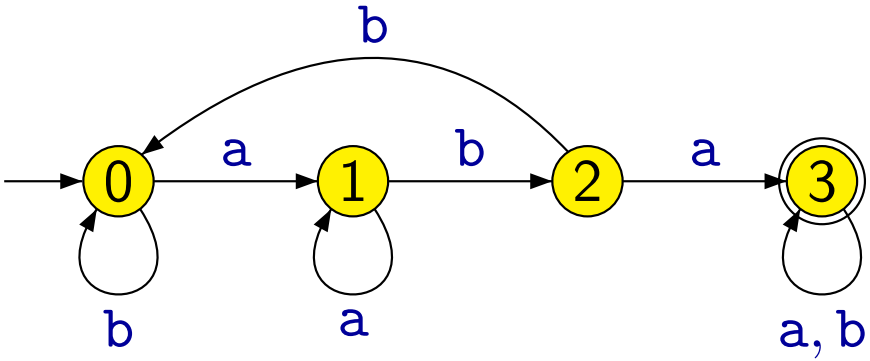


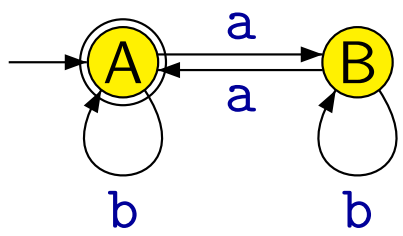


Příjmou oba slovo abbaaba?

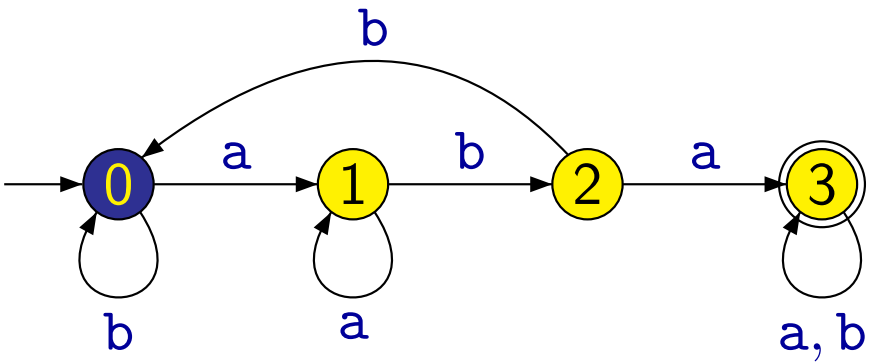


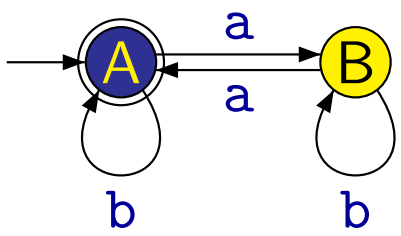
Automat pro průnik jazyků

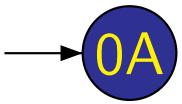




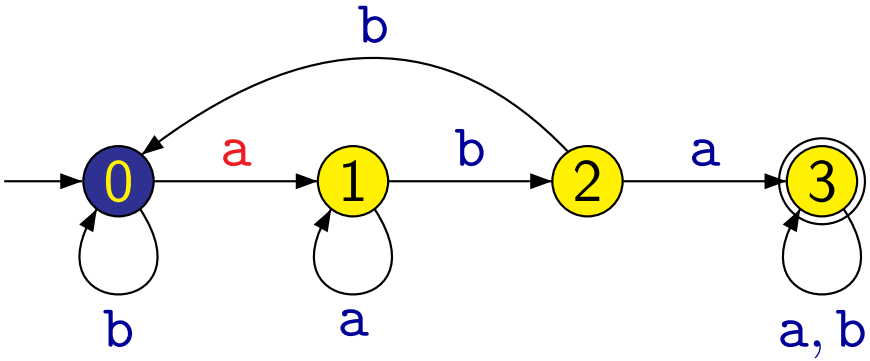
Automat pro průnik jazyků

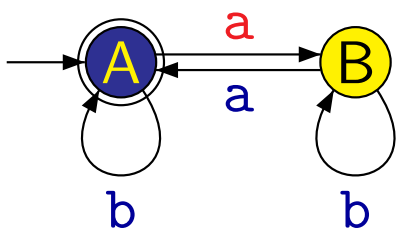


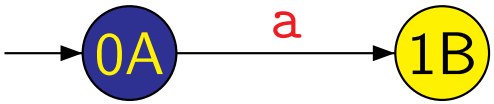




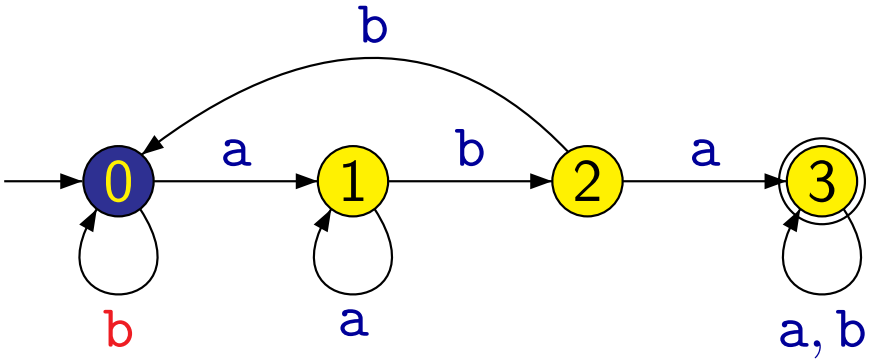
Automat pro průnik jazyků

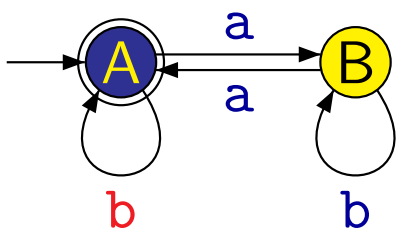


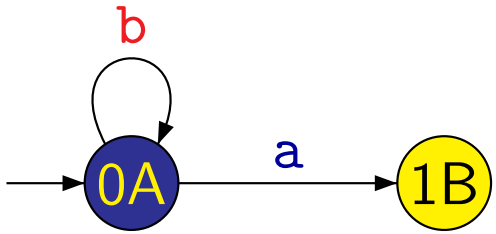




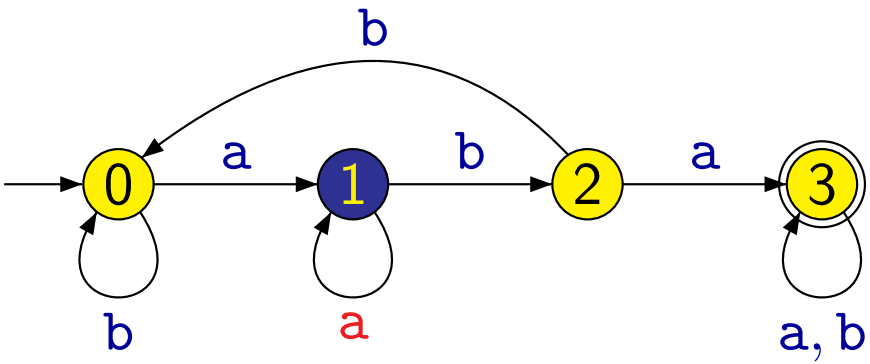
Automat pro průnik jazyků

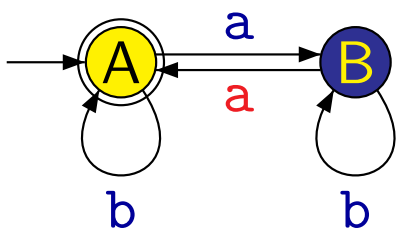


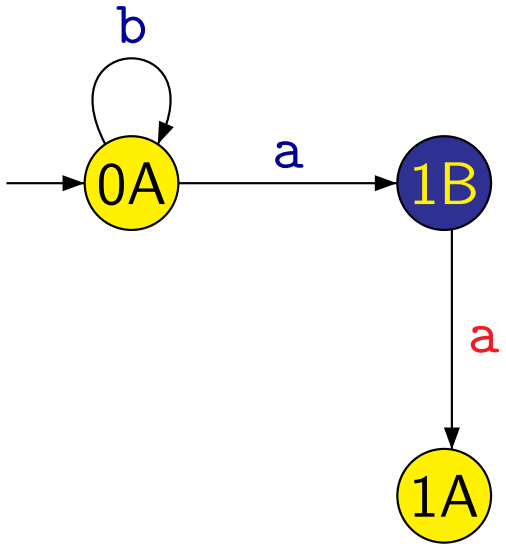




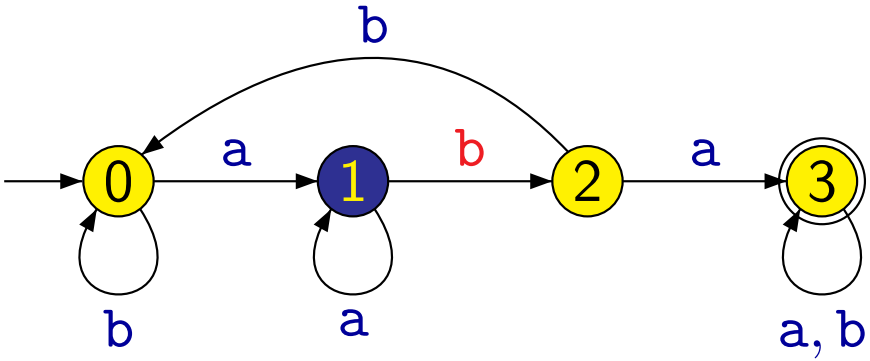
Automat pro průnik jazyků

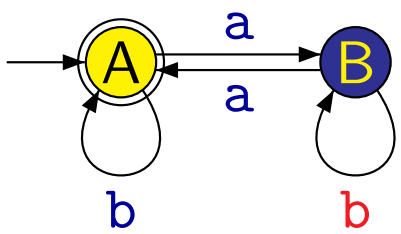


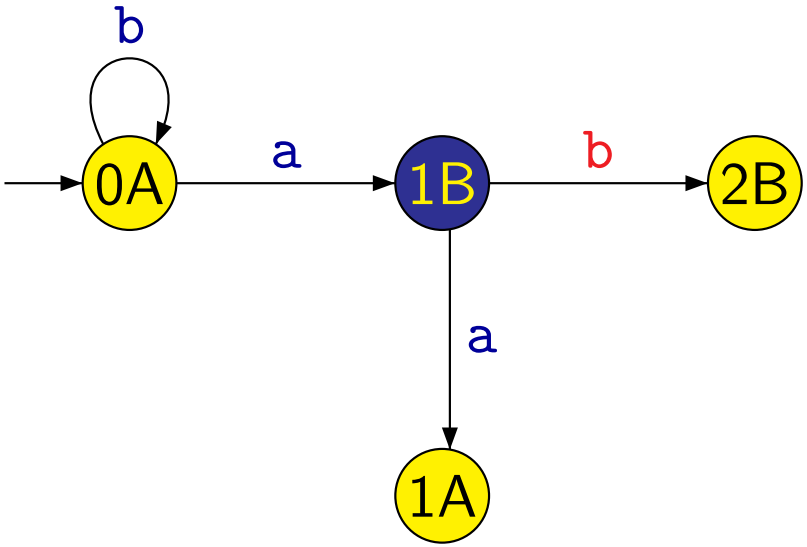




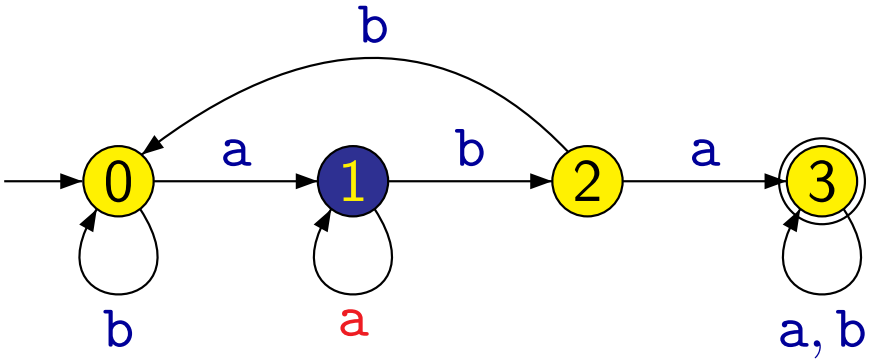
Automat pro průnik jazyků

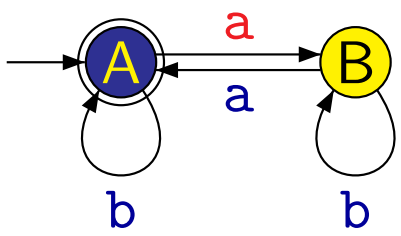


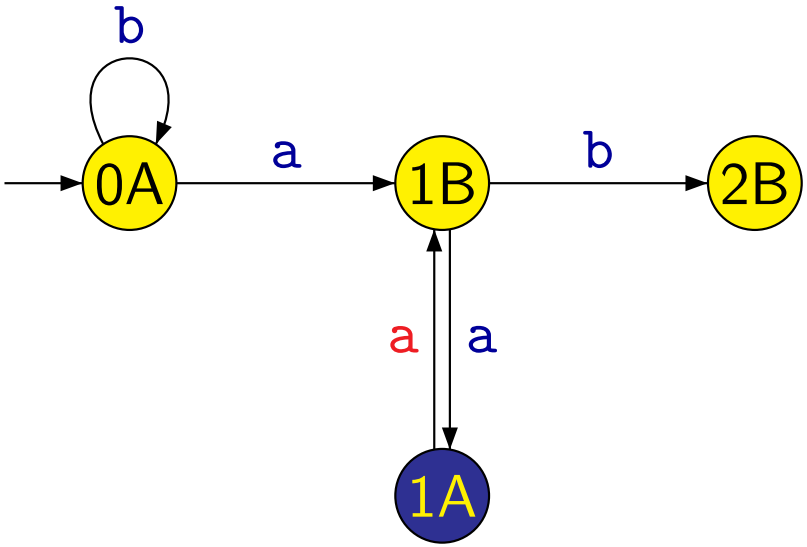




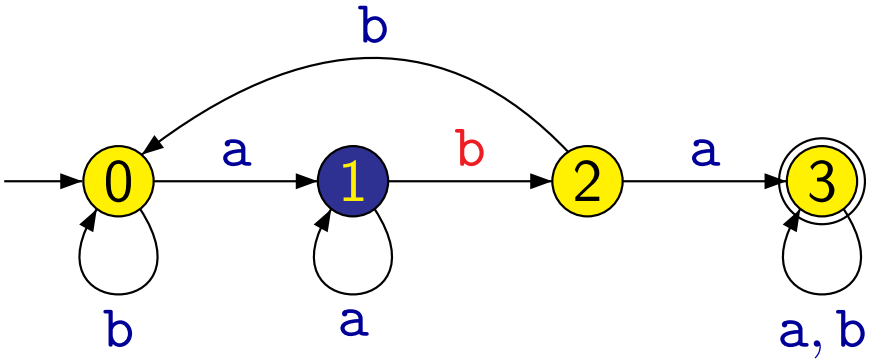
Automat pro průnik jazyků

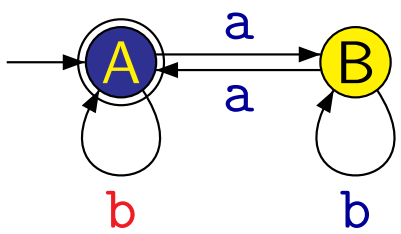


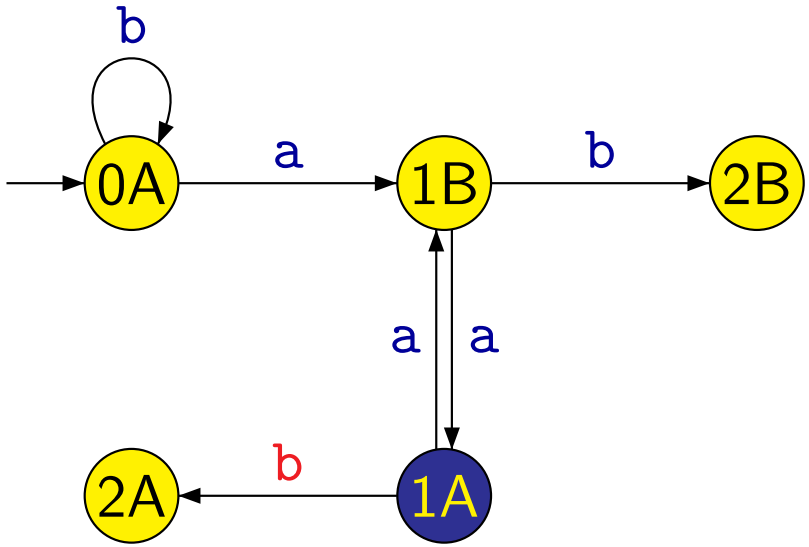




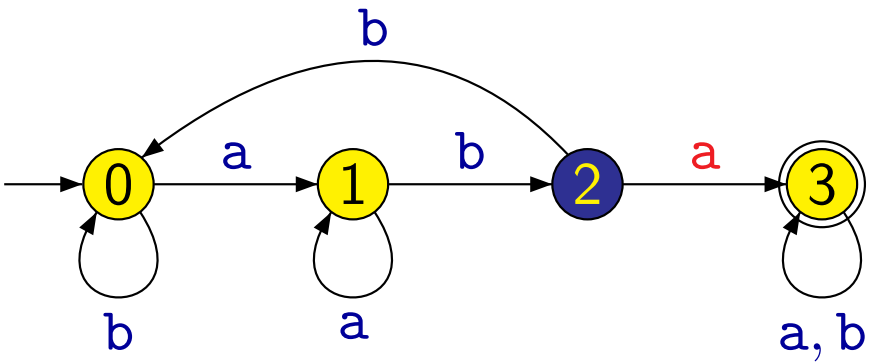
Automat pro průnik jazyků

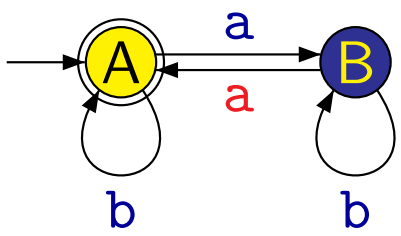


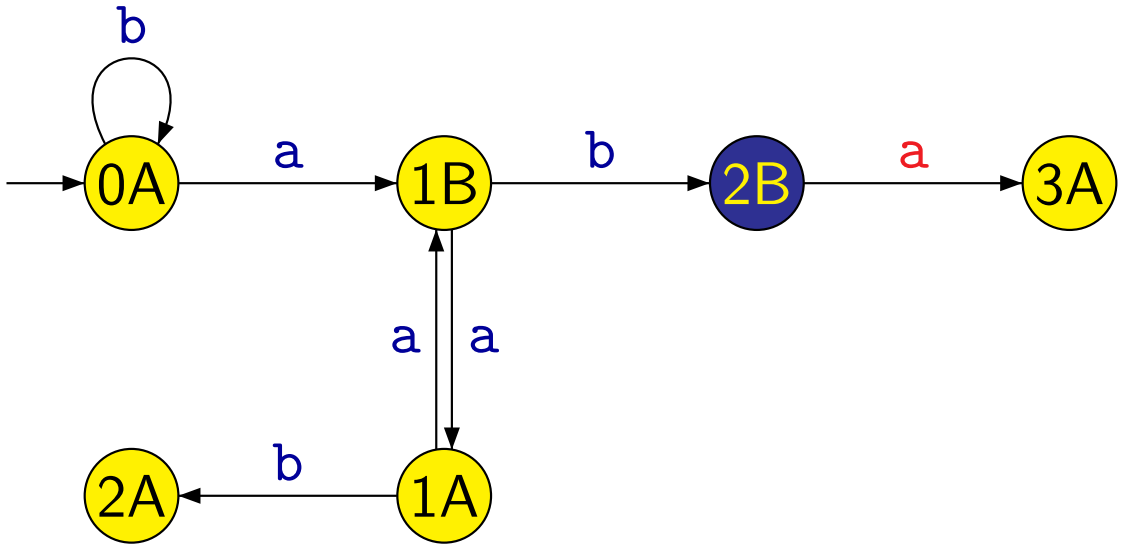




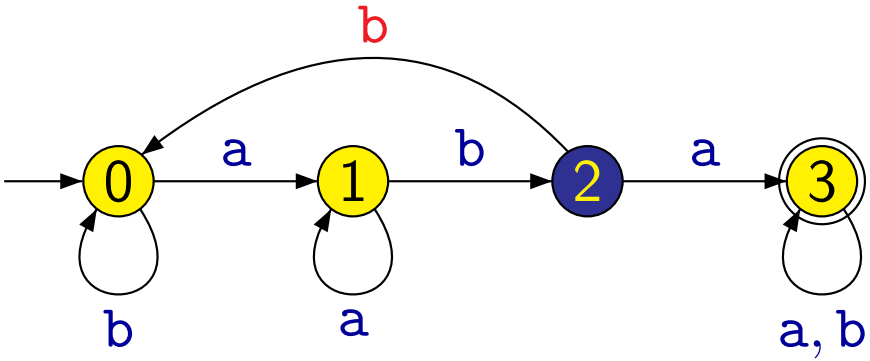
Automat pro průnik jazyků

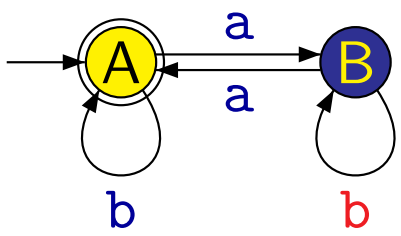


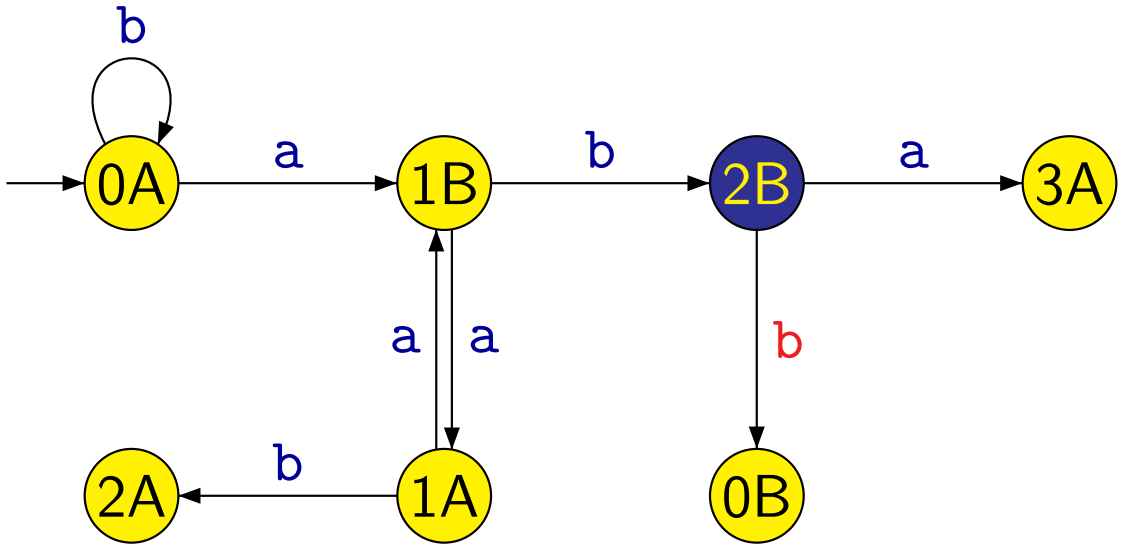




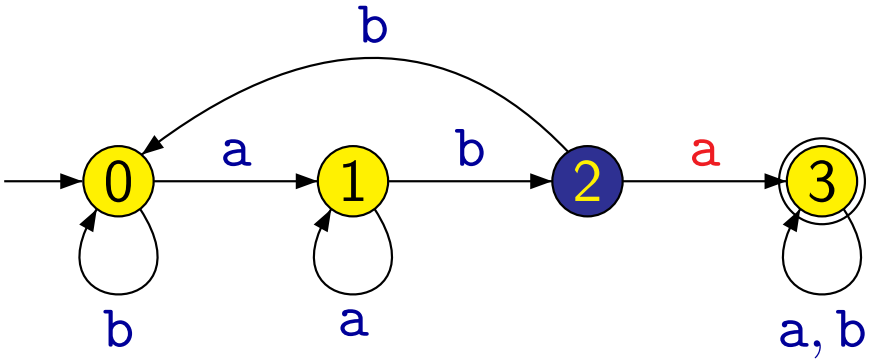
Automat pro průnik jazyků

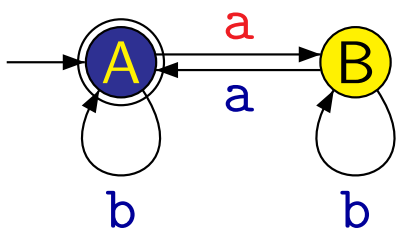


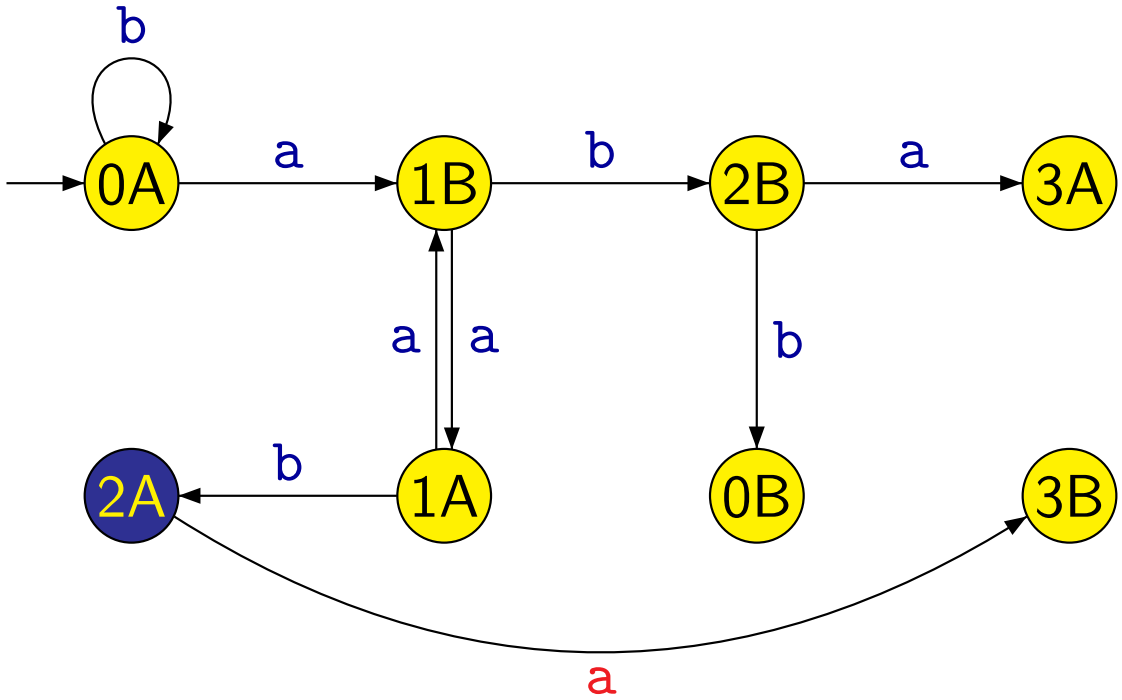




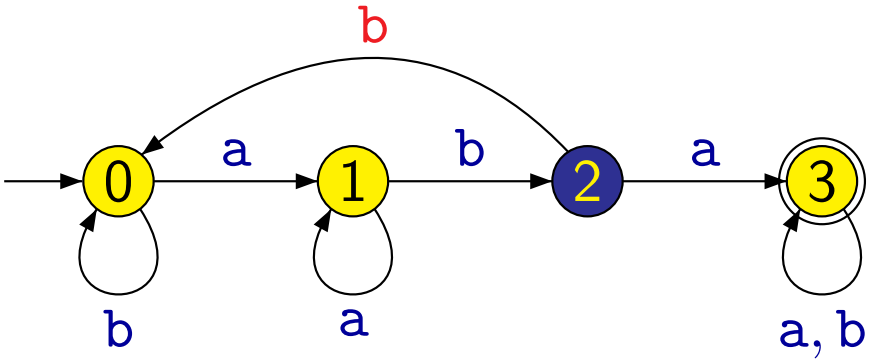
Automat pro průnik jazyků

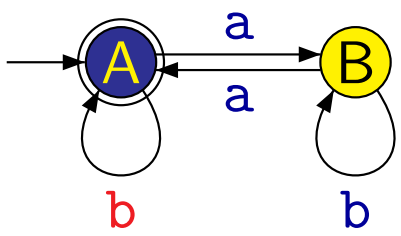


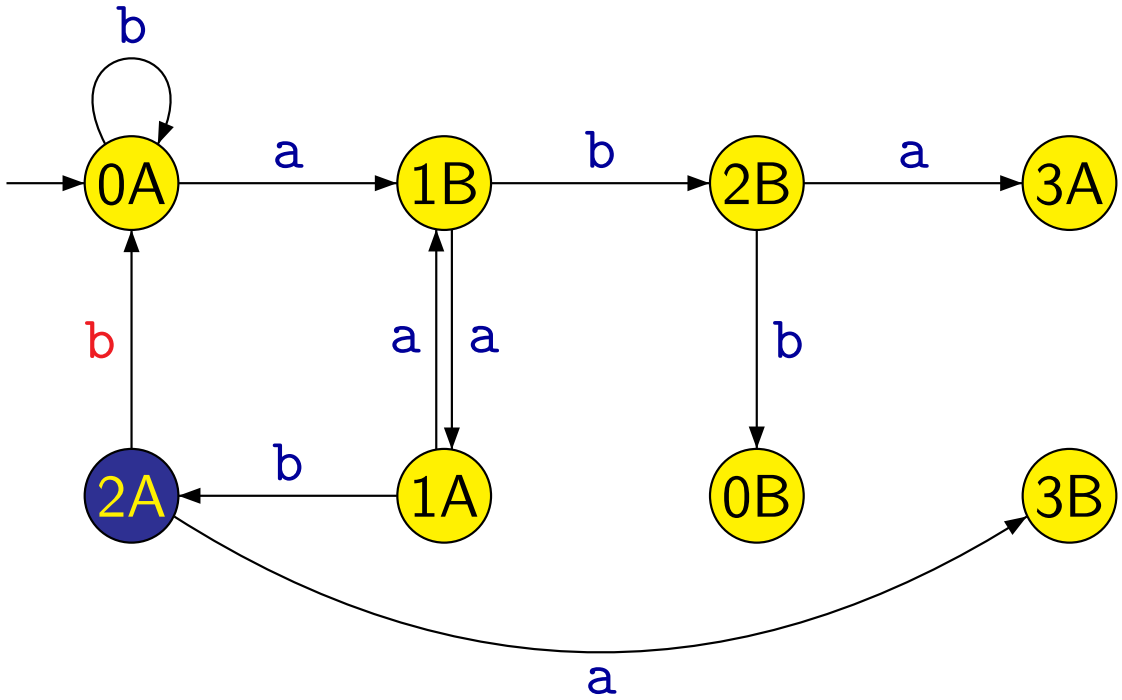




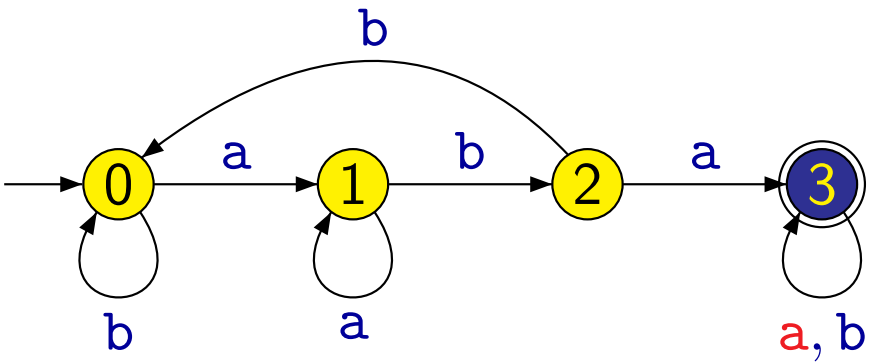
Automat pro průnik jazyků

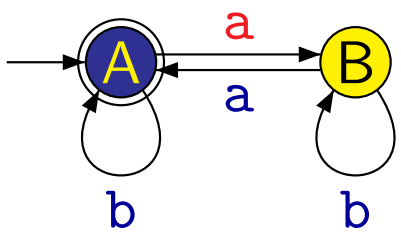


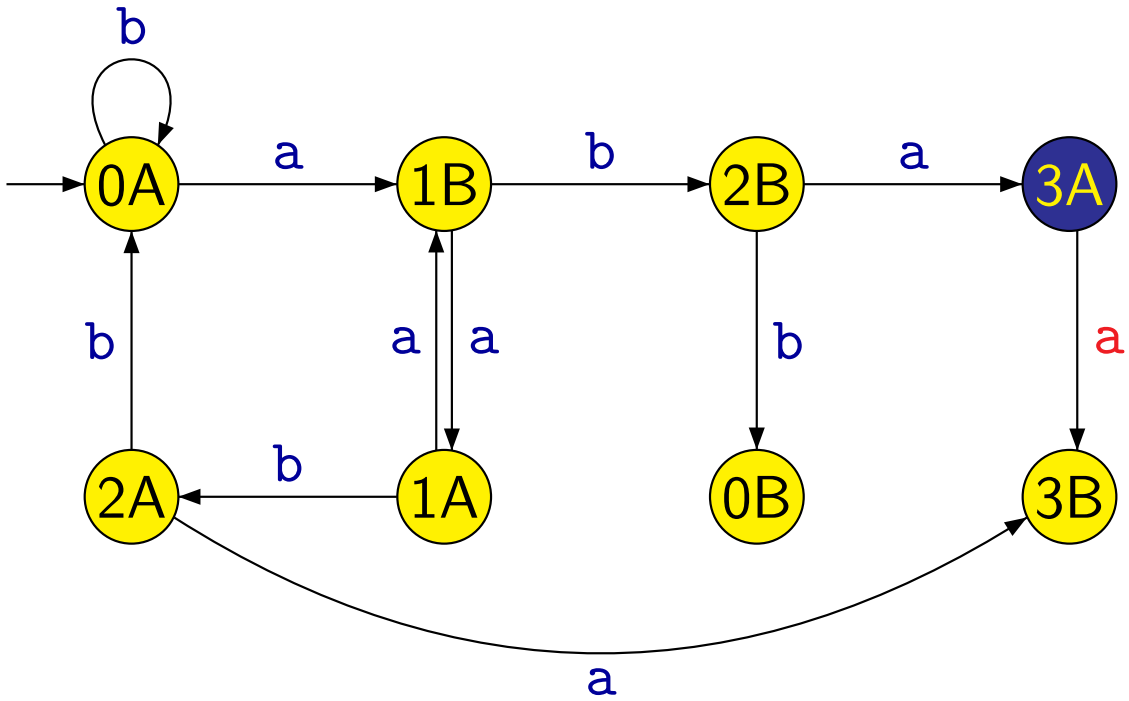




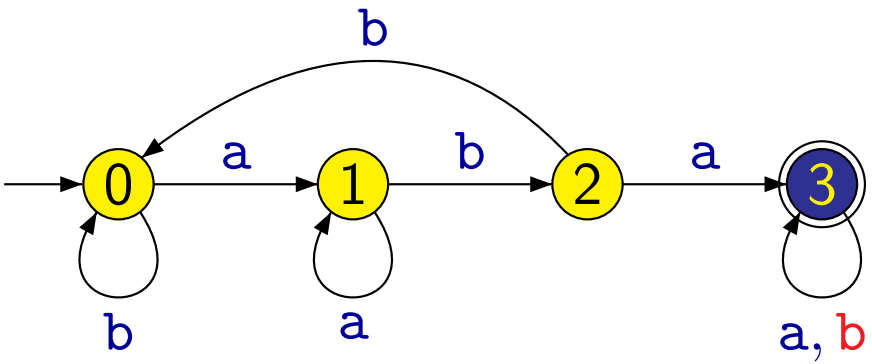
Automat pro průnik jazyků

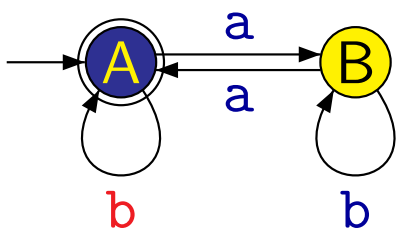


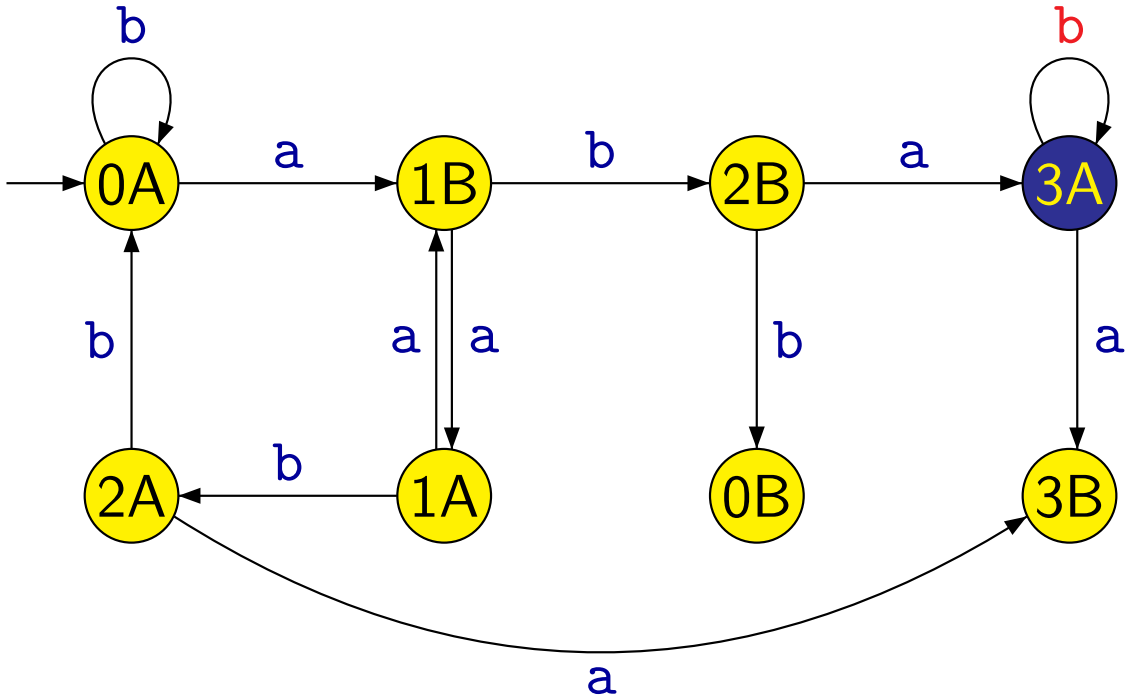




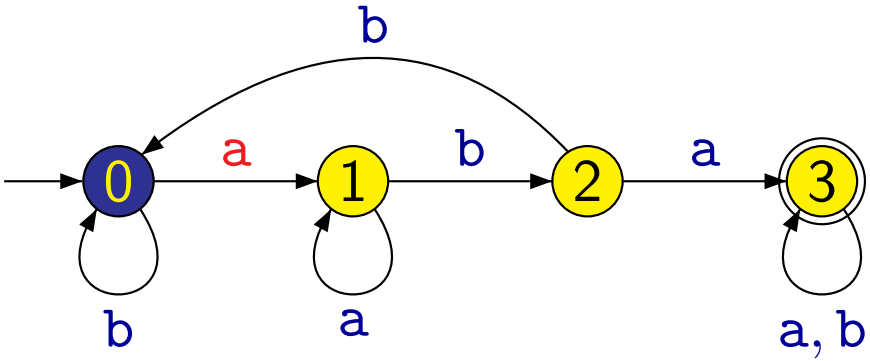
Automat pro průnik jazyků

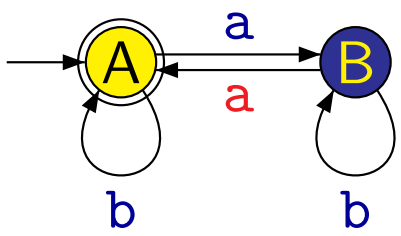


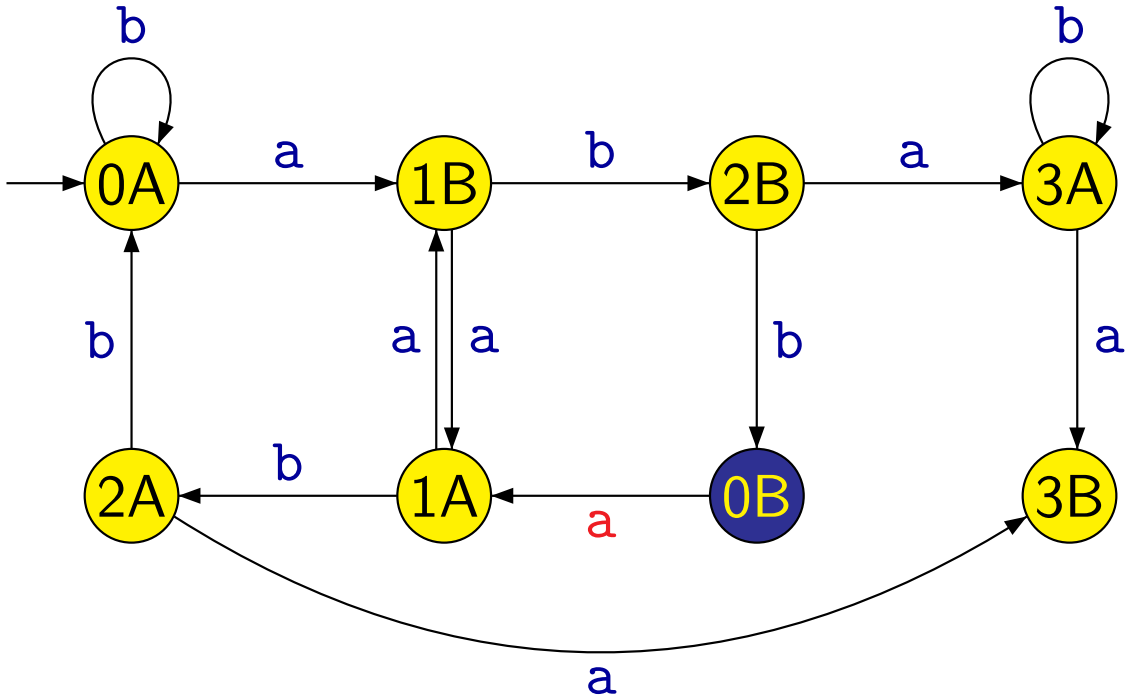




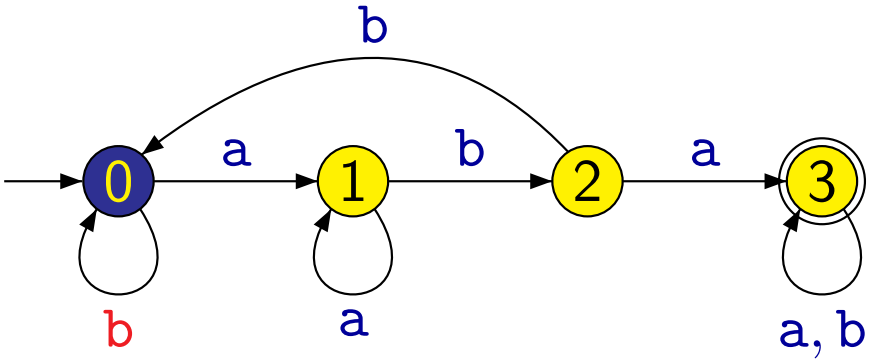
Automat pro průnik jazyků

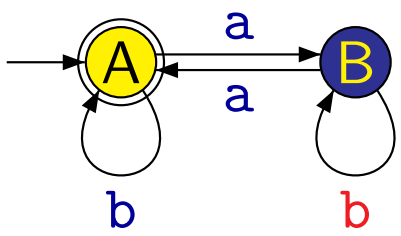


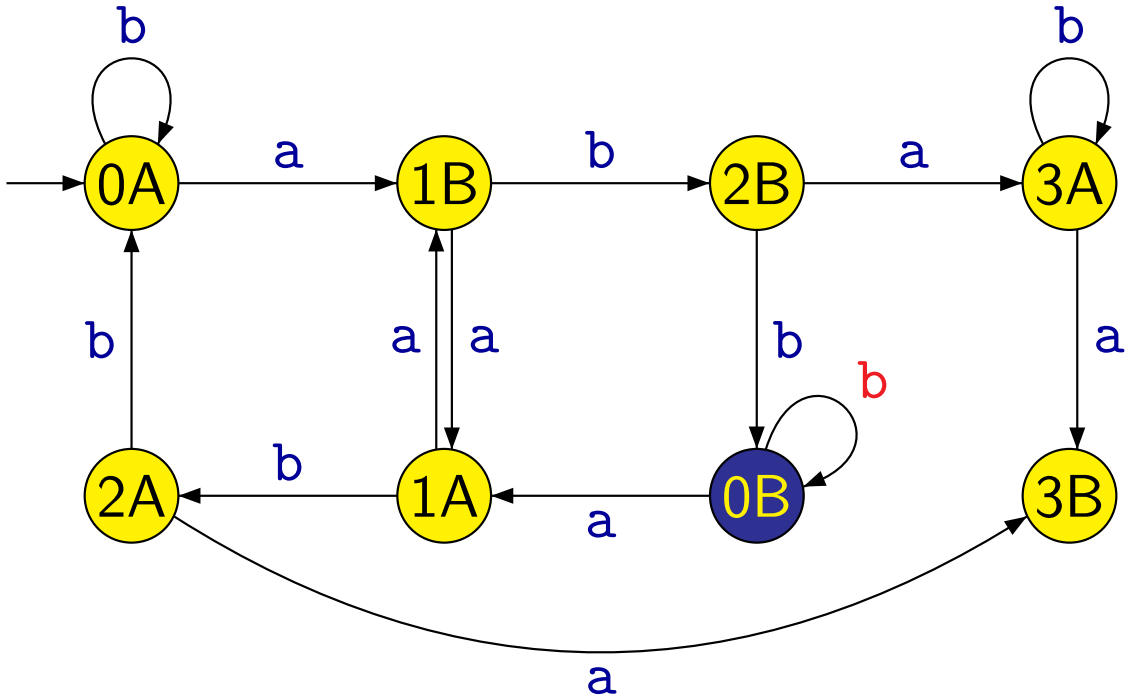




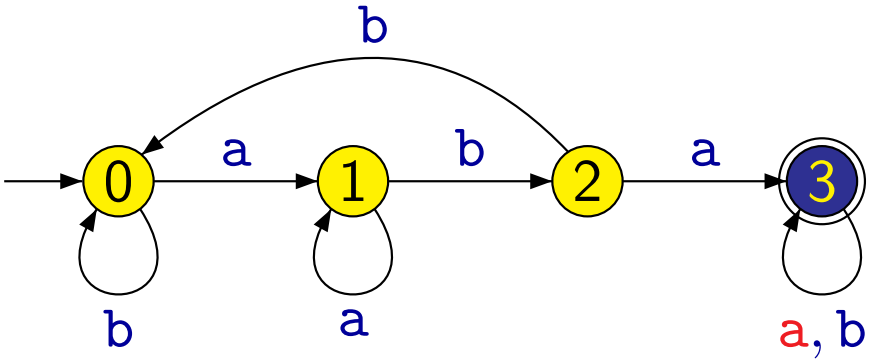
Automat pro průnik jazyků

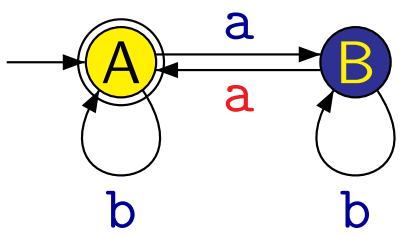


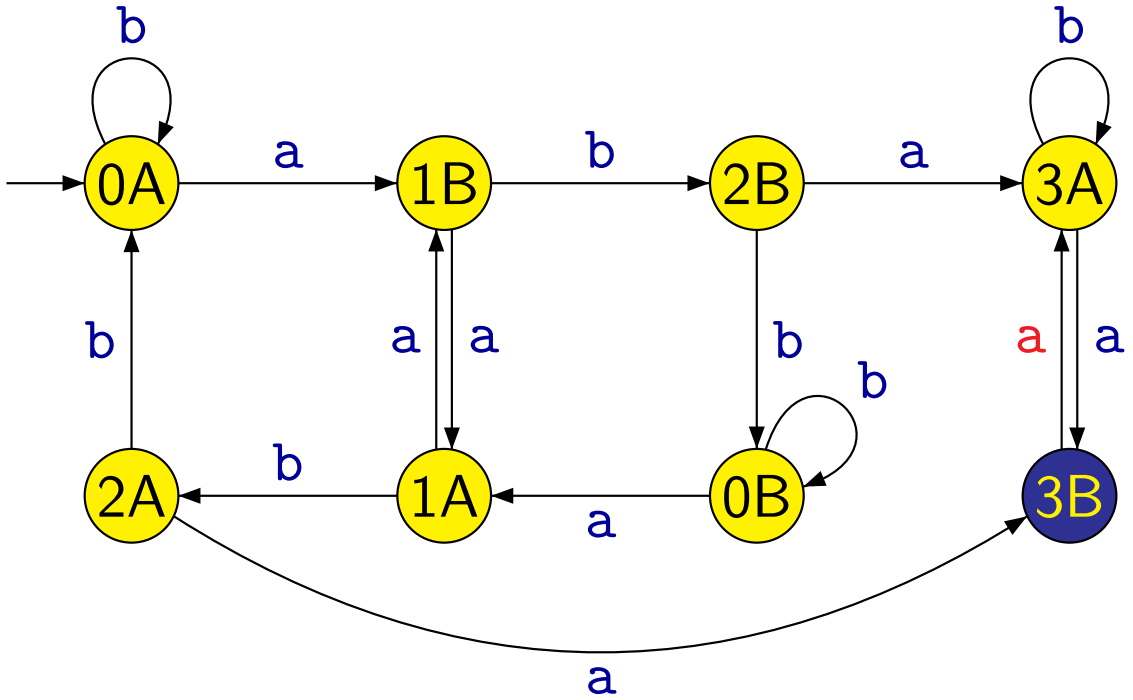




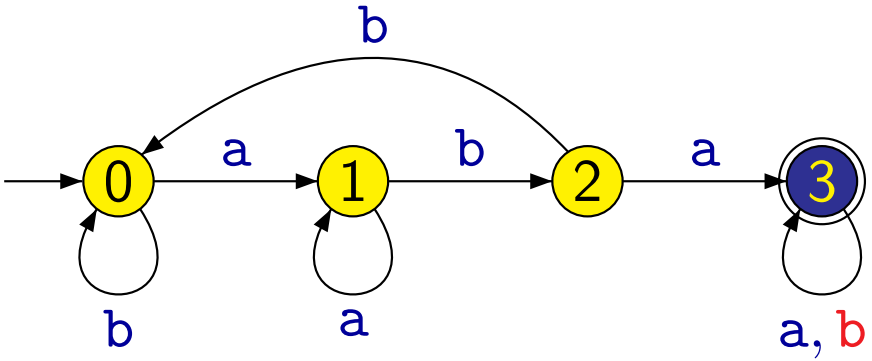
Automat pro průnik jazyků

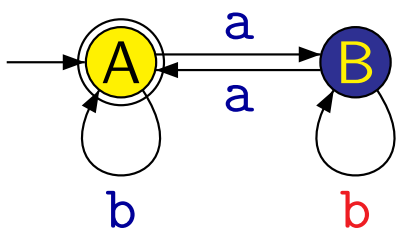


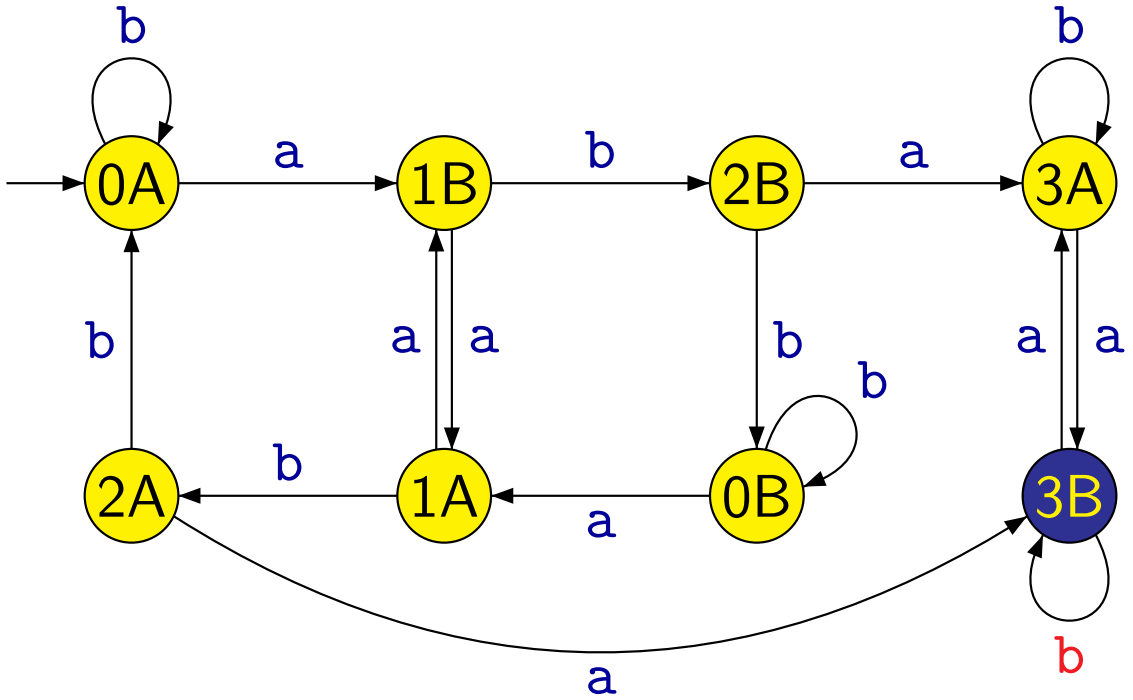




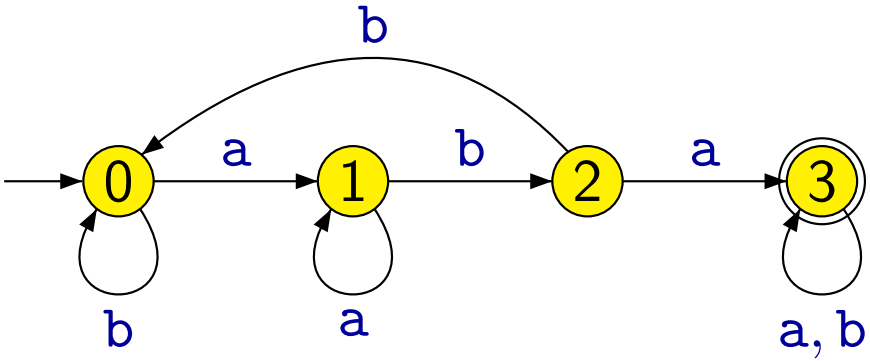
Automat pro průnik jazyků

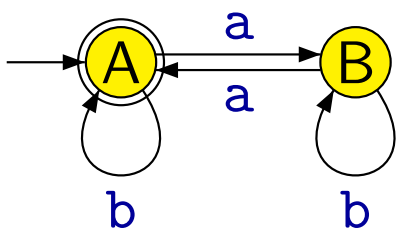


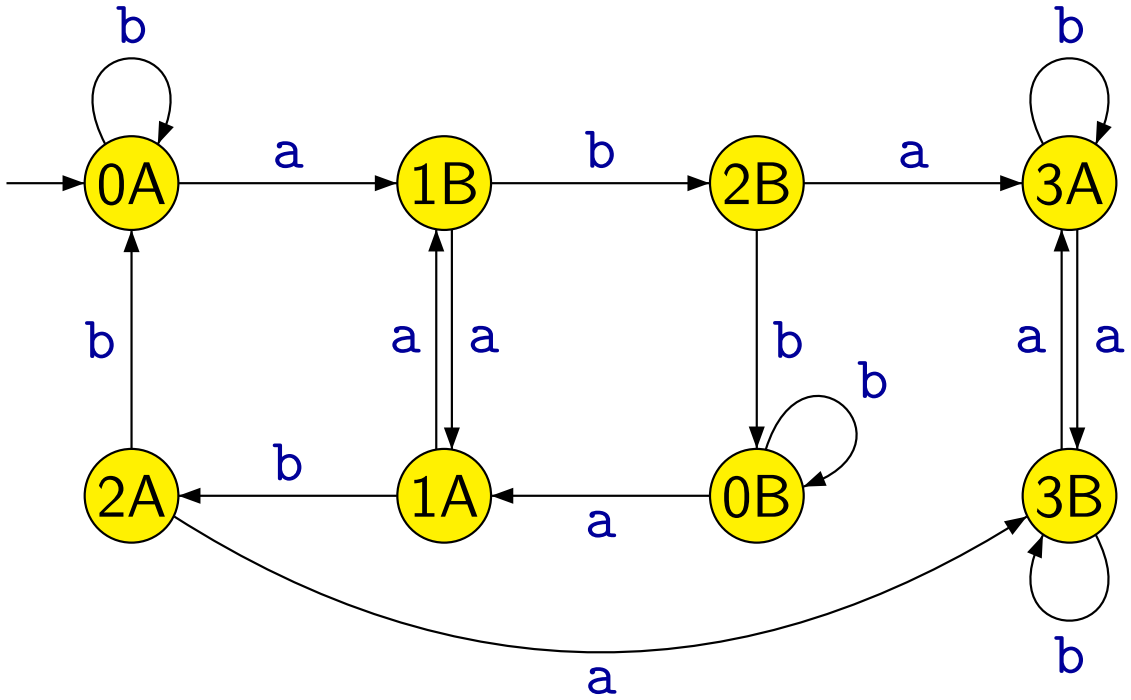




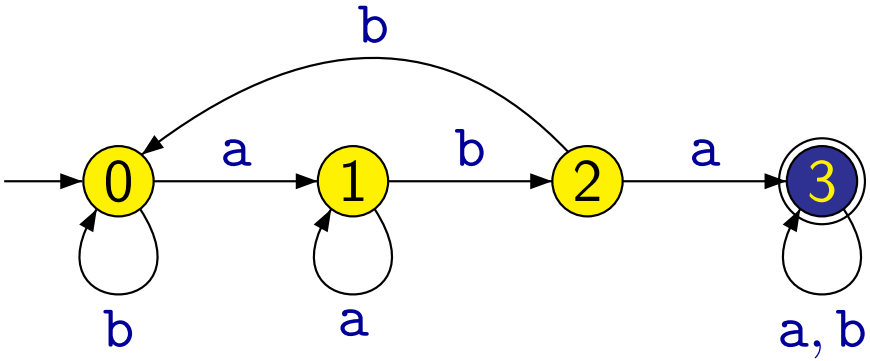
Automat pro průnik jazyků

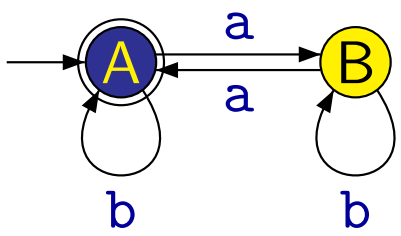


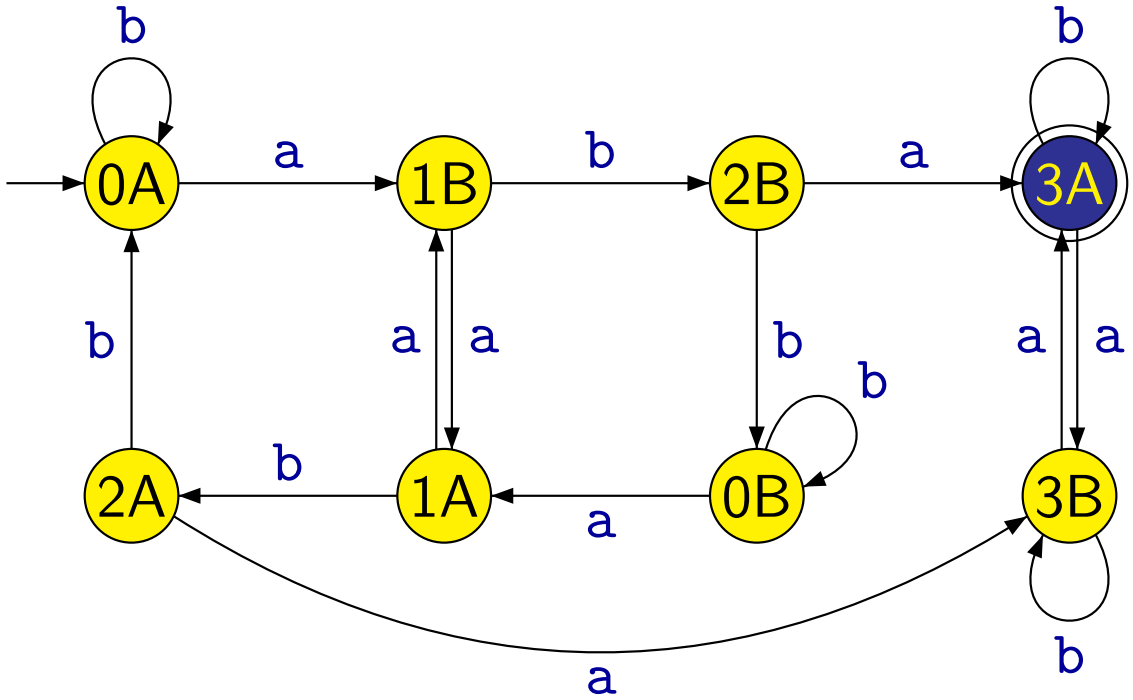




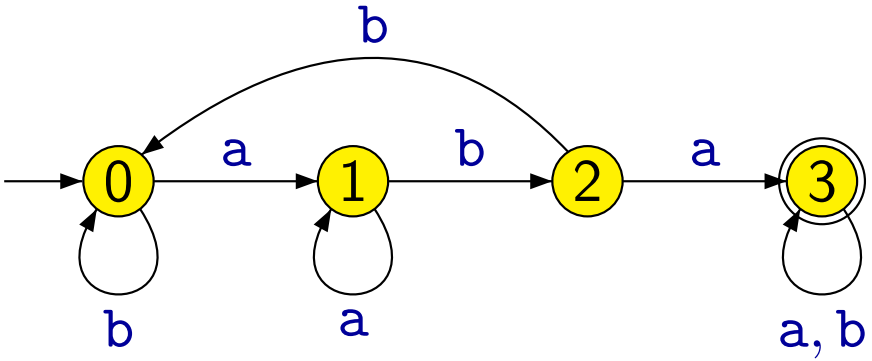
Automat pro průnik jazyků

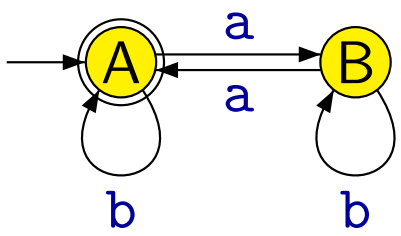


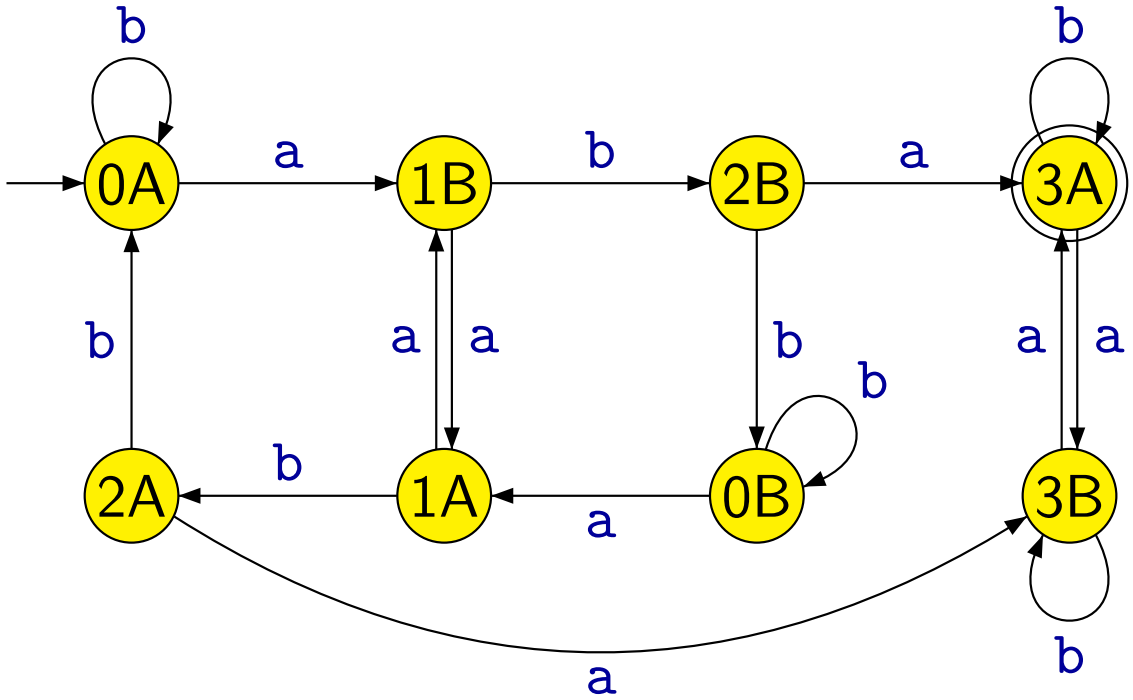




Automat pro průnik jazyků







Automat pro průnik jazyků

Formálně můžeme popsat tuto konstrukci následovně:

Předpokládáme, že máme dva deterministické konečné automaty $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ a $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$.

K nim setrojíme DKA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ kde:

- $Q = Q_1 \times Q_2$
- $\delta(\,(q_1, q_2), a\,) = (\,\delta_1(q_1, a), \delta_2(q_2, a)\,)$ pro všechna $q_1 \in Q_1$, $q_2 \in Q_2$, $a \in \Sigma$
- $q_0 = (q_{01}, q_{02})$
- $F = F_1 \times F_2$

Není težké ověřit, že pro libovolné slovo $w \in \Sigma^*$ platí, že $w \in \mathcal{L}(\mathcal{A})$ právě tehdy, když $w \in \mathcal{L}(\mathcal{A}_1)$ a $w \in \mathcal{L}(\mathcal{A}_2)$, tj.

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$$

Průnik regulárních jazyků

Věta

Jestliže jazyky $L_1, L_2 \subseteq \Sigma^*$ jsou regulární, pak také jazyk $L_1 \cap L_2$ je regulární.

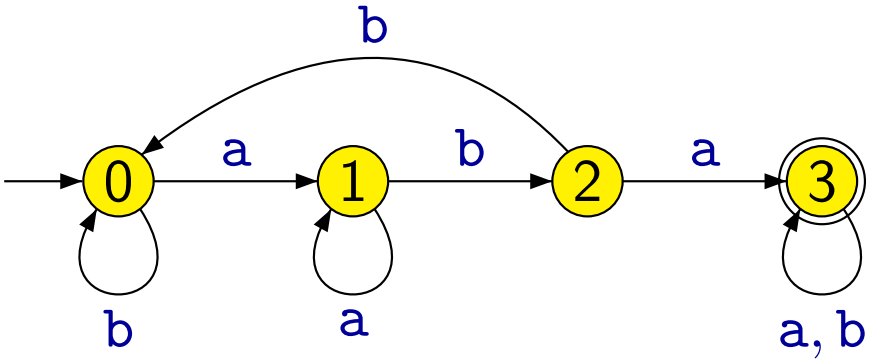
Důkaz: Předpokládejme, že $\mathcal{A}_1$ a $\mathcal{A}_2$ jsou deterministické konečné automaty takové, že

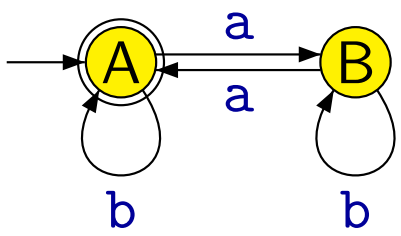
$$L_1 = \mathcal{L}(\mathcal{A}_1) \qquad L_2 = \mathcal{L}(\mathcal{A}_2)$$

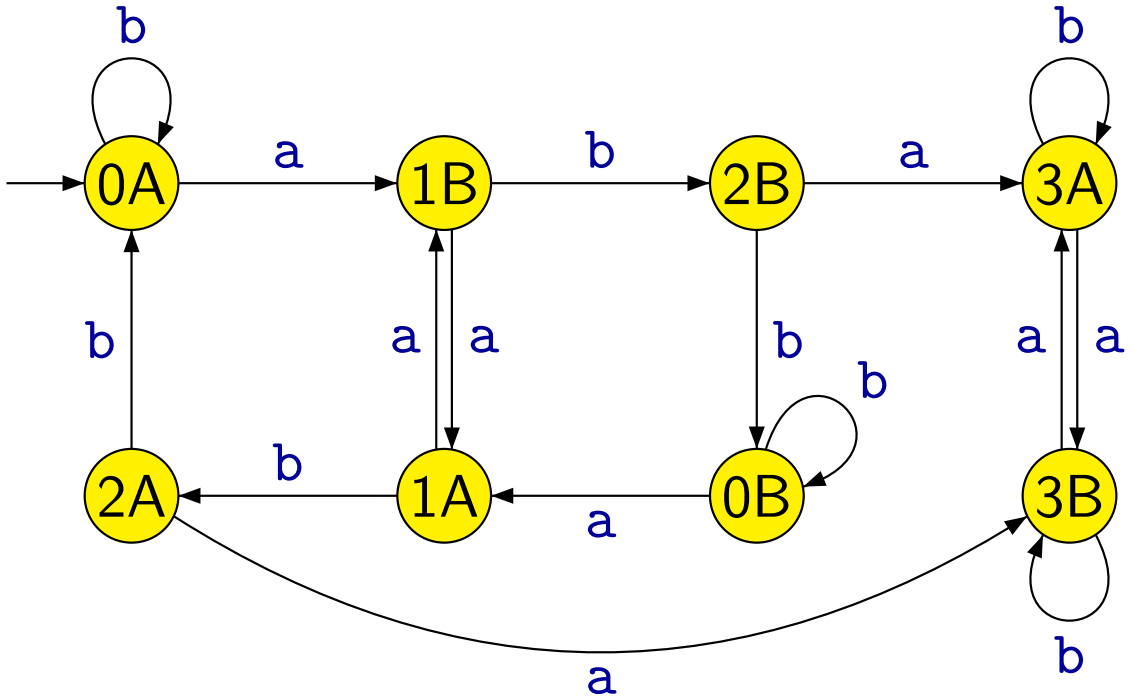
Popsanou konstrukcí k nim můžeme sestrojit deterministický konečný automat $\mathcal{A}$ takový, že

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2) = L_1 \cap L_2$$

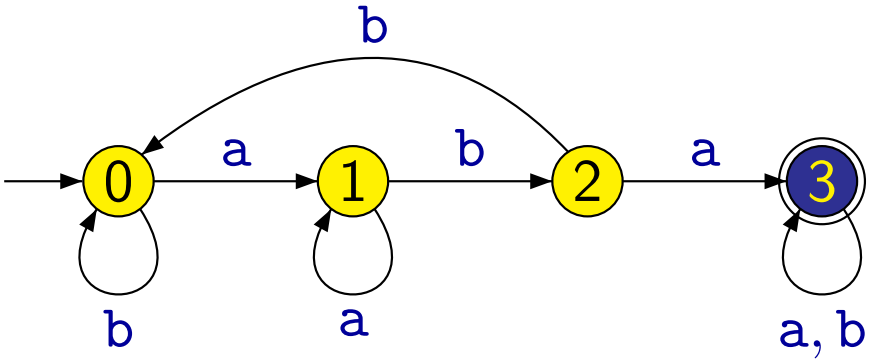
Automat pro sjednocení jazyků

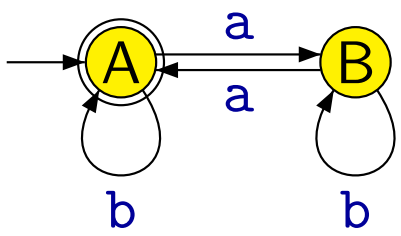


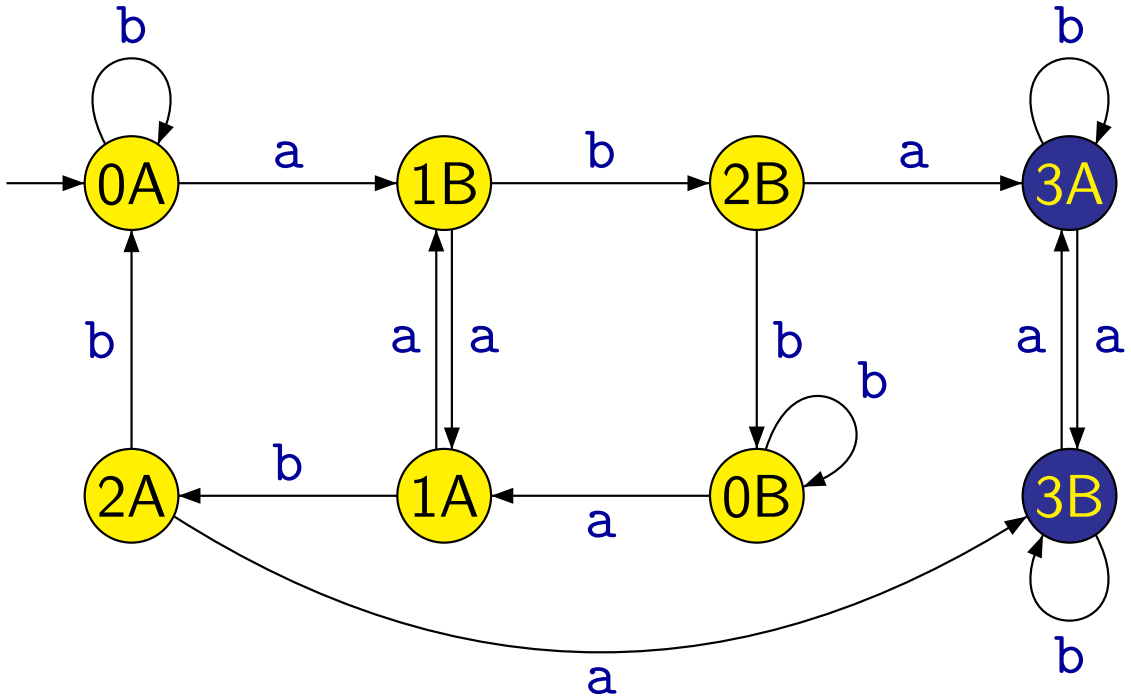




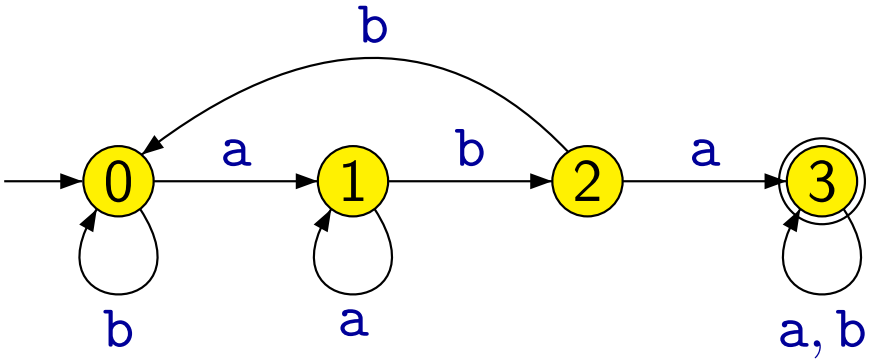
Automat pro sjednocení jazyků

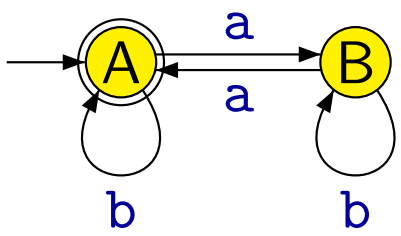


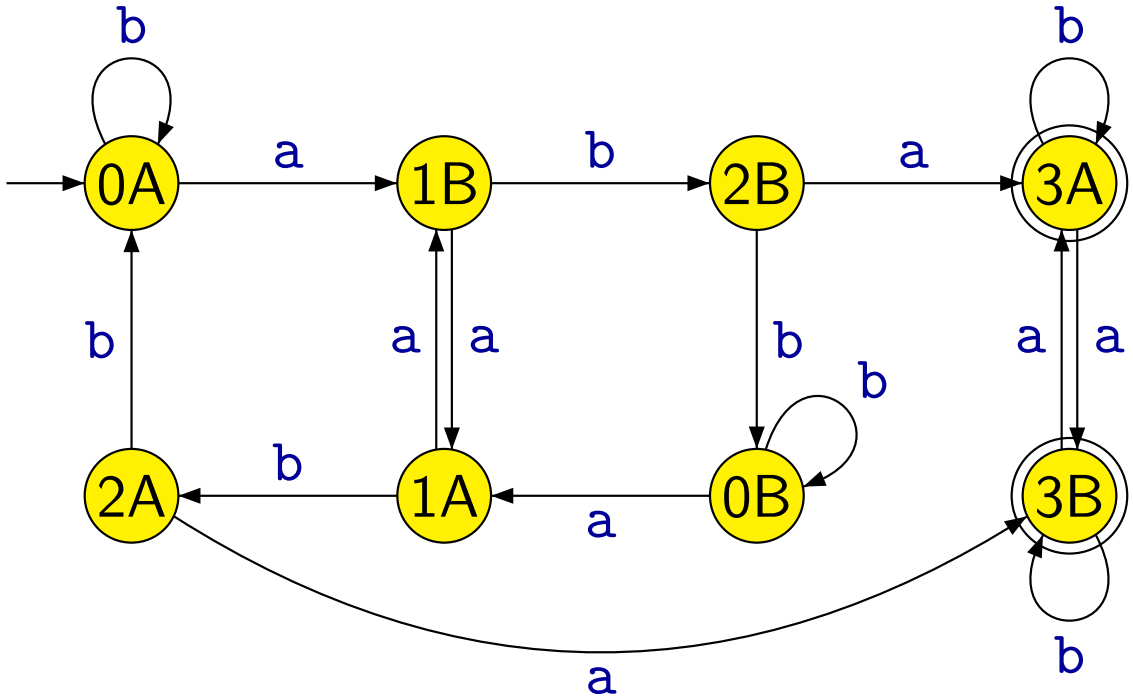




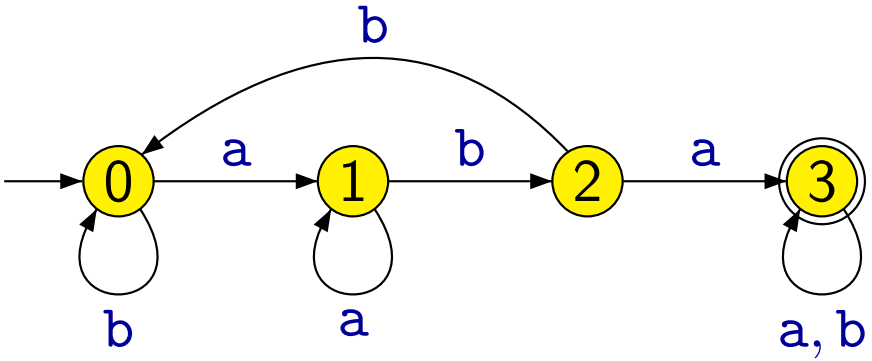
Automat pro sjednocení jazyků

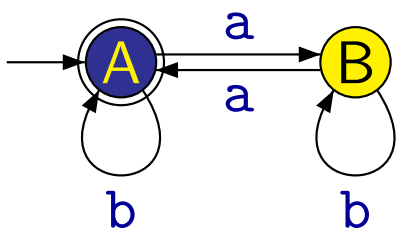


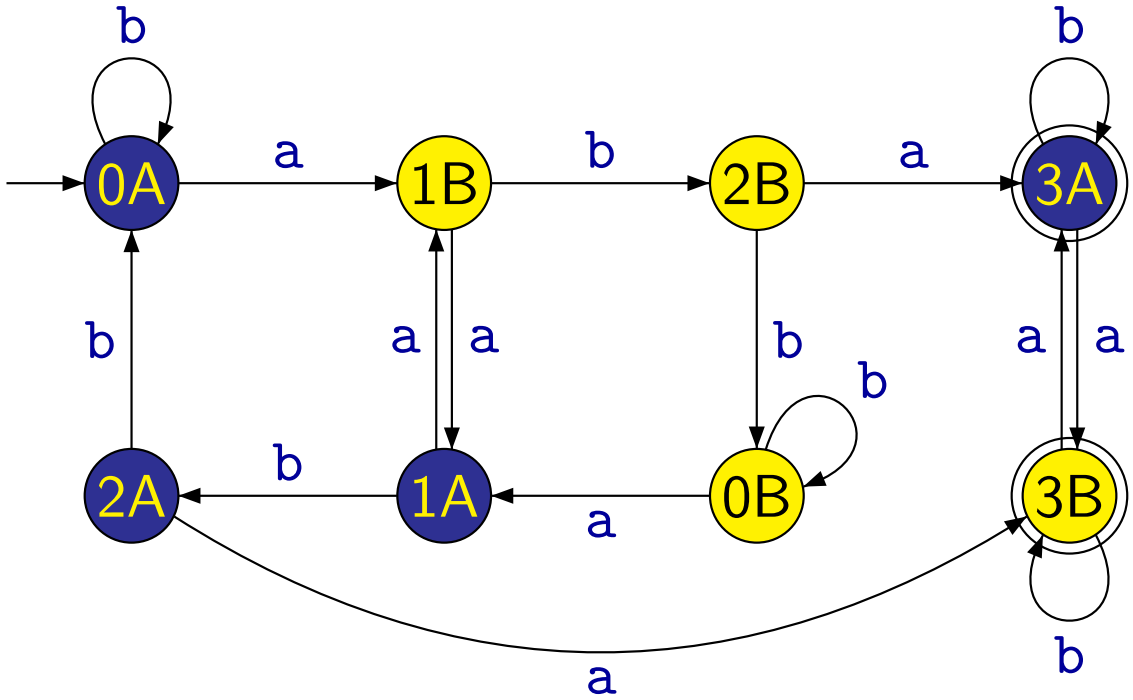




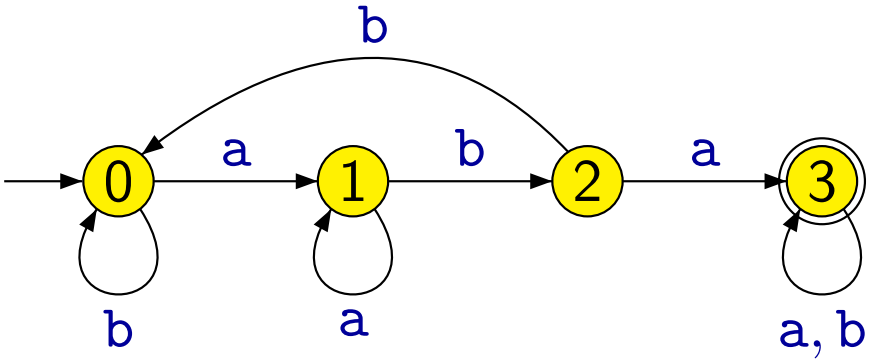
Automat pro sjednocení jazyků

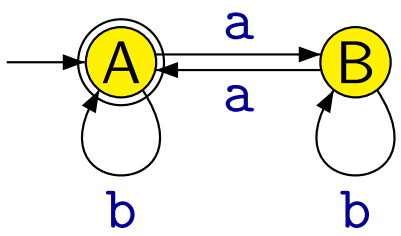


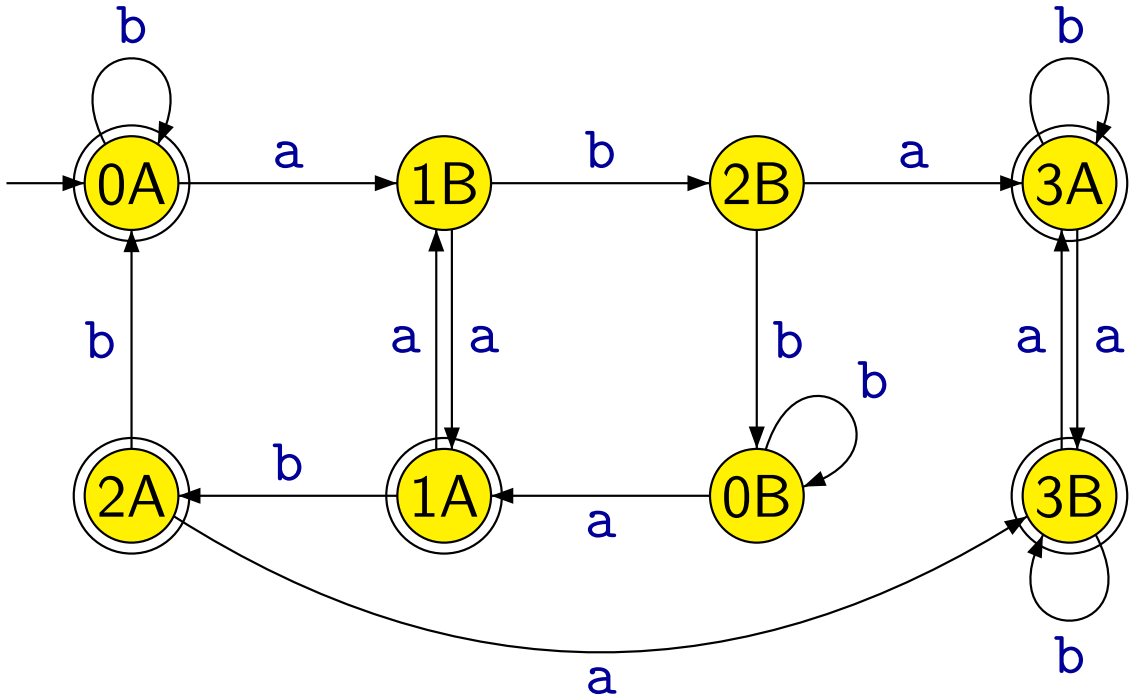




Automat pro sjednocení jazyků







Sjednocení regulárních jazyků

Konstukce automatu $\mathcal{A}$, který přijímá **sjednocení** jazyků přijímaných automaty $\mathcal{A}_1$ a $\mathcal{A}_2$, tj. jazyk

$$\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_1)$$

je téměř stejná jako v případě automatu přijímajícího $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.

Jediný rozdíl je v definici množiny přijímajících stavů:

- $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$

Sjednocení regulárních jazyků

Konstukce automatu $\mathcal{A}$, který přijímá **sjednocení** jazyků přijímaných automaty $\mathcal{A}_1$ a $\mathcal{A}_2$, tj. jazyk

$$\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_1)$$

je téměř stejná jako v případě automatu přijímajícího $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.

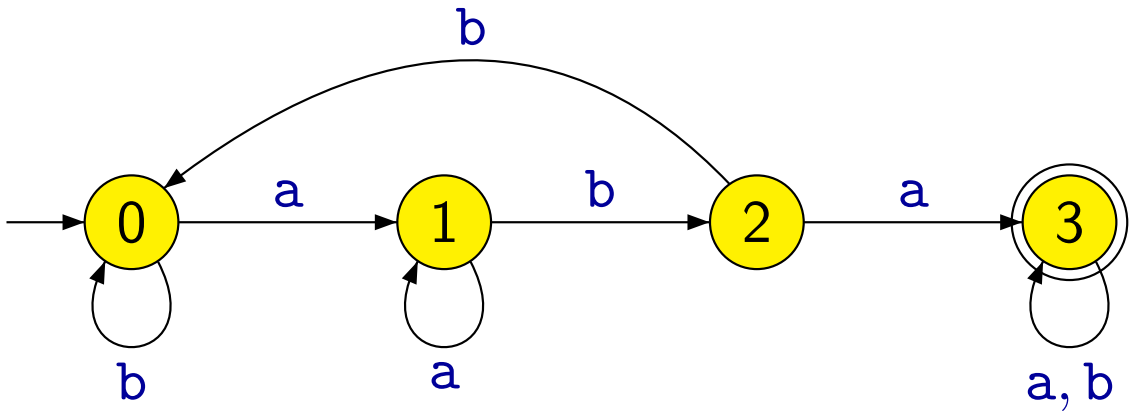
Jediný rozdíl je v definici množiny přijímajících stavů:

- $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$

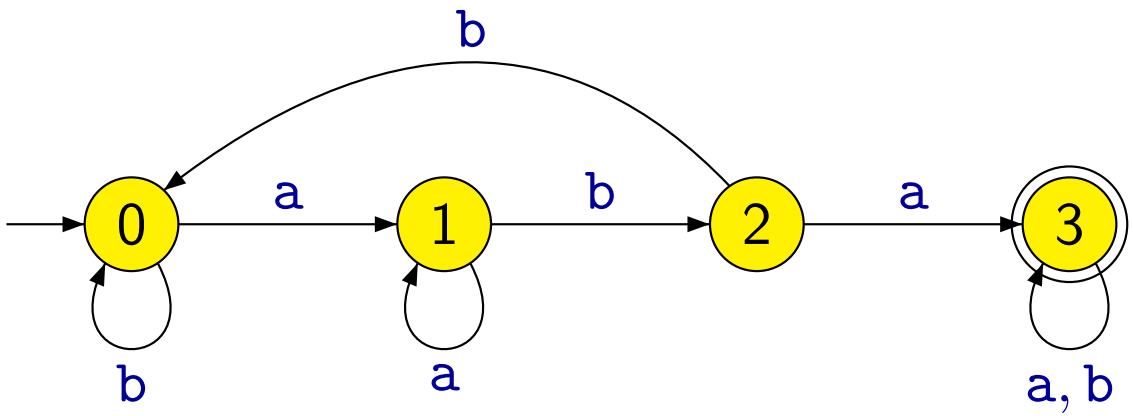
Věta

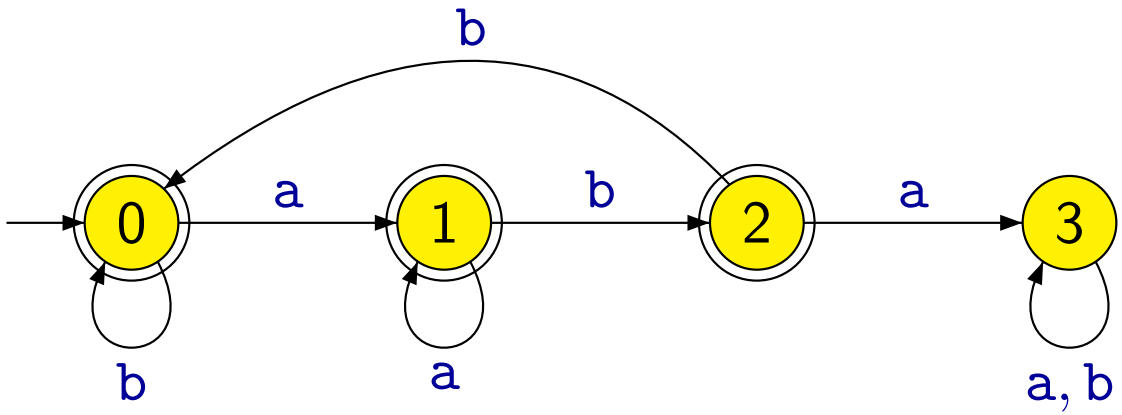
Jestliže jazyky $L_1, L_2 \subseteq \Sigma^*$ jsou regulární, pak také jazyk $L_1 \cup L_2$ je regulární.

Automat pro doplněk jazyka



Automat pro doplněk jazyka





Doplněk regulárního jazyka

K DKA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ sestrojíme DKA $\mathcal{A}' = (Q, \Sigma, \delta, q_0, Q - F)$.

Je očividné, že pro každé slovo $w \in \Sigma^*$ platí, že $w \in \mathcal{L}(\mathcal{A}')$ právě tehdy, když $w \notin \mathcal{L}(\mathcal{A})$, tj.

$$\mathcal{L}(\mathcal{A}') = \overline{\mathcal{L}(\mathcal{A})}$$

Doplněk regulárního jazyka

K DKA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ sestrojíme DKA $\mathcal{A}' = (Q, \Sigma, \delta, q_0, Q - F)$.

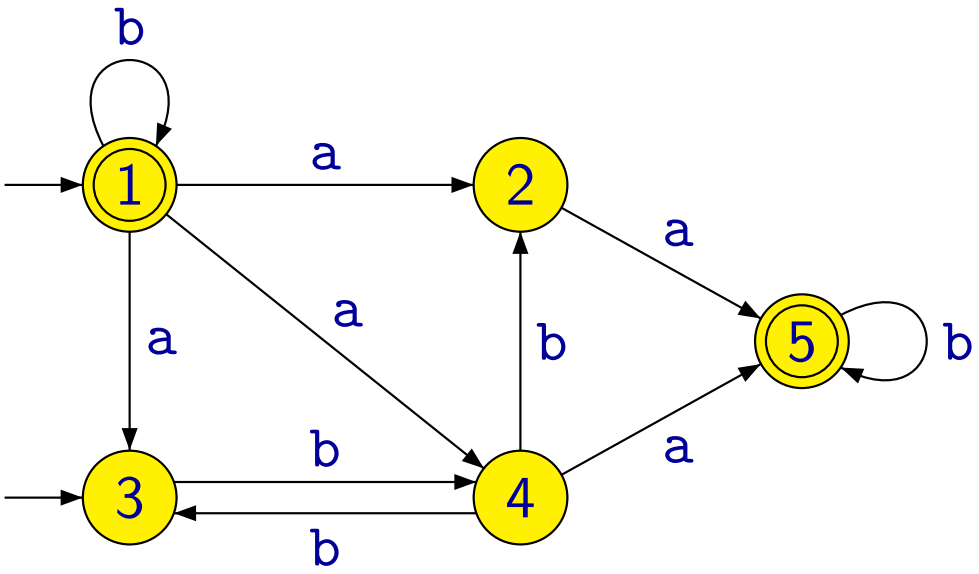
Je očividné, že pro každé slovo $w \in \Sigma^*$ platí, že $w \in \mathcal{L}(\mathcal{A}')$ právě tehdy, když $w \notin \mathcal{L}(\mathcal{A})$, tj.

$$\mathcal{L}(\mathcal{A}') = \overline{\mathcal{L}(\mathcal{A})}$$

Věta

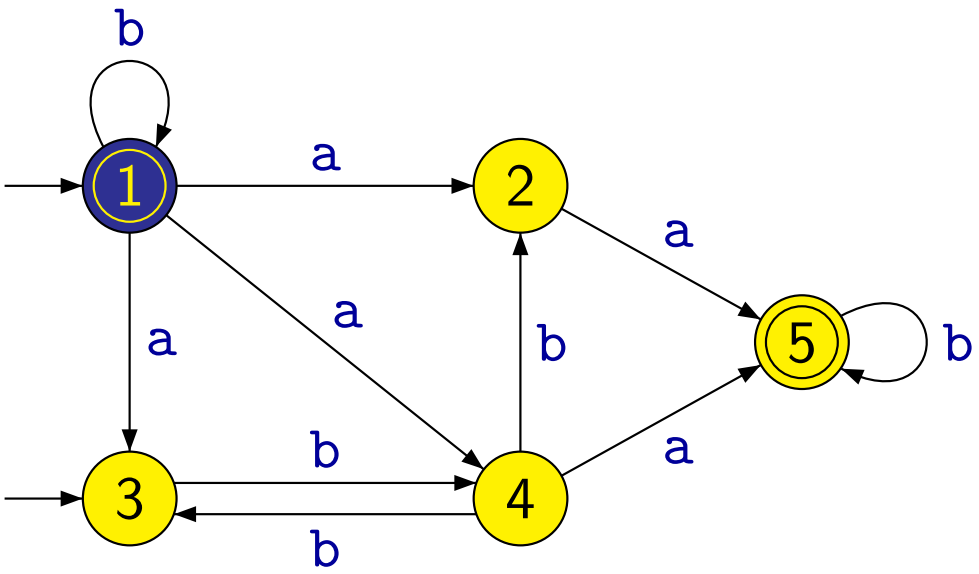
Jestliže jazyk L je regulární, pak také jeho doplněk $\overline{L}$ je regulární.

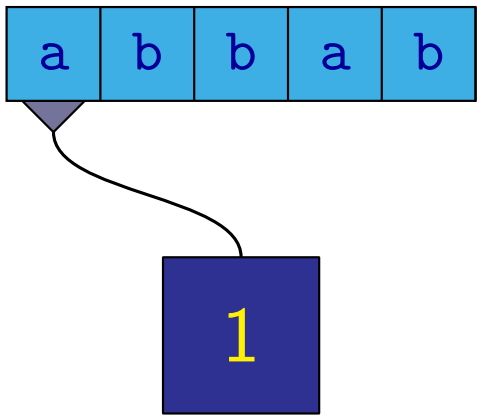
Nedeterministický konečný automat



- Z jednoho stavu může vézt libovolný (i nulový) počet přechodů označených stejným symbolem.
- V automatu může být víc než jeden počáteční stav.

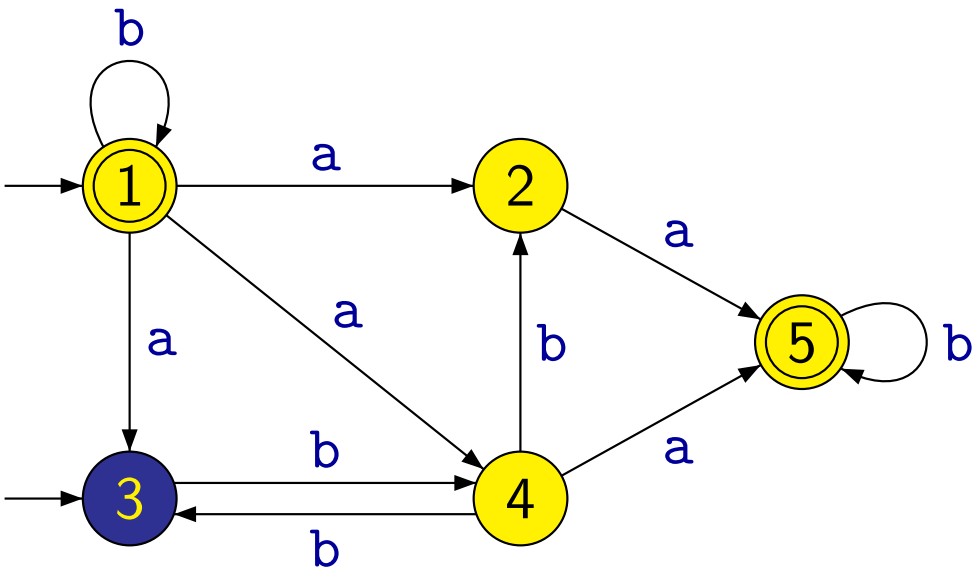
Nedeterministický konečný automat

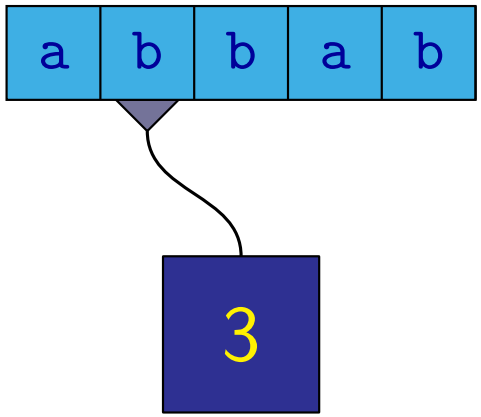




1

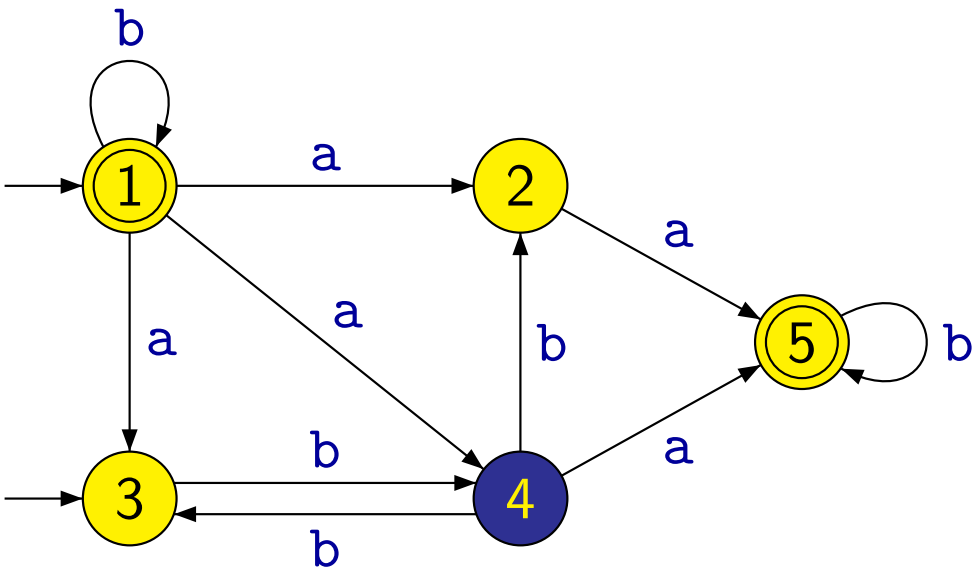
Nedeterministický konečný automat

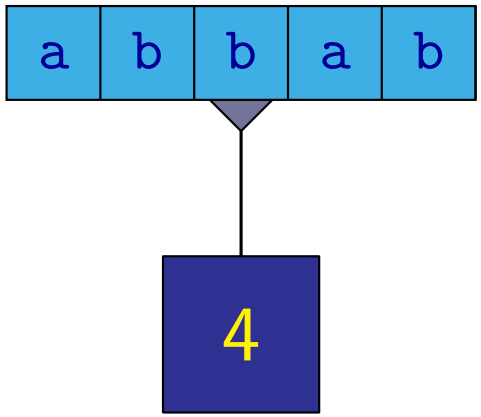




$1 \xrightarrow{a} 3$

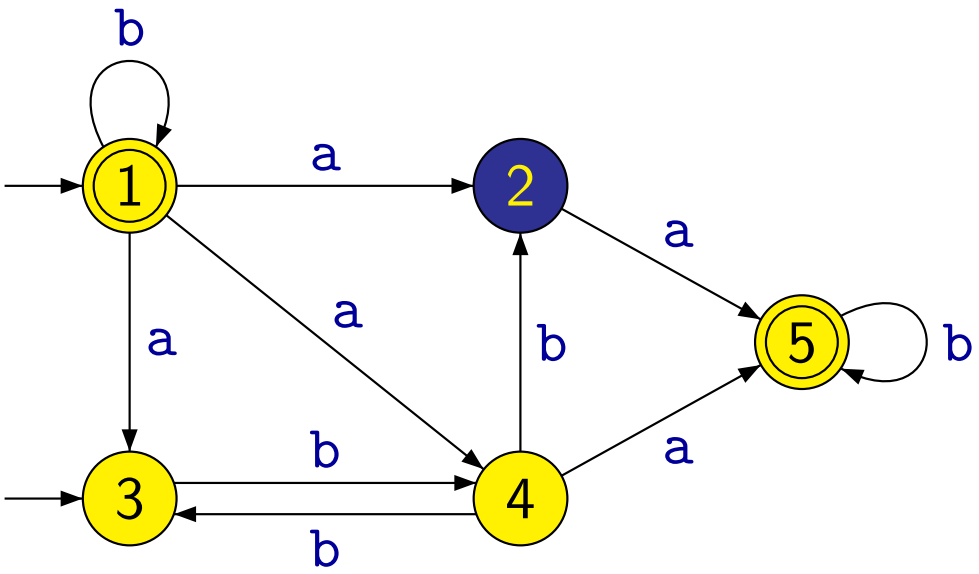
Nedeterministický konečný automat

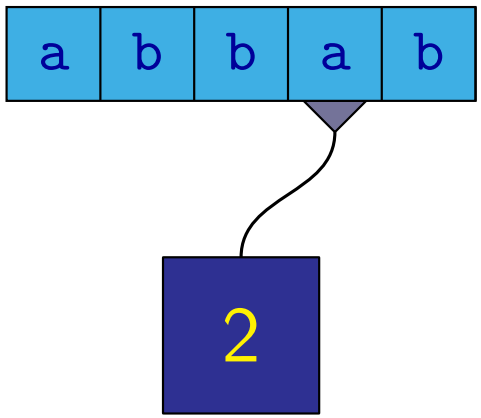




$$1 \xrightarrow{a} 3 \xrightarrow{b} 4$$

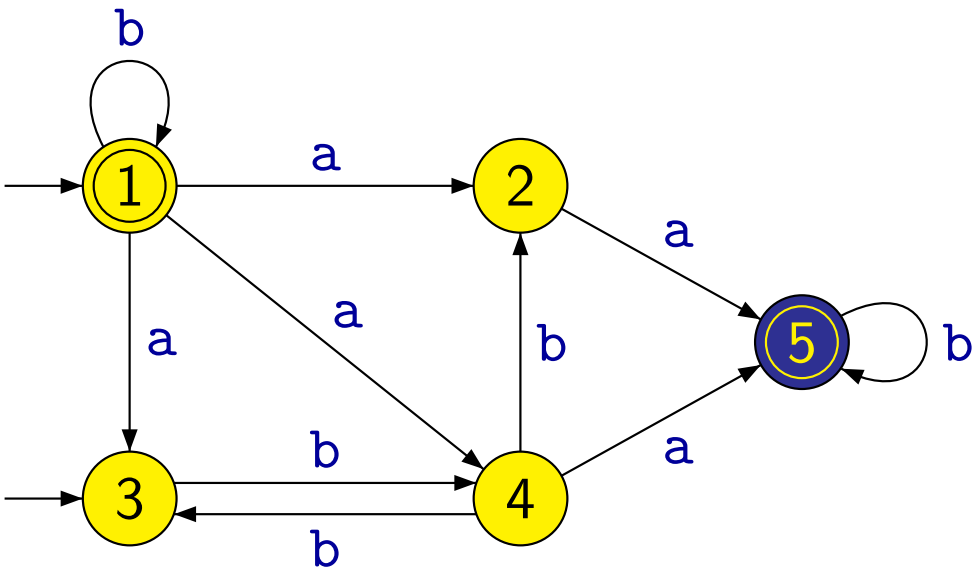
Nedeterministický konečný automat

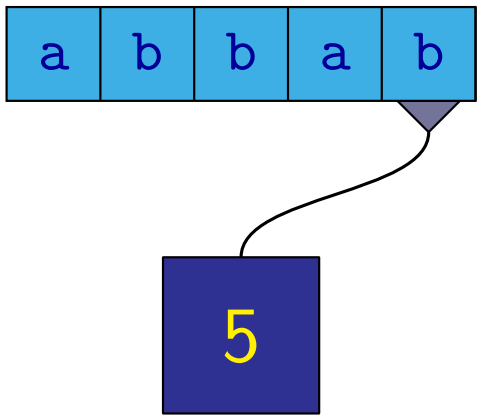




$$1 \xrightarrow{a} 3 \xrightarrow{b} 4 \xrightarrow{b} 2$$

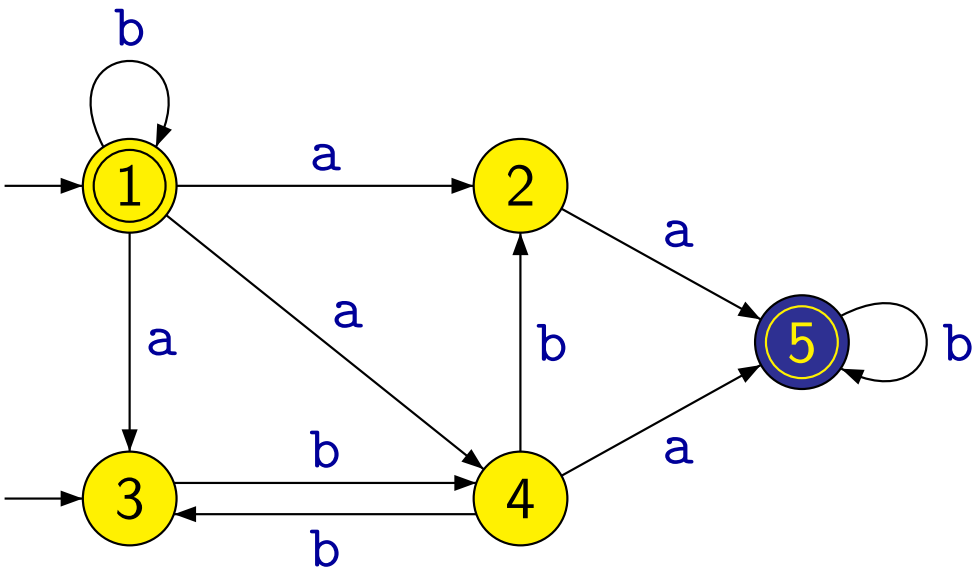
Nedeterministický konečný automat

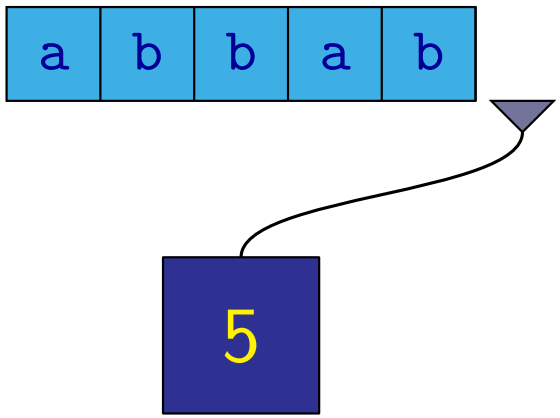




$$1 \xrightarrow{a} 3 \xrightarrow{b} 4 \xrightarrow{b} 2 \xrightarrow{a} 5$$

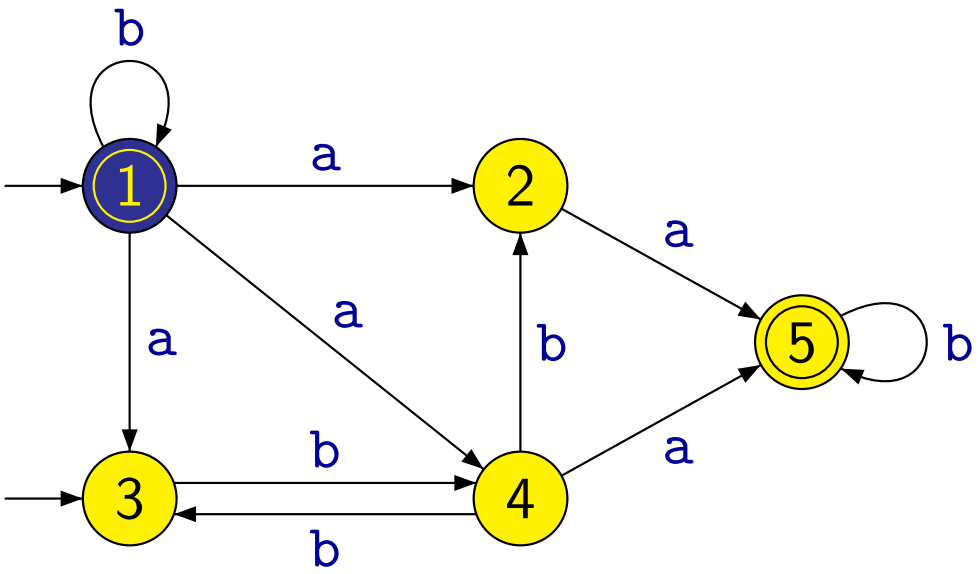
Nedeterministický konečný automat

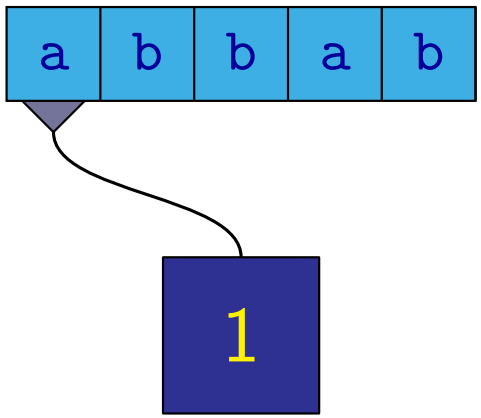




$$1 \xrightarrow{a} 3 \xrightarrow{b} 4 \xrightarrow{b} 2 \xrightarrow{a} 5 \xrightarrow{b} 5$$

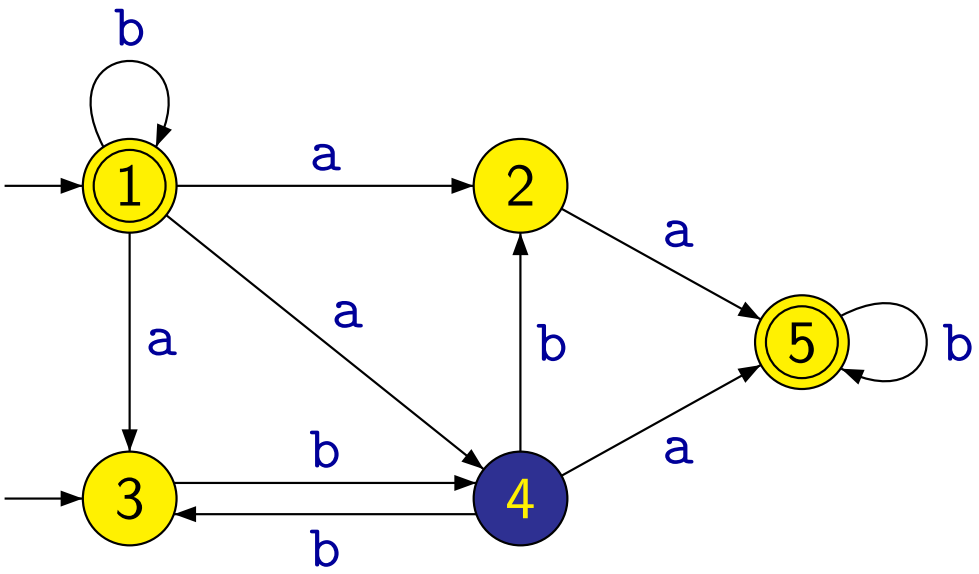
Nedeterministický konečný automat

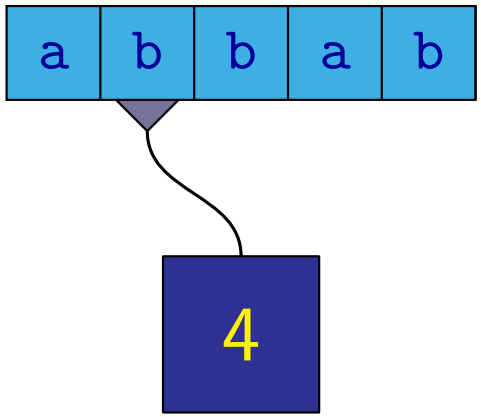




1

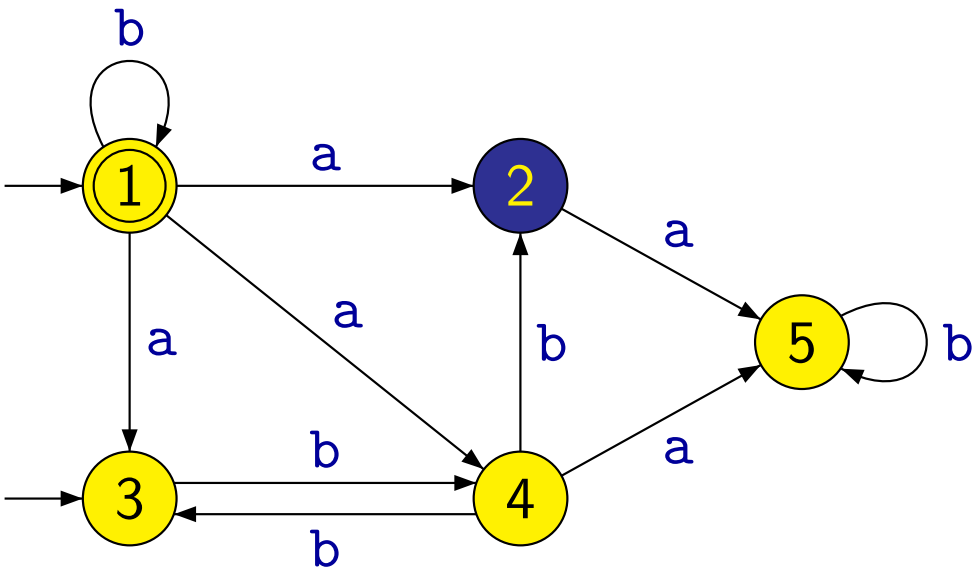
Nedeterministický konečný automat

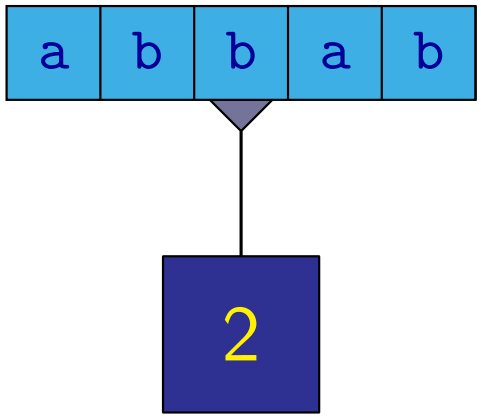




$1 \xrightarrow{a} 4$

Nedeterministický konečný automat

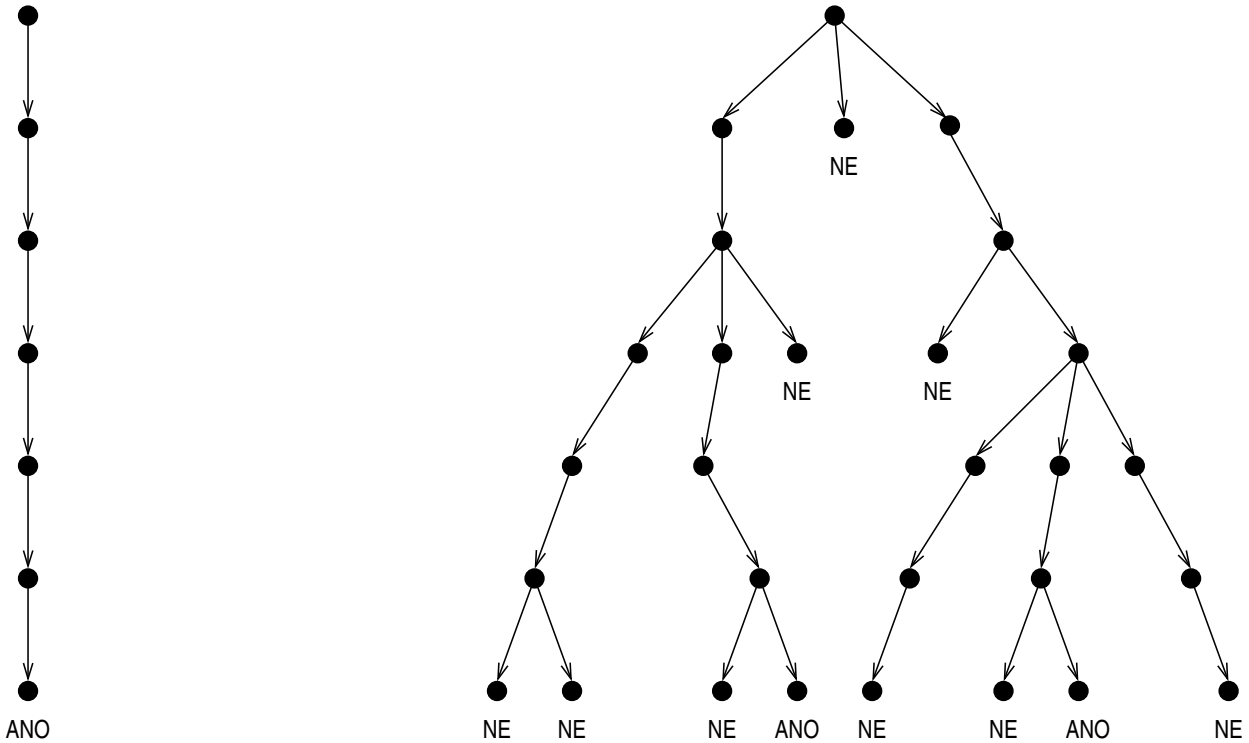




$$1 \xrightarrow{a} 4 \xrightarrow{b} 2$$

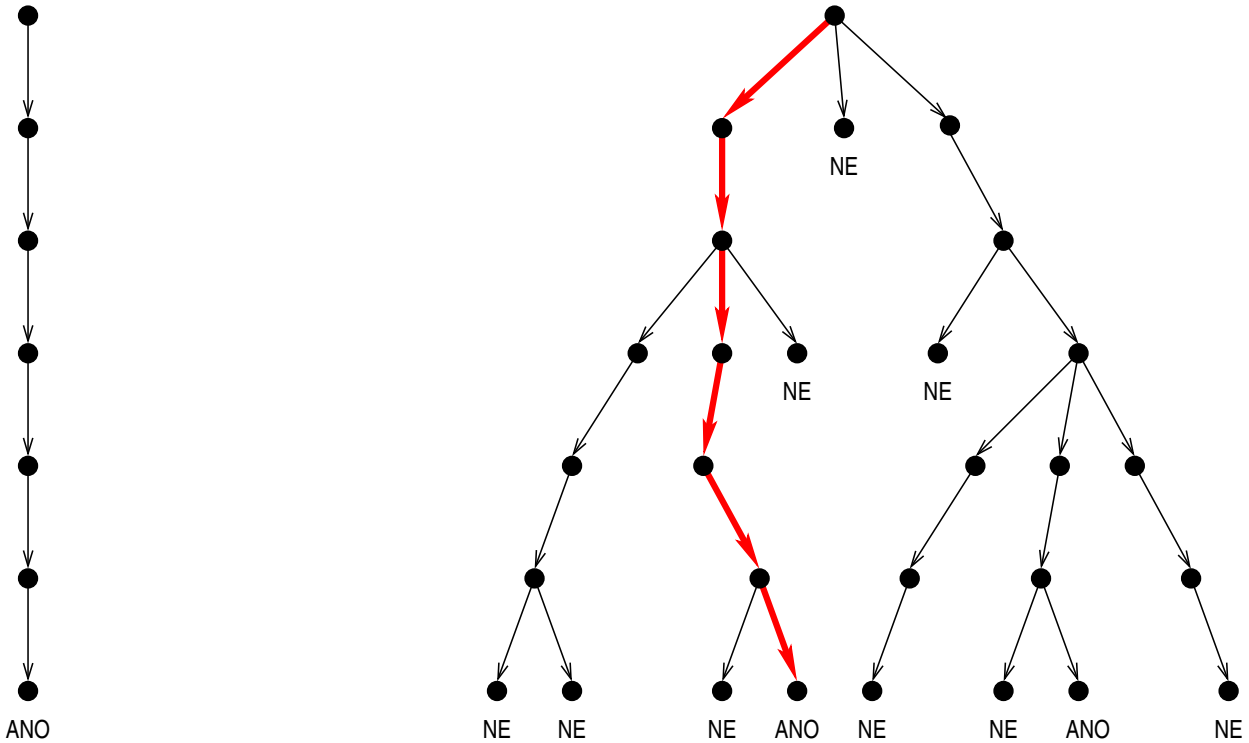
Nedeterministický konečný automat

Nedeterministický konečný automat přijímá dané slovo, jestliže **existuje** alespoň jeden jeho výpočet, který vede k přijetí tohoto slova.



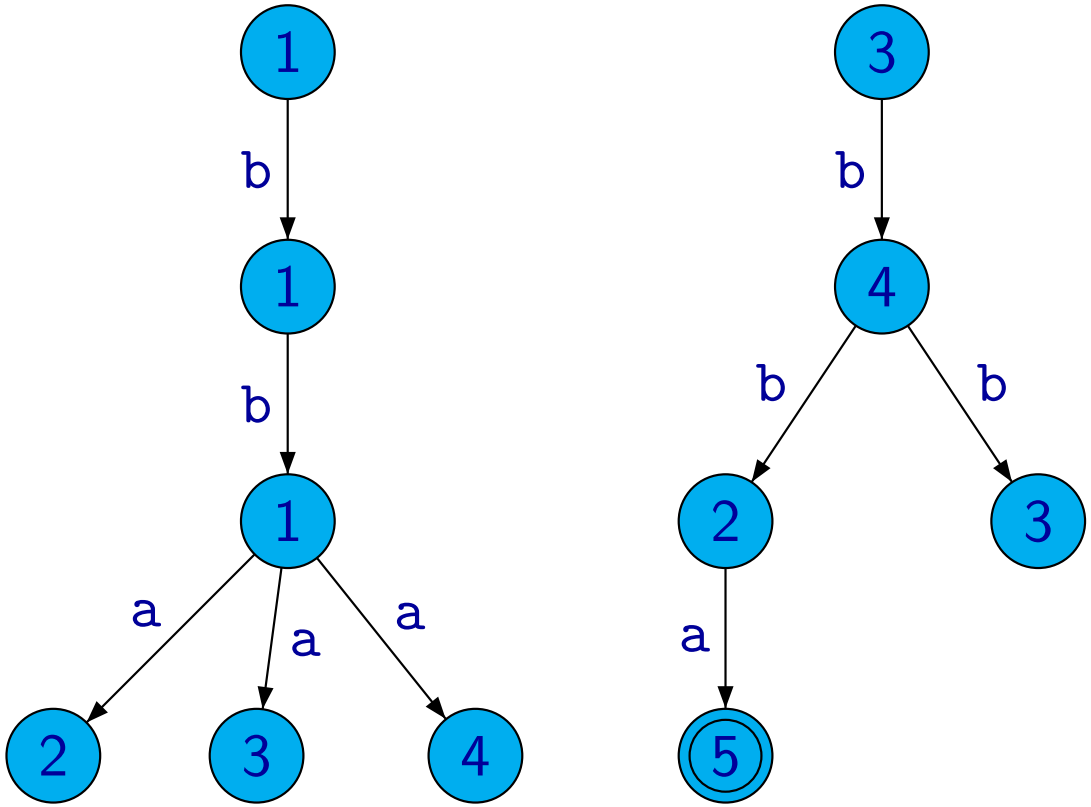
Nedeterministický konečný automat

Nedeterministický konečný automat přijímá dané slovo, jestliže **existuje** alespoň jeden jeho výpočet, který vede k přijetí tohoto slova.



Nedeterministický konečný automat

	a	b
$\leftrightarrow 1$	$2, 3, 4$	1
2	5	$-$
$\rightarrow 3$	$-$	4
4	5	$2, 3$
$\leftarrow 5$	$-$	5



Příklad: Les reprezentující všechny možné výpočty nad slovem bba.

Nedeterministický konečný automat

Formálně je **nedeterministický konečný automat** (**NKA**) definován jako pětice

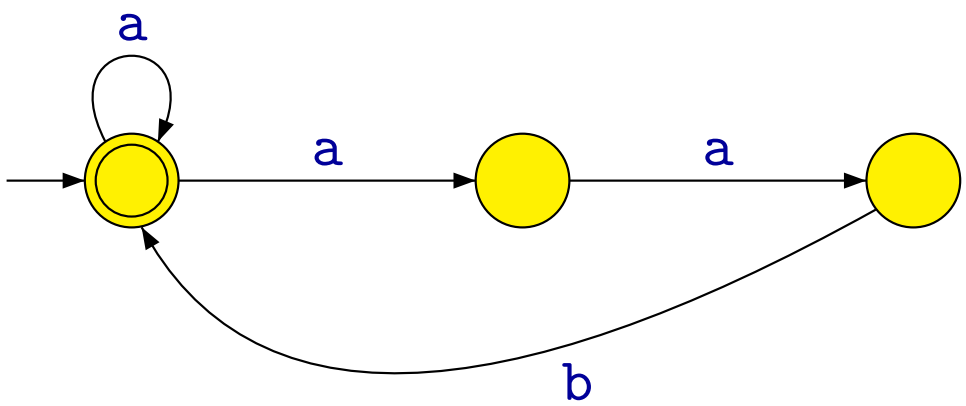
$$(Q, \Sigma, \delta, I, F)$$

kde:

- Q je konečná množina **stavů**
- Σ je konečná **abeceda**
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ je **přechodová funkce**
- $I \subseteq Q$ je množina **počátečních stavů**
- $F \subseteq Q$ je množina **přijímajících stavů**

Příklady nedeterministických konečných automatů

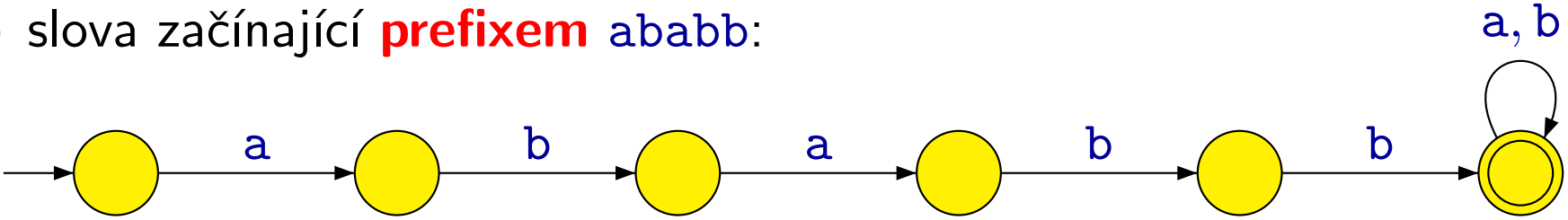
Příklad: Automat rozpoznávající jazyk nad abecedou $\{\mathtt{a}, \mathtt{b}\}$ tvořený slovy, kde každému výskytu symbolu $\mathtt{b}$ bezprostředně předchází dva symboly $\mathtt{a}$.



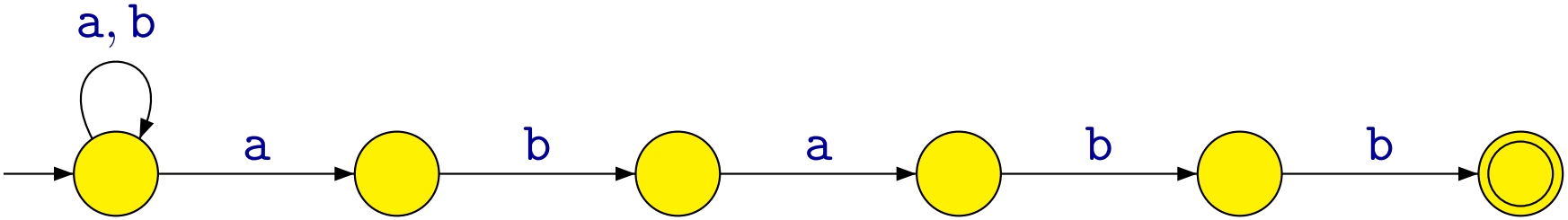
Příklady nedeterministických konečných automatů

Příklad: Automaty rozpoznávající jazyky nad abecedou $\{a, b\}$:

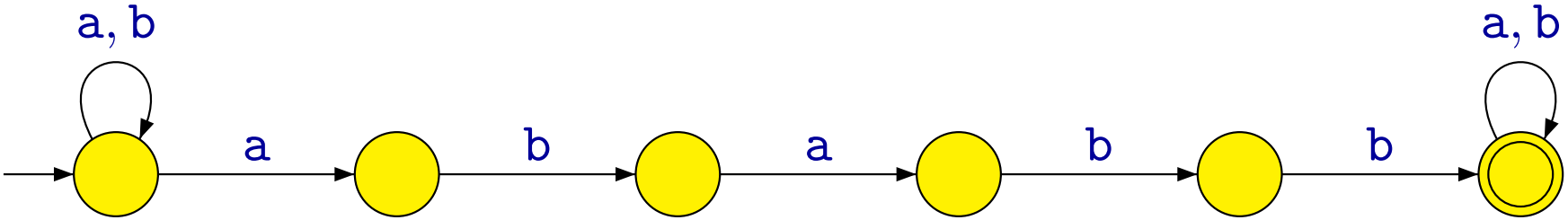
- slova začínající **prefixem** ababb:



- slova končící **sufixem** ababb:

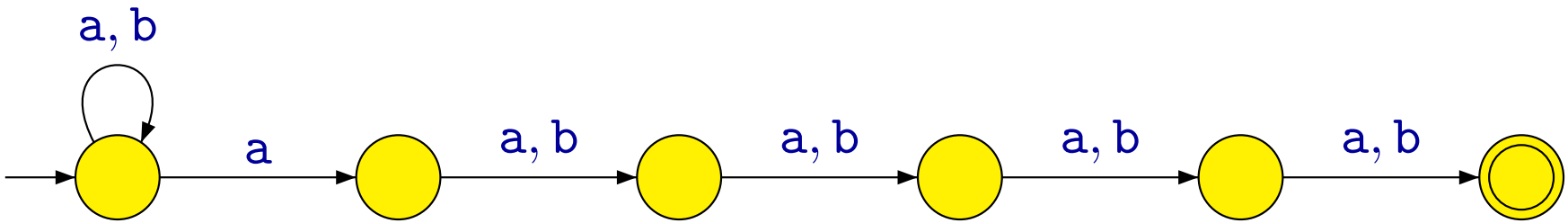


- slova obsahující **podslovo** ababb:

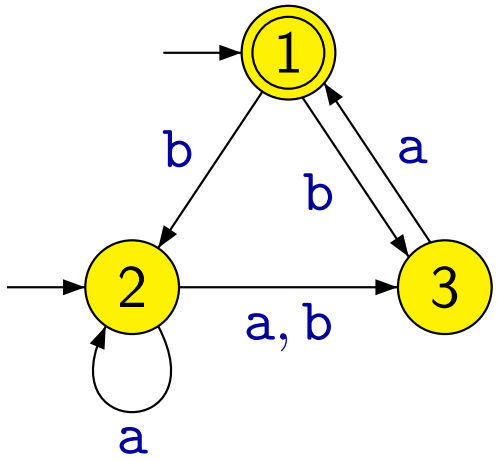


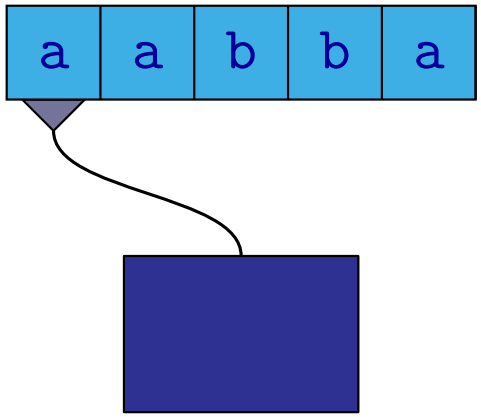
Příklady nedeterministických konečných automatů

Příklad: Automat rozpoznávající jazyk nad abecedou $\{\mathtt{a}, \mathtt{b}\}$ tvořený slovy, kde pátý symbol od konce je $\mathtt{a}$.

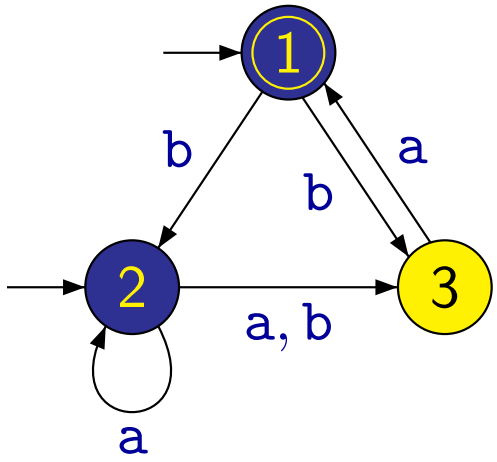


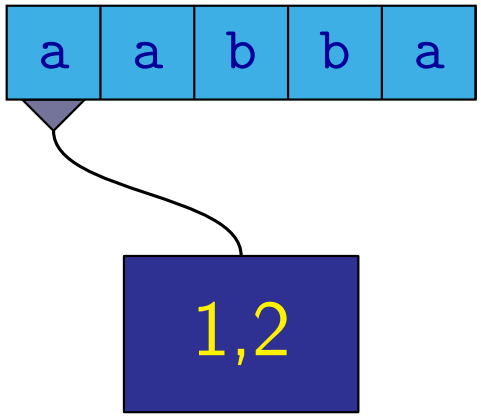
Převod NKA na DKA



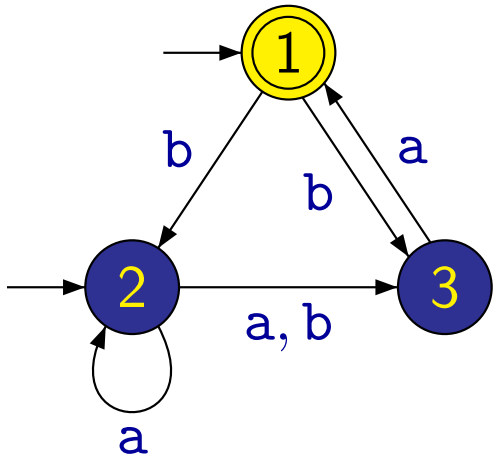


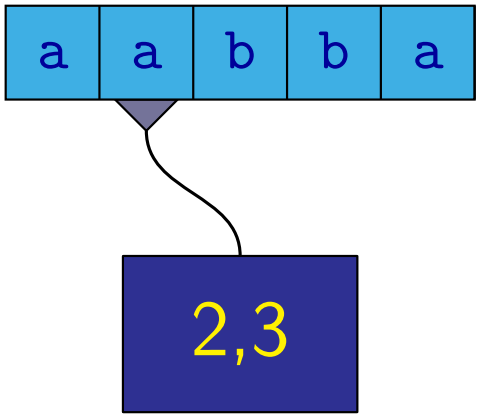
Převod NKA na DKA



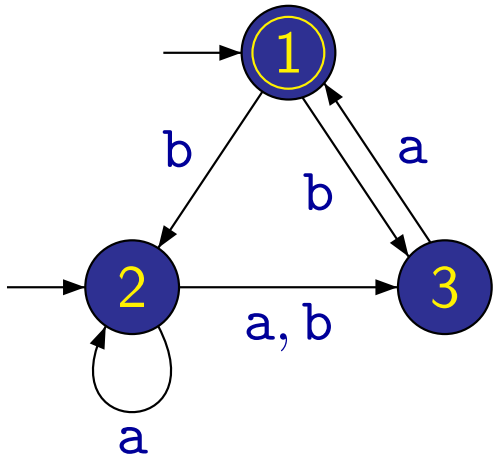


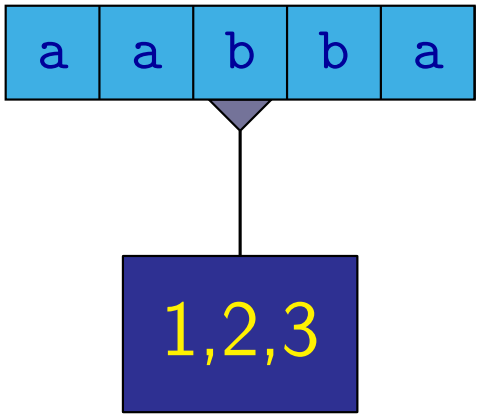
Převod NKA na DKA



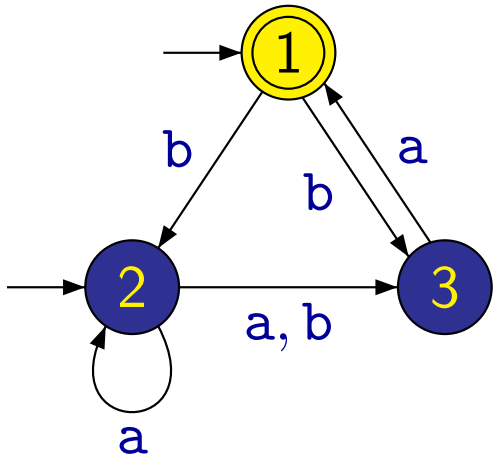


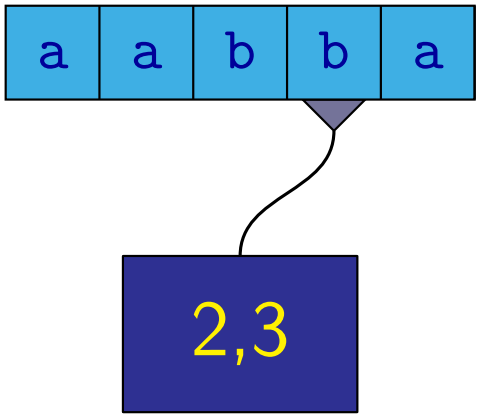
Převod NKA na DKA



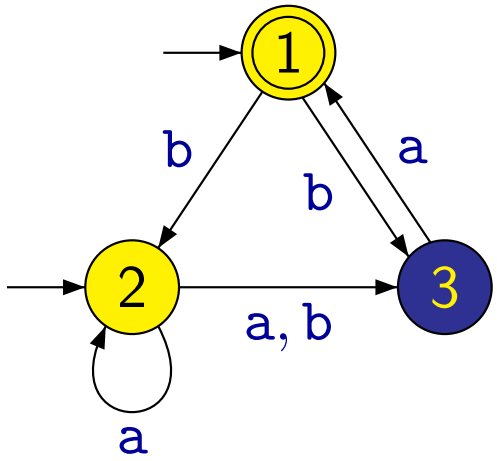


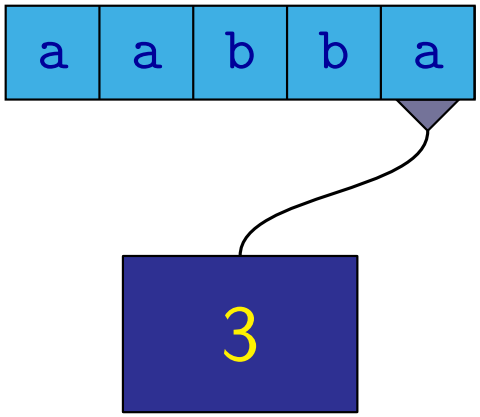
Převod NKA na DKA



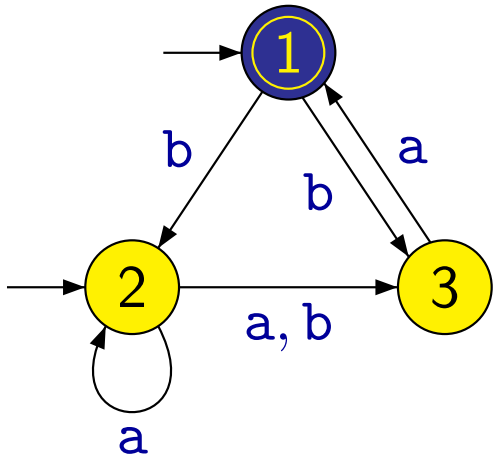


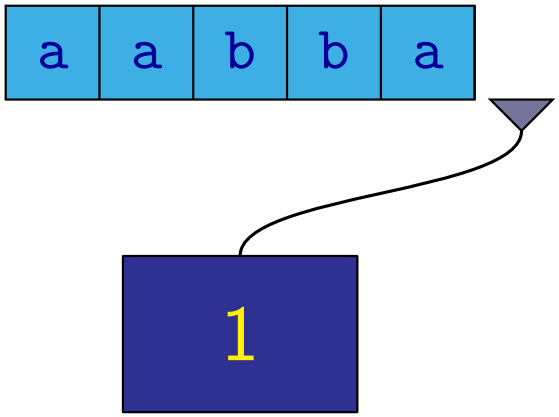
Převod NKA na DKA



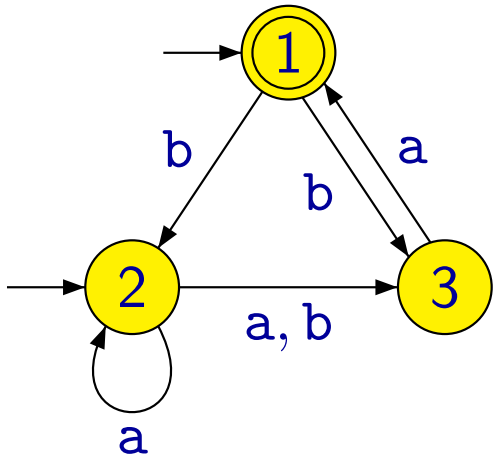


Převod NKA na DKA

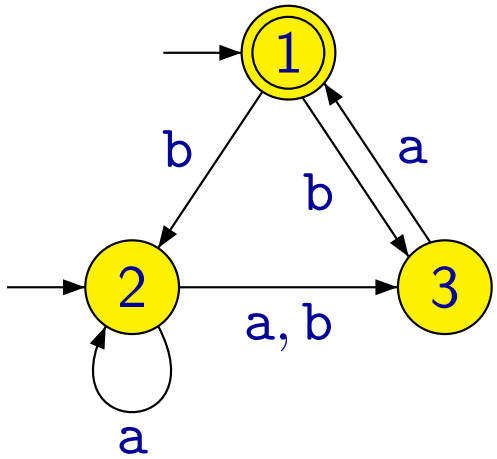




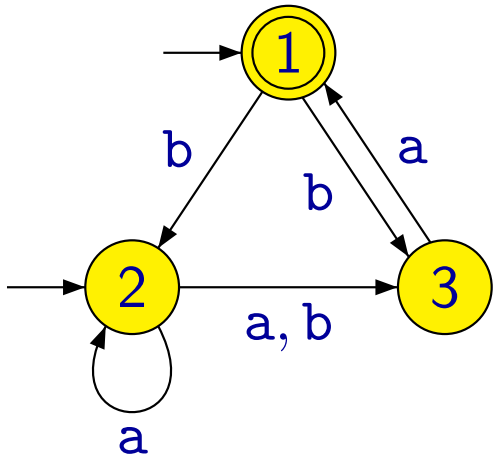
Převod NKA na DKA

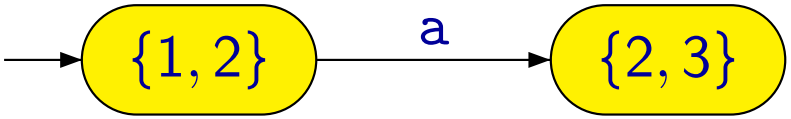


Převod NKA na DKA

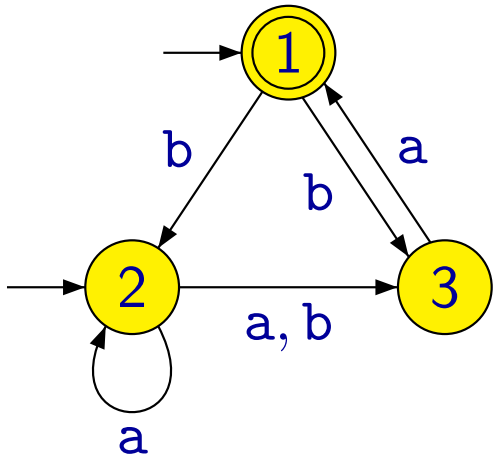


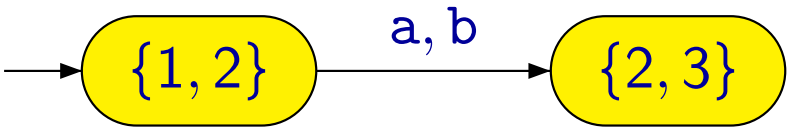
Převod NKA na DKA



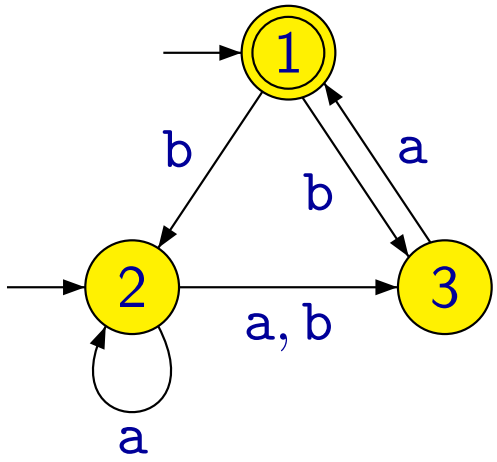


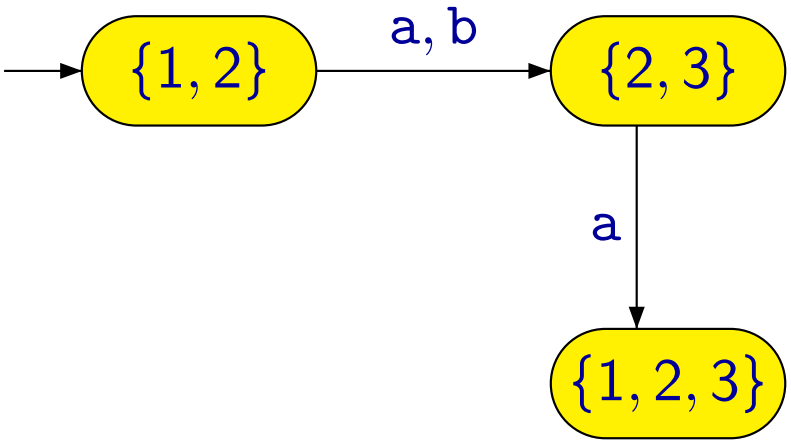
Převod NKA na DKA



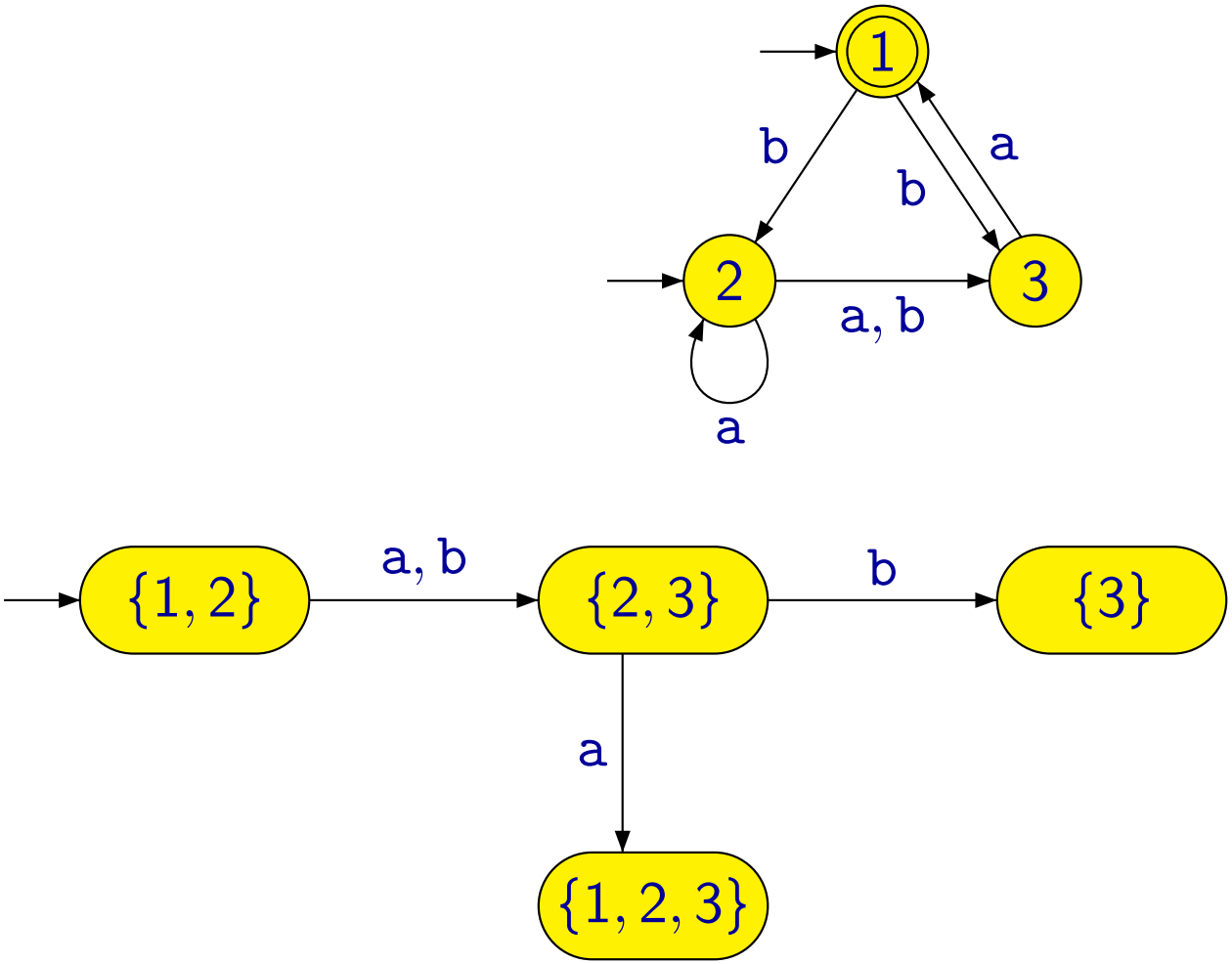


Převod NKA na DKA

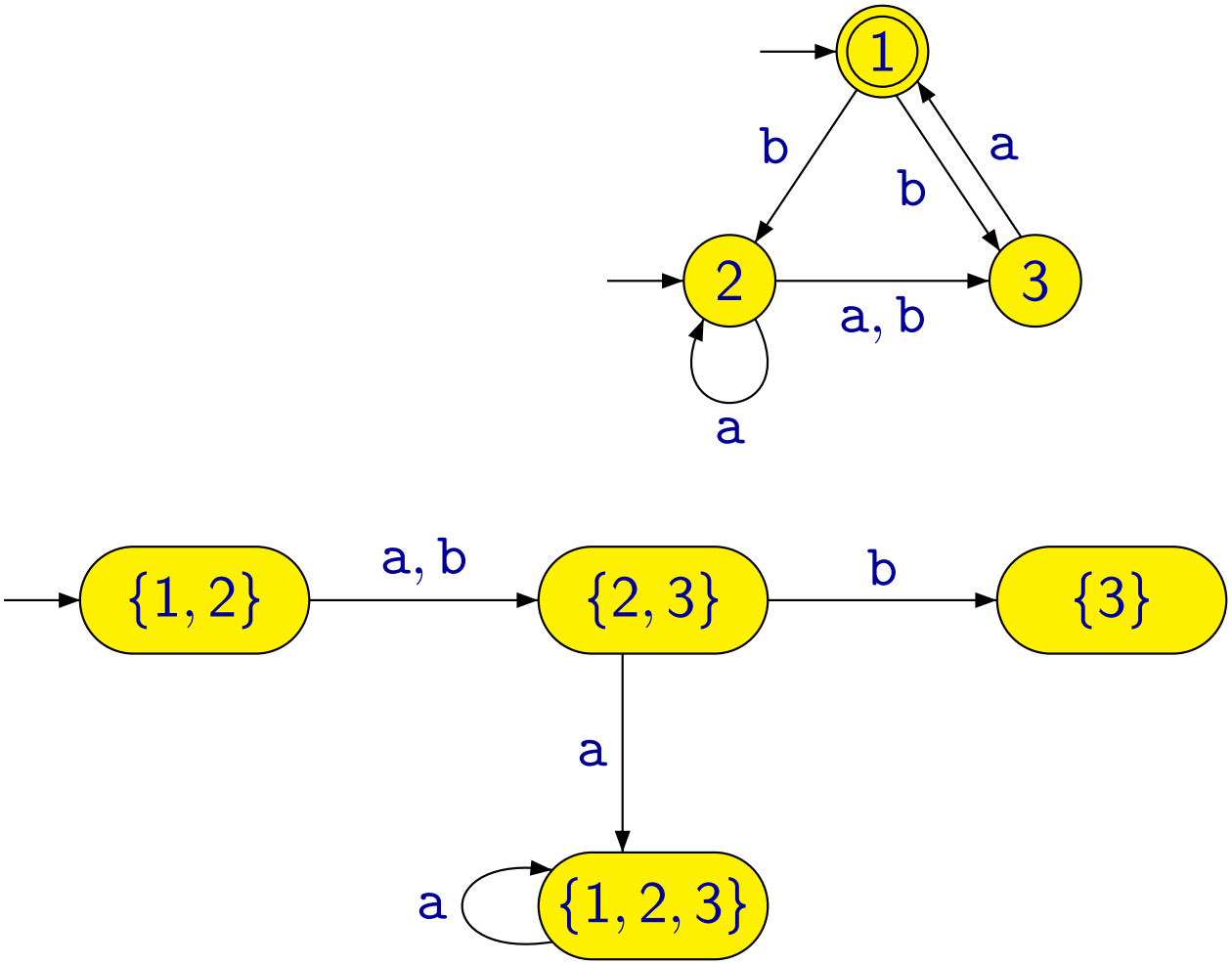




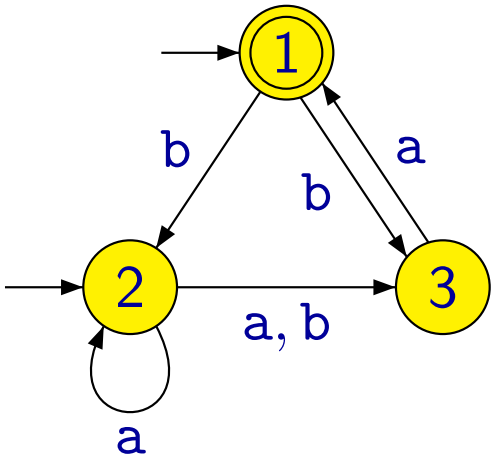
Převod NKA na DKA

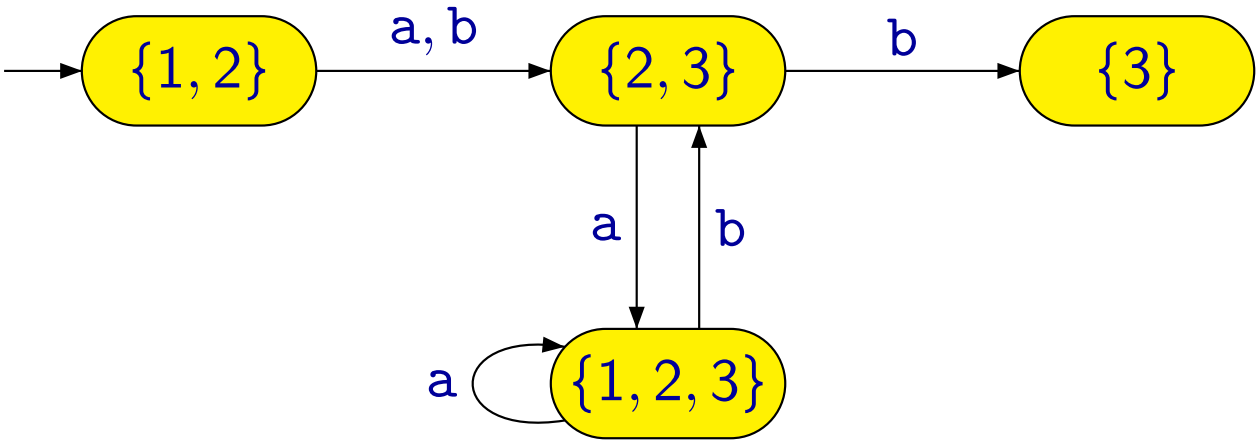


Převod NKA na DKA

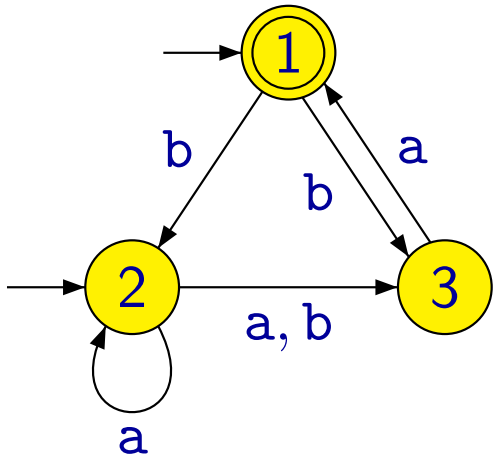


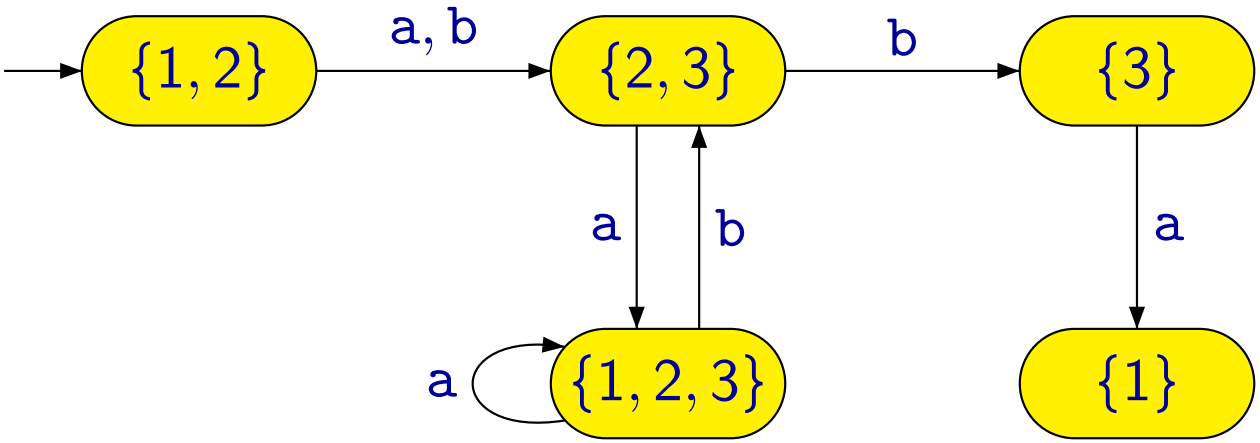
Převod NKA na DKA



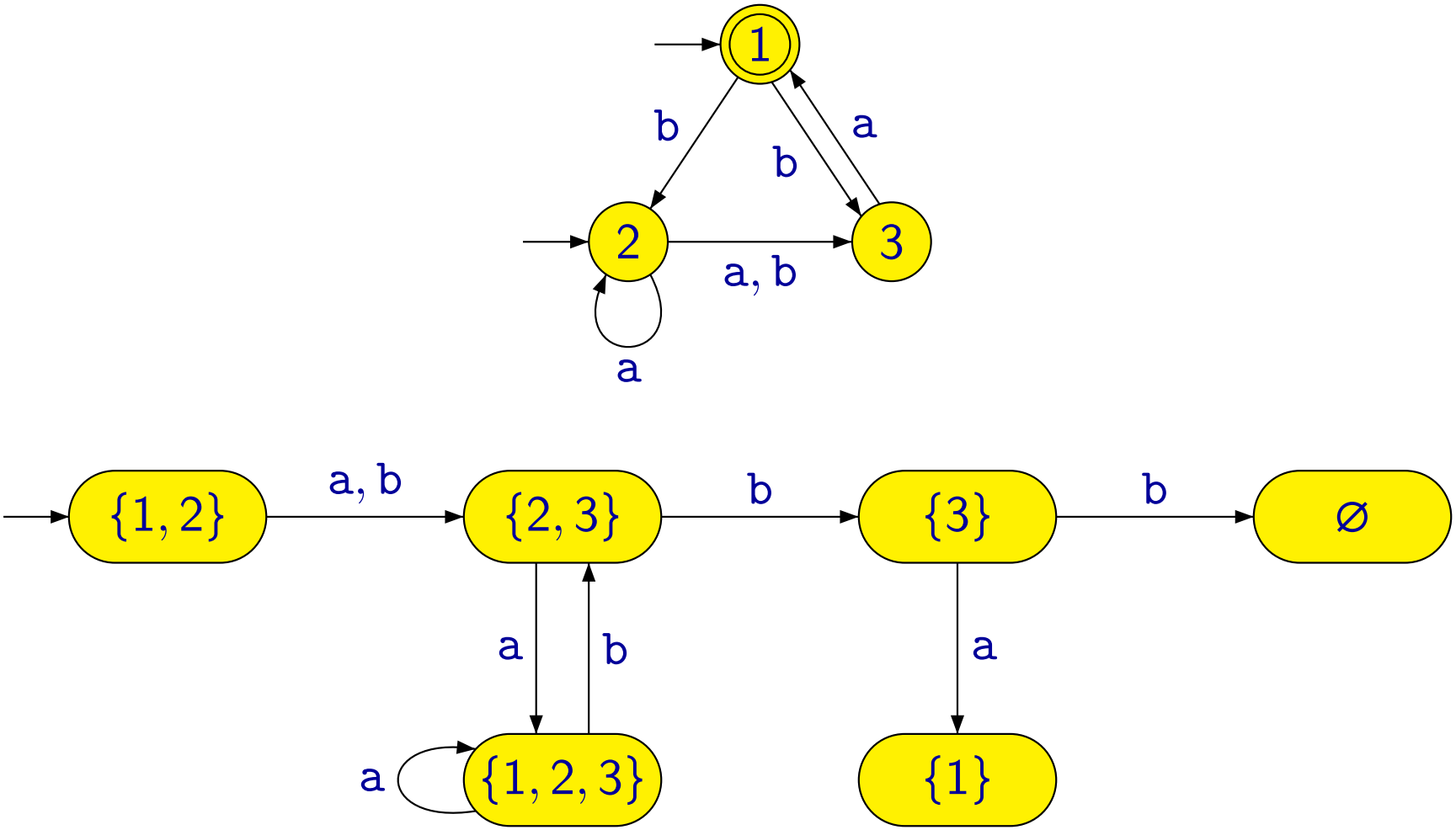


Převod NKA na DKA

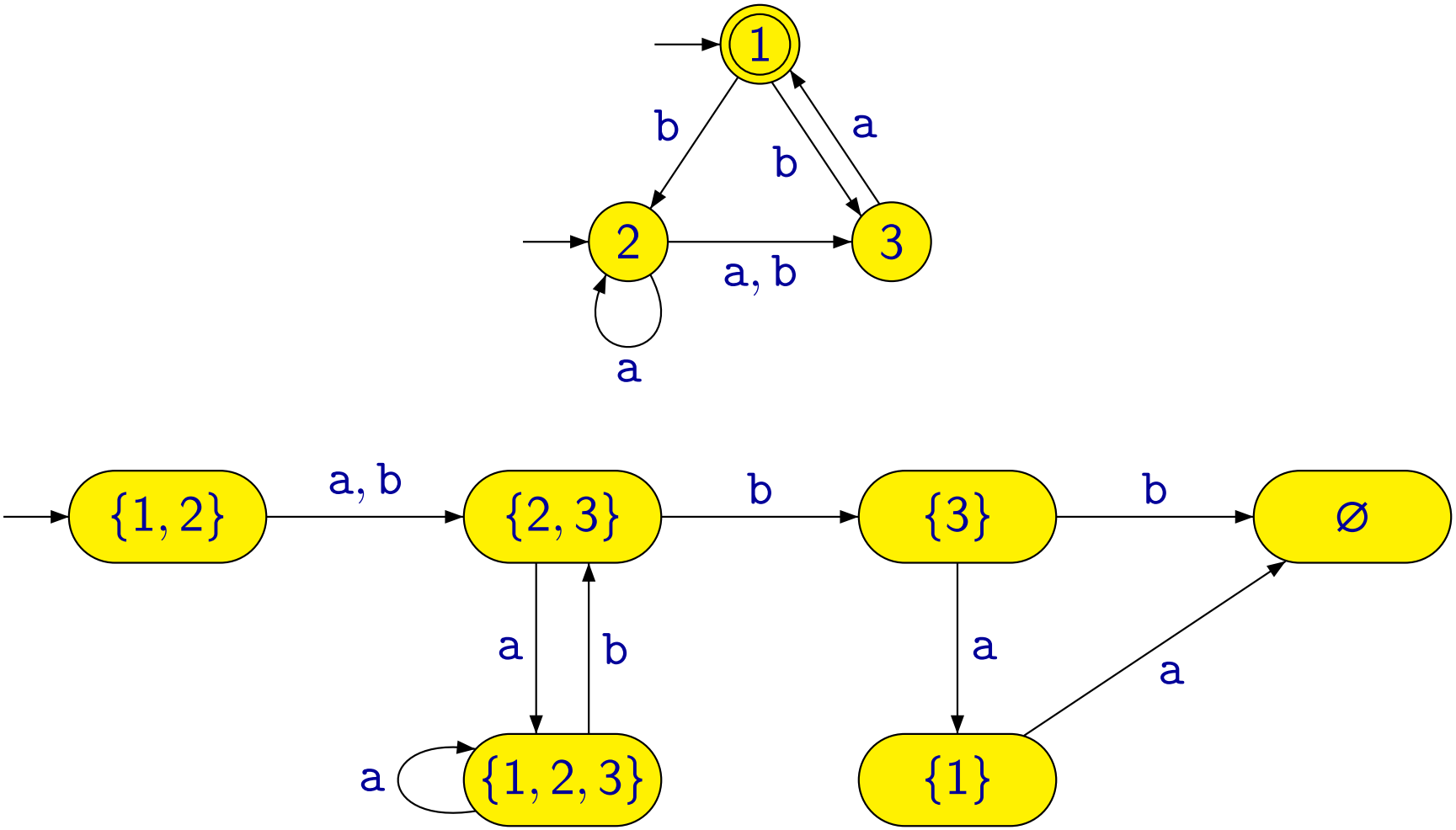




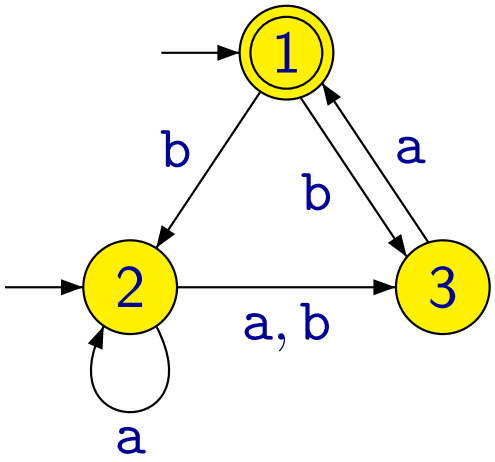
Převod NKA na DKA

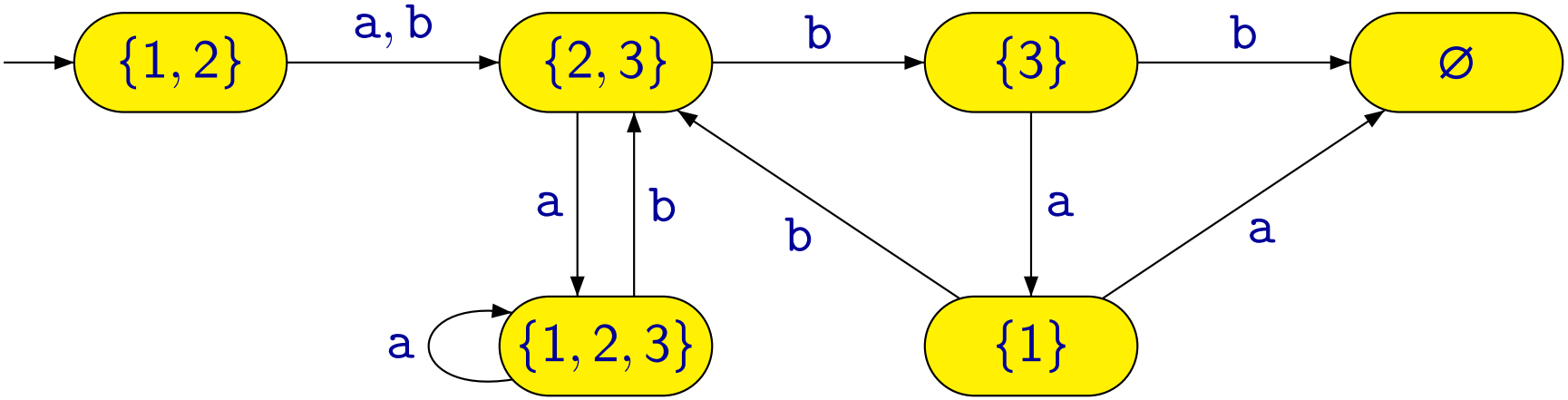


Převod NKA na DKA

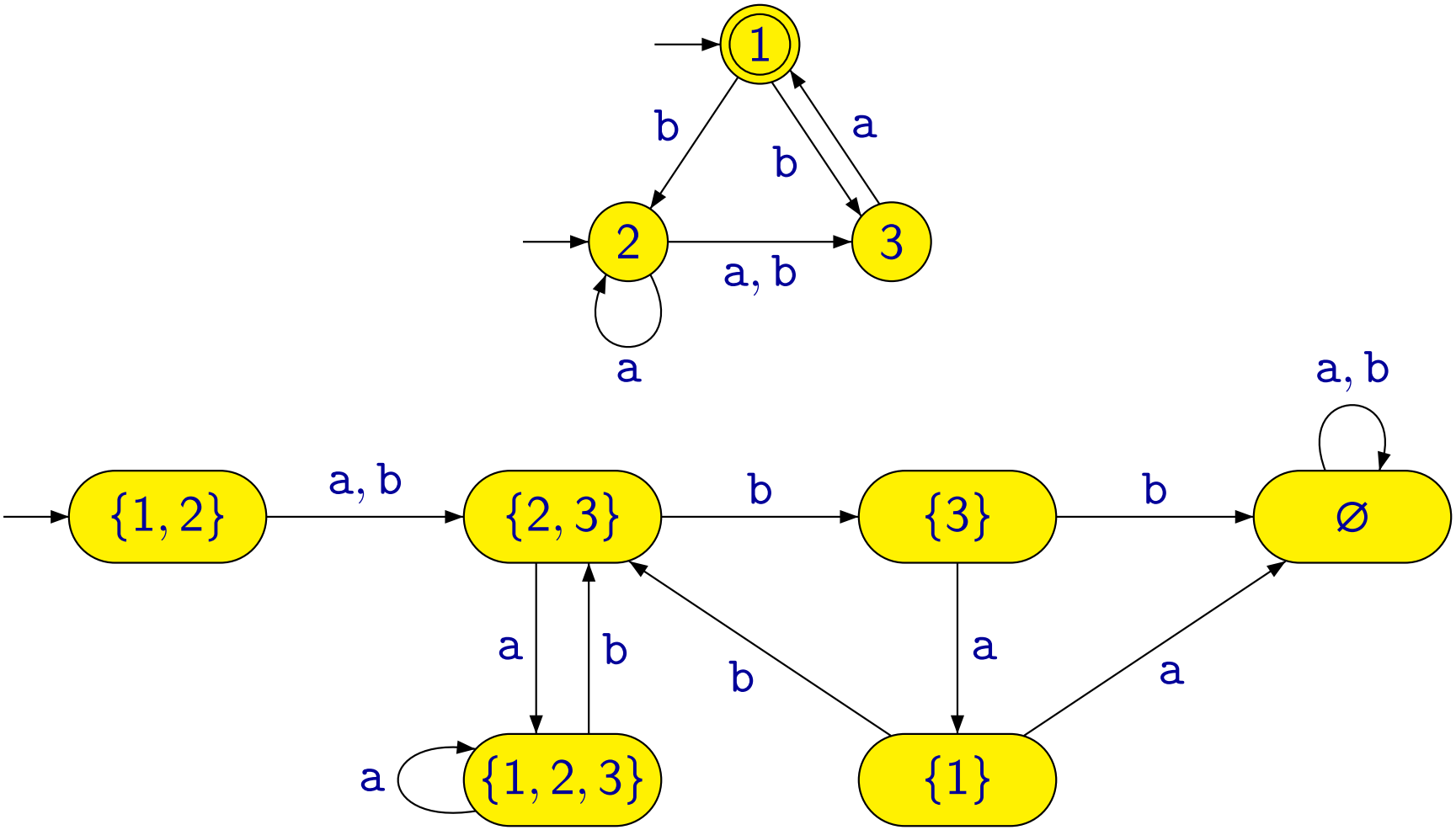


Převod NKA na DKA

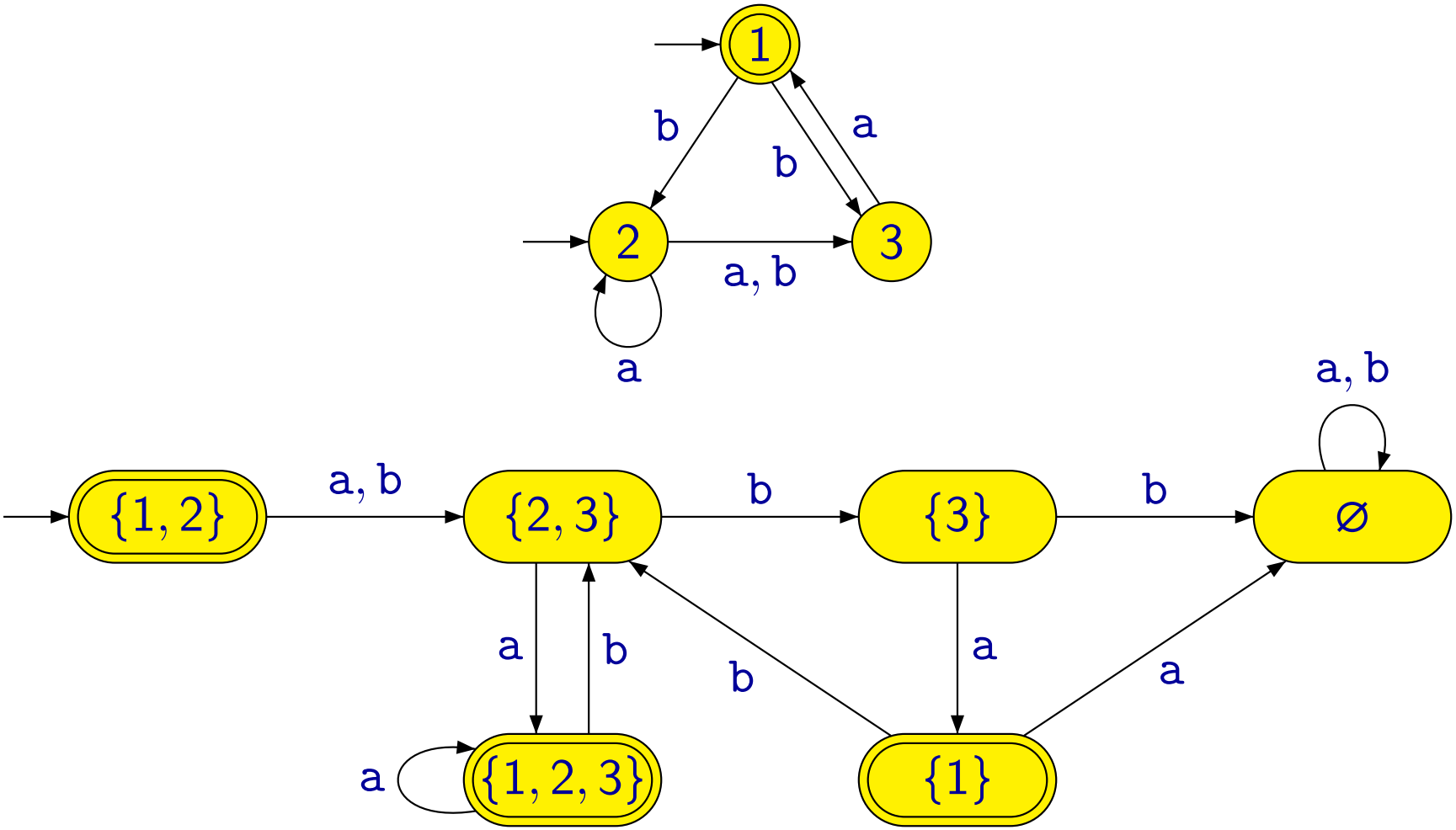




Převod NKA na DKA



Převod NKA na DKA



Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2, 3$
$\rightarrow 2$	$2, 3$	3
3	1	–

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$		

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	
$\{2,3\}$		

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$		

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	
$\leftarrow \{1,2,3\}$		

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$		

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$		
$\{3\}$		

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$	$\{1,2,3\}$	
$\{3\}$		

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\{3\}$		

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\{3\}$	$\{1\}$	

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\{3\}$	$\{1\}$	
$\leftarrow \{1\}$		

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\{3\}$	$\{1\}$	$\varnothing$
$\leftarrow \{1\}$		

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\{3\}$	$\{1\}$	$\varnothing$
$\leftarrow \{1\}$		
$\varnothing$		

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\{3\}$	$\{1\}$	$\varnothing$
$\leftarrow \{1\}$	$\varnothing$	
$\varnothing$		

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\{3\}$	$\{1\}$	$\varnothing$
$\leftarrow \{1\}$	$\varnothing$	$\{2,3\}$
$\varnothing$		

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\{3\}$	$\{1\}$	$\varnothing$
$\leftarrow \{1\}$	$\varnothing$	$\{2,3\}$
$\varnothing$	$\varnothing$	$\varnothing$

Převod NKA na DKA

	a	b
$\leftrightarrow 1$	–	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	–

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\{3\}$	$\{1\}$	$\varnothing$
$\leftarrow \{1\}$	$\varnothing$	$\{2,3\}$
$\varnothing$	$\varnothing$	$\varnothing$

	a	b
$\leftrightarrow 1$	2	2
2	3	4
$\leftarrow 3$	3	2
4	5	6
$\leftarrow 5$	6	2
6	6	6

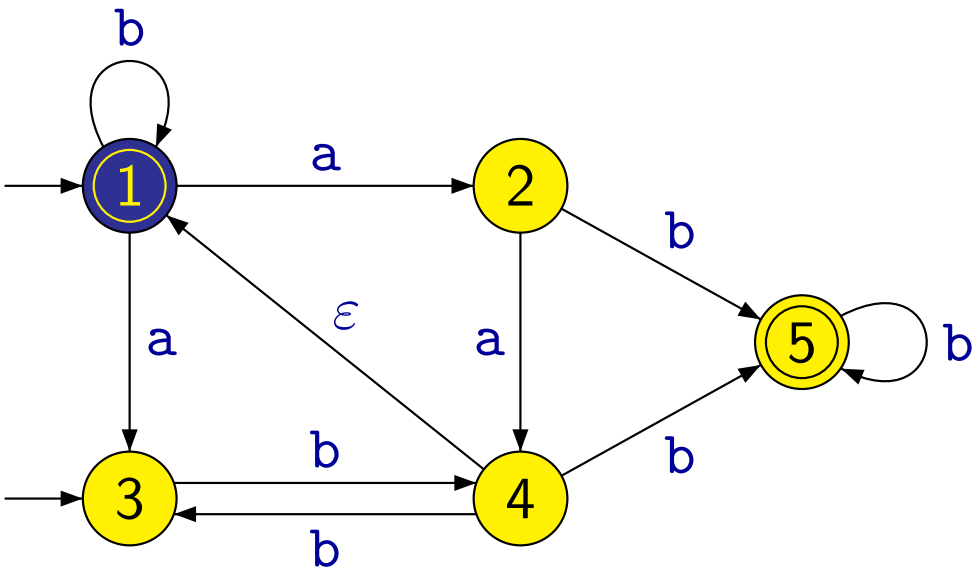
Převod NKA na DKA

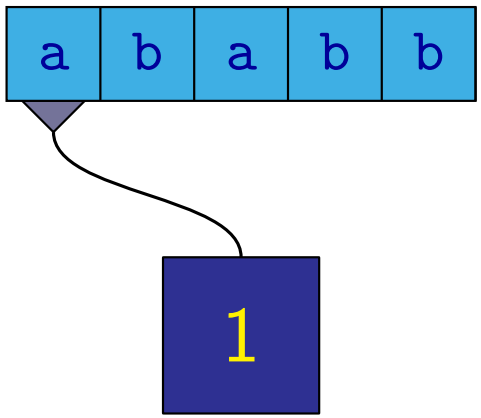
Poznámka: Při převodu nedeterministického automatu, který má n stavů, může mít výsledný deterministický automat až 2^n stavů.

Například při převodu automatu, který má 20 stavů, může vzniknout automat, který má $2^{20} = 1048576$ stavů.

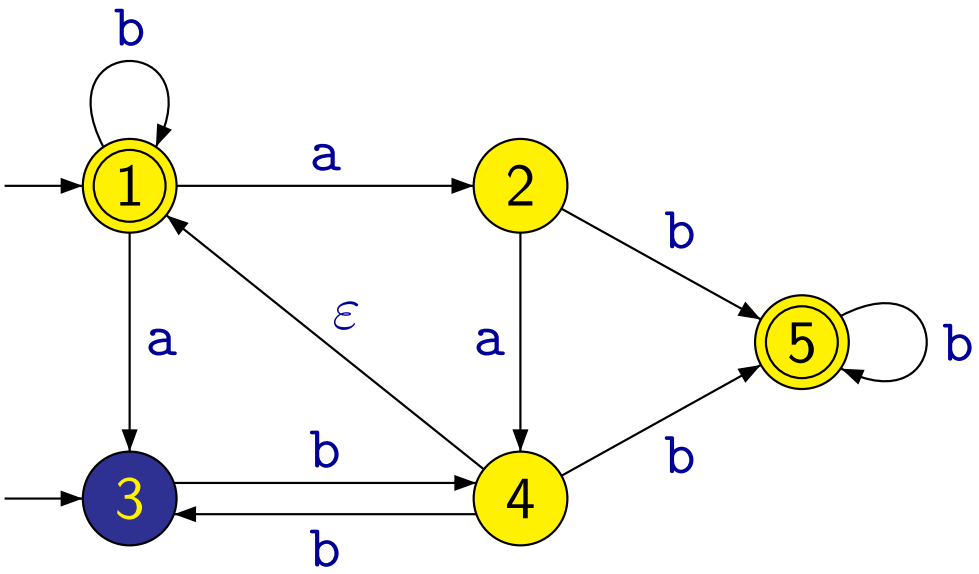
Často má sice výsledný automat podstatně méně než 2^n stavů, nicméně tyto nejhorší případy občas nastávají.

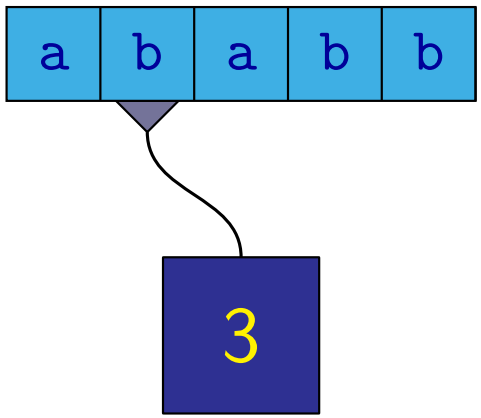
Zobecněný nedeterministický konečný automat





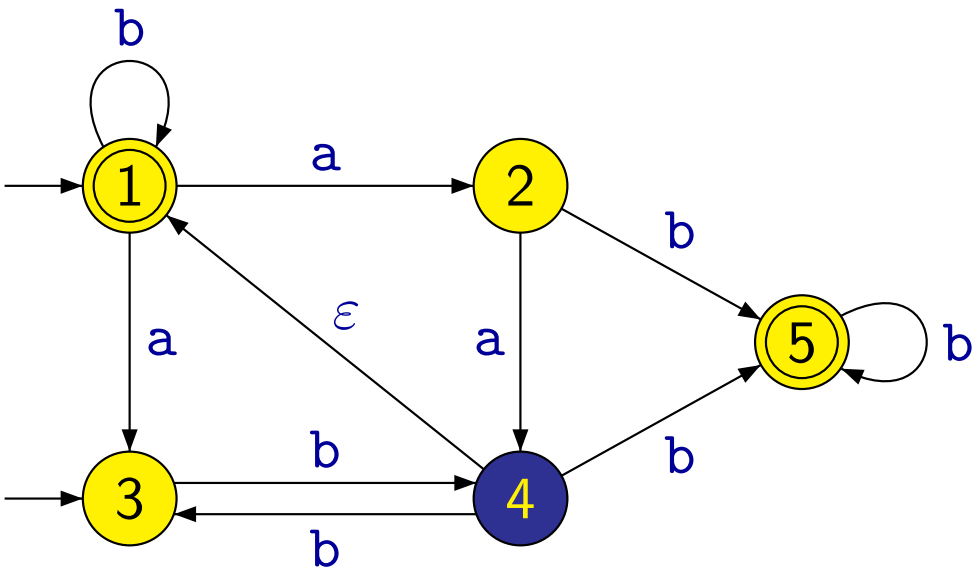
Zobecněný nedeterministický konečný automat

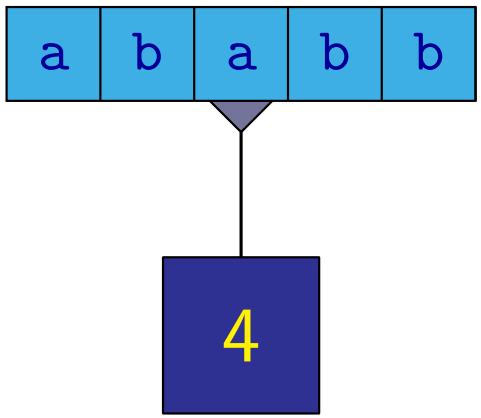




$1 \xrightarrow{a} 3$

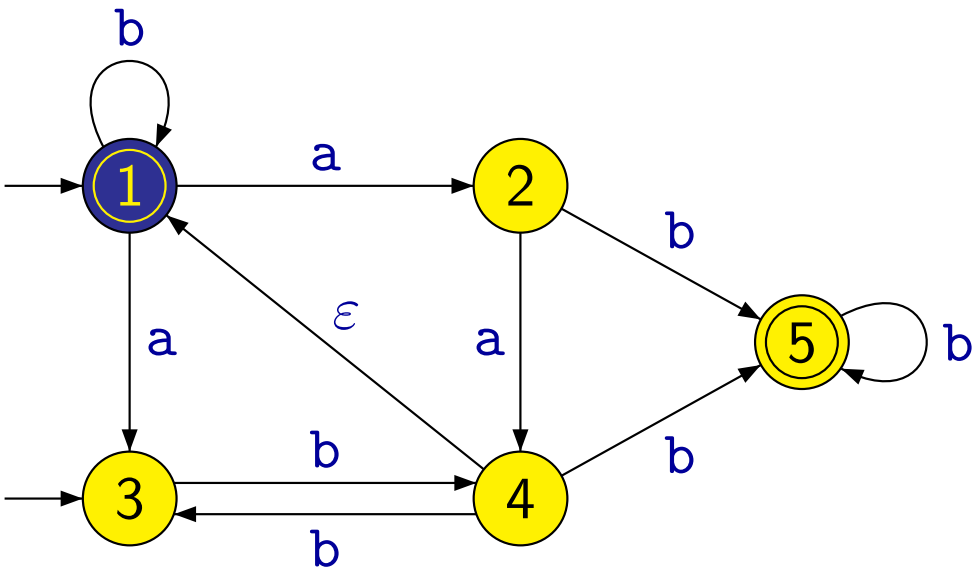
Zobecněný nedeterministický konečný automat

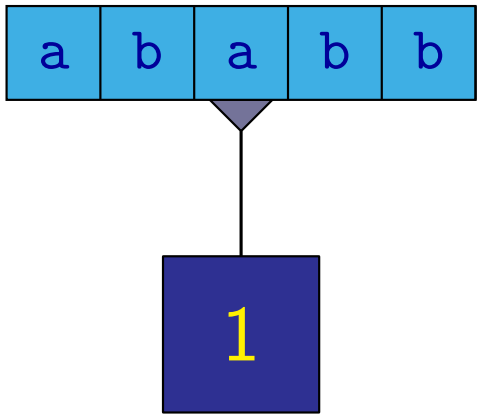




$$1 \xrightarrow{a} 3 \xrightarrow{b} 4$$

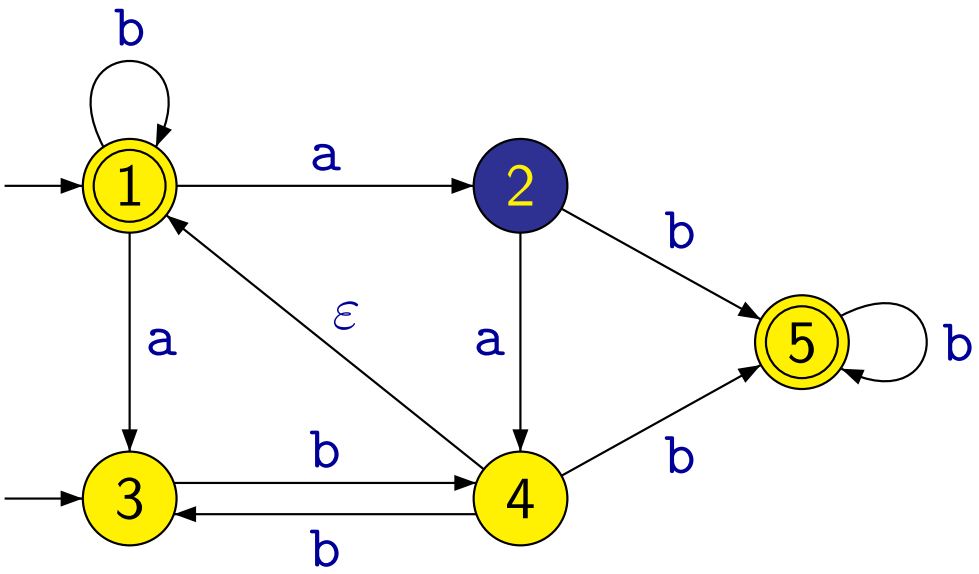
Zobecněný nedeterministický konečný automat

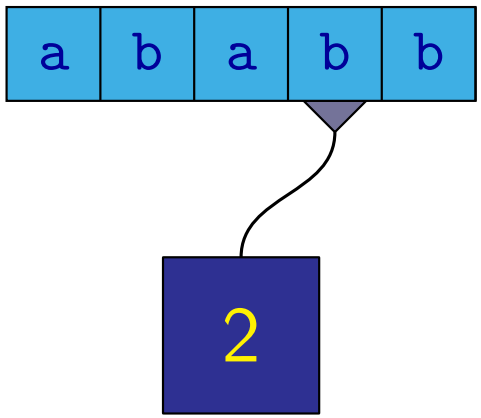




$$1 \xrightarrow{a} 3 \xrightarrow{b} 4 \xrightarrow{\varepsilon} 1$$

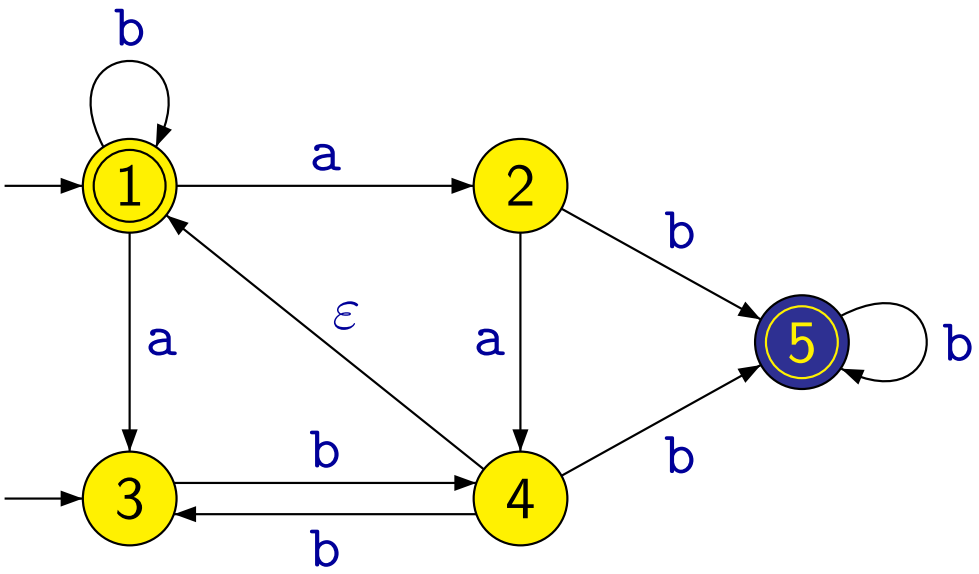
Zobecněný nedeterministický konečný automat

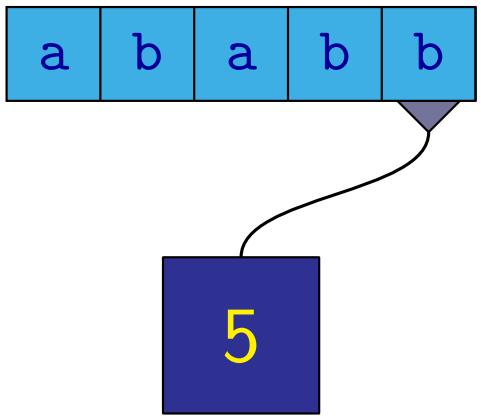




$$1 \xrightarrow{a} 3 \xrightarrow{b} 4 \xrightarrow{\varepsilon} 1 \xrightarrow{a} 2$$

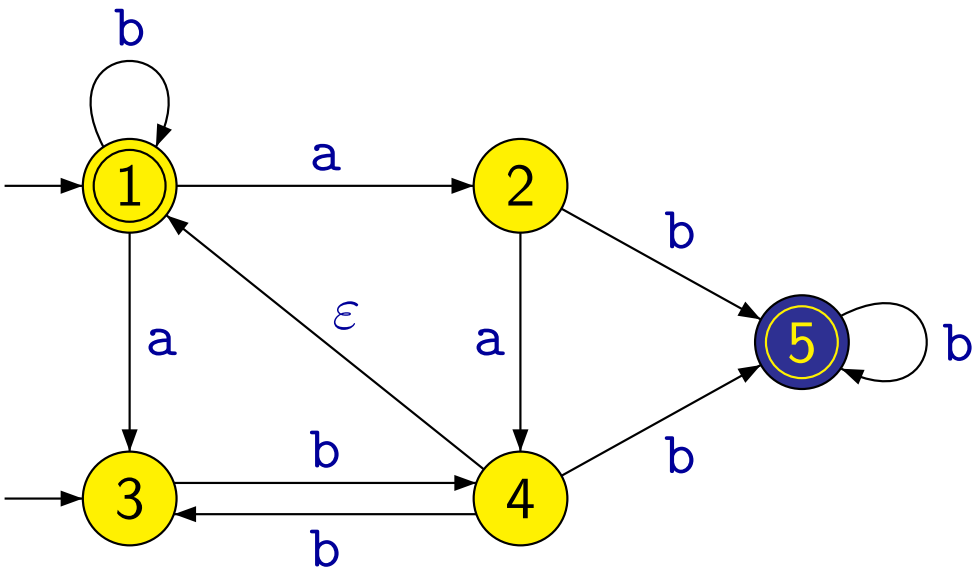
Zobecněný nedeterministický konečný automat

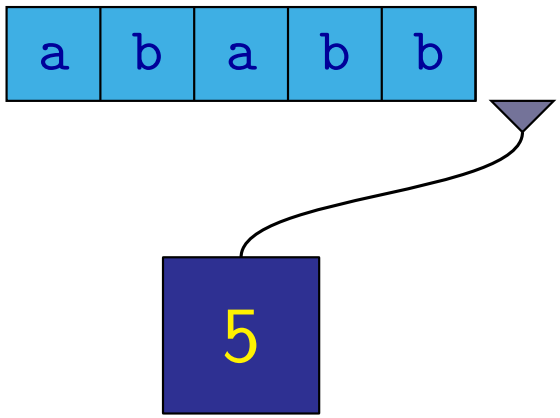




$$1 \xrightarrow{a} 3 \xrightarrow{b} 4 \xrightarrow{\varepsilon} 1 \xrightarrow{a} 2 \xrightarrow{b} 5$$

Zobecněný nedeterministický konečný automat





$$1 \xrightarrow{a} 3 \xrightarrow{b} 4 \xrightarrow{\varepsilon} 1 \xrightarrow{a} 2 \xrightarrow{b} 5 \xrightarrow{b} 5$$

Zobecněný nedeterministický konečný automat

Oproti nedeterministickému konečnému automatu má **zobecněný nedeterministický konečný automat** tzv. ε-**přechody**, tj. přechody označené symbolem ε.

Při provádění ε-přechodu se mění pouze stav řídící jednotky, ale hlava na pásce se neposouvá.

Poznámka: Výpočty zobecněného nedeterministického automatu mohou být libovolně dlouhé a dokonce i nekonečné (pokud graf obsahuje cyklus tvořený ε-přechody) bez ohledu na délku slova na pásce.

Zobecněný nedeterministický konečný automat

Formálně je **zobecněný nedeterministický konečný automat** (**ZNKA**) definován jako pětice

$$(Q, \Sigma, \delta, I, F)$$

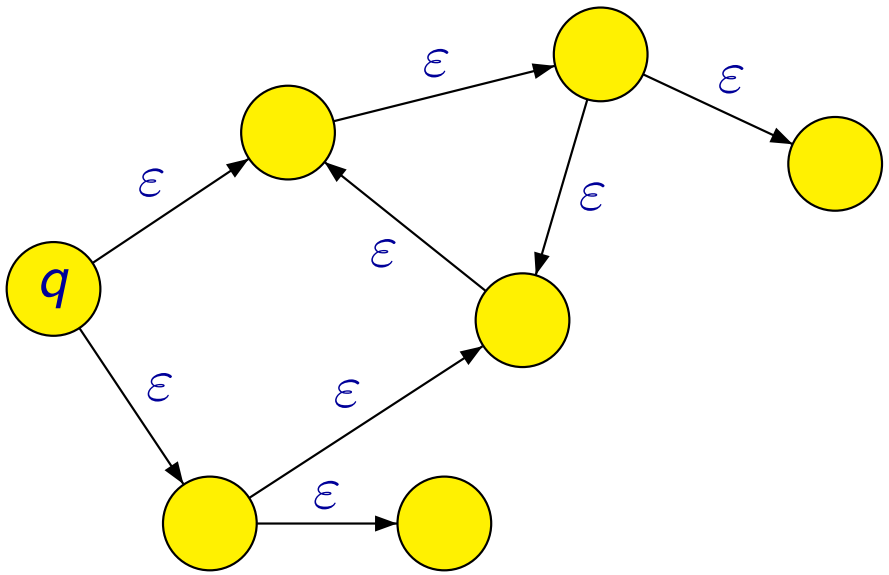
kde:

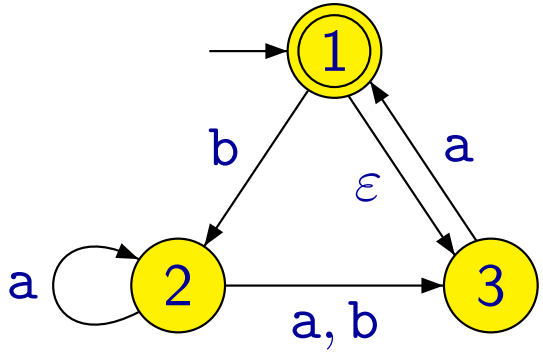
- Q je konečná množina **stavů**
- Σ je konečná **abeceda**
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$ je **přechodová funkce**
- $I \subseteq Q$ je množina **počátečních stavů**
- $F \subseteq Q$ je množina **přijímajících stavů**

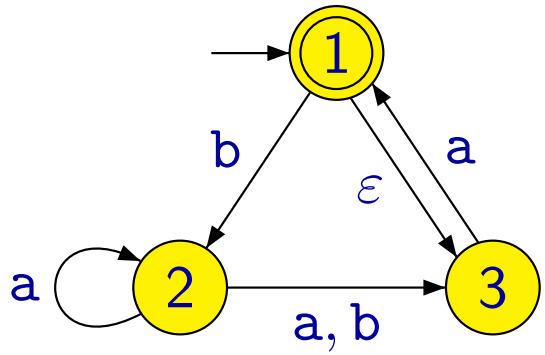
Poznámka: Na NKA můžeme nahlížet jako na speciální případ ZNKA, kde $\delta(q, \varepsilon) = \varnothing$ pro všechna $q \in Q$.

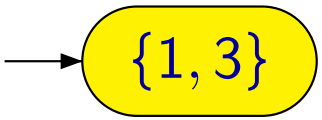
Převod na deterministický konečný automat

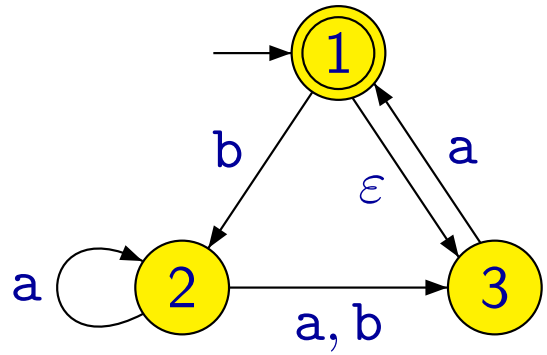
Zobecněný nedeterministický konečný automat je možné převést na deterministický podobnou konstrukcí jako nedeterministický konečný automat, s tím rozdílem, že do množin stavů musíme vždy přidat navíc i všechny stavy dosažitelné z již přidaných stavů nějakou sekvencí ε-přechodů.

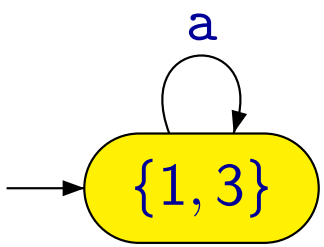


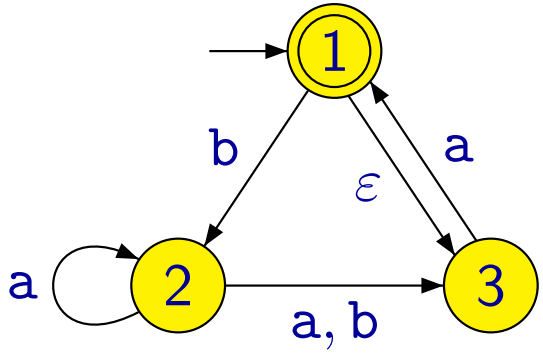

1
2
3
b
ε
a
a
a, b

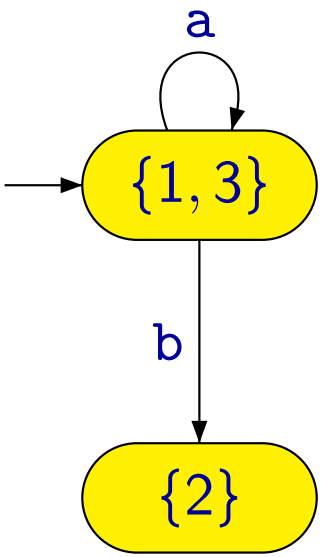

1
b
a
ε
a
2
3
a, b

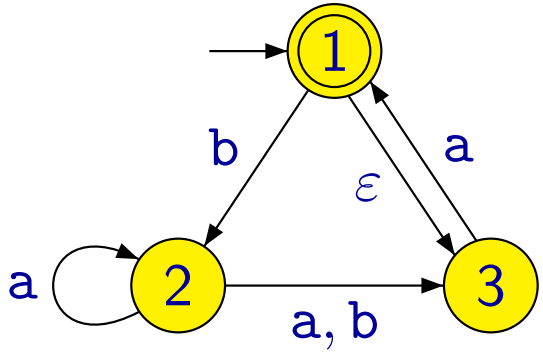

{1, 3}

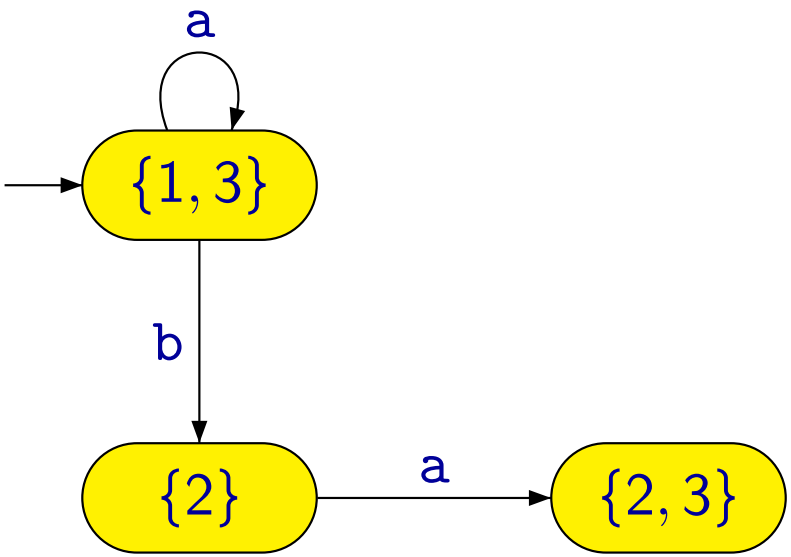

1
b
ε
a
a
2
3
a, b

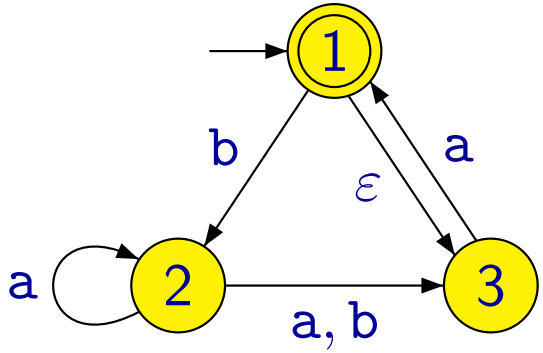

a
{1, 3}

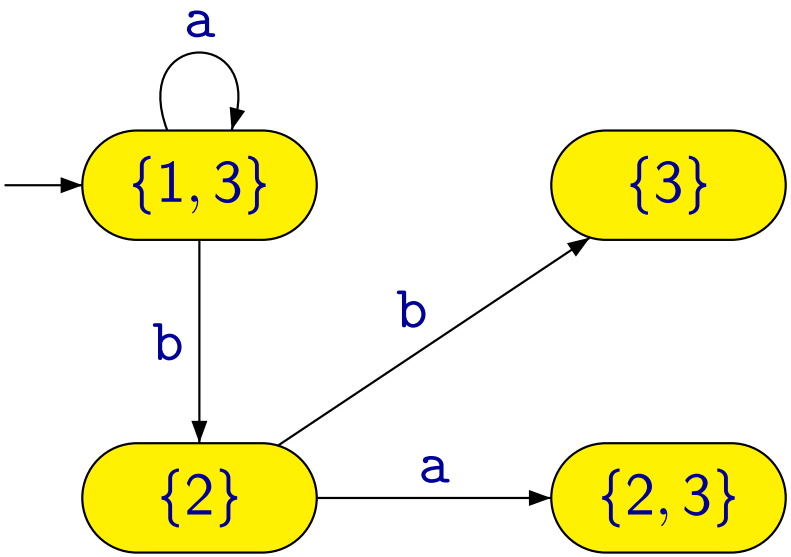

1
b
ε
a
a
2
3
a, b

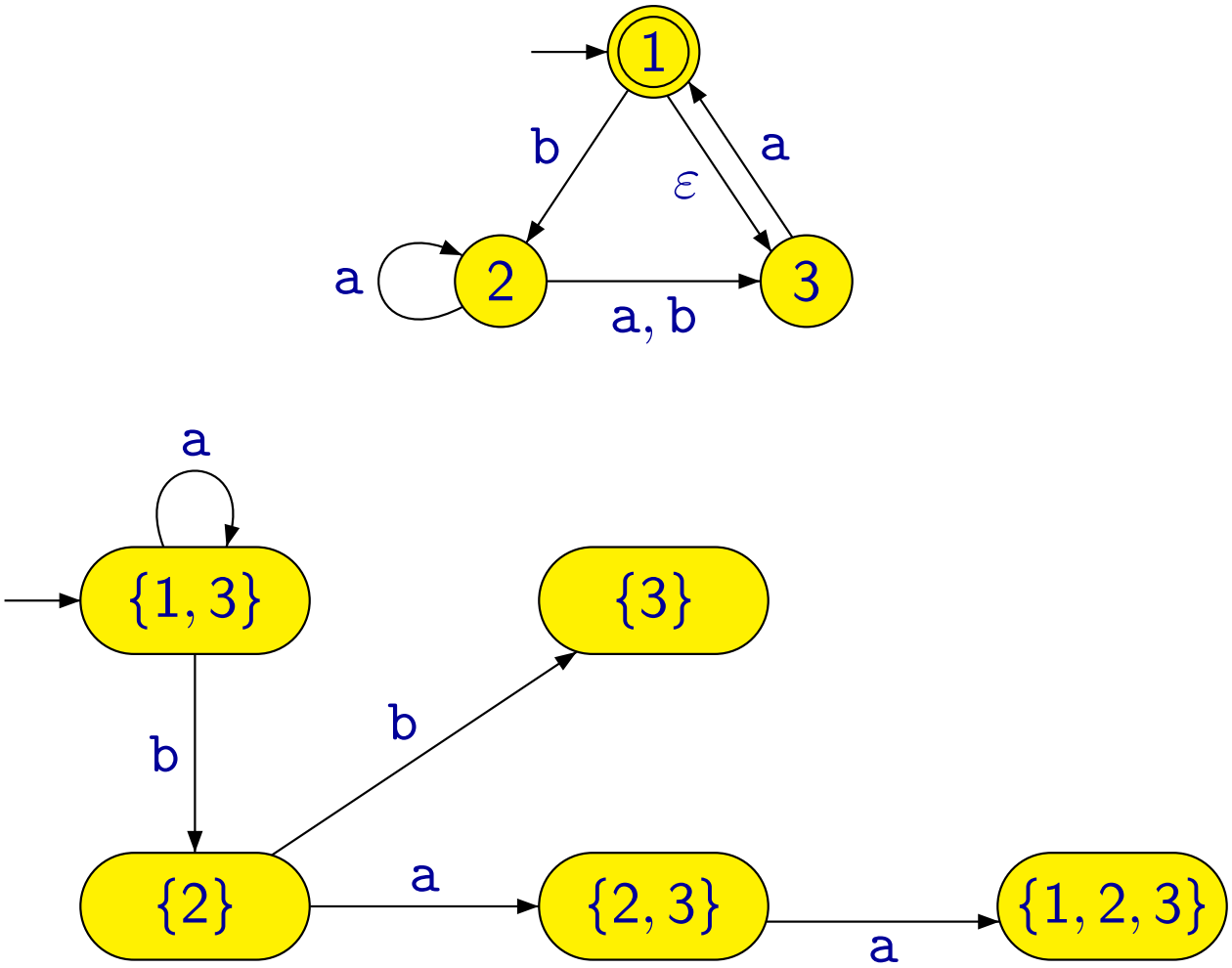

a
{1, 3}
b
{2}

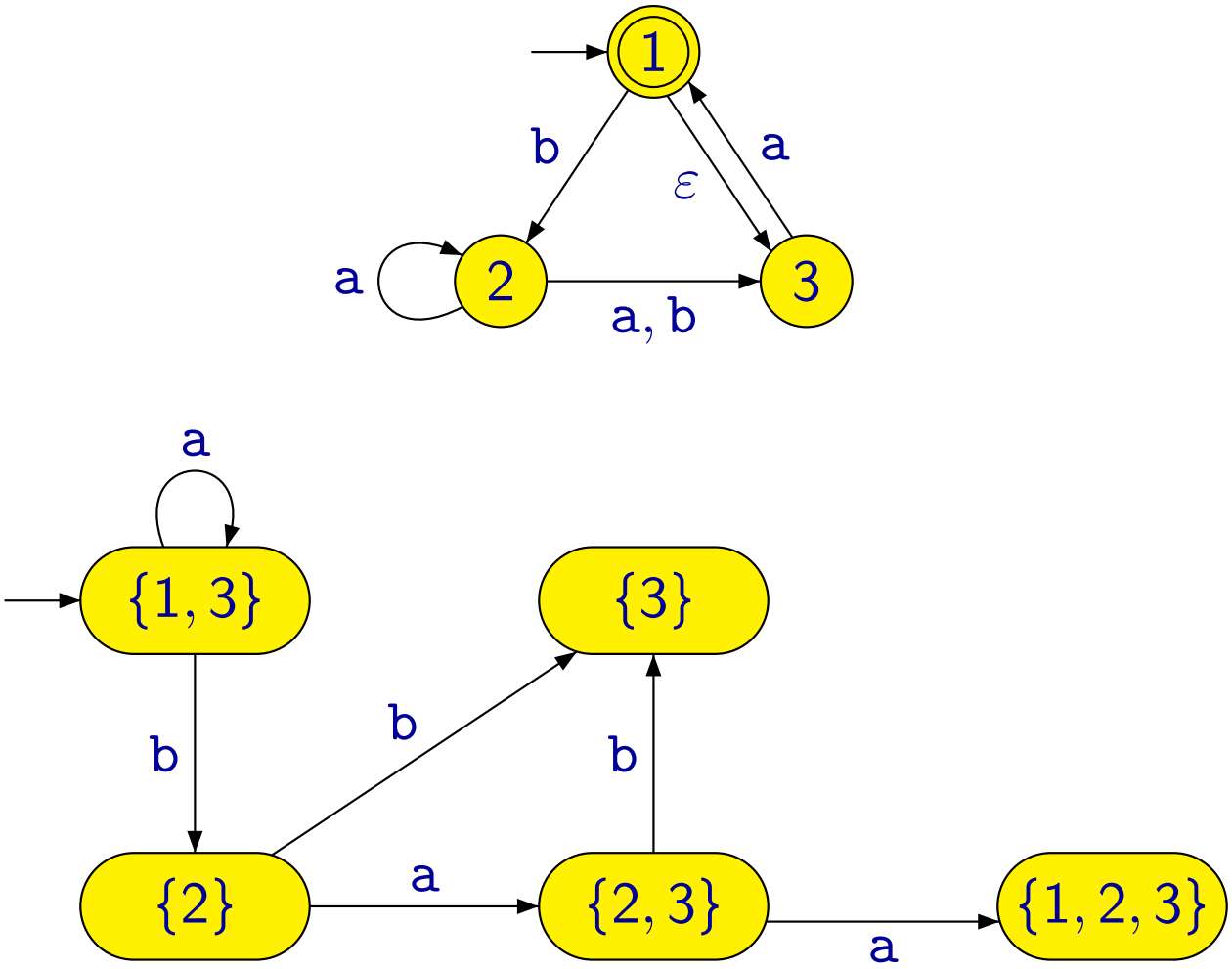

1
b
ε
a
a
2
3
a, b

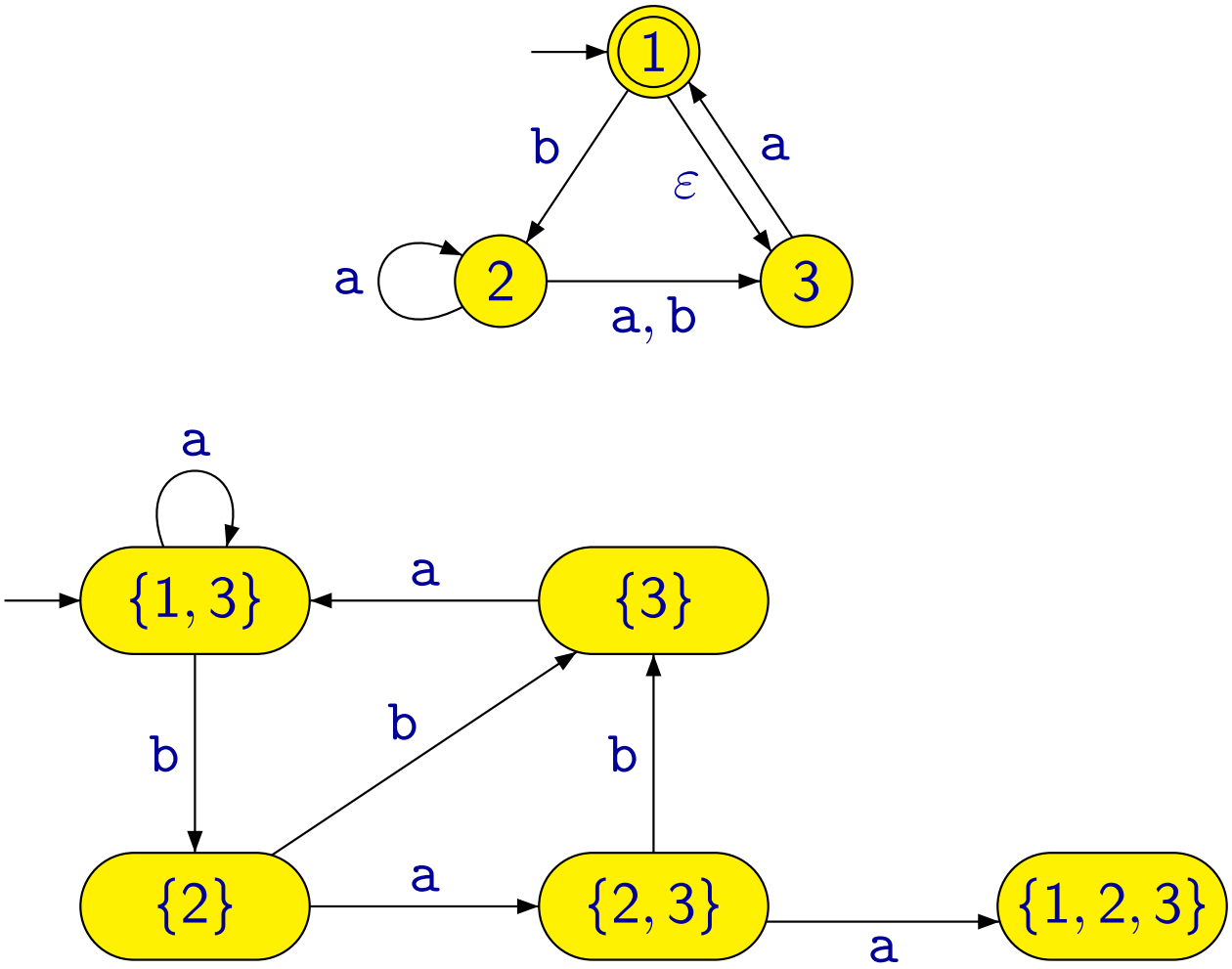

a
{1, 3}
b
{2}
a
{2, 3}

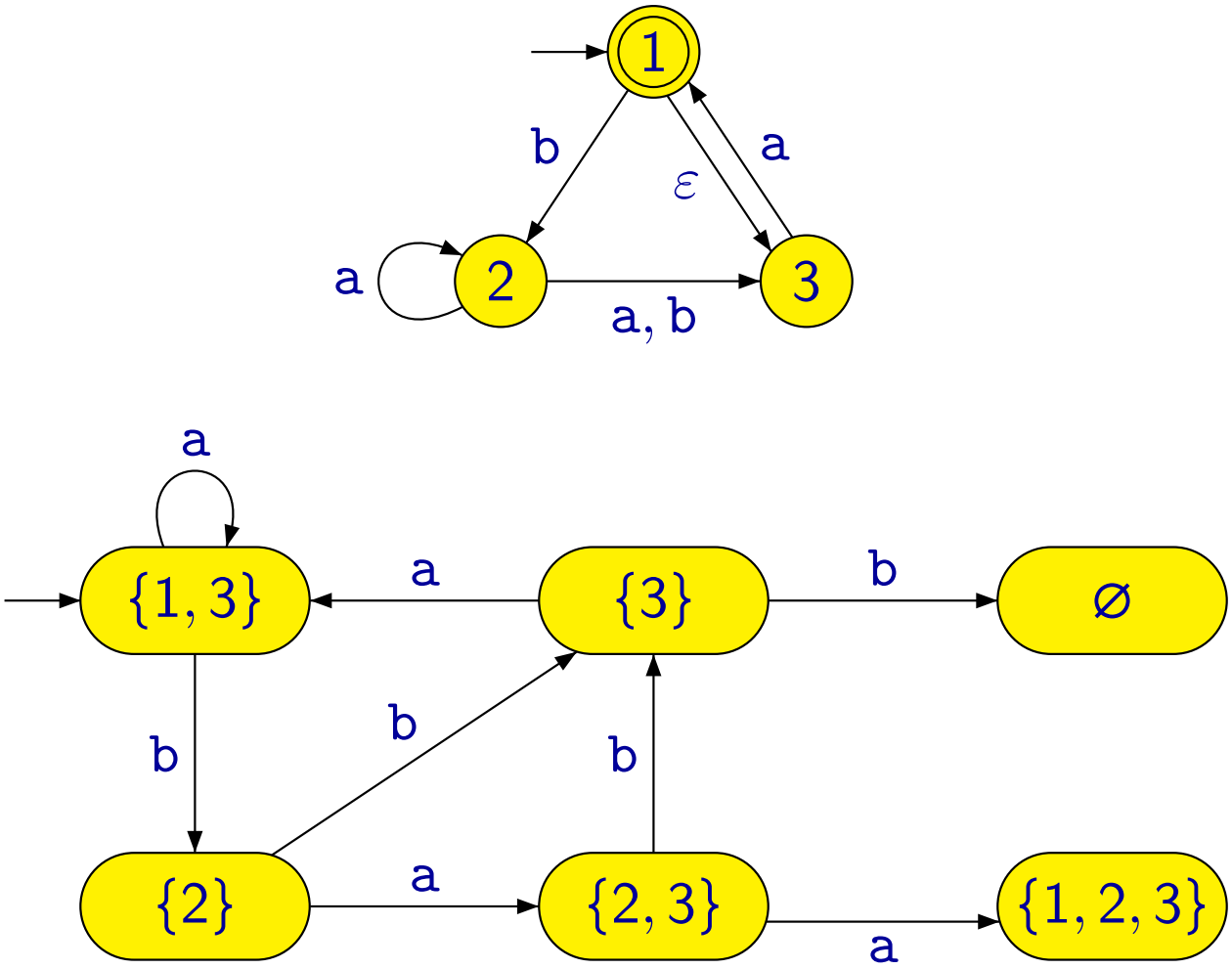

1
2
3
b
ε
a
a
a, b

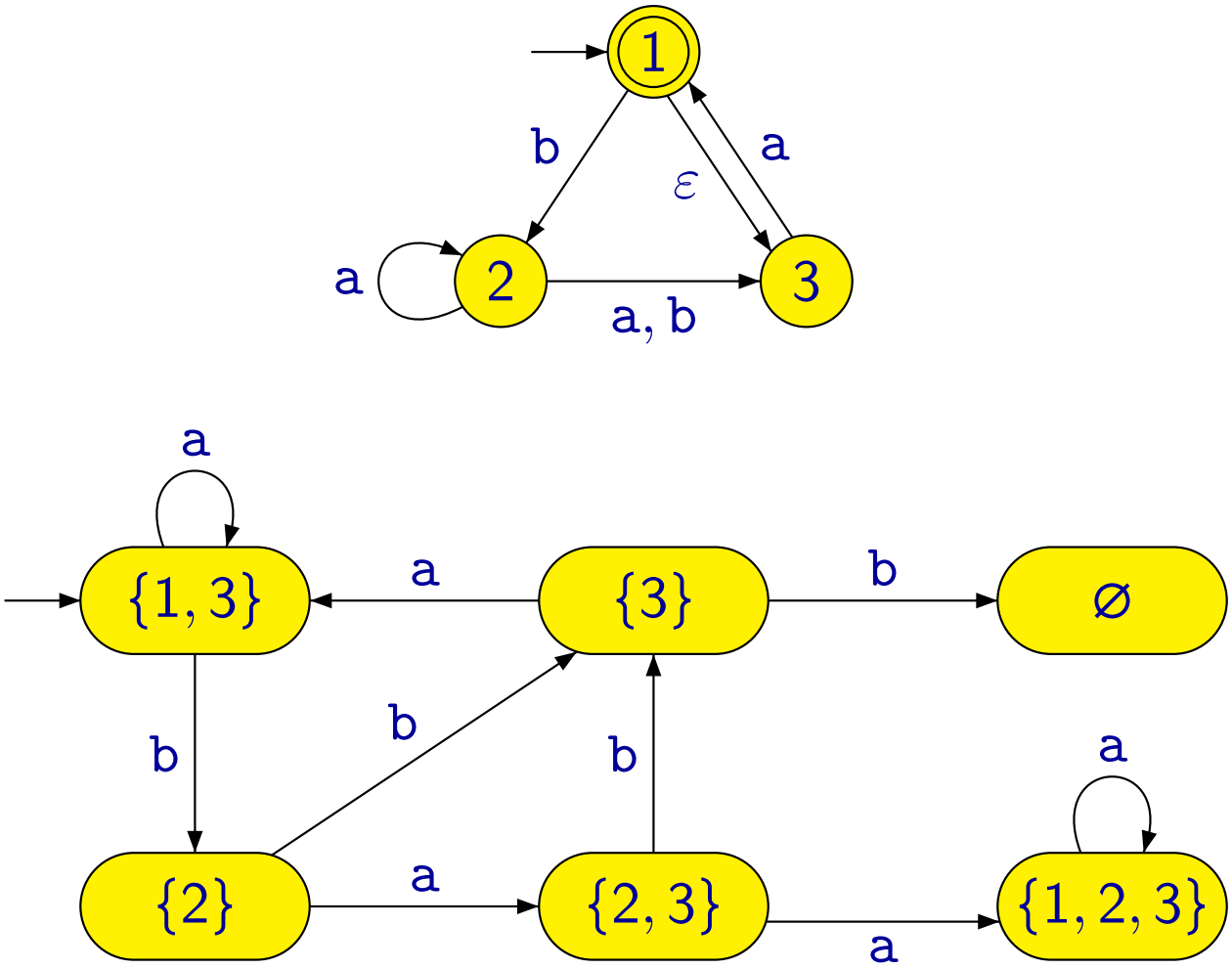

a
{1, 3}
{3}
b
b
{2}
a
{2, 3}

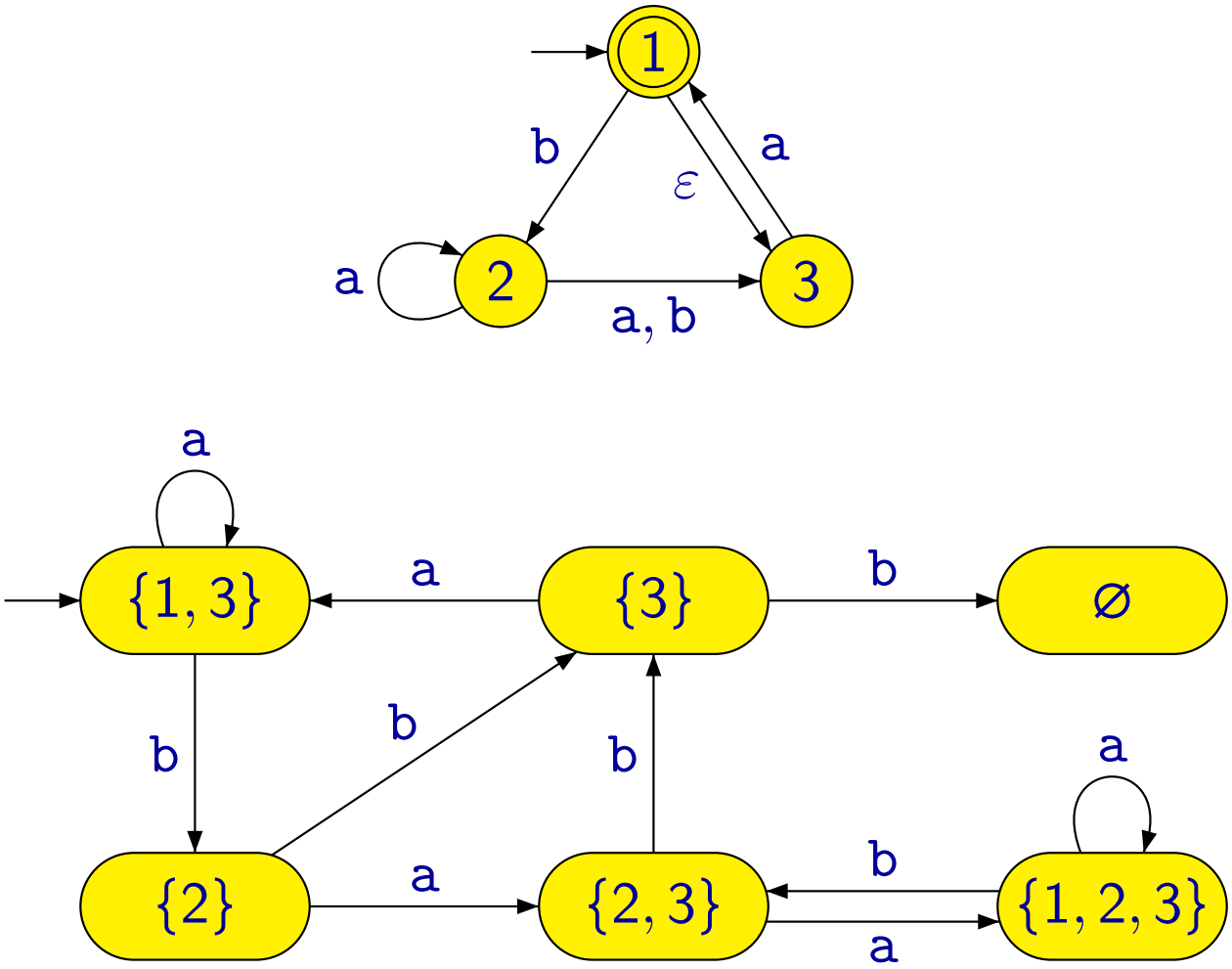

1
2
3
b
ε
a
a
a, b
a
{1, 3}
{3}
b
b
{2}
a
{2, 3}
a
{1, 2, 3}

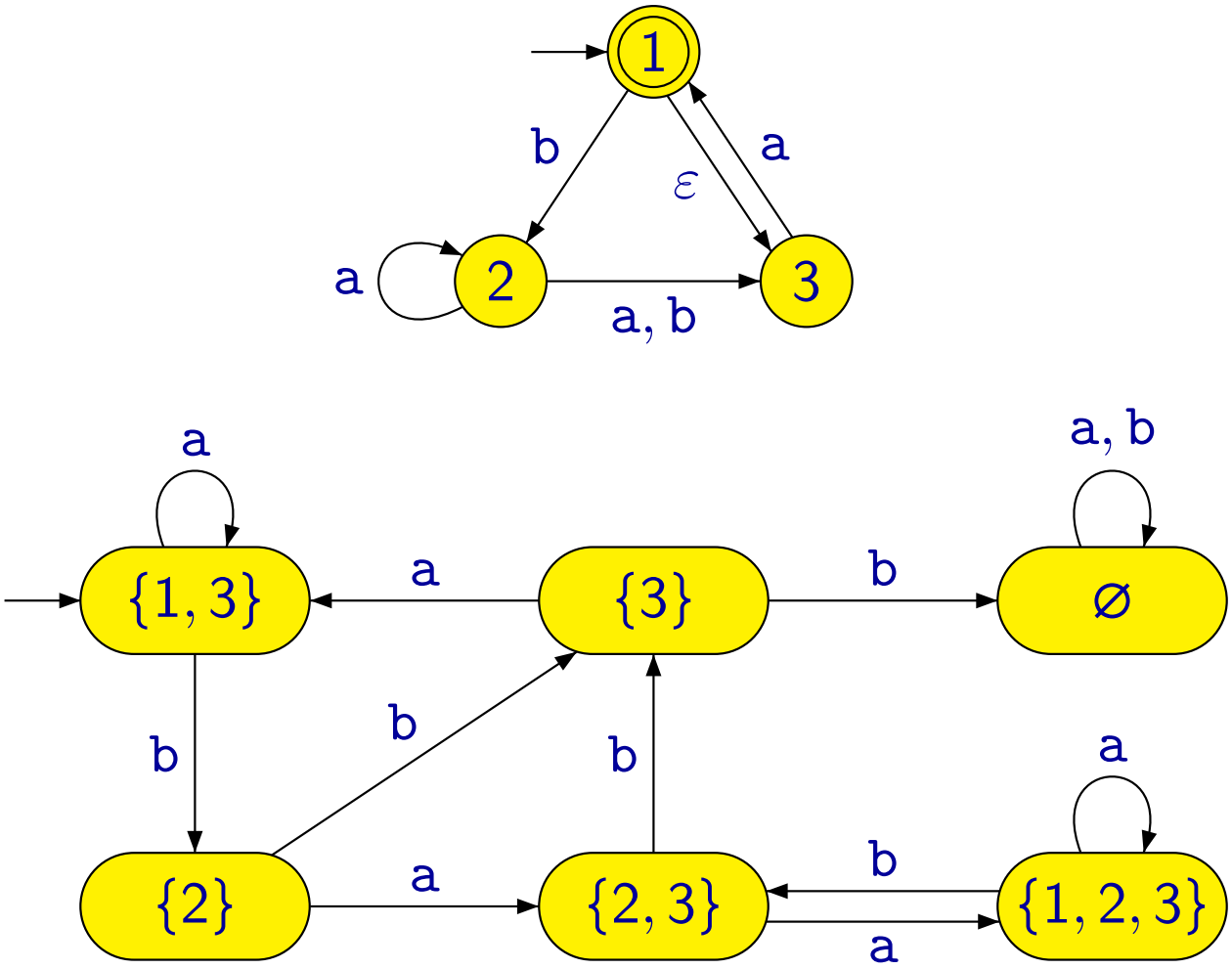

1
2
3
b
a
ε
a
a, b
a
{1, 3}
{3}
b
b
b
{2}
a
{2, 3}
a
{1, 2, 3}

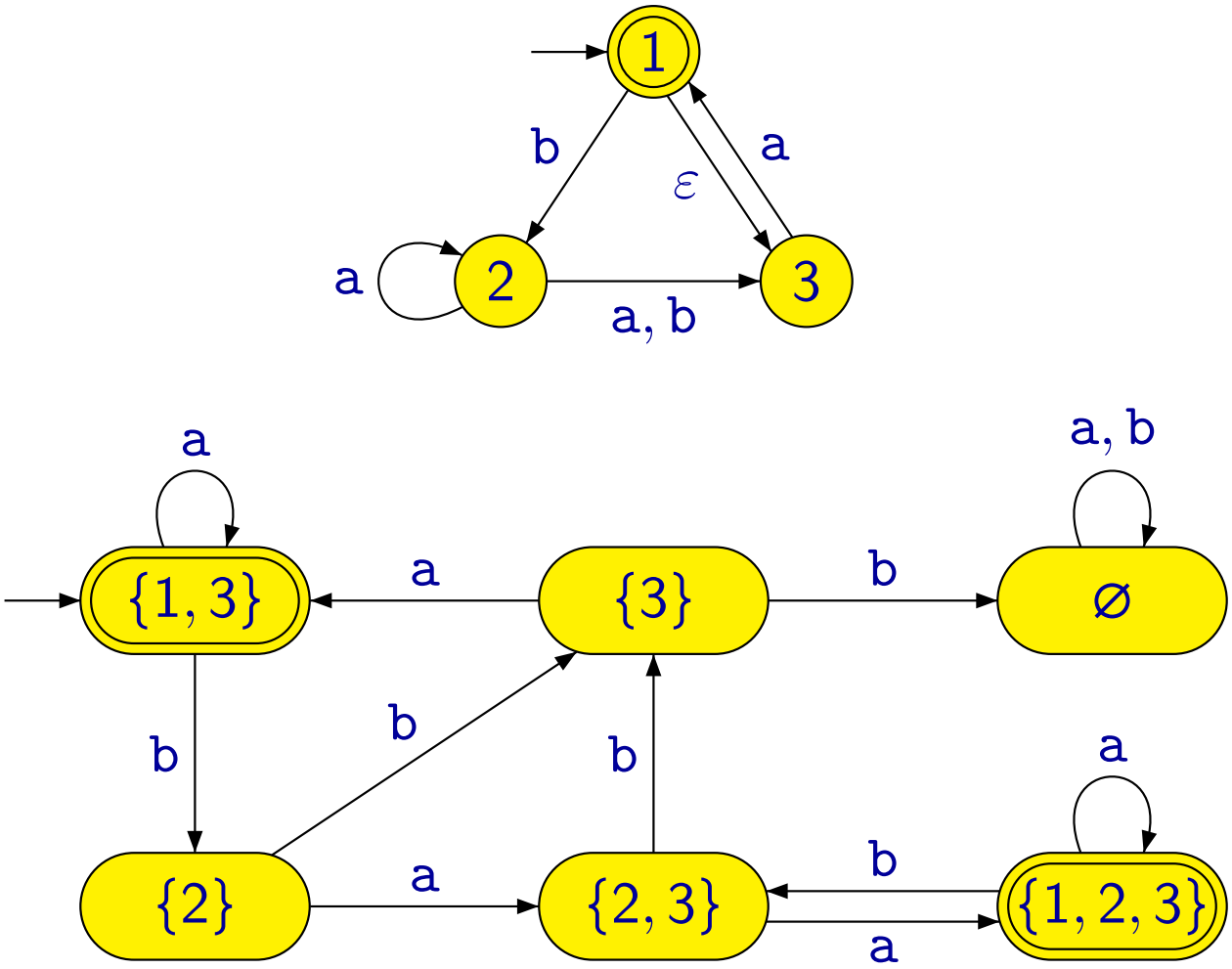

1
2
3
b
a
ε
a
a, b
a
{1, 3}
{3}
{2}
{2, 3}
{1, 2, 3}
a
b
b
b
a
a


1
2
3
b
a
ε
a
a, b
a
{1, 3}
a
{3}
b
∅
b
b
b
{2}
a
{2, 3}
a
{1, 2, 3}


1
2
3
b
a
ε
a
a, b
a
{1, 3}
a
{3}
b
∅
b
b
b
a
{2}
a
{2, 3}
a
{1, 2, 3}


1
b
a
ε
a
2
3
a, b
a
{1, 3}
a
{3}
b
∅
b
b
b
a
{2}
a
{2, 3}
b
{1, 2, 3}
a


1
b
a
ε
a
2
3
a
a, b
a
a, b
{1, 3}
a
{3}
b
∅
b
b
b
a
{2}
a
{2, 3}
b
{1, 2, 3}
a


1
b
a
ε
a
2
3
a
a, b
a
a, b
{1, 3}
a
{3}
b
∅
b
b
b
a
{2}
a
{2, 3}
b
{1, 2, 3}
a

Převod ZNKA na DKA

Předtím, než formálně popíšeme převod ZNKA na DKA, zaveďme si několik pomocných definic.

Předpokládejme nějaký daný ZNKA $\mathcal{A} = (Q, \Sigma, \delta, I, F)$.

Definujme funkci $\hat{\delta} : \mathcal{P}(Q) \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$ tak, že pro $K \subseteq Q$ a $a \in \Sigma \cup \{\varepsilon\}$ je

$$\hat{\delta}(K, a) = \bigcup_{q \in K} \delta(q, a)$$

Převod ZNKA na DKA

Pro $K \subseteq Q$ označme $Cl_\varepsilon(K)$ množinu všech stavů dosažitelných ze stavů z množiny K nějakou libovolnou sekvencí ε-přechodů.

To znamená, že funkce $Cl_\varepsilon : \mathcal{P}(Q) \to \mathcal{P}(Q)$ je definována tak, že pro $K \subseteq Q$ je $Cl_\varepsilon(K)$ nejmenší (vzledem k inkluzi) množina splňující následující dvě podmínky:

- $K \subseteq Cl_\varepsilon(K)$
- Pro každé $q \in Cl_\varepsilon(K)$ platí, že $\delta(q, \varepsilon) \subseteq Cl_\varepsilon(K)$.

Poznámka: Všimněme si, že pro libovolné K je $Cl_\varepsilon(Cl_\varepsilon(K)) = Cl_\varepsilon(K)$.

Všimněme si také, že v případě NKA (kde $\delta(q, \varepsilon) = \varnothing$ pro každé $q \in Q$) je $Cl_\varepsilon(K) = K$.

Převod ZNKA na DKA

K danému ZNKA $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ nyní můžeme sestrojit DKA $\mathcal{A}' = (Q', \Sigma, \delta', q_0', F')$, kde:

- $Q' = \mathcal{P}(Q)$ ($K \in Q'$ tedy znamená, že $K \subseteq Q$)
- $\delta' : Q' \times \Sigma \to Q'$ je definová tak, že pro $K \in Q'$ a $a \in \Sigma$ je

$$\delta'(K, a) = Cl_\varepsilon(\hat{\delta}(Cl_\varepsilon(K), a))$$

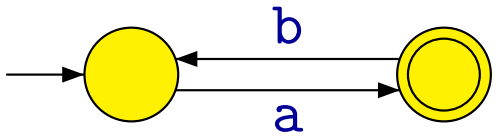
- $q_0' = Cl_\varepsilon(I)$
- $F' = \{K \in Q' \mid Cl_\varepsilon(K) \cap F \neq \varnothing\}$

Není težké ověřit, že $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.

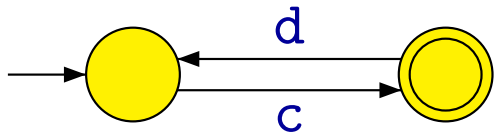
Zřetězení jazyků

$\Sigma = \{a, b, c, d\}$

$\mathcal{A}_1$:



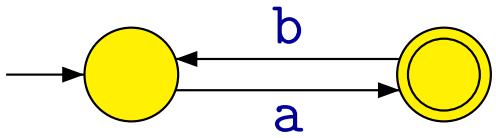
$\mathcal{A}_2$:



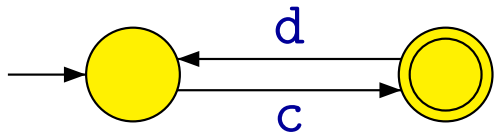
Zřetězení jazyků

$\Sigma = \{a, b, c, d\}$

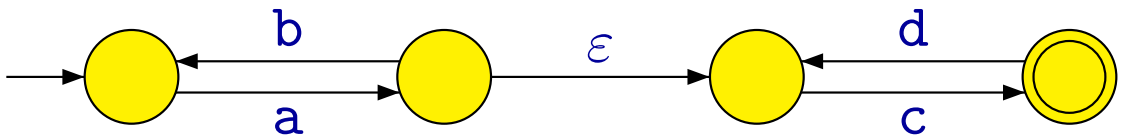
$\mathcal{A}_1$:



$\mathcal{A}_2$:



$\mathcal{A}$:

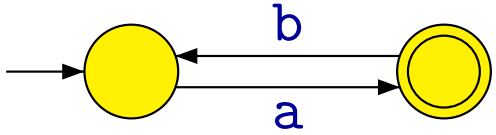


$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cdot \mathcal{L}(\mathcal{A}_2)$

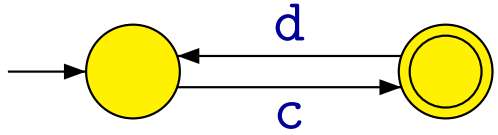
Zřetězení jazyků

$\Sigma = \{a, b, c, d\}$

$\mathcal{A}_1$:

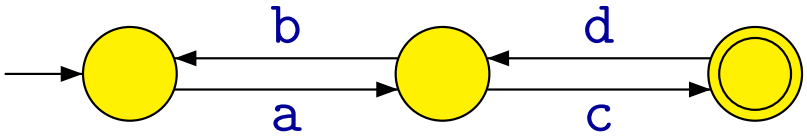


$\mathcal{A}_2$:



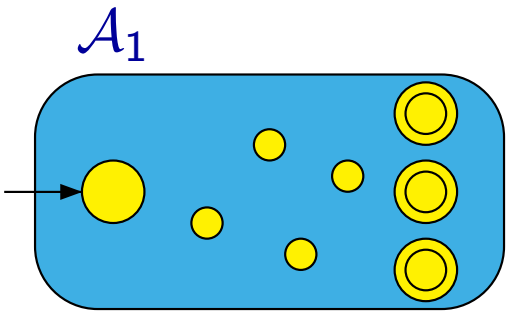
Chybná konstrukce:

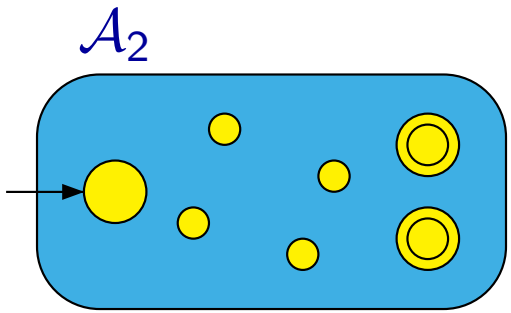
$\mathcal{A}$:



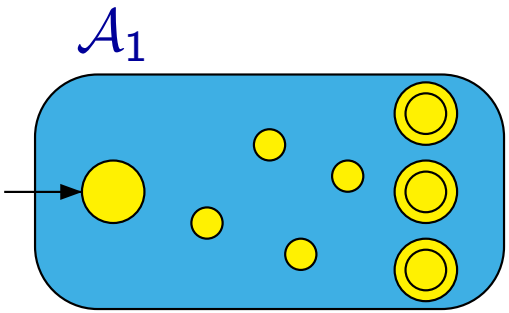
$\mathtt{acdbac} \in \mathcal{L}(\mathcal{A})$, ale $\mathtt{acdbac} \notin \mathcal{L}(\mathcal{A}_1) \cdot \mathcal{L}(\mathcal{A}_2)$

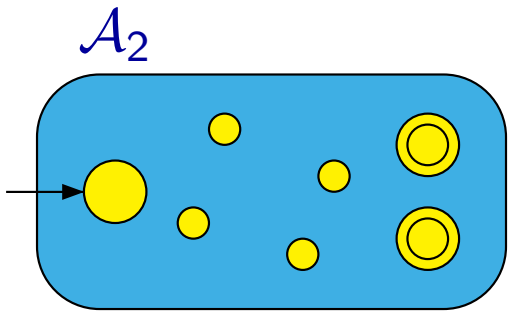
Zřetězení jazyků

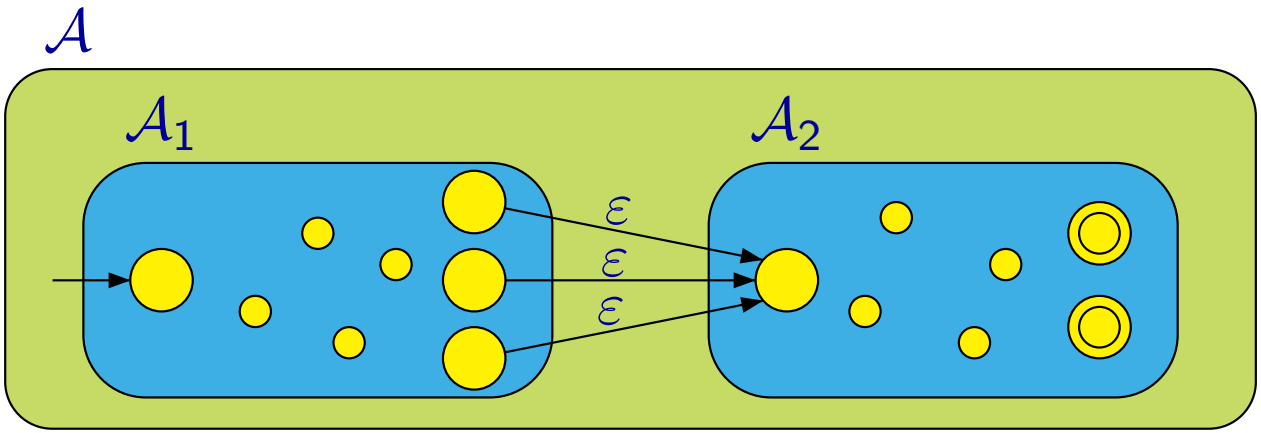




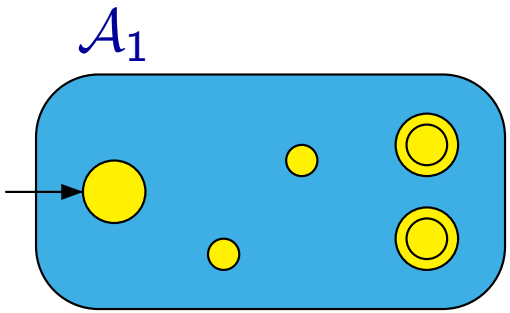
Zřetězení jazyků



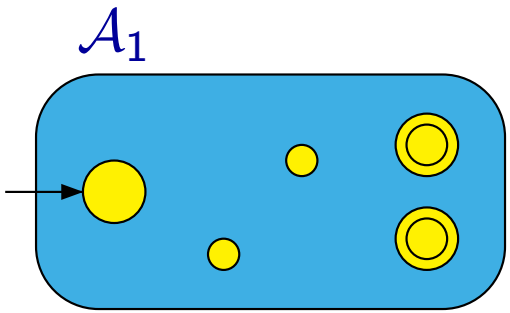


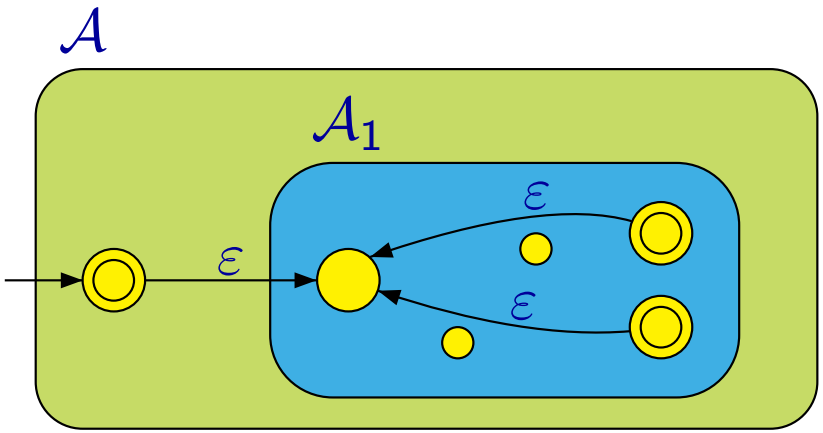


Iterace jazyka



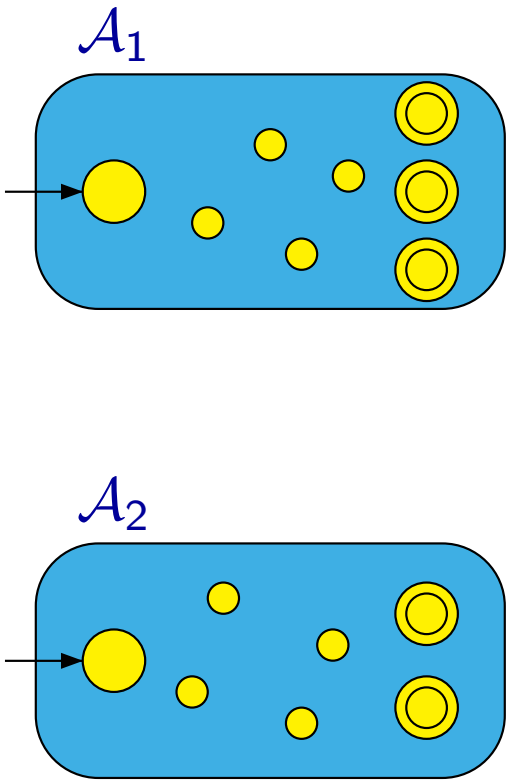
Iterace jazyka





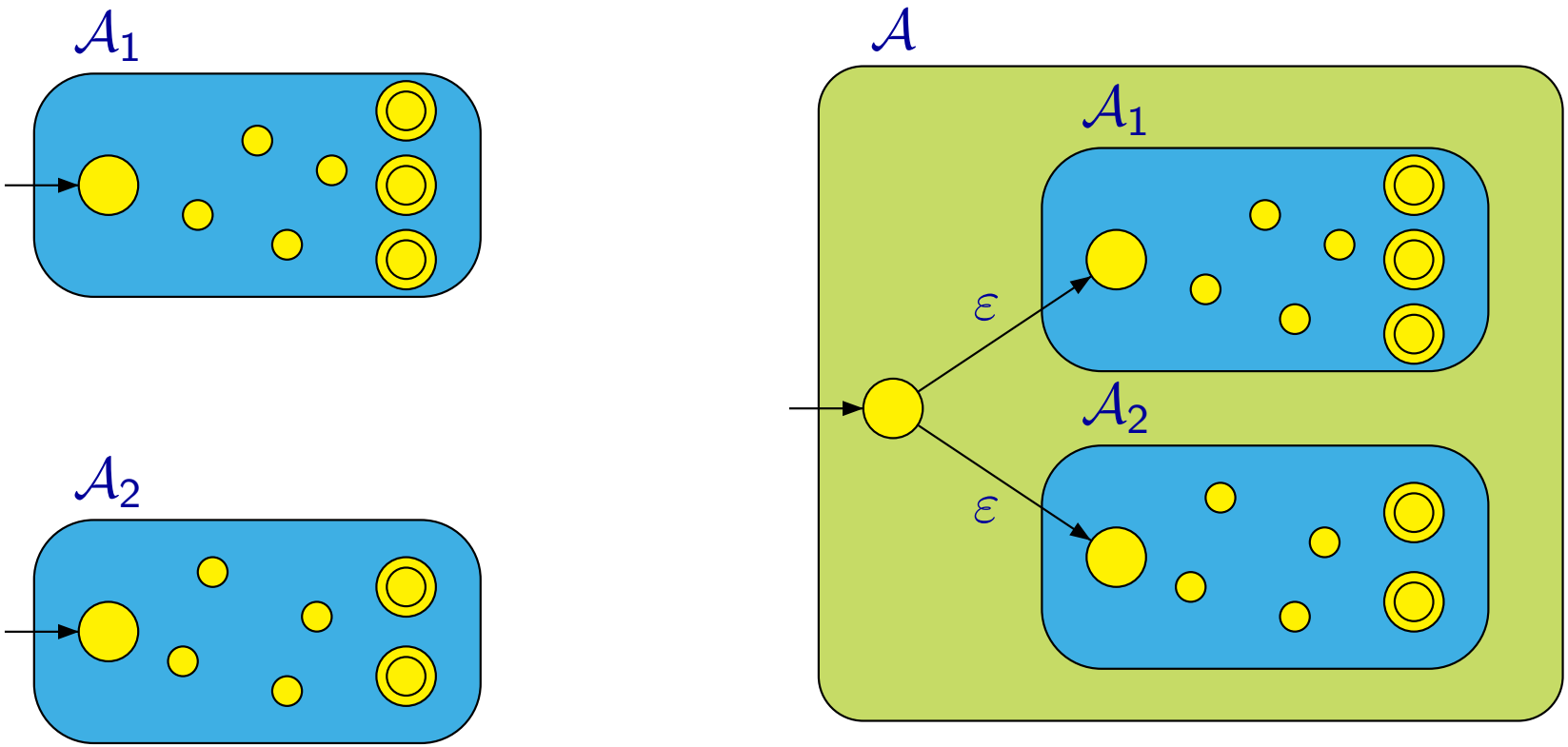
Sjednocení jazyků

Alternativní konstrukce pro sjednocení jazyků:



Sjednocení jazyků

Alternativní konstrukce pro sjednocení jazyků:



Uzavřenost třídy regulárních jazyků

Množina (všech) regulárních jazyků je uzavřená vůči operacím:

- sjednocení
- průnik
- doplněk
- zřetězení
- iterace
- ...

Převod regulárního výrazu na konečný automat

Tvrzení

Každý jazyk, který je možné vyjádřit regulárním výrazem, je regulární (tj. rozpoznávaný nějakým konečným automatem).

Důkaz: Stačí ukázat, jak k danému regulárnímu výrazu α zkonstruovat konečný automat, který rozpoznává jazyk $\mathcal{L}(\alpha)$.

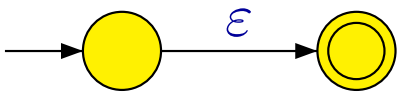
Konstrukce je rekurzivní a postupuje podle struktury výrazu α:

- Pokud je α elementární výraz (tj. $\emptyset$, ε nebo a):
  - Sestrojíme přímo odpovídající automat.
- Pokud je α tvaru $(\beta + \gamma)$, $(\beta \cdot \gamma)$ nebo (β^*):
  - Rekurzivně sestrojíme automaty rozpoznávající jazyky $\mathcal{L}(\beta)$ a $\mathcal{L}(\gamma)$.
  - Z nich sestrojíme automat rozpoznávající jazyk $\mathcal{L}(\alpha)$.

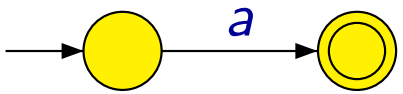
Převod regulárního výrazu na konečný automat

Automaty pro elementární výrazy:

$\emptyset$



ε

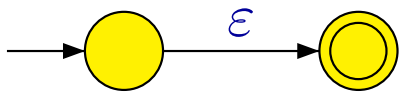


a

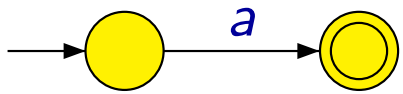
Převod regulárního výrazu na konečný automat

Automaty pro elementární výrazy:

$\emptyset$

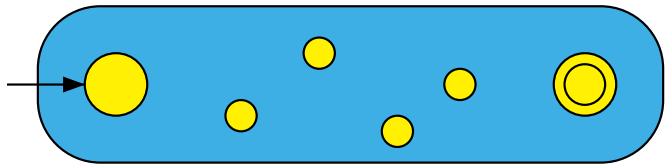


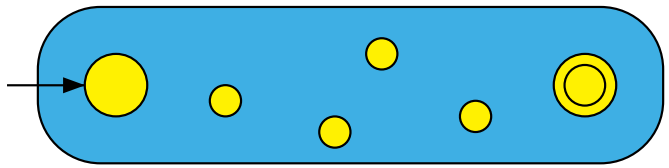
ε



a

Konstrukce pro sjednocení:

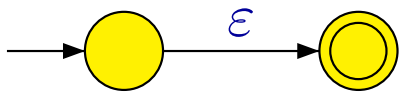




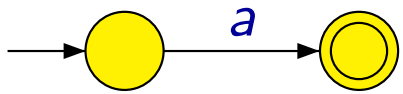
Převod regulárního výrazu na konečný automat

Automaty pro elementární výrazy:

$\emptyset$

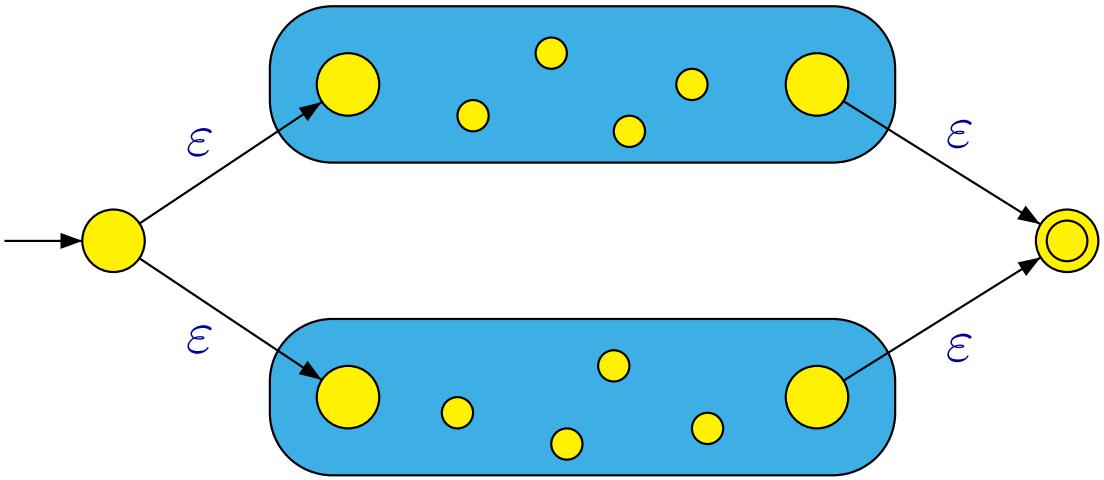


ε



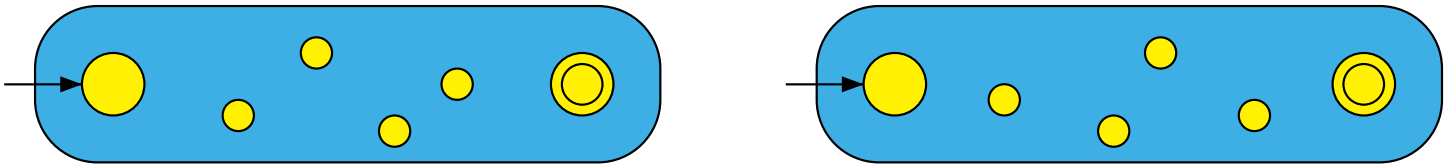
a

Konstrukce pro sjednocení:



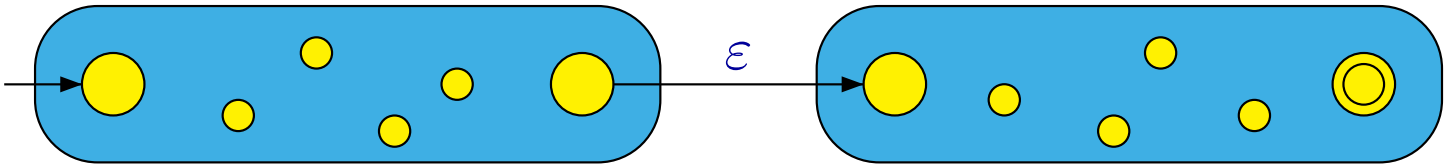
Převod regulárního výrazu na konečný automat

Konstrukce pro zřetězení:



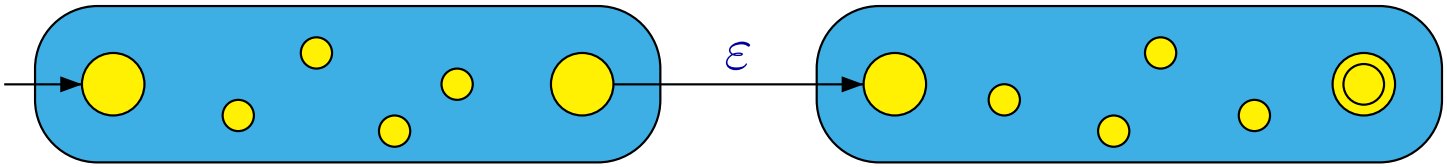
Převod regulárního výrazu na konečný automat

Konstrukce pro zřetězení:

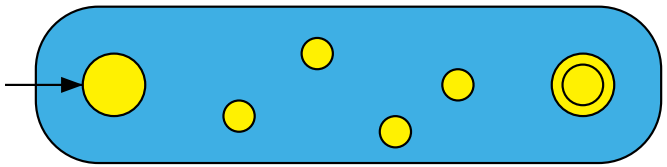


Převod regulárního výrazu na konečný automat

Konstrukce pro zřetězení:

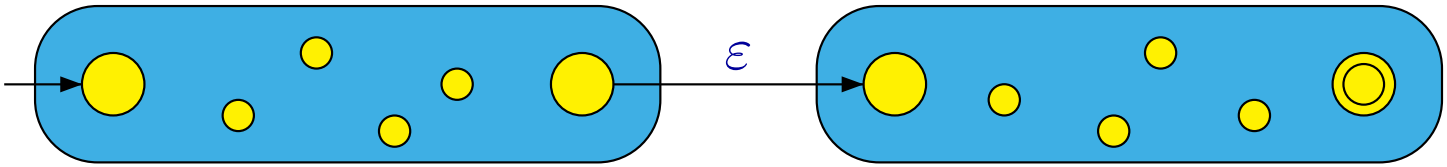


Konstrukce pro iteraci:

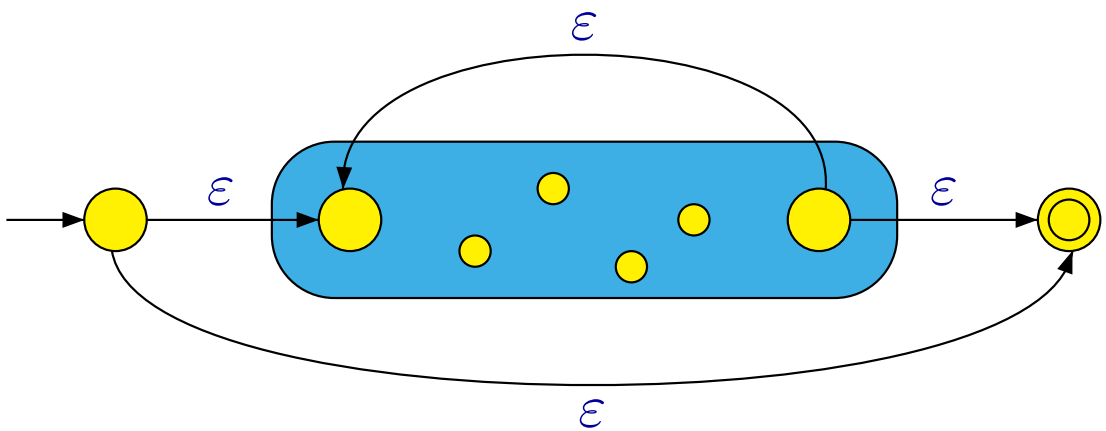


Převod regulárního výrazu na konečný automat

Konstrukce pro zřetězení:



Konstrukce pro iteraci:

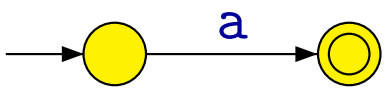


Převod regulárního výrazu na konečný automat

Příklad: Konstrukce automatu pro výraz $((\mathtt{a} + \mathtt{b}) \cdot \mathtt{b})^*$:

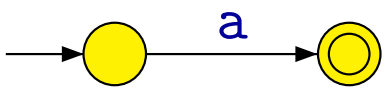
Převod regulárního výrazu na konečný automat

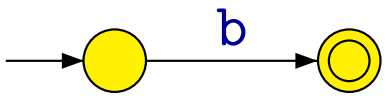
Příklad: Konstrukce automatu pro výraz $((\mathtt{a} + \mathtt{b}) \cdot \mathtt{b})^*$:



Převod regulárního výrazu na konečný automat

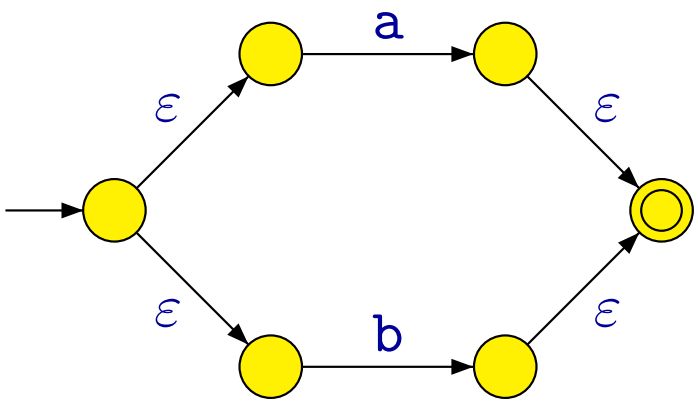
Příklad: Konstrukce automatu pro výraz $((\mathtt{a} + \mathtt{b}) \cdot \mathtt{b})^*$:





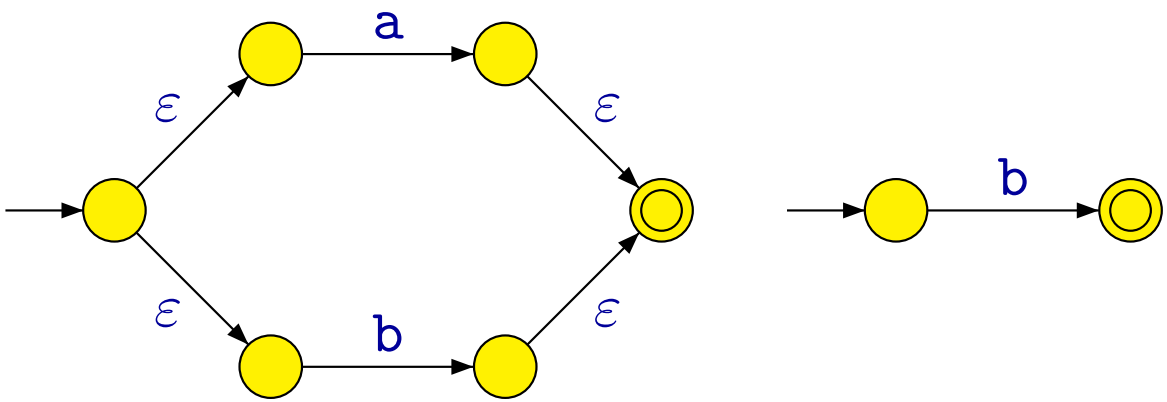
Převod regulárního výrazu na konečný automat

Příklad: Konstrukce automatu pro výraz $((\mathtt{a} + \mathtt{b}) \cdot \mathtt{b})^*$:



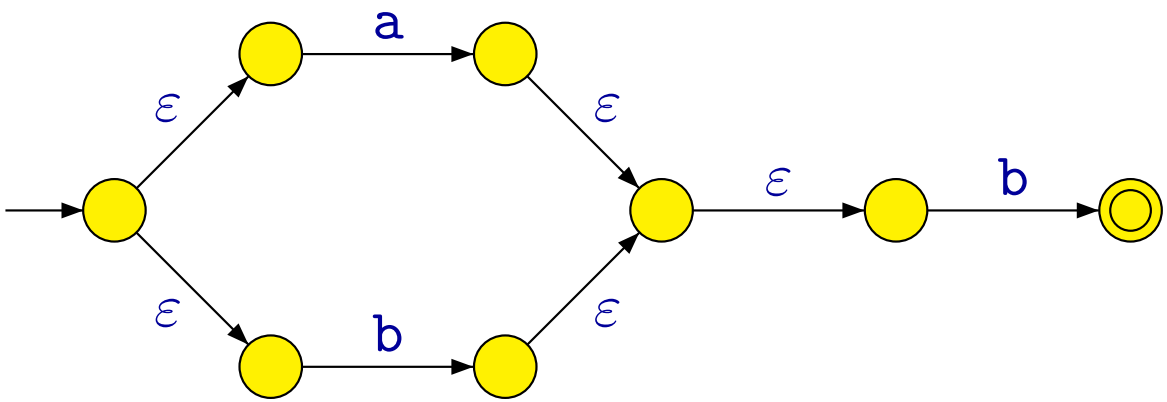
Převod regulárního výrazu na konečný automat

Příklad: Konstrukce automatu pro výraz $((\mathtt{a} + \mathtt{b}) \cdot \mathtt{b})^*$:



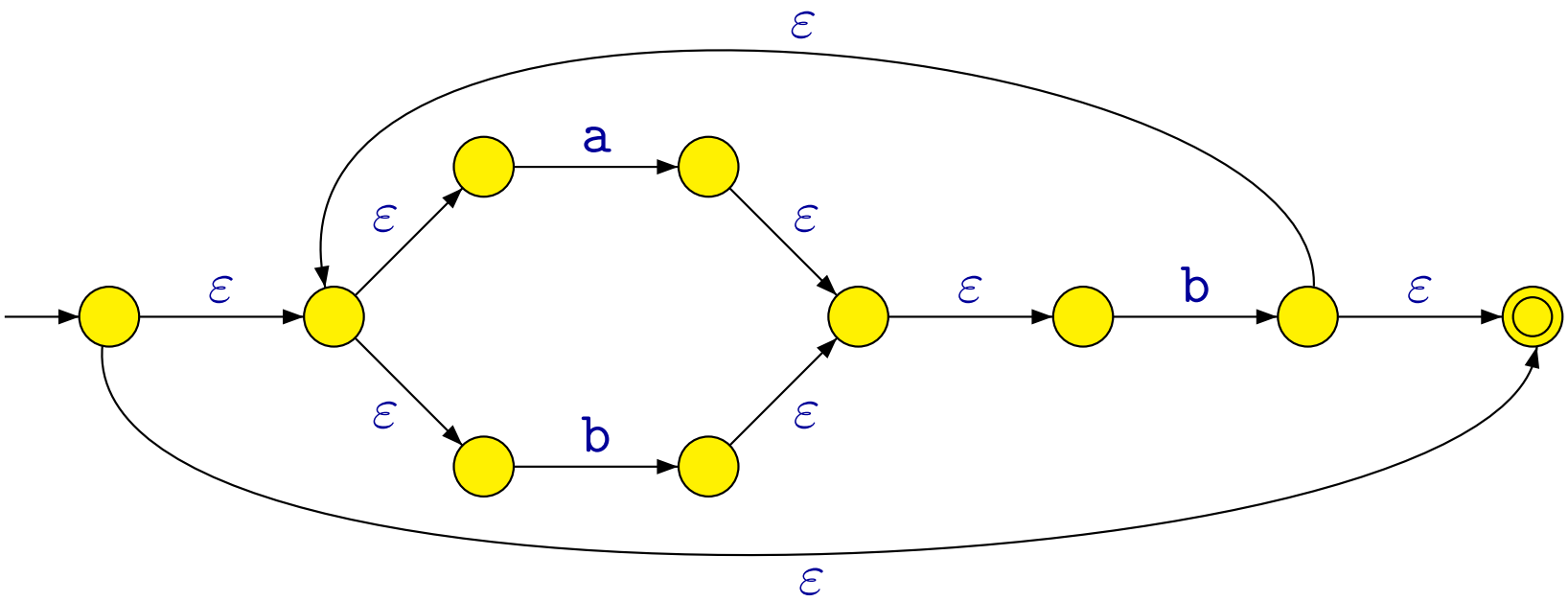
Převod regulárního výrazu na konečný automat

Příklad: Konstrukce automatu pro výraz $((\mathtt{a} + \mathtt{b}) \cdot \mathtt{b})^*$:



Převod regulárního výrazu na konečný automat

Příklad: Konstrukce automatu pro výraz $((\mathtt{a} + \mathtt{b}) \cdot \mathtt{b})^*$:



Převod regulárního výrazu na konečný automat

Pokud se výraz α skládá z n znaků (nepočítáme-li závorky), má výsledný automat:

- nejvýše $2n$ stavů,
- nejvýše $4n$ přechodů.

Poznámka: Převodem ze zobecněného nedeterministického automatu na deterministický však může počet stavů vzrůst exponenciálně, tj. výsledný automat pak může mít až $2^{2n} = 4^n$ stavů.

Převod konečného automatu na regulární výraz

Tvrzení

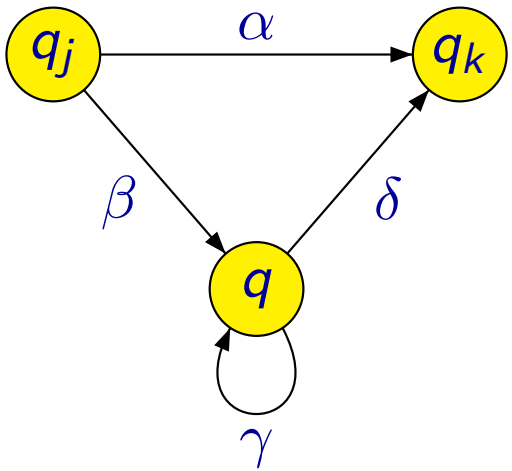
Každý regulární jazyk je možné popsat nějakým regulárním výrazem.

Důkaz: Stačí ukázat, jak pro libovolný konečný automat $\mathcal{A}$ zkonstruovat regulární výraz α takový, že $\mathcal{L}(\alpha) = \mathcal{L}(\mathcal{A})$.

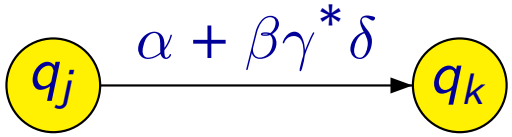
- $\mathcal{A}$ upravíme tak, aby měl právě jeden počáteční a právě jeden přijímající stav.
- Budeme postupně odebírat jednotlivé stavy.
- Přechody budou označeny regulárními výrazy.
- Zbude automat se dvěma stavy – počátečním a koncovým, a jedním přechodem ohodnoceným výsledným regulárním výrazem.

Převod konečného automatu na regulární výraz

Hlavní myšlenka: Při odstraňování stavu q nahradit pro každou dvojici zbylých stavů q_j, q_k cestu z q_j do q_k vedoucí přes q.

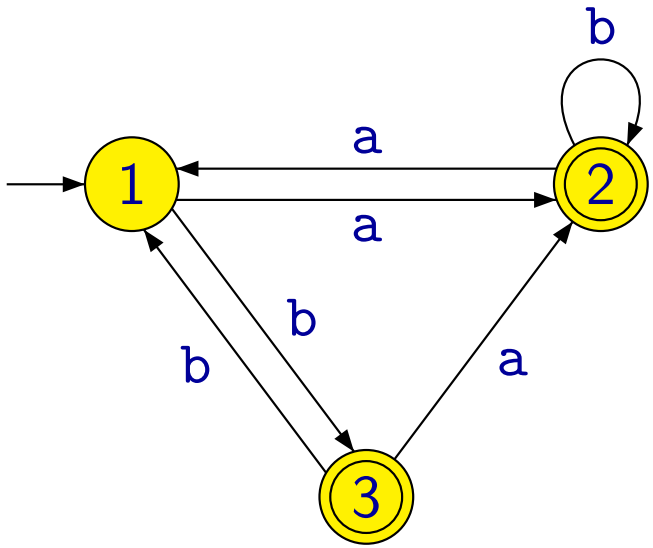


Po odstranění stavu q:



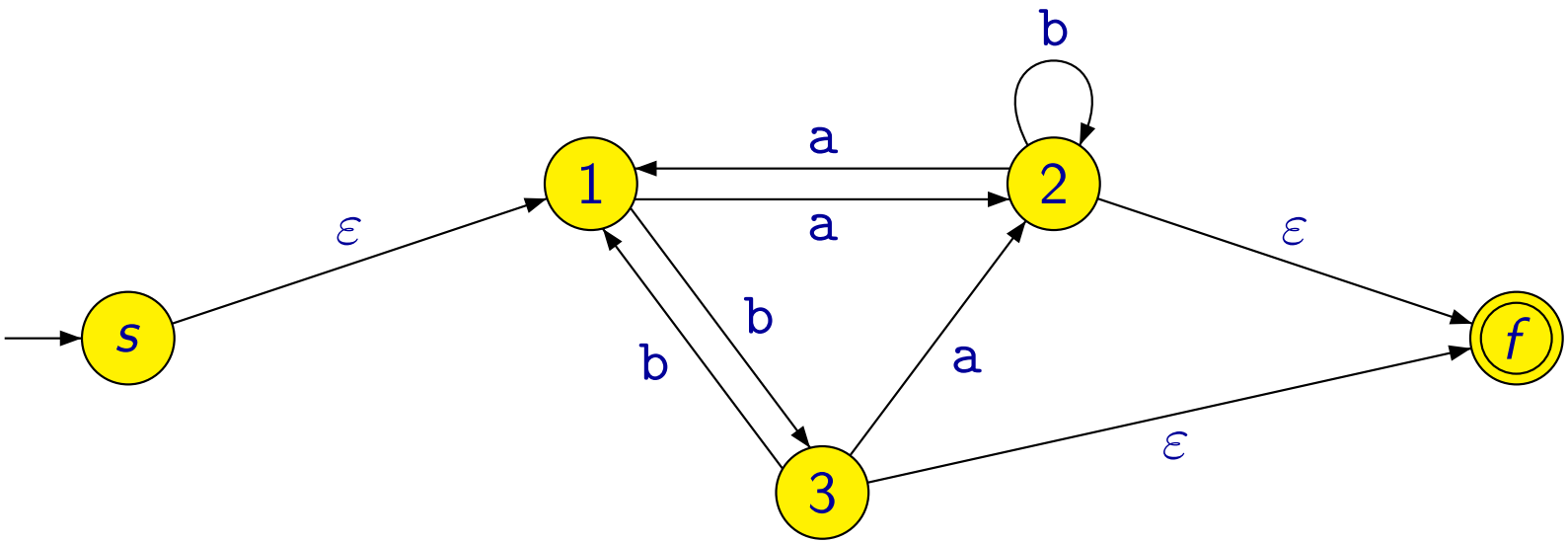
Převod konečného automatu na regulární výraz

Příklad:



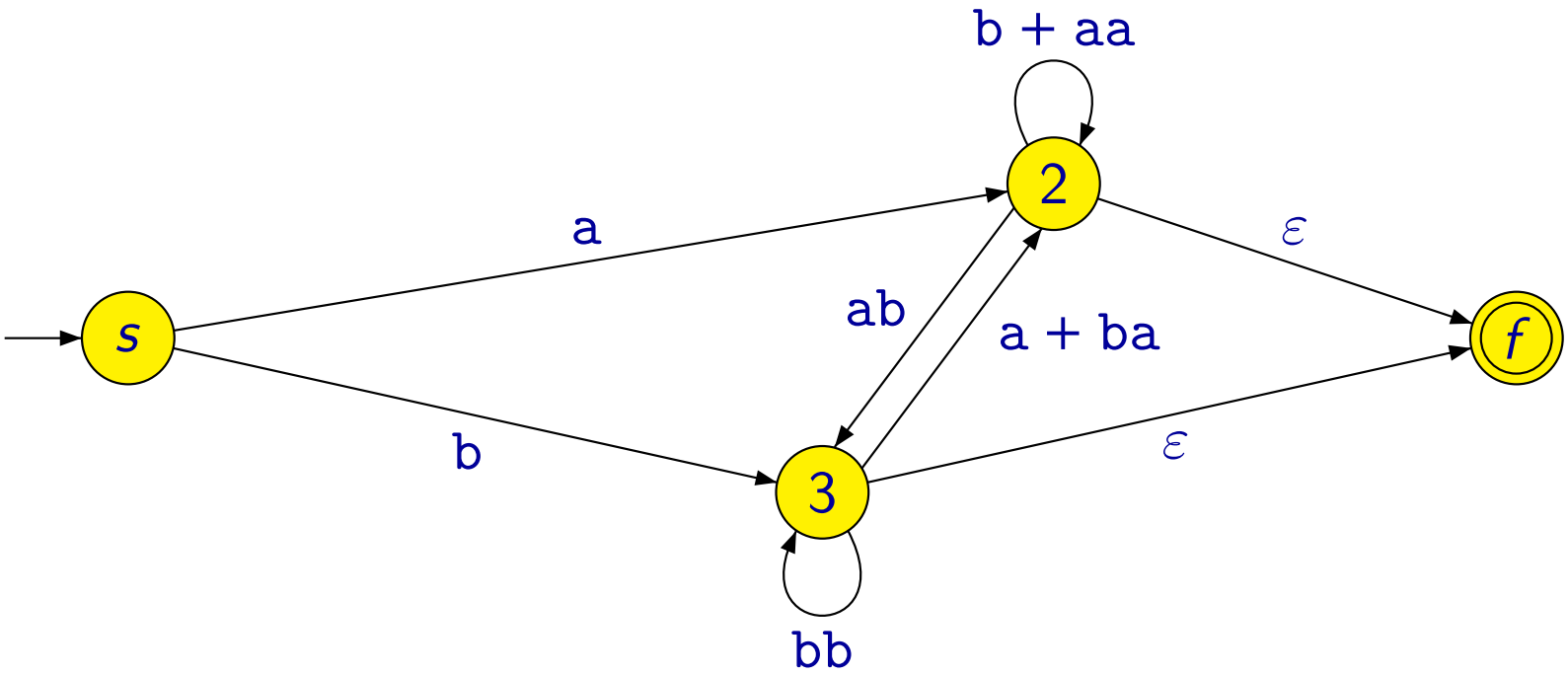
Převod konečného automatu na regulární výraz

Příklad:



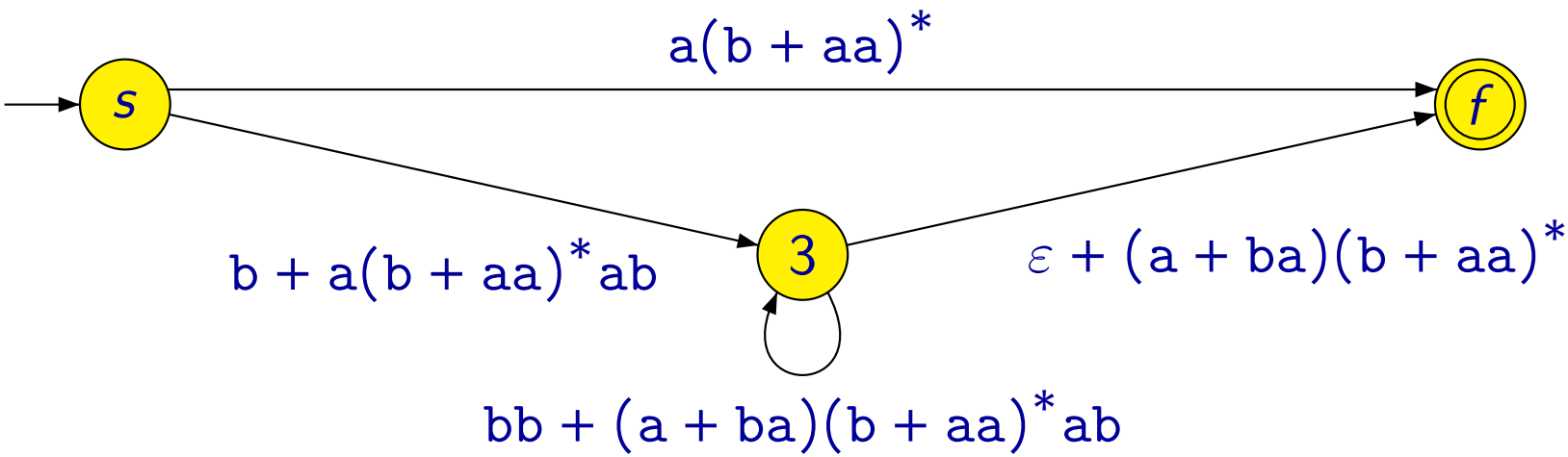
Převod konečného automatu na regulární výraz

Příklad:



Převod konečného automatu na regulární výraz

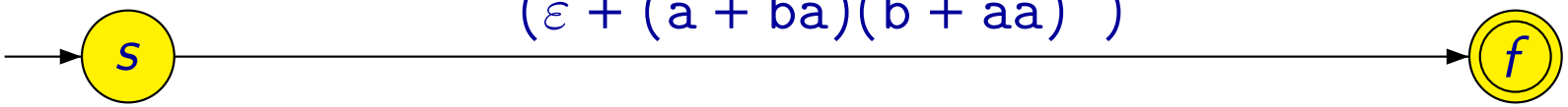
Příklad:



Převod konečného automatu na regulární výraz

Příklad:

$$\mathtt{a}(\mathtt{b}+\mathtt{aa})^*+$$
$$(\mathtt{b}+\mathtt{a}(\mathtt{b}+\mathtt{aa})^*\mathtt{ab})$$
$$(\mathtt{bb}+(\mathtt{a}+\mathtt{ba})(\mathtt{b}+\mathtt{aa})^*\mathtt{ab})^*$$
$$(\varepsilon+(\mathtt{a}+\mathtt{ba})(\mathtt{b}+\mathtt{aa})^*)$$



Ekvivalence konečných automatů a regulárních výrazů

Věta

Jazyk je regulární právě tehdy, když je ho možné popsat regulárním výrazem.

Neregulární jazyky

Ne všechny jazyky jsou regulární.

Existují jazyky, pro které neexistuje žádný konečný automat, který by je rozpoznával.

Příklady neregulárních jazyků:

- $L_1 = \{a^n b^n \mid n \geq 0\}$
- $L_2 = \{ww \mid w \in \{a, b\}^*\}$
- $L_3 = \{ww^R \mid w \in \{a, b\}^*\}$

Poznámka: Existence neregulárních jazyků vyplývá již z faktu, že automatů pracujících nad nějakou abecedou Σ je jen spočetně mnoho, zatímco jazyků nad abecedou Σ je nespočetně mnoho.

Neregulární jazyky

Jak dokázat o nějakém jazyce L, že není regulární?

Jazyk není regulární, jestliže neexistuje (tj. není možné sestrojit) konečný automat, který by ho rozpoznával.

Jak ale dokázat, že něco neexistuje?

Neregulární jazyky

Jak dokázat o nějakém jazyce L, že není regulární?

Jazyk není regulární, jestliže neexistuje (tj. není možné sestrojit) konečný automat, který by ho rozpoznával.

Jak ale dokázat, že něco neexistuje?

Odpověď: Sporem.

Např. předpokládat, že existuje nějaký automat $\mathcal{A}$ rozpoznávající jazyk L, a ukázat, že tento předpoklad vede k logickému sporu.

Neregulární jazyky

Ukážeme, že jazyk $L = \{a^n b^n \mid n \geq 0\}$ není regulární.

Důkaz sporem.

Předpokládejme, že existuje DKA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ takový, že $\mathcal{L}(\mathcal{A}) = L$.

Neregulární jazyky

Ukážeme, že jazyk $L = \{a^n b^n \mid n \geq 0\}$ není regulární.

Důkaz sporem.

Předpokládejme, že existuje DKA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ takový, že $\mathcal{L}(\mathcal{A}) = L$.

Řekněme, že $|Q| = n$.

Neregulární jazyky

Ukážeme, že jazyk $L = \{a^n b^n \mid n \geq 0\}$ není regulární.

Důkaz sporem.

Předpokládejme, že existuje DKA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ takový, že $\mathcal{L}(\mathcal{A}) = L$.

Řekněme, že $|Q| = n$.

Vezměme si slovo $z = a^n b^n$.

Neregulární jazyky

Ukážeme, že jazyk $L = \{a^n b^n \mid n \geq 0\}$ není regulární.

Důkaz sporem.

Předpokládejme, že existuje DKA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ takový, že $\mathcal{L}(\mathcal{A}) = L$.

Řekněme, že $|Q| = n$.

Vezměme si slovo $z = a^n b^n$.

Protože $z \in L$, musí existovat přijímající výpočet automatu $\mathcal{A}$

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} \cdots \xrightarrow{a} q_{n-1} \xrightarrow{a} q_n \xrightarrow{b} q_{n+1} \xrightarrow{b} \cdots \xrightarrow{b} q_{2n-1} \xrightarrow{b} q_{2n}$$

kde q_0 je počáteční stav a $q_{2n} \in F$.

Neregulární jazyky

Vezměme si nyní prvních $n+1$ stavů ve výpočtu

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} \cdots \xrightarrow{a} q_{n-1} \xrightarrow{a} q_n \xrightarrow{b} q_{n+1} \xrightarrow{b} \cdots \xrightarrow{b} q_{2n-1} \xrightarrow{b} q_{2n}$$

tj. posloupnost stavů $q_0, q_1, \ldots, q_n$.

Je zřejmé, že všechny stavy v této posloupnosti nemohou být navzájem různé, protože $|Q| = n$ a tato posloupnost má $n+1$ prvků.

To znamená, že existuje nějaký stav $q \in Q$, který se v této posloupnosti vyskytuje (alespoň) dvakrát.

Neregulární jazyky

Vezměme si nyní prvních $n+1$ stavů ve výpočtu

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} \cdots \xrightarrow{a} q_{n-1} \xrightarrow{a} q_n \xrightarrow{b} q_{n+1} \xrightarrow{b} \cdots \xrightarrow{b} q_{2n-1} \xrightarrow{b} q_{2n}$$

tj. posloupnost stavů $q_0, q_1, \ldots, q_n$.

Je zřejmé, že všechny stavy v této posloupnosti nemohou být navzájem různé, protože $|Q| = n$ a tato posloupnost má $n+1$ prvků.

To znamená, že existuje nějaký stav $q \in Q$, který se v této posloupnosti vyskytuje (alespoň) dvakrát.

Jde o aplikaci tzv. **holubníkového principu (pigeonhole principle)**.

Holubníkový princip

Jestliže mám $n+1$ holubů rozmístěných do n klecí, pak jsou alespoň v jedné kleci minimálně dva holubi.

Neregulární jazyky

Vezměme si nyní prvních $n+1$ stavů ve výpočtu

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} \cdots \xrightarrow{a} q_{n-1} \xrightarrow{a} q_n \xrightarrow{b} q_{n+1} \xrightarrow{b} \cdots \xrightarrow{b} q_{2n-1} \xrightarrow{b} q_{2n}$$

tj. posloupnost stavů $q_0, q_1, \ldots, q_n$.

Je zřejmé, že všechny stavy v této posloupnosti nemohou být navzájem různé, protože $|Q| = n$ a tato posloupnost má $n+1$ prvků.

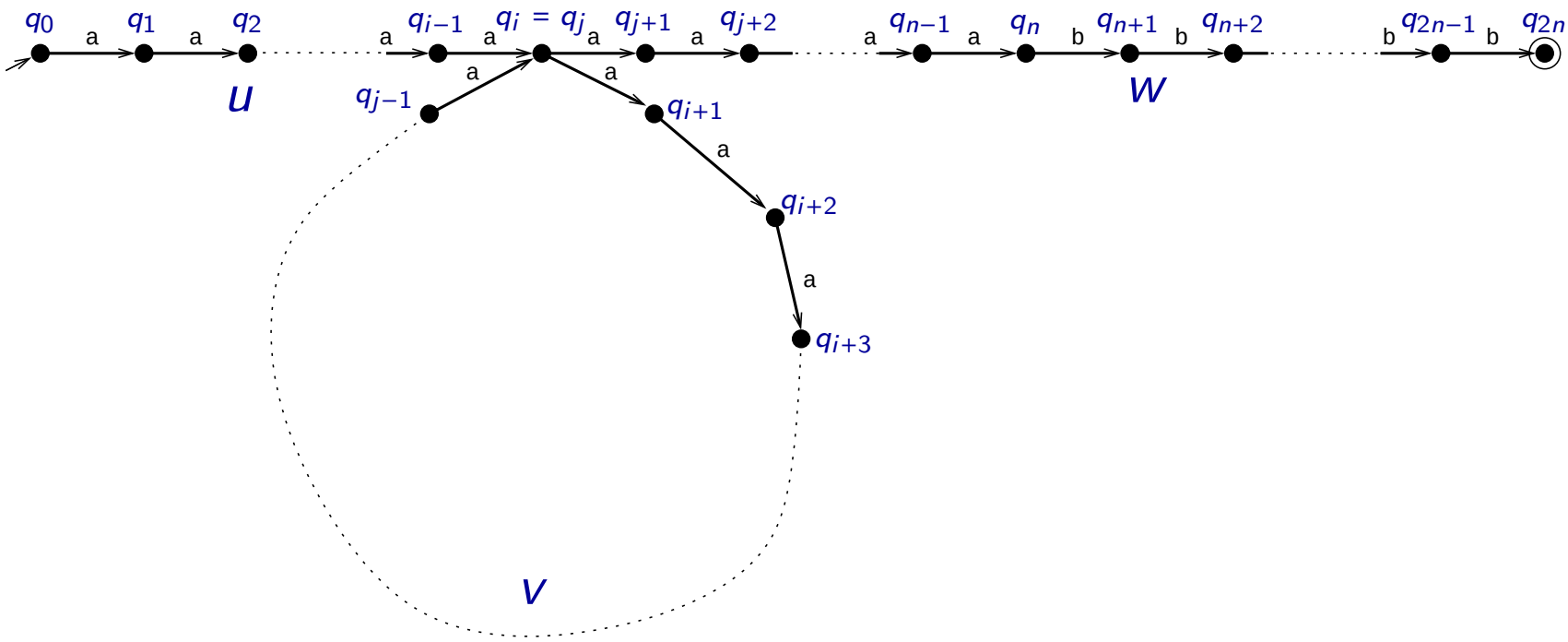
To znamená, že existuje nějaký stav $q \in Q$, který se v této posloupnosti vyskytuje (alespoň) dvakrát.

Tj. existují indexy i, j takové, že $0 \leq i < j \leq n$ a

$$q_i = q_j$$

což znamená, že automat $\mathcal{A}$ při čtení symbolů a ve slově $z = a^n b^n$ projde cyklem.

Neregulární jazyky



Slovo $z = a^n b^n$ můžeme rozdělit na tři části u, v, w takové, že $z = uvw$:

$$u = a^i \qquad v = a^{j-i} \qquad w = a^{n-j} b^n$$

Neregulární jazyky

Pro slova $u = a^i$, $v = a^{j-i}$ a $w = a^{n-j}b^n$ platí

$$q_0 \xrightarrow{u} q_i \qquad q_i \xrightarrow{v} q_j \qquad q_j \xrightarrow{w} q_{2n}$$

Označme r délku slova v, tj. $r = j - i$ (zjevně $r > 0$, protože $i < j$).

Protože $q_i = q_j$, tak automat přijme slovo $uw = a^{n-r}b^n$, které nepatří do jazyka L:

$$q_0 \xrightarrow{u} q_i \xrightarrow{w} q_{2n}$$

Rovněž slovo $uvvw = a^{n+r}b^n$, které také nepatří do L, bude přijato:

$$q_0 \xrightarrow{u} q_i \xrightarrow{v} q_i \xrightarrow{v} q_i \xrightarrow{w} q_{2n}$$

Neregulární jazyky

Podobně můžeme zdůvodnit, že každé slovo tvaru $uvvvv\cdots vvw$, tj. tvaru $uv^k w$ pro nějaké $k \geq 0$, bude automatem $\mathcal{A}$ přijato:

$$q_0 \xrightarrow{u} q_i \xrightarrow{v} q_i \xrightarrow{v} q_i \xrightarrow{v} \cdots \xrightarrow{v} q_i \xrightarrow{v} q_i \xrightarrow{w} q_{2n}$$

Slovo tvaru $uv^k w$ vypadá následovně: $a^{n-r+rk} b^n$.

Protože $r > 0$, tak následující rovnost platí jen pro $k = 1$:

$$n - r + rk = n$$

Pokud je tedy $k \neq 1$, tak slovo $uv^k w$ nepatří do jazyka L.

Automat $\mathcal{A}$ však každé takové slovo přijme, což je spor s předpokladem, že $\mathcal{L}(\mathcal{A}) = \{a^n b^n \mid n \geq 0\}$.

Bezkontextové gramatiky

Bezkontextové gramatiky

Příklad: Chtěli bychom popsat jazyk aritmetických výrazů obsahující výrazy jako například:

`175` `(9+15)` `(((10-4)*((1+34)+2))/(3+(-37)))`

Pro jednoduchost předpokládejme:

- Výrazy jsou plně uzávorkované.
- Jediné aritmetické operace jsou "`+`", "`-`", "`*`", "`/`" a unární "`-`".
- Hodnoty operandů jsou přirozená čísla zapsaná v desítkové soustavě — zápis čísla je neprázdná posloupnost číslic.

Abeceda jazyka: $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, *, /, (,)\}$

Bezkontextové gramatiky

Příklad (pokr.): Popis pomocí induktivní definice:

- **Číslice** je libovolný ze znaků 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- **Číslo** je neprázdná posloupnost číslic, tj.:
  - Pokud je α číslice, tak α je číslo.
  - Pokud α je číslice a β je číslo, tak i $\alpha\beta$ je číslo.
- **Výraz** je libovolná posloupnost symbolů vytvořená podle následujících pravidel:
  - Pokud je α číslo, tak α je výraz.
  - Pokud α je výraz, tak i $(-\alpha)$ je výraz.
  - Pokud α a β jsou výrazy, tak i $(\alpha+\beta)$ je výraz.
  - Pokud α a β jsou výrazy, tak i $(\alpha-\beta)$ je výraz.
  - Pokud α a β jsou výrazy, tak i $(\alpha*\beta)$ je výraz.
  - Pokud α a β jsou výrazy, tak i (α/β) je výraz.

Bezkontextové gramatiky

Příklad (pokr.): Způsob zápisu téže informace jako v předchozí induktivní definici pomocí **bezkontextové gramatiky**:

Zavedeme následující pomocné symboly — těmto symbolům se říká **neterminály**:

- D — zastupuje libovolnou číslici
- C — zastupuje libovolné číslo
- E — zastupuje libovolný výraz

$$D \to 0 \qquad D \to 5$$
$$D \to 1 \qquad D \to 6$$
$$D \to 2 \qquad D \to 7$$
$$D \to 3 \qquad D \to 8$$
$$D \to 4 \qquad D \to 9$$

$$C \to D$$
$$C \to DC$$

$$E \to C$$
$$E \to (-E)$$
$$E \to (E{+}E)$$
$$E \to (E{-}E)$$
$$E \to (E{*}E)$$
$$E \to (E/E)$$

Příklad (pokr.): Stručnější způsob zápisu:

$$D \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$
$$C \to D \mid DC$$
$$E \to C \mid (-E) \mid (E+E) \mid (E-E) \mid (E*E) \mid (E/E)$$

Bezkontextové gramatiky

Příklad: Jazyk, kde slova jsou (případně i prázdné) posloupnosti výrazů popsaných v předchozím příkladě, kde jednotlivé výrazy jsou odděleny čárkami (abecedu je třeba rozšířit o symbol “,”):

$$
\begin{aligned}
S &\to T \mid \varepsilon \\
T &\to E \mid E\,,T \\
D &\to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
C &\to D \mid DC \\
E &\to C \mid (-E) \mid (E{+}E) \mid (E{-}E) \mid (E{*}E) \mid (E/E)
\end{aligned}
$$

Bezkontextové gramatiky

Příklad: Příkazy nějakého programovacího jazyka (fragment gramatiky):

$$
\begin{aligned}
S &\to E\texttt{;} \mid T \mid \texttt{if (}E\texttt{)}\ S \mid \texttt{if (}E\texttt{)}\ S\ \texttt{else}\ S \\
&\quad \mid \texttt{while (}E\texttt{)}\ S \mid \texttt{do}\ S\ \texttt{while (}E\texttt{);} \mid \texttt{for (}F\texttt{;}\,F\texttt{;}\,F\texttt{)}\ S \\
&\quad \mid \texttt{return}\ F\texttt{;} \\
T &\to \texttt{\{}\ U\ \texttt{\}} \\
U &\to \varepsilon \mid SU \\
F &\to \varepsilon \mid E \\
E &\to \quad \ldots \\
&\qquad \ldots
\end{aligned}
$$

Poznámka:

- S — příkaz
- T — blok příkazů
- U — sekvence příkazů
- E — výraz
- F — výraz, který je možno vynechat

Bezkontextové gramatiky

Formálně je **bezkontextová gramatika** definována jako čtveřice

$$\mathcal{G} = (\Pi, \Sigma, S, P)$$

kde:

- Π je konečná množina **neterminálních symbolů** (**neterminálů**)
- Σ je konečná množina **terminálních symbolů** (**terminálů**), přičemž $\Pi \cap \Sigma = \varnothing$
- $S \in \Pi$ je **počáteční neterminál**
- $P \subseteq \Pi \times (\Pi \cup \Sigma)^*$ je konečná množina **přepisovacích pravidel**

Bezkontextové gramatiky

Poznámky:

- Pro označení neterminálních symbolů budeme používat velká písmena A, B, C, ...
- Pro označení terminálních symbolů budeme používat malá písmena a, b, c, ... nebo číslice 0, 1, 2, ...
- Pro označení řetězců z $(\Pi \cup \Sigma)^*$ budeme používat malá písmena řecké abecedy α, β, γ, ...
- Místo zápisu (A, α) budeme pro pravidla používat zápis

$$A \to \alpha$$

A – levá strana pravidla
α – pravá strana pravidla

Bezkontextové gramatiky

Příklad: Gramatika $\mathcal{G} = (\Pi, \Sigma, S, P)$, kde

- $\Pi = \{A, B, C\}$
- $\Sigma = \{a, b\}$
- $S = A$
- P obsahuje pravidla

$$A \to aBBb$$
$$A \to AaA$$
$$B \to \varepsilon$$
$$B \to bCA$$
$$C \to AB$$
$$C \to a$$
$$C \to b$$

Bezkontextové gramatiky

Poznámka: Pokud máme více pravidel se stejnou levou stranou, jako třeba

$$A \to \alpha_1 \qquad A \to \alpha_2 \qquad A \to \alpha_3$$

můžeme je stručněji zapsat jako

$$A \to \alpha_1 \mid \alpha_2 \mid \alpha_3$$

Například pravidla dříve uvedené gramatiky můžeme zapsat jako

$$\begin{aligned} A &\to aBBb \mid AaA \\ B &\to \varepsilon \mid bCA \\ C &\to AB \mid a \mid b \end{aligned}$$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo *abbabb* je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

A

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$\underline{A} \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo *abbabb* je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$\underline{A}$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$\underline{A} \to \underline{aBBb} \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$\underline{A} \Rightarrow \underline{aBBb}$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Longrightarrow aBBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$\underline{B} \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow a\underline{B}Bb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$\underline{A} \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abC\underline{A}Bb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$\underline{A} \to \underline{aBBb} \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abC\underline{A}Bb \Rightarrow abC\underline{aBBb}Bb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaB\underline{B}bBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$\underline{B} \to \underline{\varepsilon} \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaB\underline{B}bBb \Rightarrow abCaBbBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$\underline{C} \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow ab\underline{C}aBbBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$\underline{C} \to AB \mid a \mid \underline{b}$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow ab\underline{C}aBbBb \Rightarrow ab\underline{b}aBbBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb$$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$\underline{B} \to \underline{\varepsilon} \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBb\underline{B}b \Rightarrow abbaBbb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abbaBbb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abbaBbb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$\underline{B} \to \underline{\varepsilon} \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abba\underline{B}bb \Rightarrow abbabb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $\mathcal{G} = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice $\mathcal{G}$ vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abbaBbb \Rightarrow abbabb$

Bezkontextové gramatiky

Na řetězcích z $(\Pi \cup \Sigma)^*$ definujeme relaci $\Rightarrow \subseteq (\Pi \cup \Sigma)^* \times (\Pi \cup \Sigma)^*$ takovou, že

$$\alpha \Rightarrow \alpha'$$

právě když $\alpha = \beta_1 A \beta_2$ a $\alpha' = \beta_1 \gamma \beta_2$ pro nějaká $\beta_1, \beta_2, \gamma \in (\Pi \cup \Sigma)^*$ a $A \in \Pi$, kde $(A \to \gamma) \in P$.

Příklad: Jestliže $(B \to bCA) \in P$, pak

$$aCBbA \Rightarrow aCbCAbA$$

Poznámka: Neformálně řečeno zápis $\alpha \Rightarrow \alpha'$ znamená, že z α je možné jedním krokem odvodit α', a to tak, že výskyt nějakého neterminálu A v α nahradíme pravou stranou nějakého pravidla $A \to \gamma$, kde se A vyskytuje na levé straně.

Bezkontextové gramatiky

Na řetězcích z $(\Pi \cup \Sigma)^*$ definujeme relaci $\Rightarrow \subseteq (\Pi \cup \Sigma)^* \times (\Pi \cup \Sigma)^*$ takovou, že

$$\alpha \Rightarrow \alpha'$$

právě když $\alpha = \beta_1 A \beta_2$ a $\alpha' = \beta_1 \gamma \beta_2$ pro nějaká $\beta_1, \beta_2, \gamma \in (\Pi \cup \Sigma)^*$ a $A \in \Pi$, kde $(A \to \gamma) \in P$.

Příklad: Jestliže $(B \to bCA) \in P$, pak

$$aC\underline{B}bA \Rightarrow aC\underline{bCA}bA$$

Poznámka: Neformálně řečeno zápis $\alpha \Rightarrow \alpha'$ znamená, že z α je možné jedním krokem odvodit α', a to tak, že výskyt nějakého neterminálu A v α nahradíme pravou stranou nějakého pravidla $A \to \gamma$, kde se A vyskytuje na levé straně.

Bezkontextové gramatiky

Derivace délky n je posloupnost $\beta_0, \beta_1, \beta_2, \cdots, \beta_n$, kde $\beta_i \in (\Pi \cup \Sigma)^*$ a kde $\beta_{i-1} \Rightarrow \beta_i$ pro všechna $1 \leq i \leq n$, což můžeme stručněji zapsat

$$\beta_0 \Rightarrow \beta_1 \Rightarrow \beta_2 \Rightarrow \ldots \Rightarrow \beta_{n-1} \Rightarrow \beta_n$$

Skutečnost, že pro dané $\alpha, \alpha' \in (\Pi \cup \Sigma)^*$ a $n \in \mathbb{N}$ existuje nějaká derivace $\beta_0 \Rightarrow \beta_1 \Rightarrow \beta_2 \Rightarrow \ldots \Rightarrow \beta_{n-1} \Rightarrow \beta_n$, kde $\alpha = \beta_0$ a $\alpha' = \beta_n$, zapisujeme

$$\alpha \Rightarrow^n \alpha'$$

Skutečnost, že $\alpha \Rightarrow^n \alpha'$ pro nějaké $n \geq 0$, zapisujeme

$$\alpha \Rightarrow^* \alpha'$$

Poznámka: Relace $\Rightarrow^*$ je reflexivním a tranzitivním uzávěrem relace $\Rightarrow$ (tj. nejmenší reflexivní a tranzitivní relací obsahující relaci $\Rightarrow$).

Bezkontextové gramatiky

Větné formy jsou ty $\alpha \in (\Pi \cup \Sigma)^*$, pro které platí

$$S \Rightarrow^* \alpha$$

kde S je počáteční neterminál.

Bezkontextové gramatiky

Jazyk $\mathcal{L}(\mathcal{G})$ generovaný gramatikou $\mathcal{G} = (\Pi, \Sigma, S, P)$ je množina všech slov v abecedě Σ, která lze odvodit nějakou derivací z počátečního neterminálu S pomocí pravidel z P, tj.

$$\mathcal{L}(\mathcal{G}) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$

Definice

Jazyk L je **bezkontextový**, jestliže existuje bezkontextová gramatika $\mathcal{G}$ taková, že $L = \mathcal{L}(\mathcal{G})$.

Derivační strom

$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$

Derivační strom

A

$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$

A

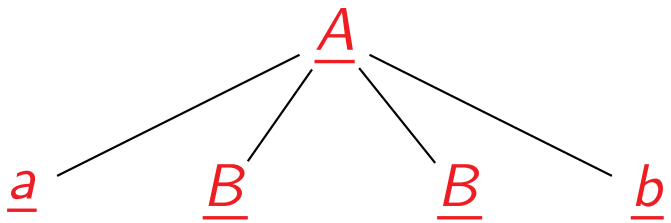
Derivační strom

$\underline{A}$

$\underline{A} \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$

$\underline{A}$

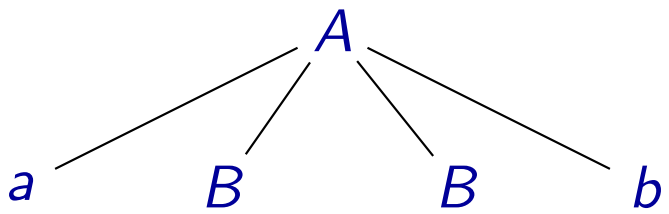
Derivační strom



$\underline{A} \to \underline{aBBb} \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$

$\underline{A} \Rightarrow \underline{aBBb}$

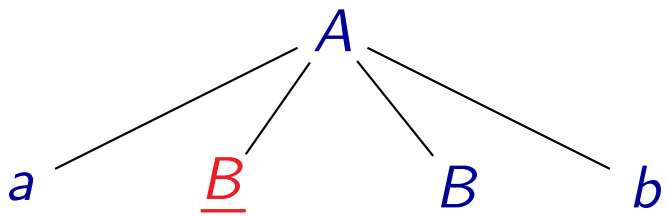
Derivační strom



$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$

$A \Longrightarrow aBBb$

Derivační strom

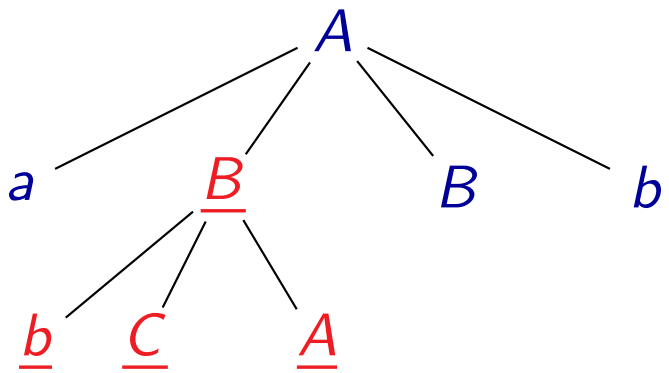


$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$

$A \Longrightarrow aBBb$

Derivační strom

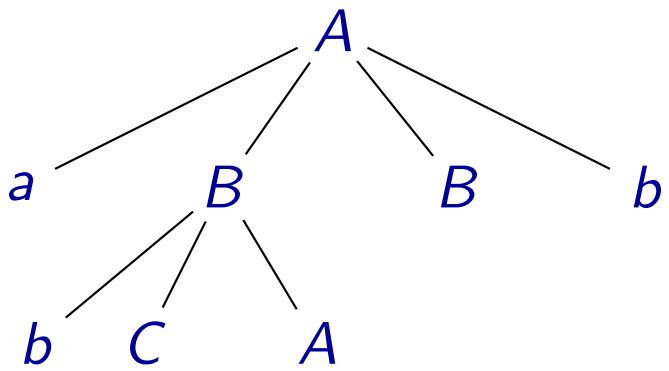
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb$

Derivační strom

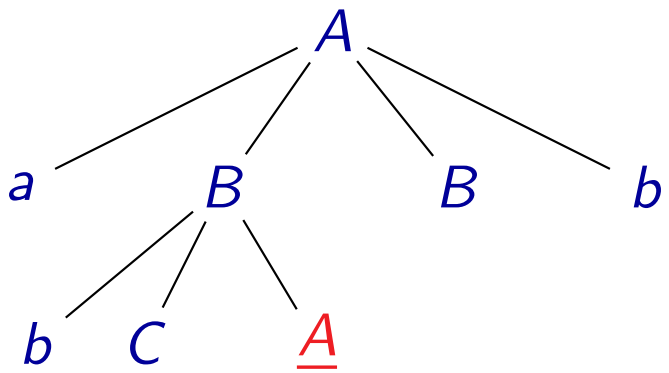
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb$

Derivační strom

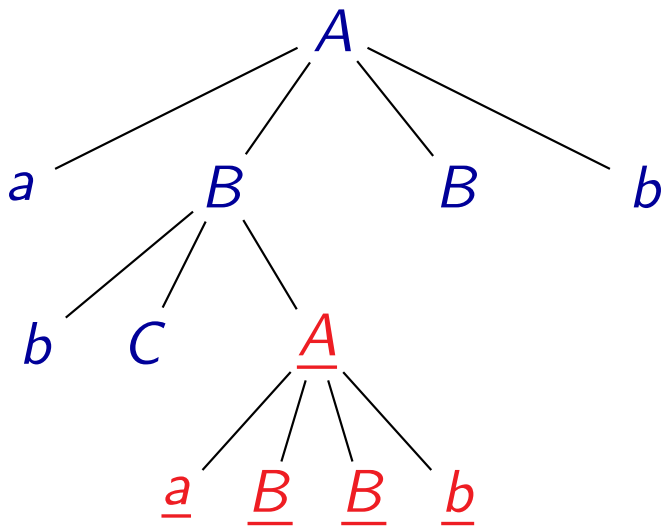
$\underline{A} \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abC\underline{A}Bb$

Derivační strom

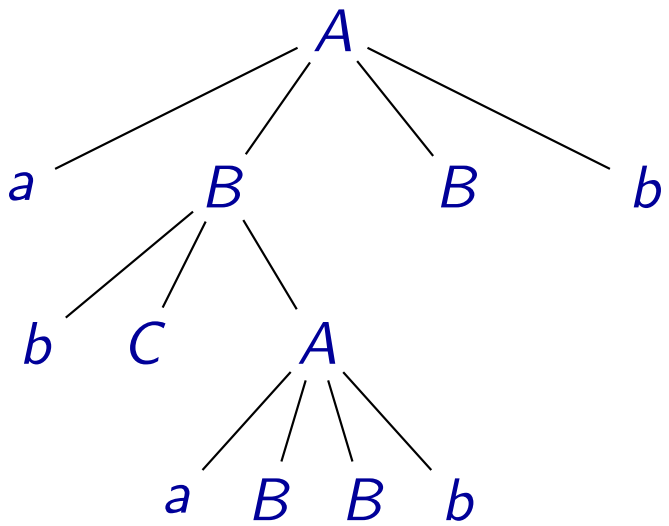
$\underline{A} \to \underline{aBBb} \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abC\underline{A}Bb \Rightarrow abC\underline{aBBb}Bb$

Derivační strom

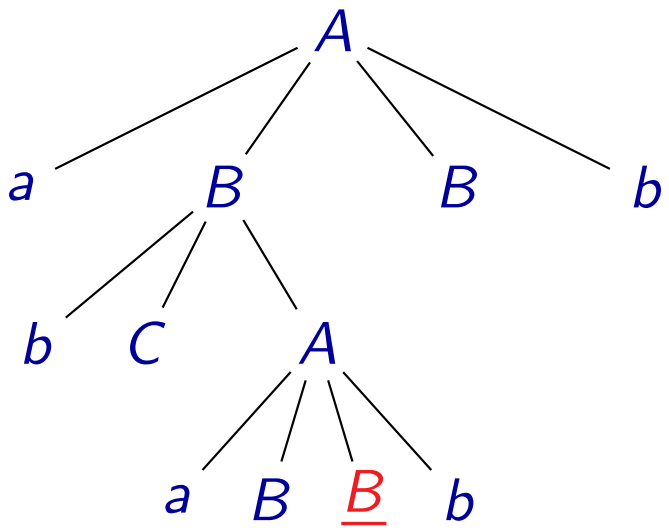
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb$

Derivační strom

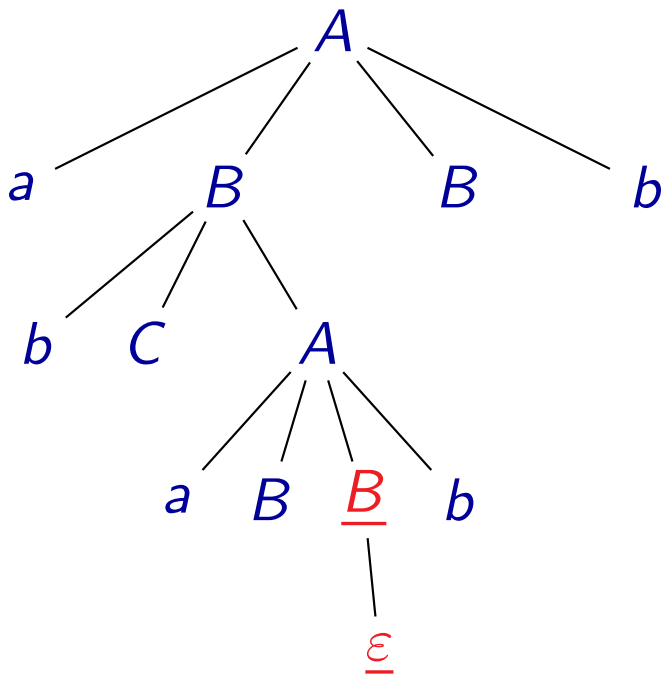
$A \to aBBb \mid AaA$
$\underline{B} \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaB\underline{B}bBb$

Derivační strom

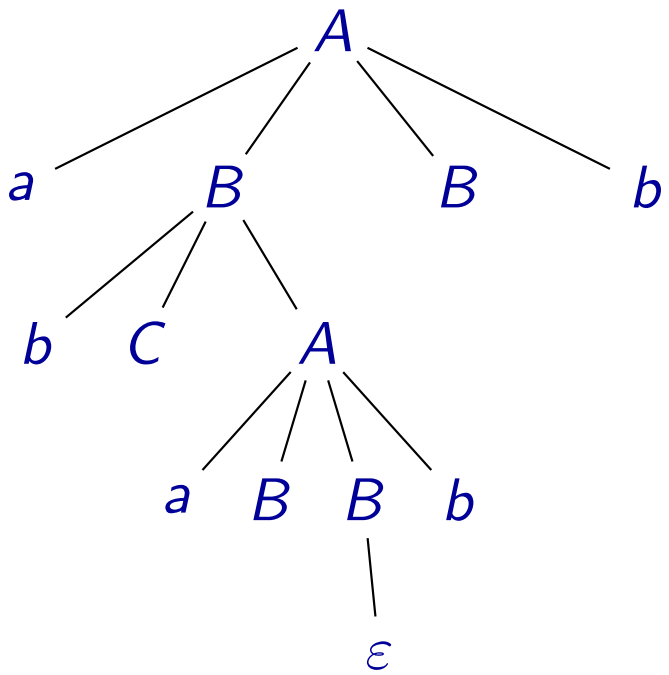
$A \to aBBb \mid AaA$
$\underline{B} \to \underline{\varepsilon} \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaB\underline{B}bBb \Rightarrow abCaBbBb$

Derivační strom

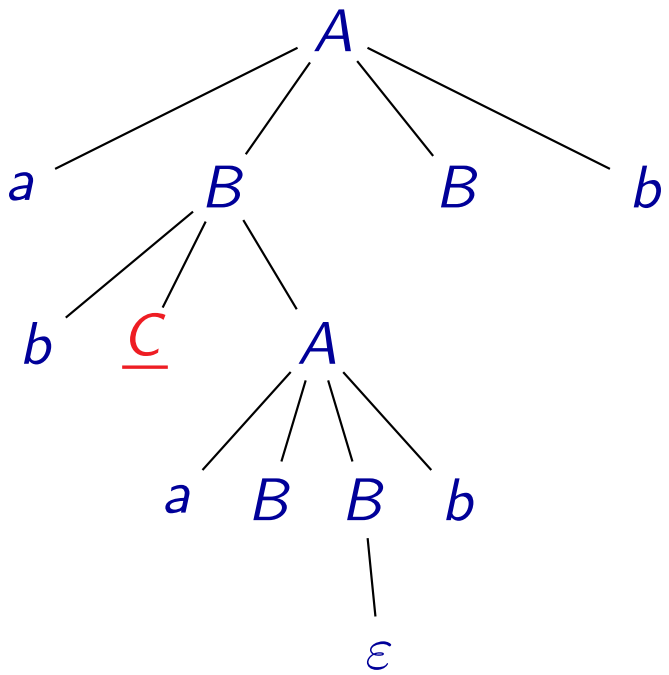
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb$

Derivační strom

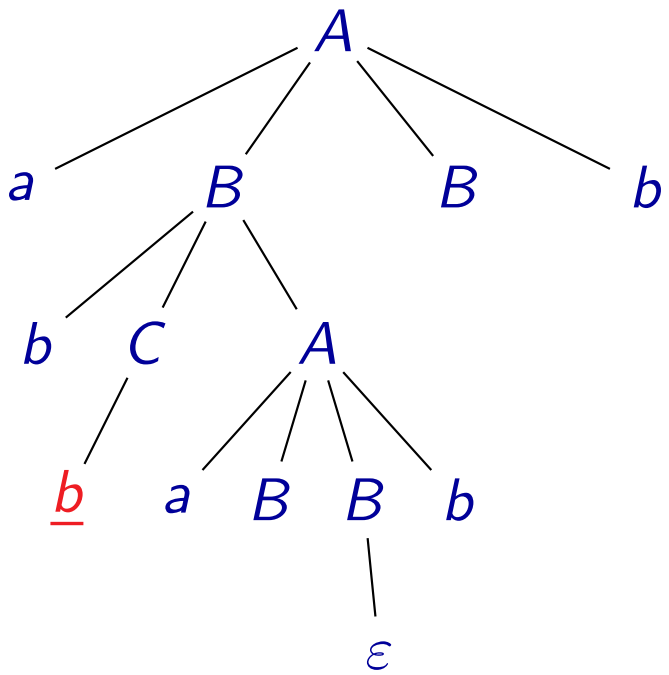
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$\underline{C} \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow ab\underline{C}aBbBb$

Derivační strom

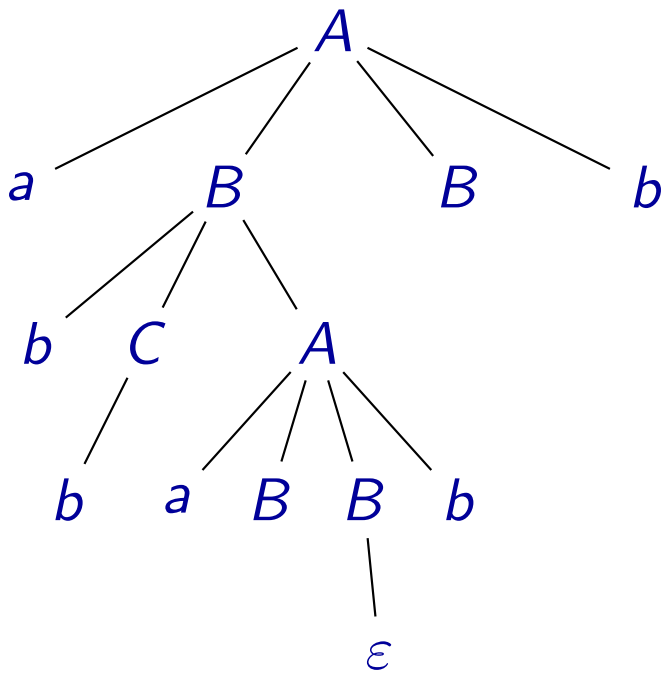
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$\underline{C} \to AB \mid a \mid \underline{b}$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow ab\underline{C}aBbBb \Rightarrow ab\underline{b}aBbBb$

Derivační strom

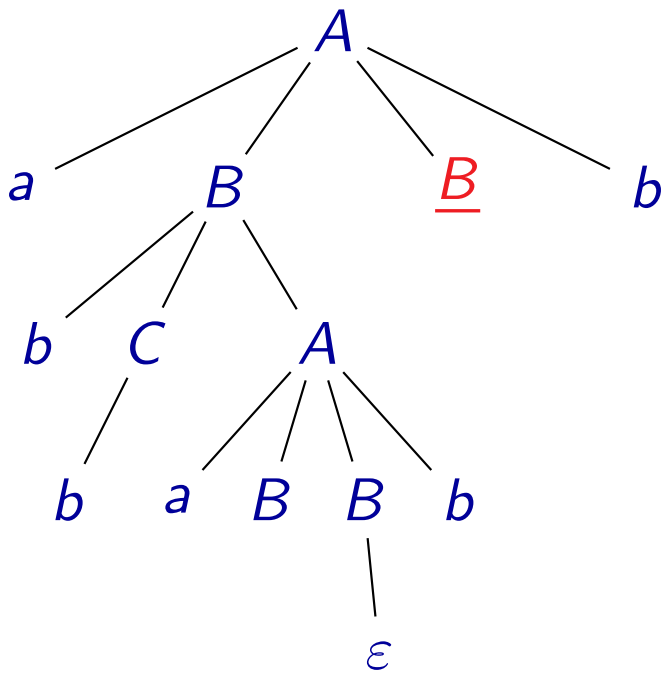
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb$

Derivační strom

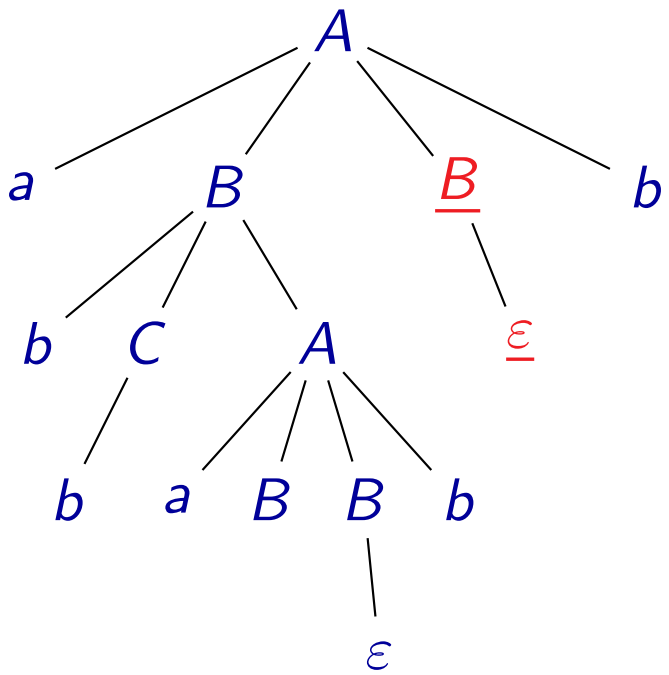
$A \to aBBb \mid AaA$
$\underline{B} \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBb\underline{B}b$

Derivační strom

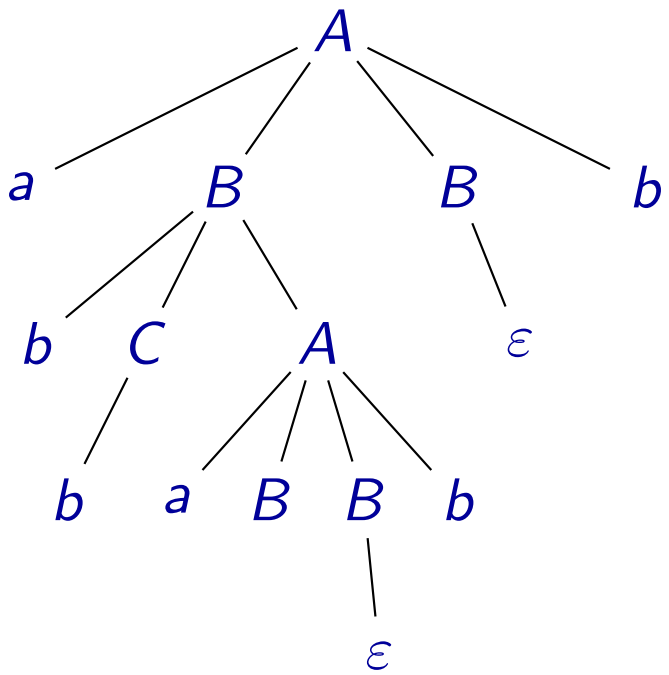
$A \to aBBb \mid AaA$
$\underline{B} \to \underline{\varepsilon} \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBb\underline{B}b \Rightarrow abbaBbb$

Derivační strom

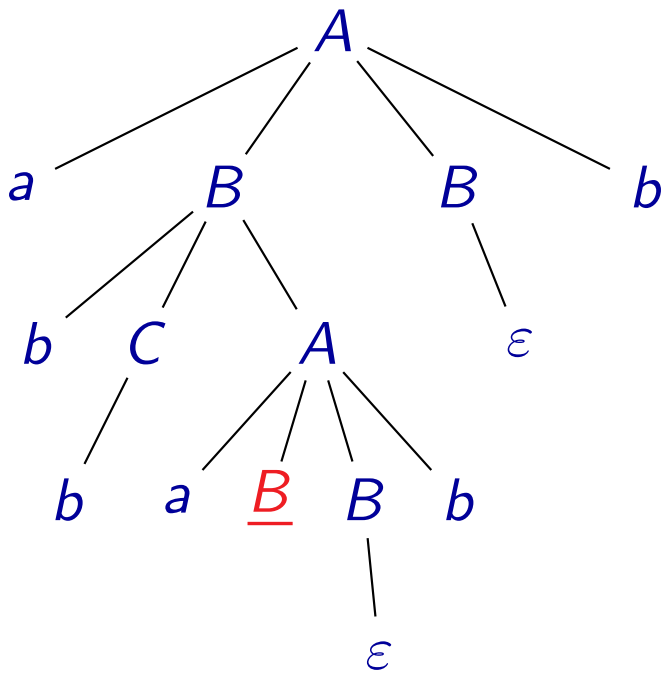
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abbaBbb$

Derivační strom

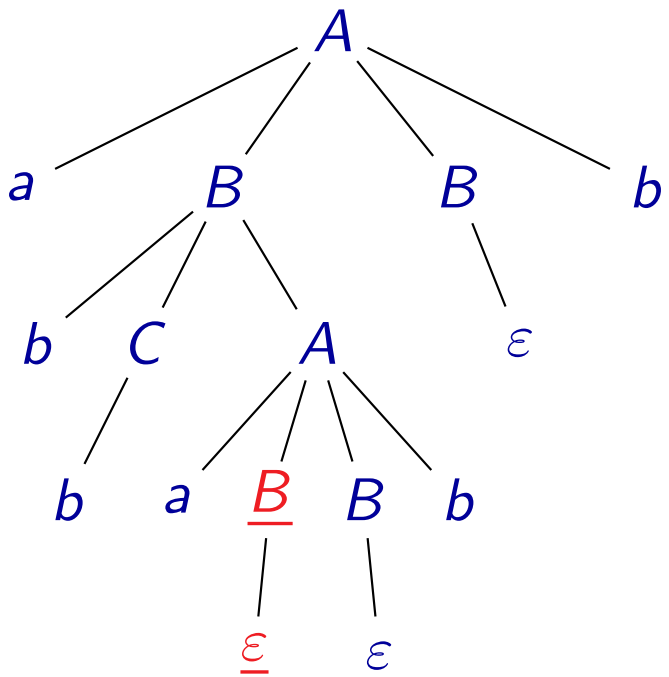
$A \to aBBb \mid AaA$
$\underline{B} \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abba\underline{B}bb$

Derivační strom

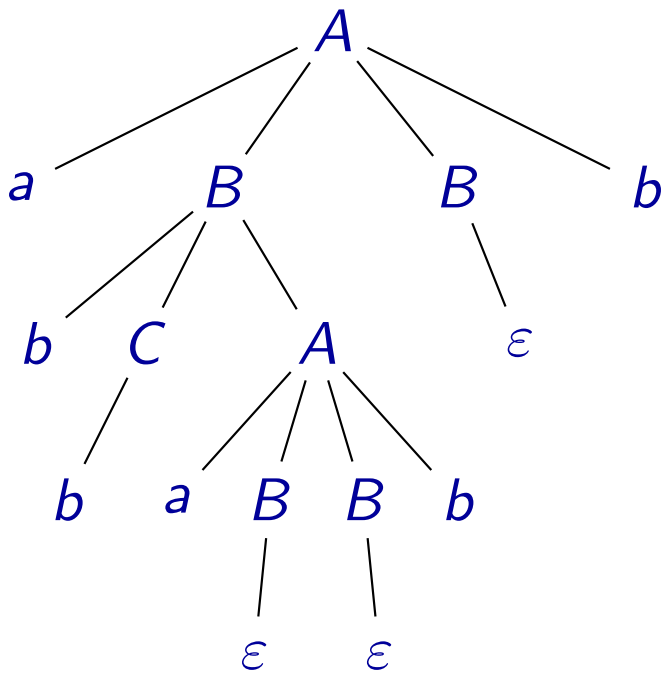
$A \to aBBb \mid AaA$
$\underline{B} \to \underline{\varepsilon} \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abba\underline{B}bb \Rightarrow abbabb$

Derivační strom

$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abbaBbb \Rightarrow abbabb$

Derivační strom

Každé derivaci odpovídá nějaký **derivační strom**:

- Vrcholy stromu jsou ohodnoceny terminály a neterminály.
- Kořen stromu je ohodnocen počátečním neterminálem.
- Listy stromu jsou ohodnoceny terminály nebo symboly ε.
- Ostatní vrcholy stromu jsou ohodnoceny neterminály.
- Pokud je vrchol ohodnocen neterminálem A, pak jeho potomci jsou ohodnoceni symboly pravé strany nějakého přepisovacího pravidla $A \to \alpha$.

Bezkontextové gramatiky

Příklad: Gramatika generující jazyk

$$L = \{a^n b^n \mid n \geq 0\}$$

Bezkontextové gramatiky

Příklad: Gramatika generující jazyk

$$L = \{a^n b^n \mid n \geq 0\}$$

Gramatika $\mathcal{G} = (\Pi, \Sigma, S, P)$, kde $\Pi = \{S\}$, $\Sigma = \{a, b\}$ a P obsahuje

$$S \to \varepsilon \mid aSb$$

Bezkontextové gramatiky

Příklad: Gramatika generující jazyk

$$L = \{a^n b^n \mid n \geq 0\}$$

Gramatika $\mathcal{G} = (\Pi, \Sigma, S, P)$, kde $\Pi = \{S\}$, $\Sigma = \{a, b\}$ a P obsahuje

$$S \to \varepsilon \mid aSb$$

$S \Rightarrow \varepsilon$
$S \Rightarrow aSb \Rightarrow ab$
$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$
$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$
$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaaaSbbbb \Rightarrow aaaabbbb$
$\cdots$

Bezkontextové gramatiky

Příklad: Gramatika generující jazyk L tvořený všemi palindromy nad abecedou $\{a, b\}$, tj.

$$L = \{w \in \{a, b\}^* \mid w = w^R\}$$

Poznámka: w^R označuje tzv. **zrcadlový obraz** slova w, tj. slovo w zapsané pozpátku.

Bezkontextové gramatiky

Příklad: Gramatika generující jazyk L tvořený všemi palindromy nad abecedou $\{a, b\}$, tj.

$$L = \{w \in \{a, b\}^* \mid w = w^R\}$$

Poznámka: w^R označuje tzv. **zrcadlový obraz** slova w, tj. slovo w zapsané pozpátku.

Řešení:

$$S \to \varepsilon \mid a \mid b \mid aSa \mid bSb$$

Bezkontextové gramatiky

Příklad: Gramatika generující jazyk L tvořený všemi palindromy nad abecedou $\{a, b\}$, tj.

$$L = \{w \in \{a, b\}^* \mid w = w^R\}$$

Poznámka: w^R označuje tzv. **zrcadlový obraz** slova w, tj. slovo w zapsané pozpátku.

Řešení:

$$S \to \varepsilon \mid a \mid b \mid aSa \mid bSb$$

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abaSaba \Rightarrow abaaaba$$

Bezkontextové gramatiky

Příklad: Gramatika generující jazyk L tvořený všemi dobře uzávorkovanými sekvencemi symbolů '(' a ')'.

Například $(()())(()) \in L$, ale $)()) \notin L$.

Bezkontextové gramatiky

Příklad: Gramatika generující jazyk L tvořený všemi dobře uzávorkovanými sekvencemi symbolů '(' a ')'.

Například $(()())(()) \in L$, ale $)()) \notin L$.

Řešení:

$$A \to \varepsilon \mid (A) \mid AA$$

Bezkontextové gramatiky

Příklad: Gramatika generující jazyk L tvořený všemi dobře uzávorkovanými sekvencemi symbolů '(' a ')'.

Například $(()())(()) \in L$, ale $)()) \notin L$.

Řešení:

$$A \to \varepsilon \mid (A) \mid AA$$

$A \Rightarrow AA \Rightarrow (A)A \Rightarrow (A)(A) \Rightarrow (AA)(A) \Rightarrow ((A)A)(A) \Rightarrow$
$(()A)(A) \Rightarrow (()(A))(A) \Rightarrow (()())(A) \Rightarrow (()())((A)) \Rightarrow$
$(()())(())$

Bezkontextové gramatiky

Příklad: Gramatika generující jazyk L tvořený všemi dobře vytvořenými aritmetickými výrazy, kde operandy jsou vždy tvaru 'a', a kde jako operátory můžeme používat symboly $+$ a $*$.

Například $(a + a) * a + (a * a) \in L$.

Bezkontextové gramatiky

Příklad: Gramatika generující jazyk L tvořený všemi dobře vytvořenými aritmetickými výrazy, kde operandy jsou vždy tvaru 'a', a kde jako operátory můžeme používat symboly $+$ a $*$.

Například $(a + a) * a + (a * a) \in L$.

Řešení:

$$E \to a \mid E + E \mid E * E \mid (E)$$

Bezkontextové gramatiky

Příklad: Gramatika generující jazyk L tvořený všemi dobře vytvořenými aritmetickými výrazy, kde operandy jsou vždy tvaru 'a', a kde jako operátory můžeme používat symboly $+$ a $*$.

Například $(a + a) * a + (a * a) \in L$.

Řešení:

$$E \to a \mid E + E \mid E * E \mid (E)$$

$$E \Rightarrow E + E \Rightarrow E * E + E \Rightarrow (E) * E + E \Rightarrow (E + E) * E + E \Rightarrow$$
$$(a + E) * E + E \Rightarrow (a + a) * E + E \Rightarrow (a + a) * a + E \Rightarrow (a + a) * a + (E) \Rightarrow$$
$$(a + a) * a + (E * E) \Rightarrow (a + a) * a + (a * E) \Rightarrow (a + a) * a + (a * a)$$

Levá a pravá derivace

$$E \to a \mid E + E \mid E * E \mid (E)$$

Levá derivace je derivace, ve které v každém kroku nahrazujeme vždy nejlevější neterminál.

$$\underline{E} \Rightarrow \underline{E} + E \Rightarrow \underline{E} * E + E \Rightarrow a * \underline{E} + E \Rightarrow a * a + \underline{E} \Rightarrow a * a + a$$

Pravá derivace je derivace, ve které v každém kroku nahrazujeme vždy nejpravější neterminál.

$$\underline{E} \Rightarrow E + \underline{E} \Rightarrow \underline{E} + a \Rightarrow E * \underline{E} + a \Rightarrow \underline{E} * a + a \Rightarrow a * a + a$$

Derivace však nemusí být ani levá ani pravá:

$$\underline{E} \Rightarrow \underline{E} + E \Rightarrow E * \underline{E} + E \Rightarrow E * a + \underline{E} \Rightarrow \underline{E} * a + a \Rightarrow a * a + a$$

Levá a pravá derivace

- Jednomu derivačnímu stromu může odpovídat více různých derivací.
- Každému derivačnímu stromu odpovídá právě jedna levá a právě jedna pravá derivace.

Ekvivalence gramatik

Gramatiky $\mathcal{G}_1$ a $\mathcal{G}_2$ jsou **ekvivalentní**, jestliže generují tentýž jazyk, tj. jestliže $\mathcal{L}(\mathcal{G}_1) = \mathcal{L}(\mathcal{G}_2)$.

Poznámka: Problém ekvivalence bezkontextových gramatik je algoritmicky nerozhodnutelný. Dá se dokázat, že není možné vytvořit algoritmus, který by pro libovolné dvě bezkontextové gramatiky rozhodl, zda jsou ekvivalentní či ne.

Dokonce je algoritmicky nerozhodnutelný i problém, zda gramatika generuje jazyk Σ^*.

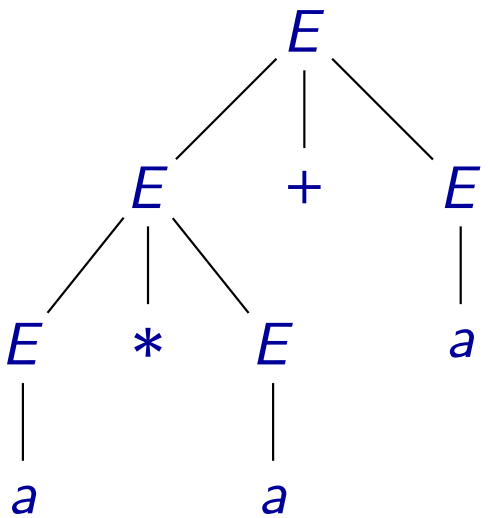
Nejednoznačné gramatiky

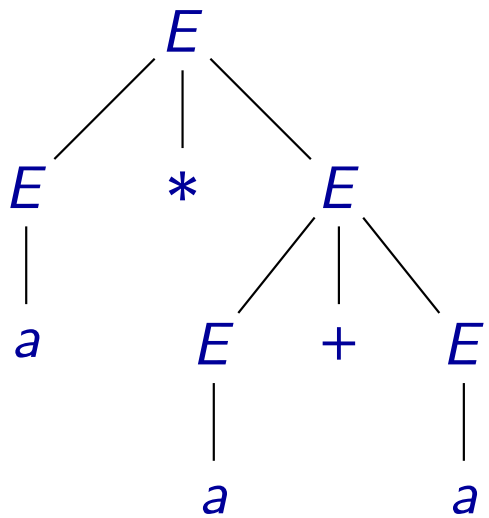
Gramatika $\mathcal{G}$ je **nejednoznačná**, jestliže existuje nějaké slovo $w \in \mathcal{L}(\mathcal{G})$, kterému přísluší dva různé derivační stromy, resp. dvě různé levé či dvě různé pravé derivace.

Příklad:

$E \Rightarrow E + E \Rightarrow E * E + E \Rightarrow a * E + E \Rightarrow a * a + E \Rightarrow a * a + a$

$E \Rightarrow E * E \Rightarrow E * E + E \Rightarrow a * E + E \Rightarrow a * a + E \Rightarrow a * a + a$





Nejednoznačné gramatiky

Někdy je možné nejednoznačnou gramatiku nahradit gramatikou, která generuje tentýž jazyk, ale není nejednoznačná.

Příklad: Gramatiku

$$E \to a \mid E + E \mid E * E \mid (E)$$

můžeme nahradit ekvivalentní gramatikou

$$\begin{aligned} E &\to T \mid T + E \\ T &\to F \mid F * T \\ F &\to a \mid (E) \end{aligned}$$

Poznámka: Pokud se nejednoznačná gramatika žádnou ekvivalentní jednoznačnou gramatikou nahradit nedá, říkáme, že je **podstatně nejednoznačná**.

Bezkontextové jazyky

Třída bezkontextových jazyků je uzavřená vůči:

- zřetězení
- sjednocení
- iteraci

Třída bezkontextových jazyků však není uzavřená vůči:

- doplňku
- průniku

Bezkontextové jazyky

Máme dány gramatiky $\mathcal{G}_1 = (\Pi_1, \Sigma, S_1, P_1)$ a $\mathcal{G}_2 = (\Pi_2, \Sigma, S_2, P_2)$, přičemž můžeme předpokládat, že $\Pi_1 \cap \Pi_2 = \varnothing$ a $S \notin \Pi_1 \cup \Pi_2$.

- Gramatika $\mathcal{G}$ taková, že $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}_1) \cdot \mathcal{L}(\mathcal{G}_2)$:

$$\mathcal{G} = (\Pi_1 \cup \Pi_2 \cup \{S\},\ \Sigma,\ S,\ P_1 \cup P_2 \cup \{S \to S_1 S_2\})$$

- Gramatika $\mathcal{G}$ taková, že $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}_1) \cup \mathcal{L}(\mathcal{G}_2)$:

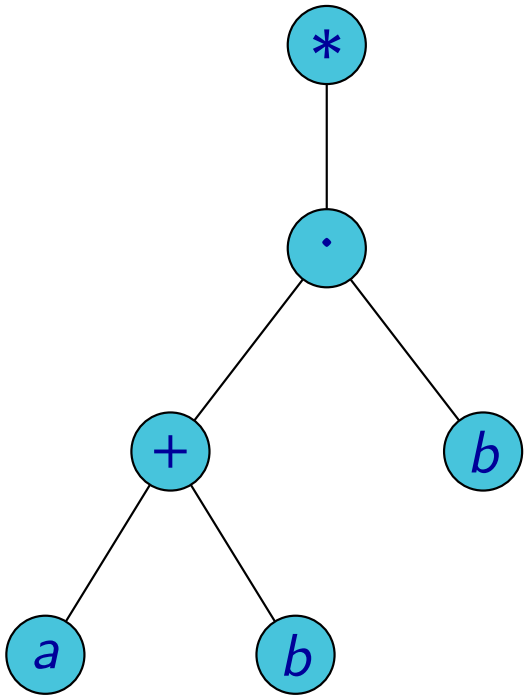
$$\mathcal{G} = (\Pi_1 \cup \Pi_2 \cup \{S\},\ \Sigma,\ S,\ P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\})$$

- Gramatika $\mathcal{G}$ taková, že $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}_1)^*$:

$$\mathcal{G} = (\Pi_1 \cup \{S\},\ \Sigma,\ S,\ P_1 \cup \{S \to \varepsilon, S \to S_1 S\})$$

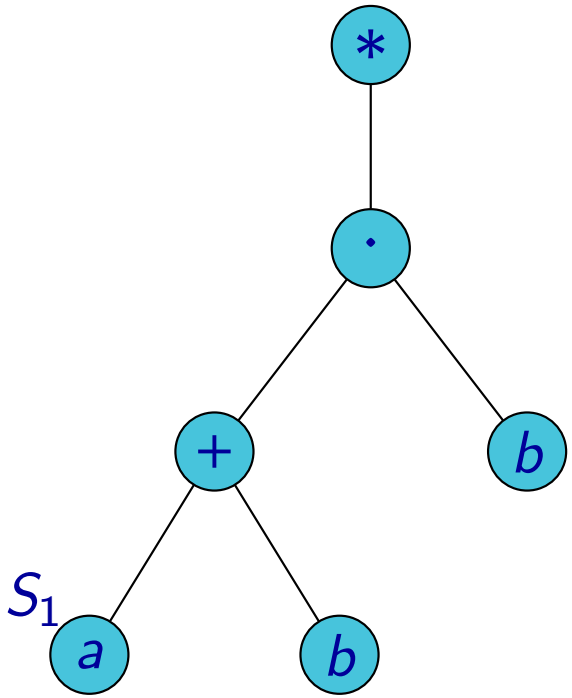
Převod regulárního výrazu na bezkontextovou gramatiku

Příklad: Konstrukce bezkontextové gramatiky k regulárnímu výrazu $((a + b) \cdot b)^*$:



Převod regulárního výrazu na bezkontextovou gramatiku

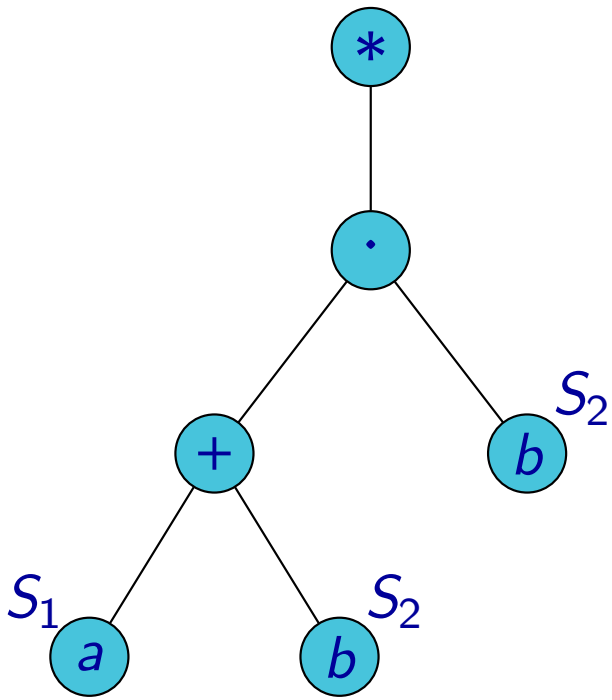
Příklad: Konstrukce bezkontextové gramatiky k regulárnímu výrazu $((a + b) \cdot b)^*$:



$S_1 \to a$

Převod regulárního výrazu na bezkontextovou gramatiku

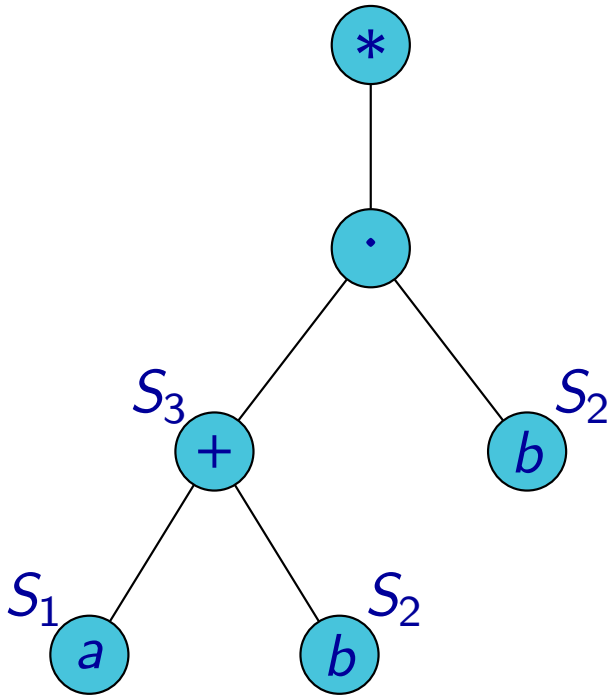
Příklad: Konstrukce bezkontextové gramatiky k regulárnímu výrazu $((a + b) \cdot b)^*$:



$S_2 \to b$
$S_1 \to a$

Převod regulárního výrazu na bezkontextovou gramatiku

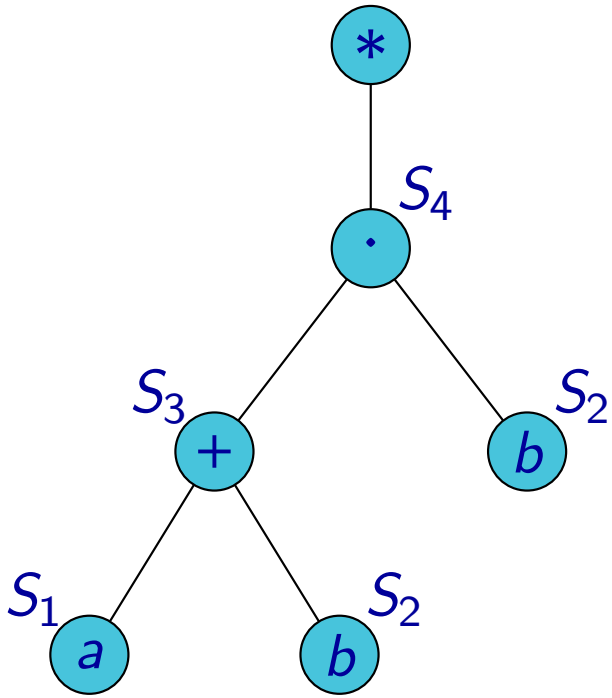
Příklad: Konstrukce bezkontextové gramatiky k regulárnímu výrazu $((a + b) \cdot b)^*$:



$$S_3 \to S_1 \mid S_2$$
$$S_2 \to b$$
$$S_1 \to a$$

Převod regulárního výrazu na bezkontextovou gramatiku

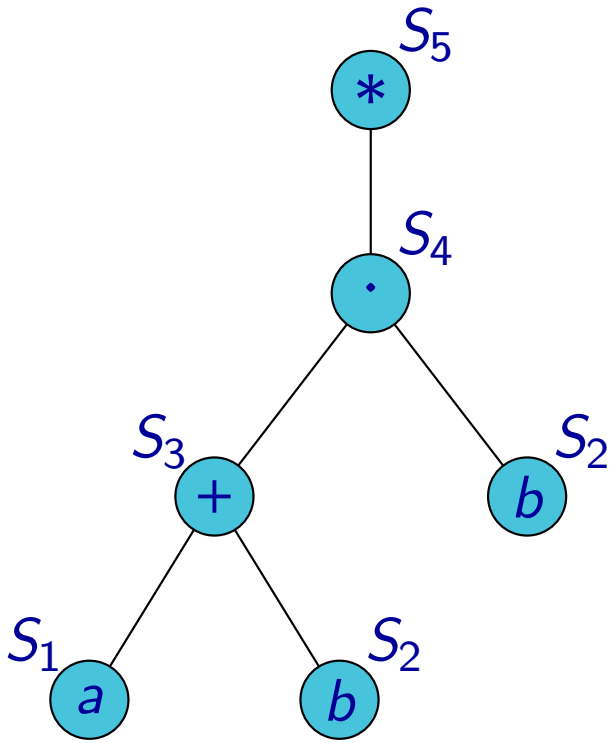
Příklad: Konstrukce bezkontextové gramatiky k regulárnímu výrazu $((a + b) \cdot b)^*$:



$$
\begin{aligned}
S_4 &\to S_3 S_2 \\
S_3 &\to S_1 \mid S_2 \\
S_2 &\to b \\
S_1 &\to a
\end{aligned}
$$

Převod regulárního výrazu na bezkontextovou gramatiku

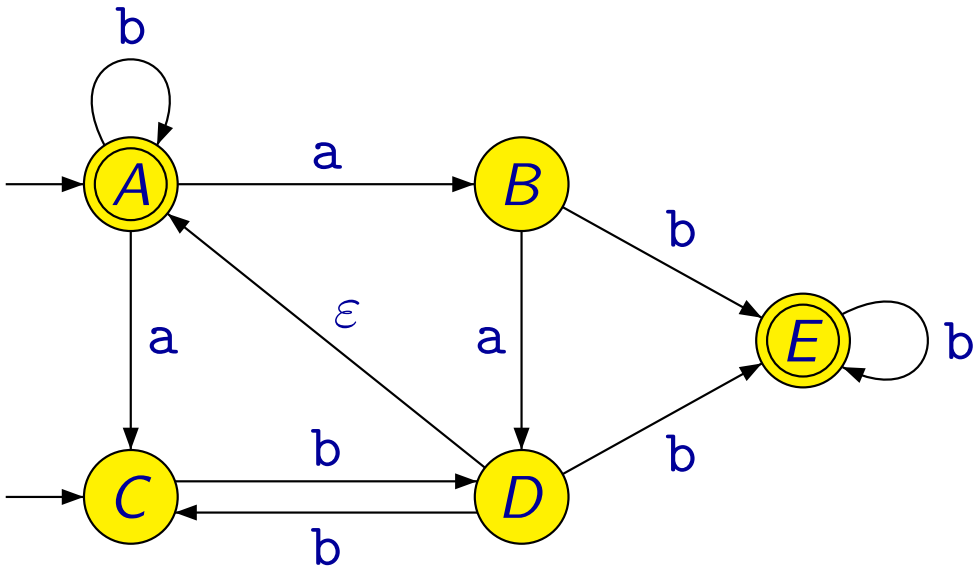
Příklad: Konstrukce bezkontextové gramatiky k regulárnímu výrazu $((a + b) \cdot b)^*$:



$$
\begin{aligned}
S_5 &\to \varepsilon \mid S_4 S_5 \\
S_4 &\to S_3 S_2 \\
S_3 &\to S_1 \mid S_2 \\
S_2 &\to b \\
S_1 &\to a
\end{aligned}
$$

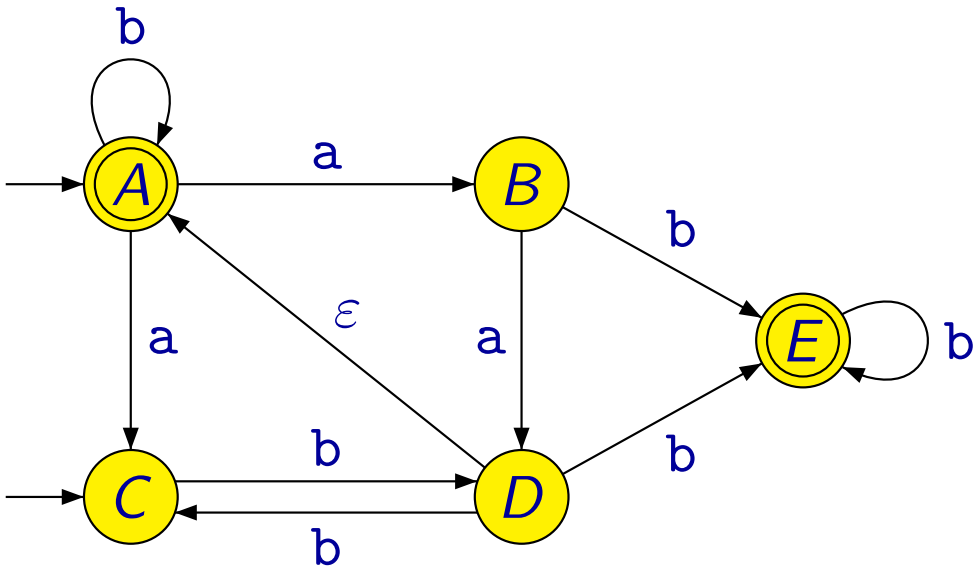
Převod konečného automatu na bezkontextovou gramatiku

Příklad:



Převod konečného automatu na bezkontextovou gramatiku

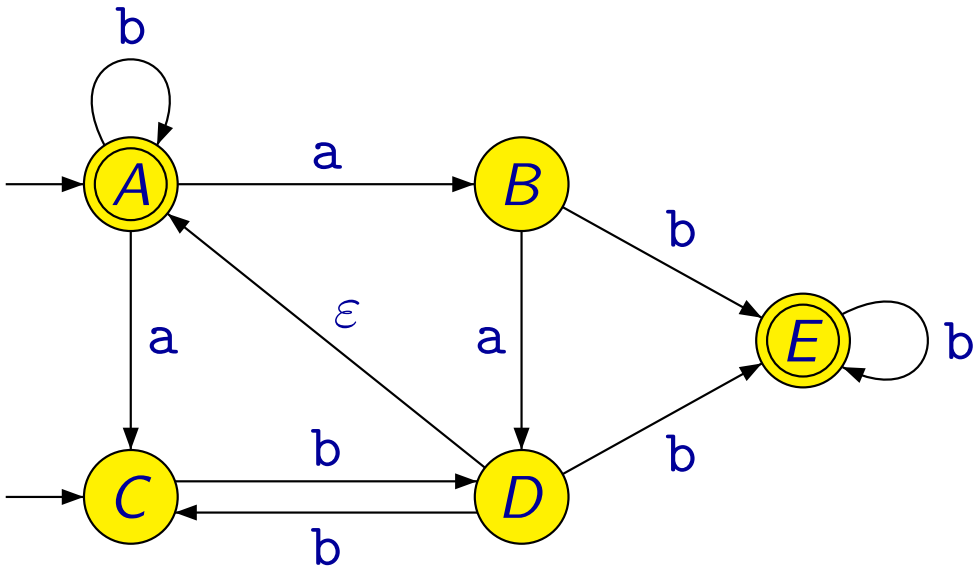
Příklad:



$S \to A \mid C$

Převod konečného automatu na bezkontextovou gramatiku

Příklad:

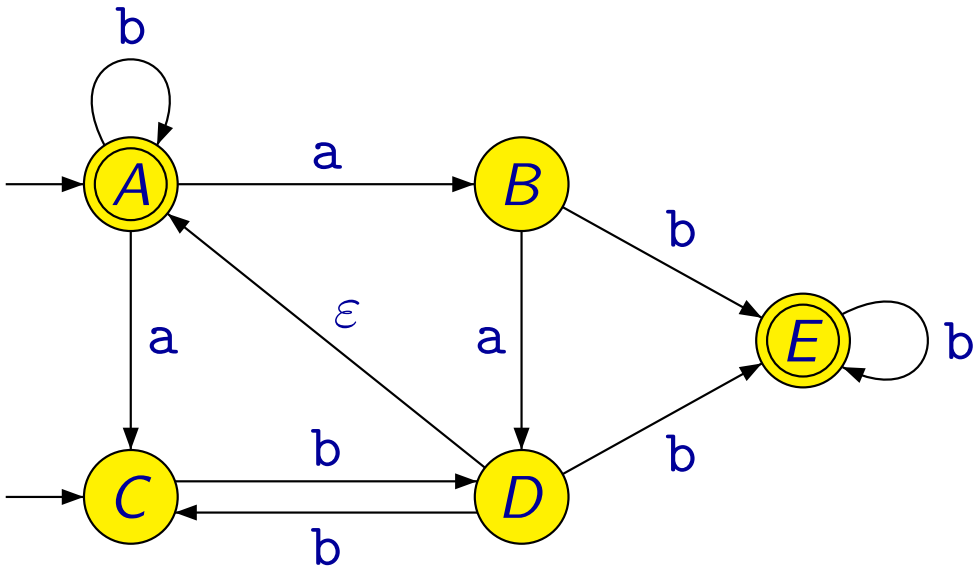


$$S \to A \mid C$$

$$A \to aB \mid aC \mid bA$$
$$B \to aD \mid bE$$
$$C \to bD$$
$$D \to bC \mid bE \mid A$$
$$E \to bE$$

Převod konečného automatu na bezkontextovou gramatiku

Příklad:



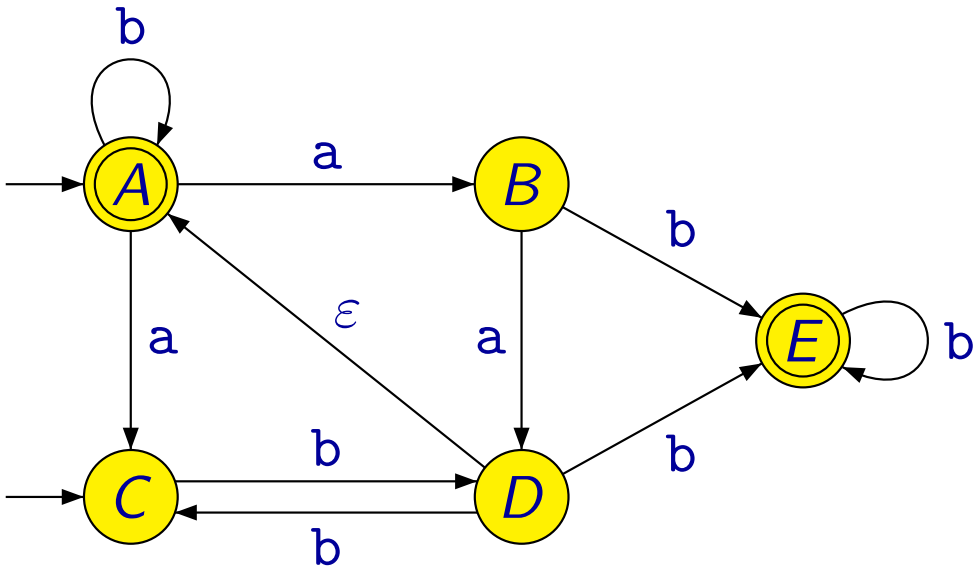
$$S \to A \mid C$$

$$A \to aB \mid aC \mid bA$$
$$B \to aD \mid bE$$
$$C \to bD$$
$$D \to bC \mid bE \mid A$$
$$E \to bE$$

$$A \to \varepsilon$$
$$E \to \varepsilon$$

Převod konečného automatu na bezkontextovou gramatiku

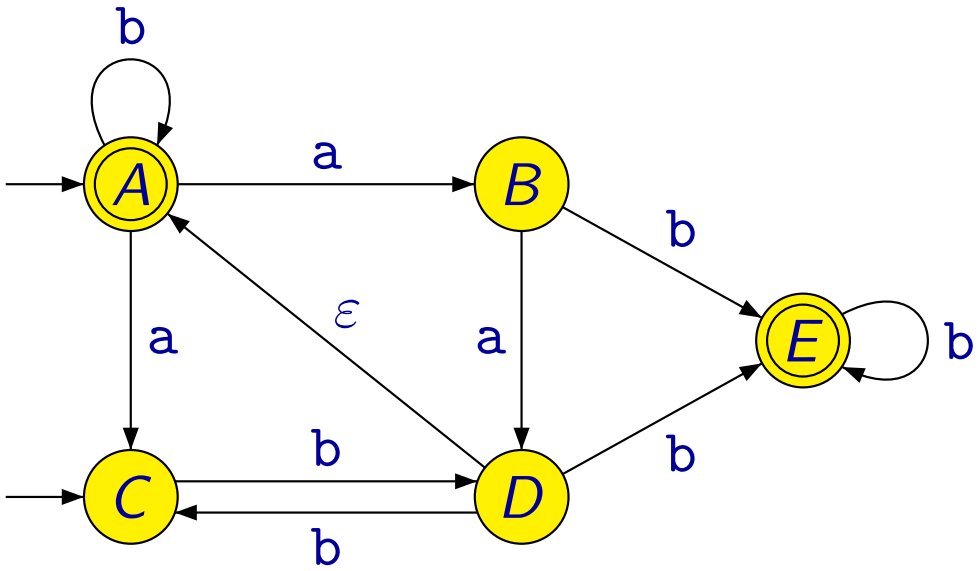
Příklad:



Alternativní konstrukce:

Převod konečného automatu na bezkontextovou gramatiku

Příklad:

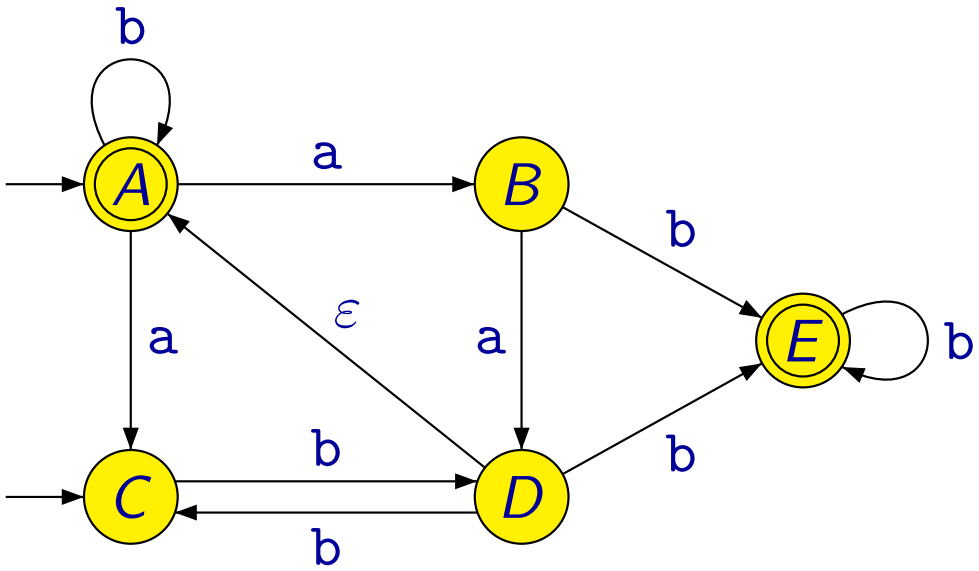


Alternativní konstrukce:

$$S \to A \mid E$$

Převod konečného automatu na bezkontextovou gramatiku

Příklad:



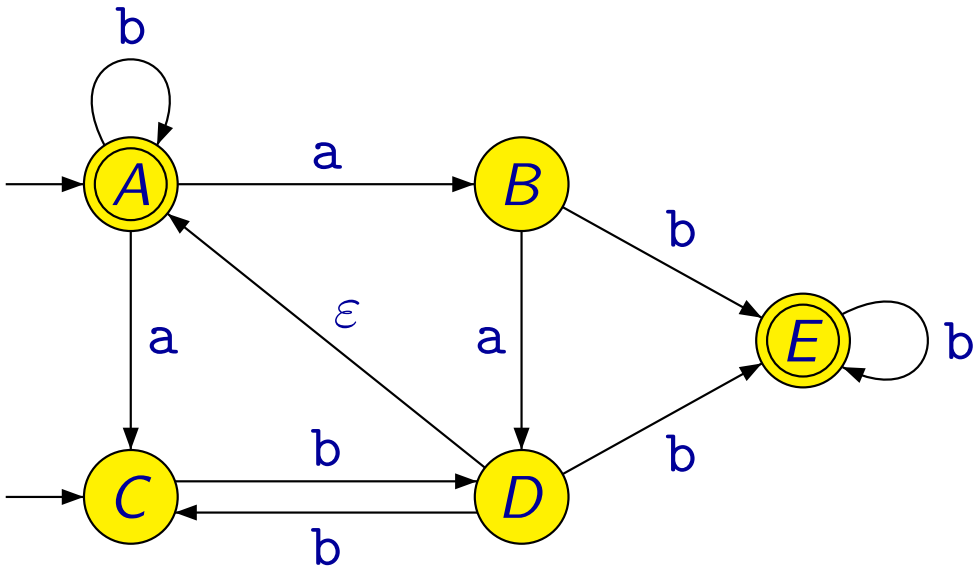
Alternativní konstrukce:

$$S \to A \mid E$$

$$A \to Ab \mid D$$
$$B \to Aa$$
$$C \to Aa \mid Db$$
$$D \to Ba \mid Cb$$
$$E \to Bb \mid Db \mid Eb$$

Převod konečného automatu na bezkontextovou gramatiku

Příklad:



Alternativní konstrukce:

$$S \to A \mid E$$

$$A \to Ab \mid D$$
$$B \to Aa$$
$$C \to Aa \mid Db$$
$$D \to Ba \mid Cb$$
$$E \to Bb \mid Db \mid Eb$$

$$A \to \varepsilon$$
$$C \to \varepsilon$$

Regulární gramatiky

Definice

Gramatika $\mathcal{G} = (\Pi, \Sigma, S, P)$ je **pravá regulární gramatika**, jestliže všechna pravidla v P jsou některého z následujících tvarů (kde $A, B \in \Pi$, $a \in \Sigma$):

- $A \to B$
- $A \to aB$
- $A \to \varepsilon$

Definice

Gramatika $\mathcal{G} = (\Pi, \Sigma, S, P)$ je **levá regulární gramatika**, jestliže všechna pravidla v P jsou některého z následujících tvarů (where $A, B \in \Pi$, $a \in \Sigma$):

- $A \to B$
- $A \to Ba$
- $A \to \varepsilon$

Regulární gramatiky

Definice

Gramatika $\mathcal{G}$ je **regulární**, jestliže je pravá regulární nebo levá regulární.

Poznámka: Někdy se též uvádí poněkud obecnější definice pravé (resp. levé) regulární gramatiky, kde jsou povolena pravidla následujících tvarů:

- $A \to wB$ (resp. $A \to Bw$)
- $A \to w$

kde $A, B \in \Pi$, $w \in \Sigma^*$.

Taková pravidla je možné snadno „rozložit“ na pravidla odpovídají dříve uvedené definici.

Příklad: Pravidlo $A \to abbB$ je možno nahradit pravidly

$$A \to aX_1 \qquad X_1 \to bX_2 \qquad X_2 \to bB$$

kde X_1, X_2 jsou nové neterminály nepoužité nikde jinde v gramatice.

Regulární gramatiky

Tvrzení

Ke každému regulárnímu jazyku L existuje levá regulární gramatika $\mathcal{G}$ taková, že $\mathcal{L}(\mathcal{G}) = L$, a pravá regulární gramatika $\mathcal{G}'$ taková, že $\mathcal{L}(\mathcal{G}') = L$.

Tvrzení

Ke každé regulární gramatice $\mathcal{G}$ existuje konečný automat $\mathcal{A}$ takový, že $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{G})$.

Zásobníkové automaty

Zásobníkový automat

Příklad: Vezměme si jazyk nad abecedou $\Sigma = \{(,), [,], <, >\}$ tvořený „správně uzávorkovanými“ sekvencemi, tj. sekvencemi, kde každá levá závorka má odpovídající pravou a naopak každá pravá má odpovídající levou, přičemž se závorky „nekříží“ (jako třeba ve slově <[>]).

Tento jazyk je možné popsat bezkontextovou gramatikou

$$A \to \varepsilon \mid (A) \mid [A] \mid <A> \mid AA$$

Typický příklad slova, které patří do tohoto jazyka:

$$<[]([()[<>]])>[]$$

Není těžké ukázat, že tento jazyk není regulární.

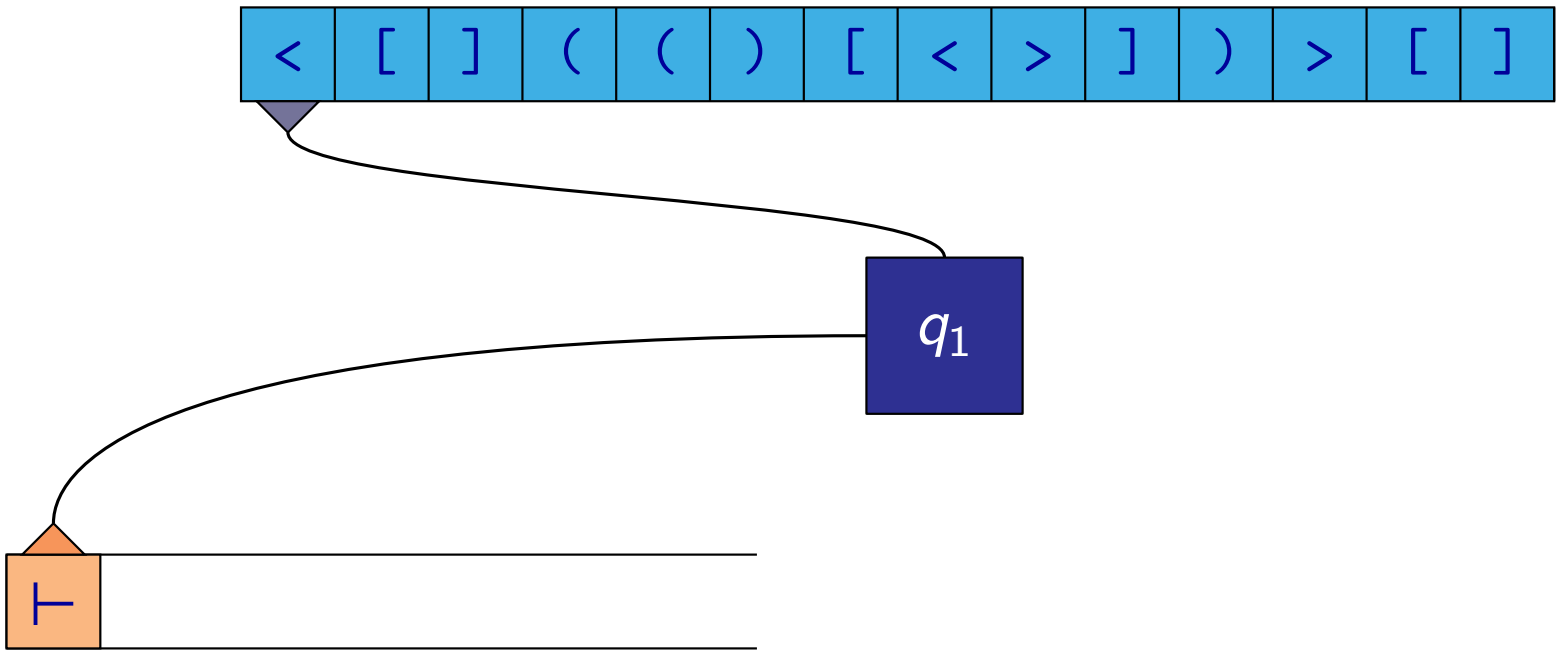
Zásobníkový automat

Chtěli bychom navrhnout zařízení podobné konečnému automatu, které by bylo schopno rozpoznávat slova z tohoto jazyka.

Jako vhodná možnost se nabízí využít při tomto rozpoznávání (neomezeně velký) **zásobník**.

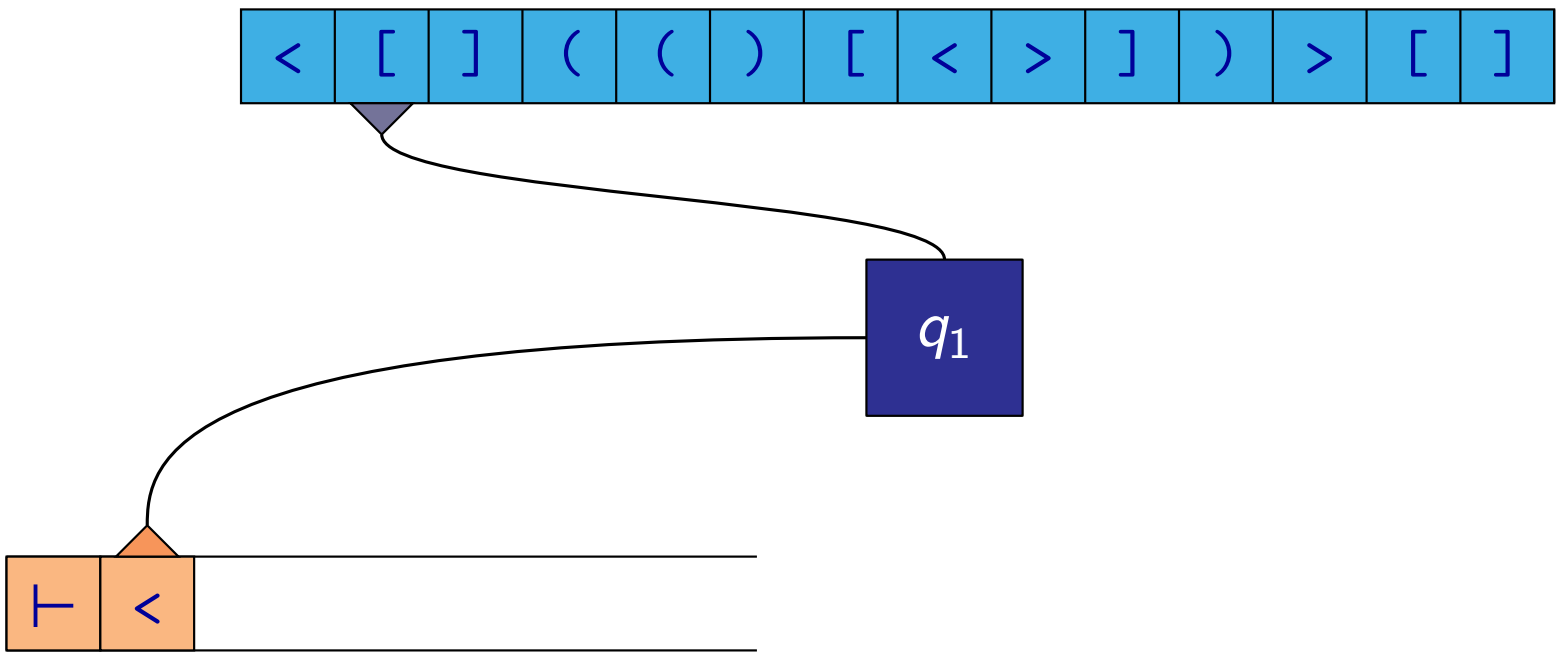
Zásobníkový automat

- Slovo `<[](()[<>])>[]` patří do jazyka.



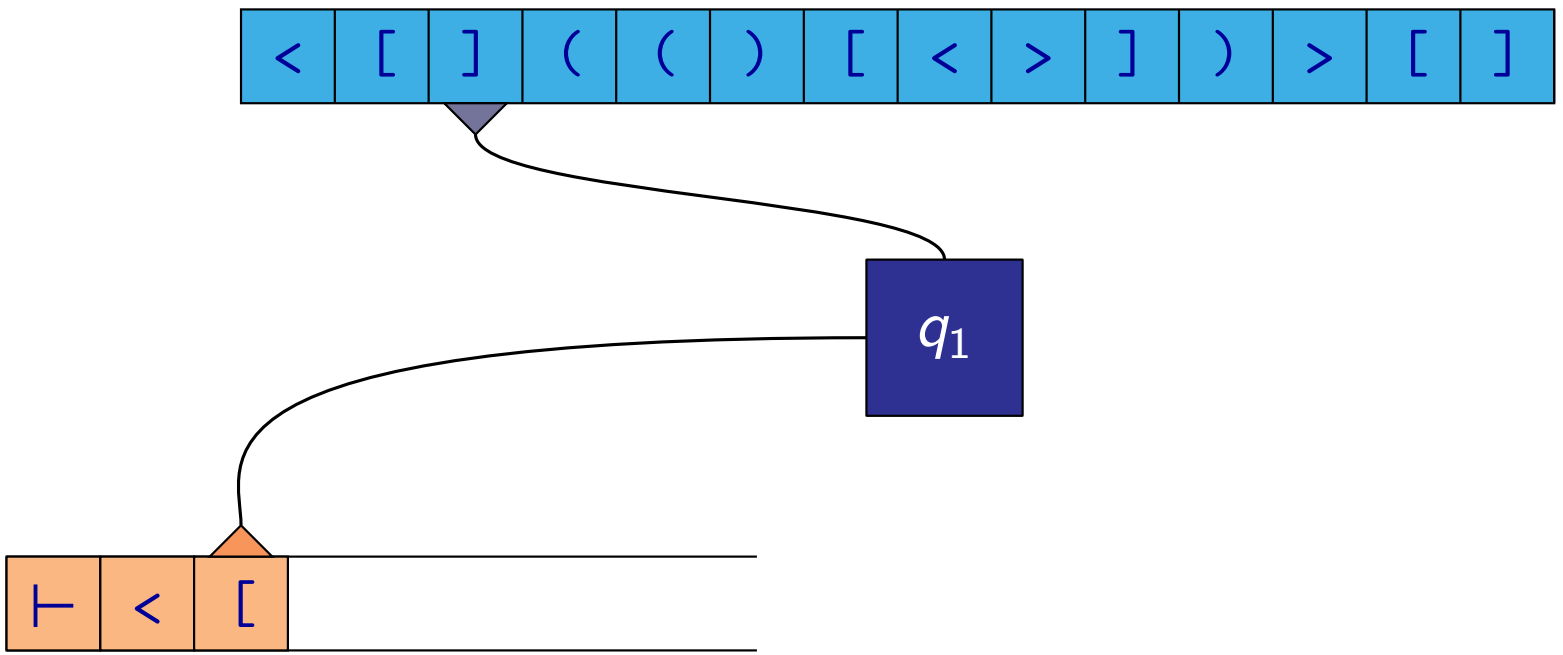
Zásobníkový automat

- Slovo <[](()[<>])>[] patří do jazyka.



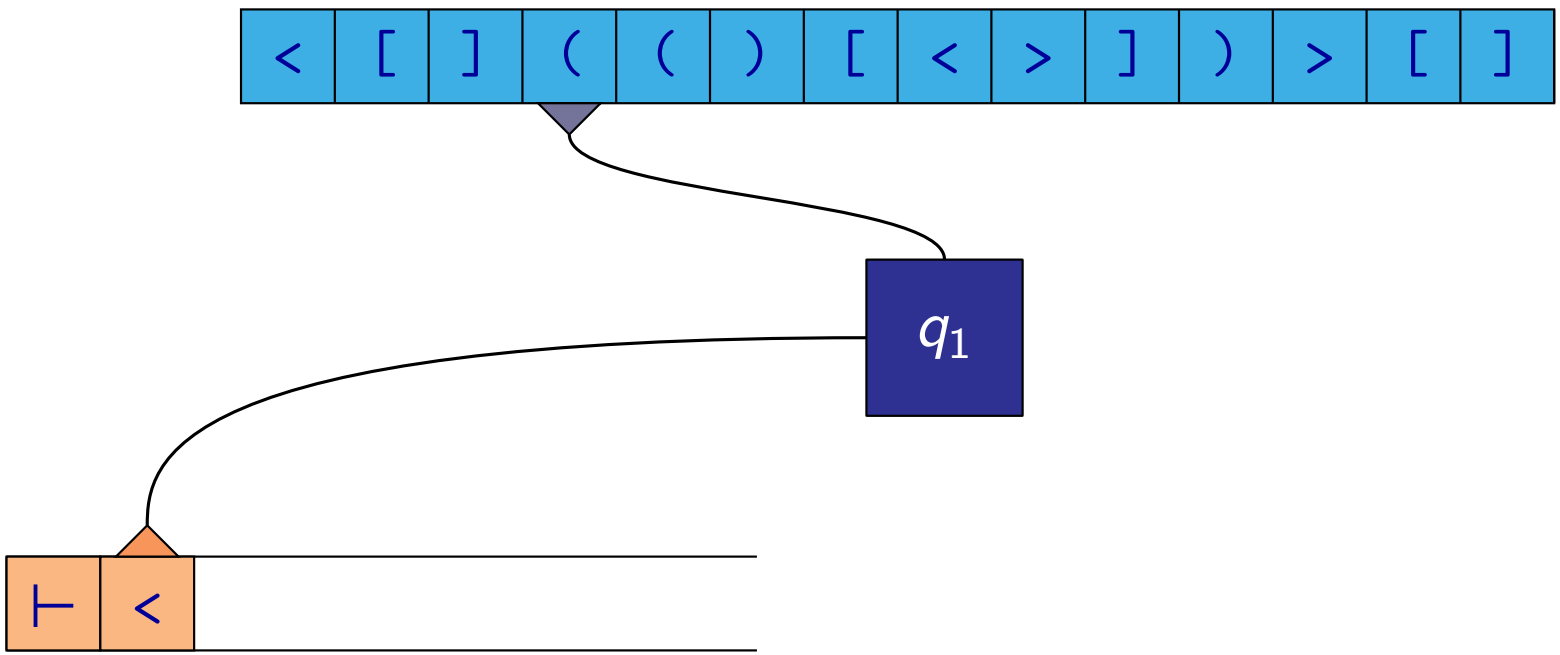
Zásobníkový automat

- Slovo <[](()[<>])>[] patří do jazyka.



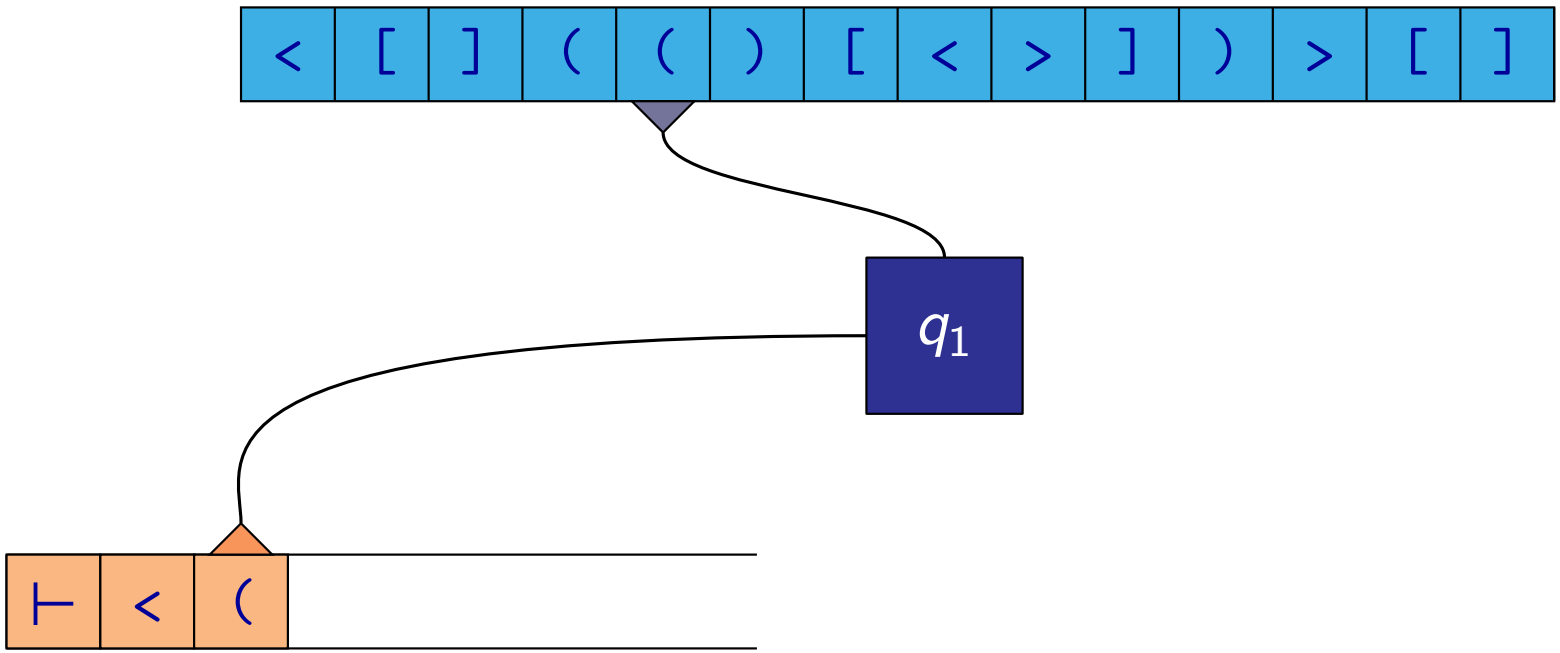
Zásobníkový automat

- Slovo <[](()[<>])>[] patří do jazyka.



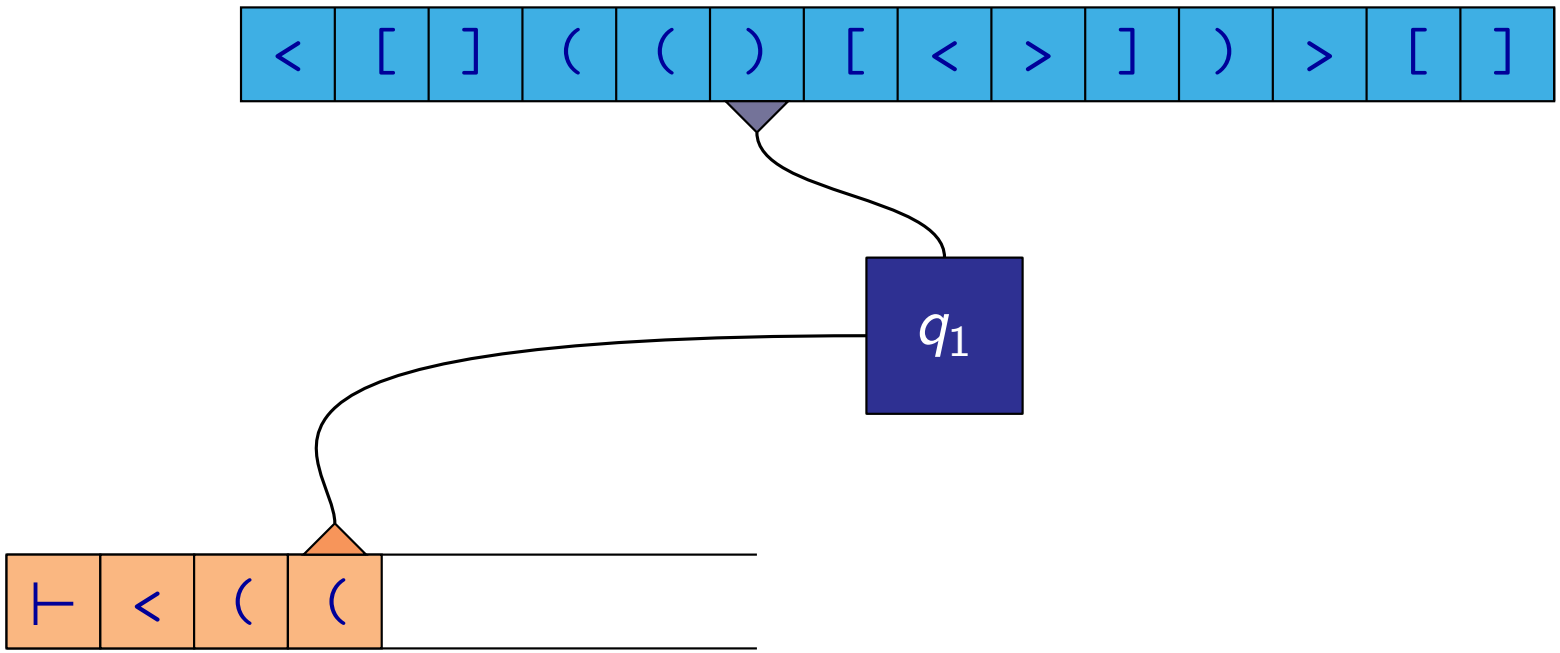
Zásobníkový automat

- Slovo <[](()[<>])>[] patří do jazyka.



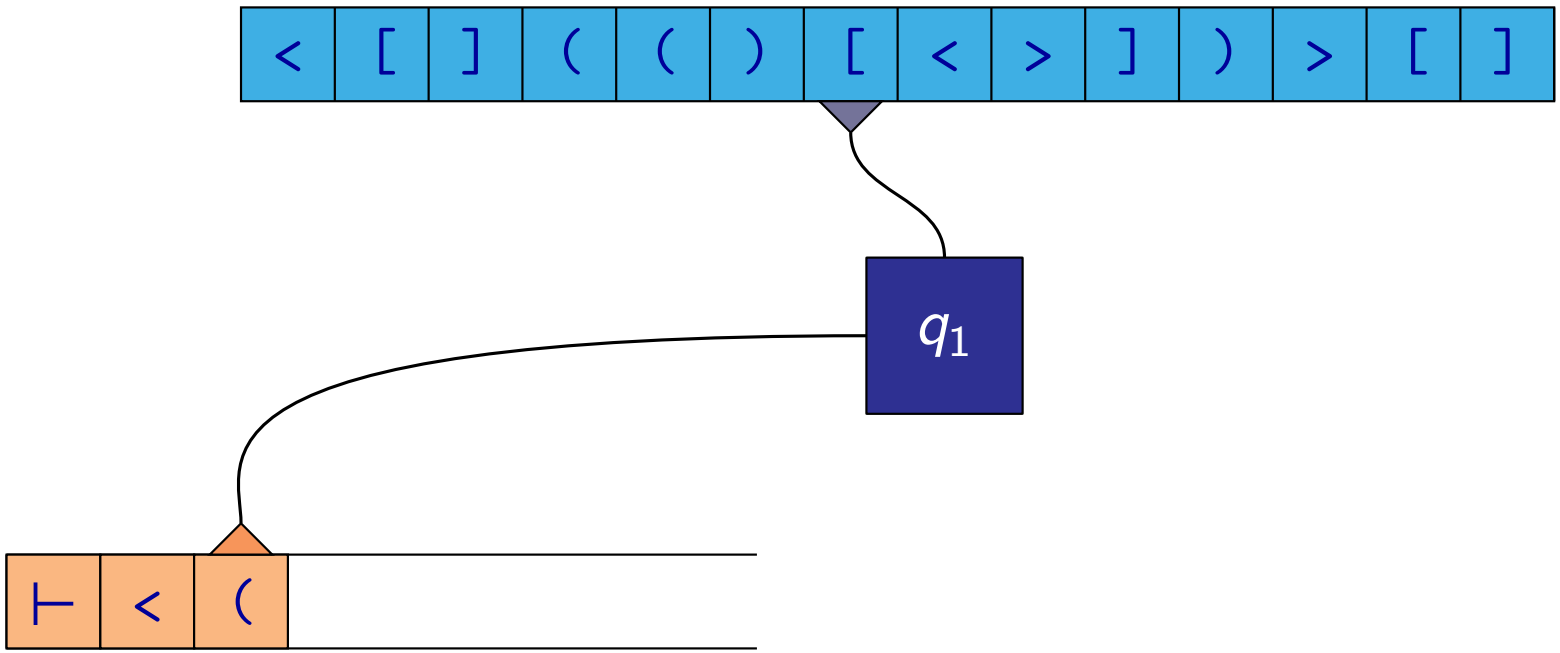
Zásobníkový automat

- Slovo <[](()[<>])>[] patří do jazyka.



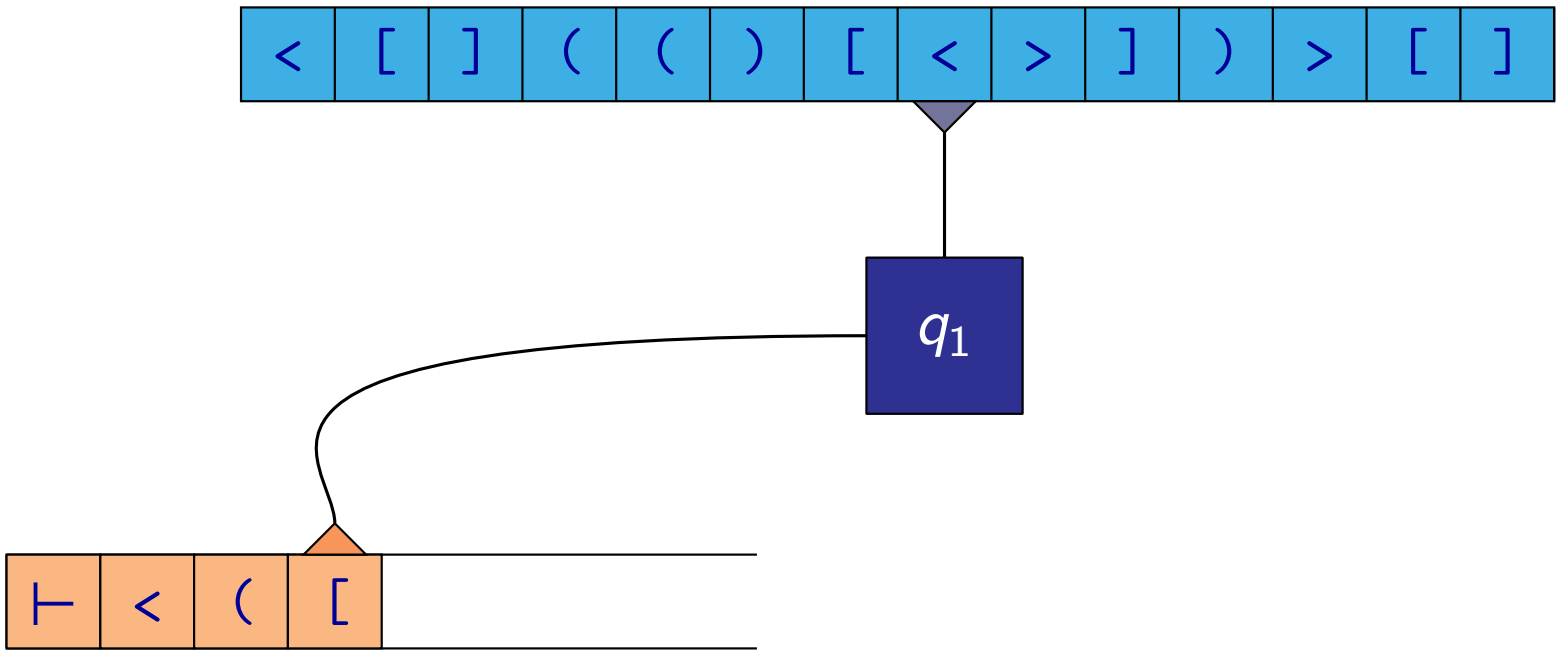
Zásobníkový automat

- Slovo <[](()[<>])>[] patří do jazyka.



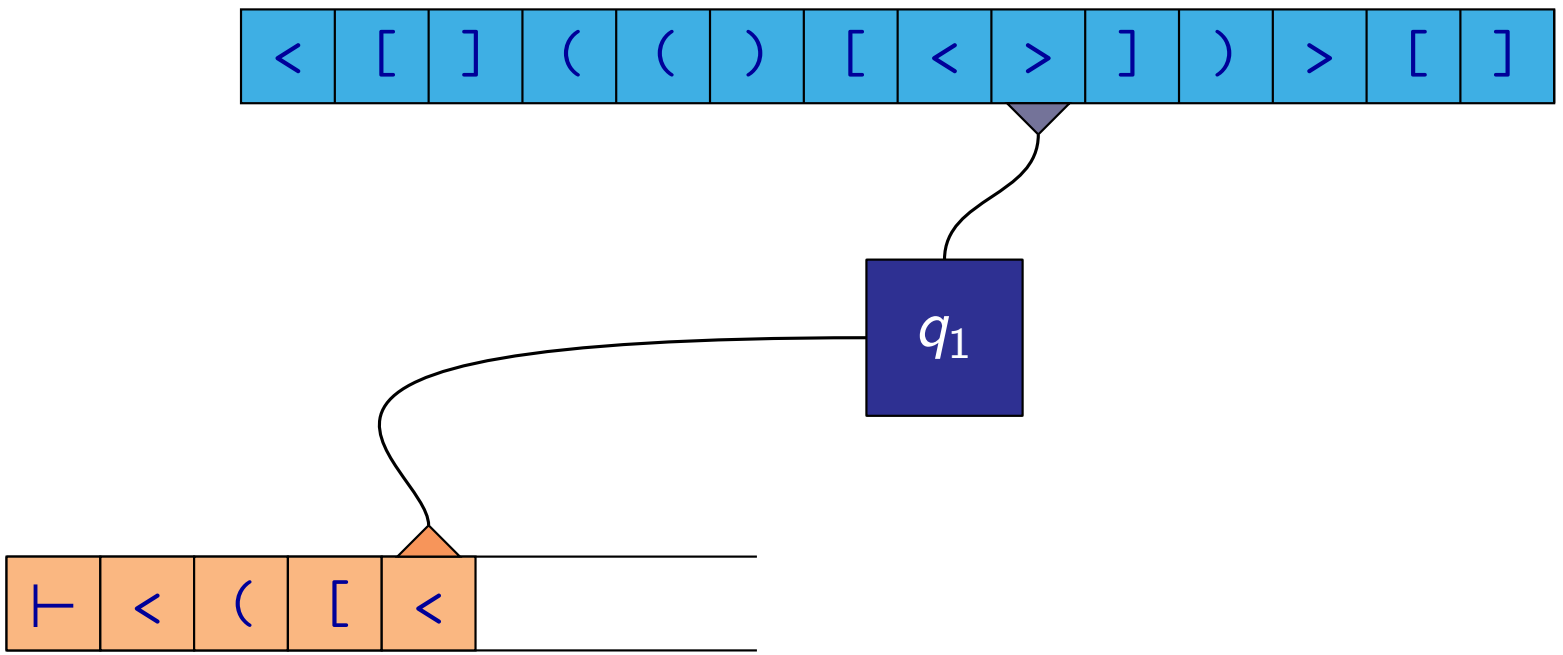
Zásobníkový automat

- Slovo <[](()[<>])>[] patří do jazyka.



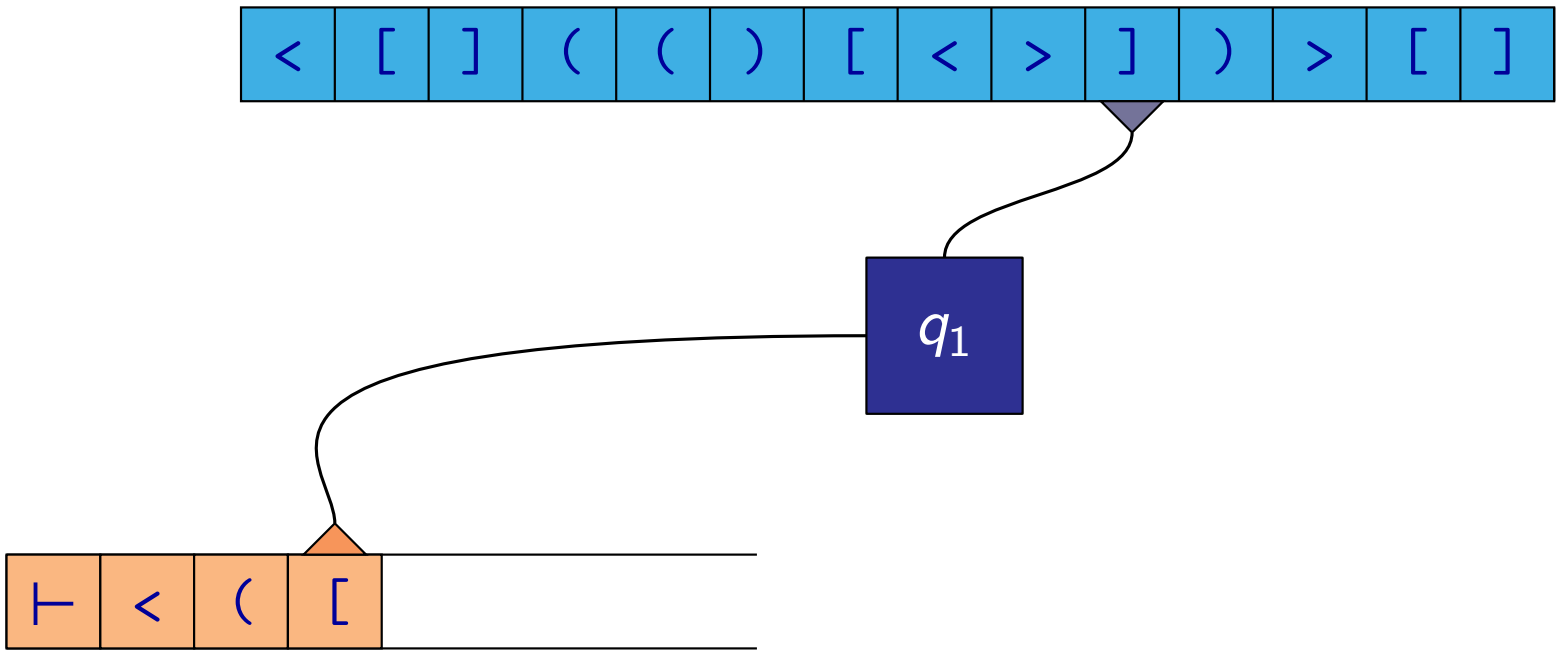
Zásobníkový automat

- Slovo <[](()[<>])>[] patří do jazyka.



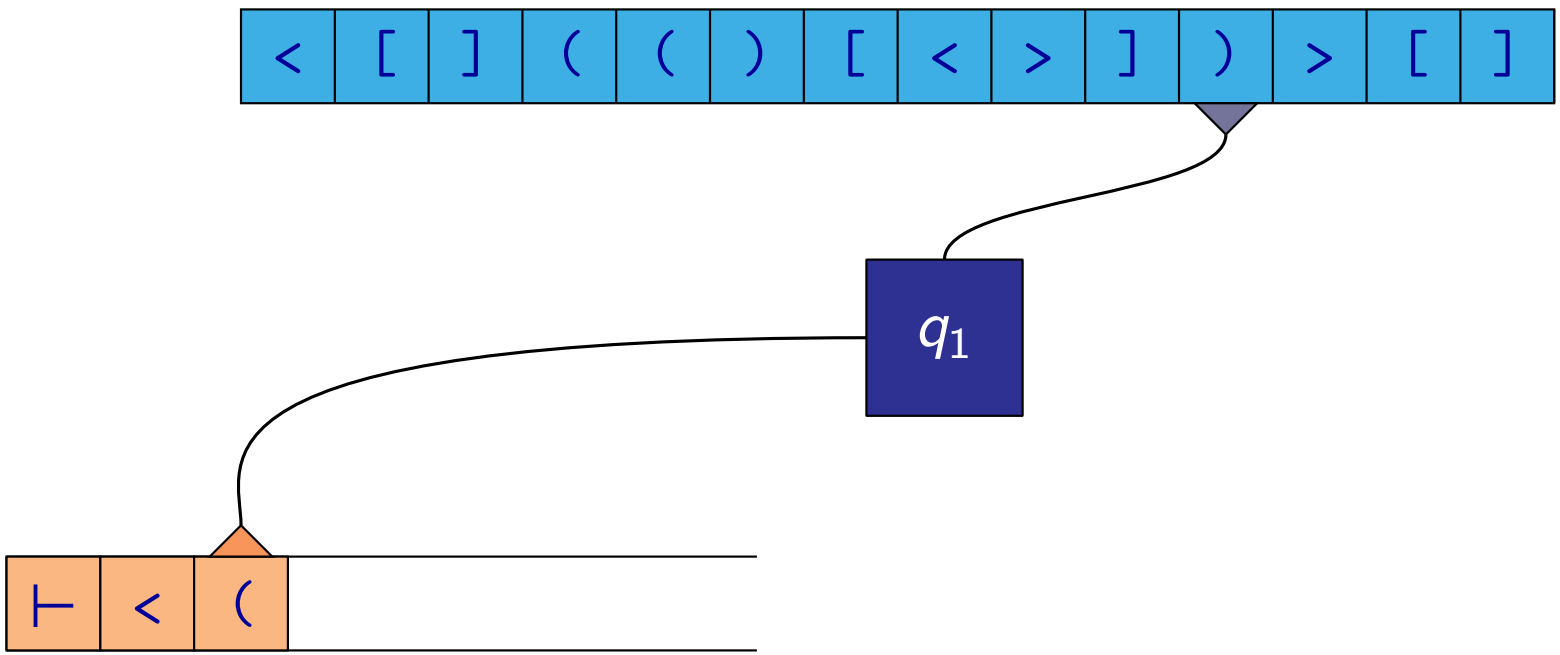
Zásobníkový automat

- Slovo <[](()[<>])>[] patří do jazyka.



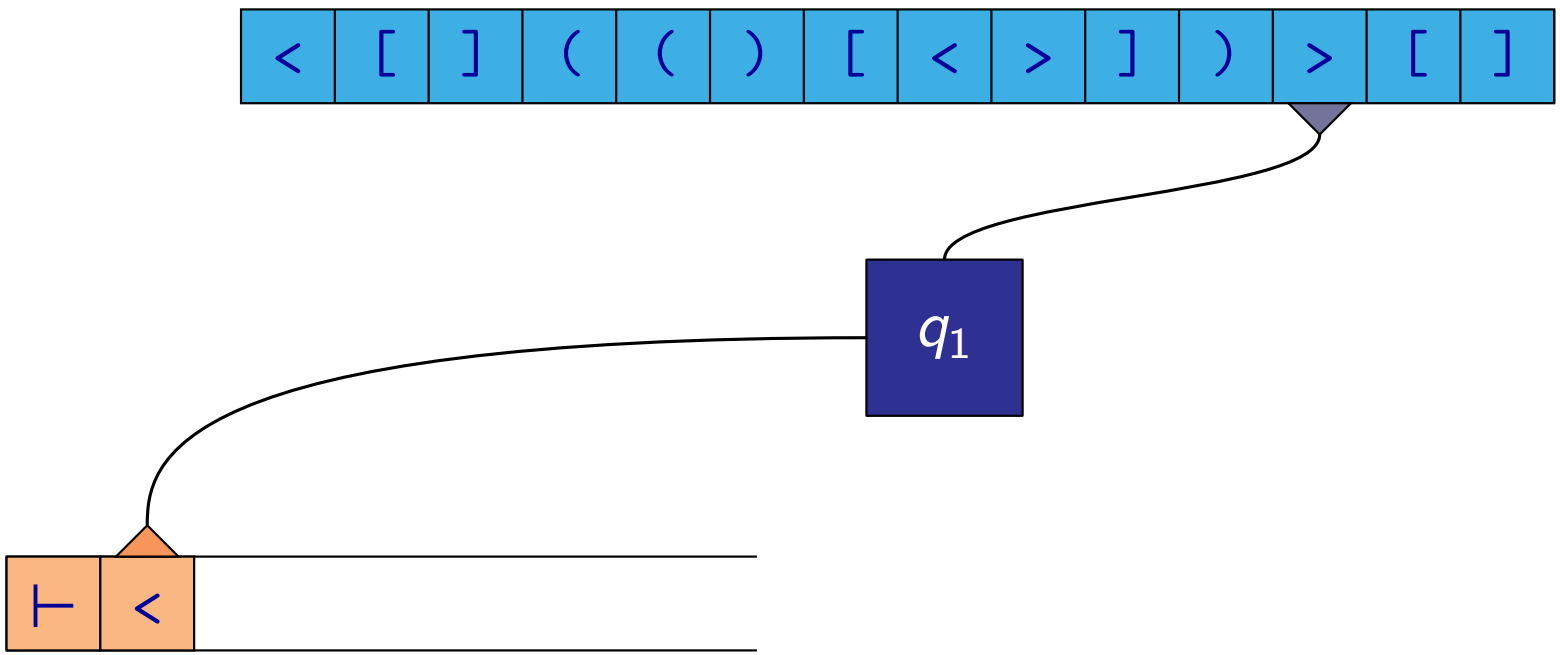
Zásobníkový automat

- Slovo <[](()[<>])>[] patří do jazyka.



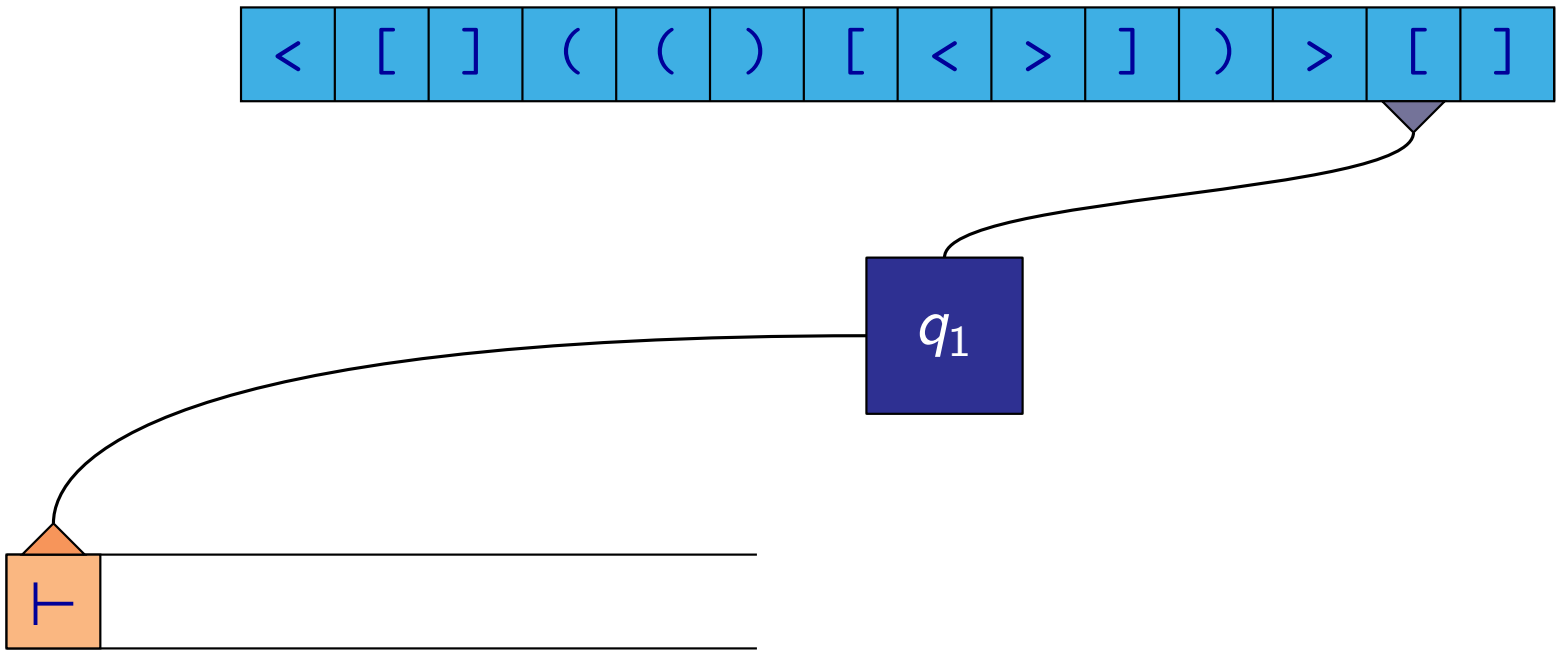
Zásobníkový automat

- Slovo <[](()[<>])>[] patří do jazyka.



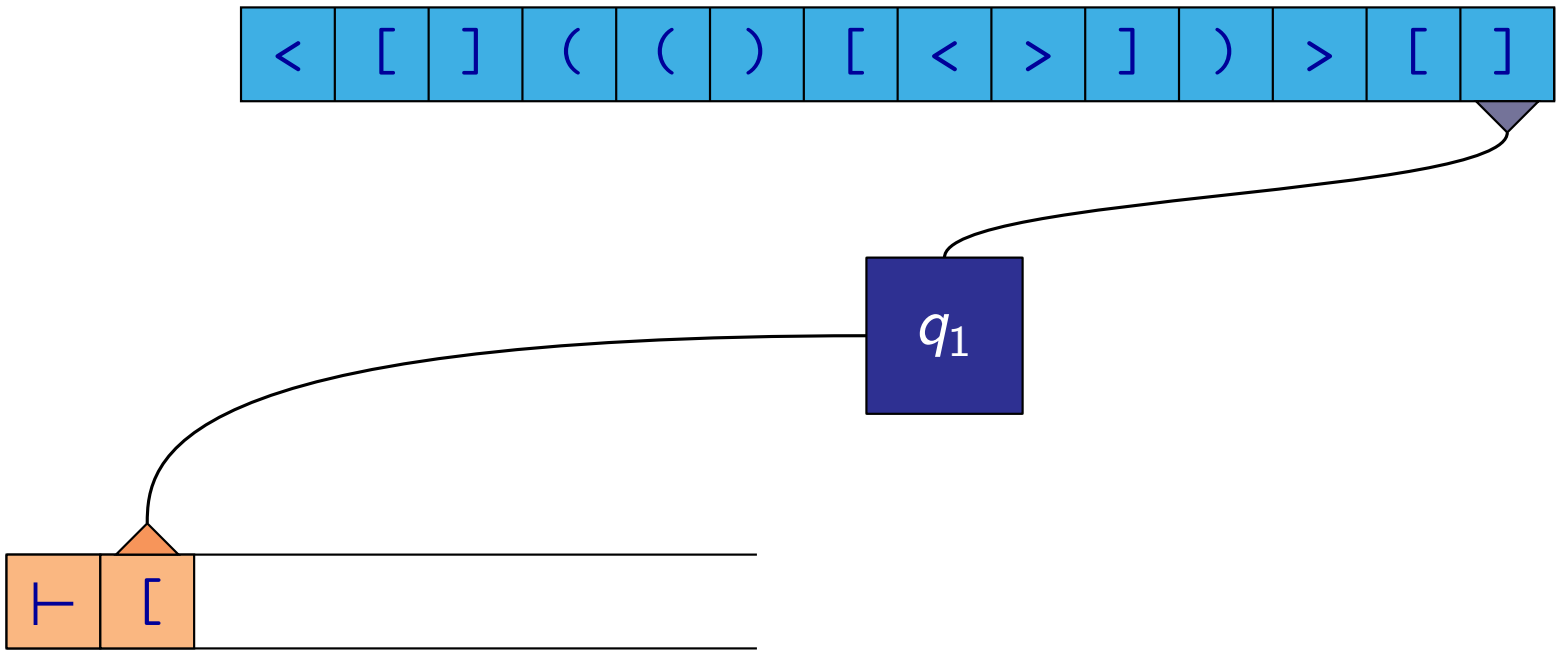
Zásobníkový automat

- Slovo <[](()[<>])>[] patří do jazyka.



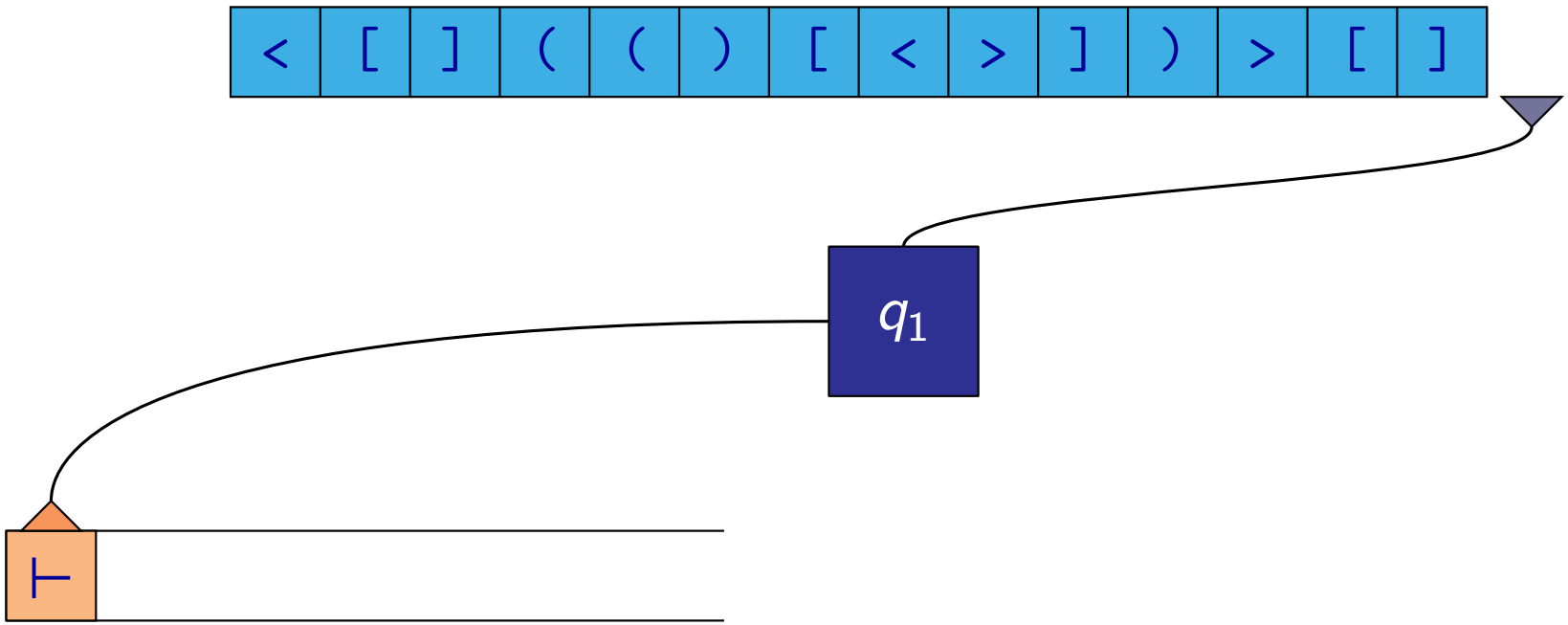
Zásobníkový automat

- Slovo <[](()[<>])>[] patří do jazyka.



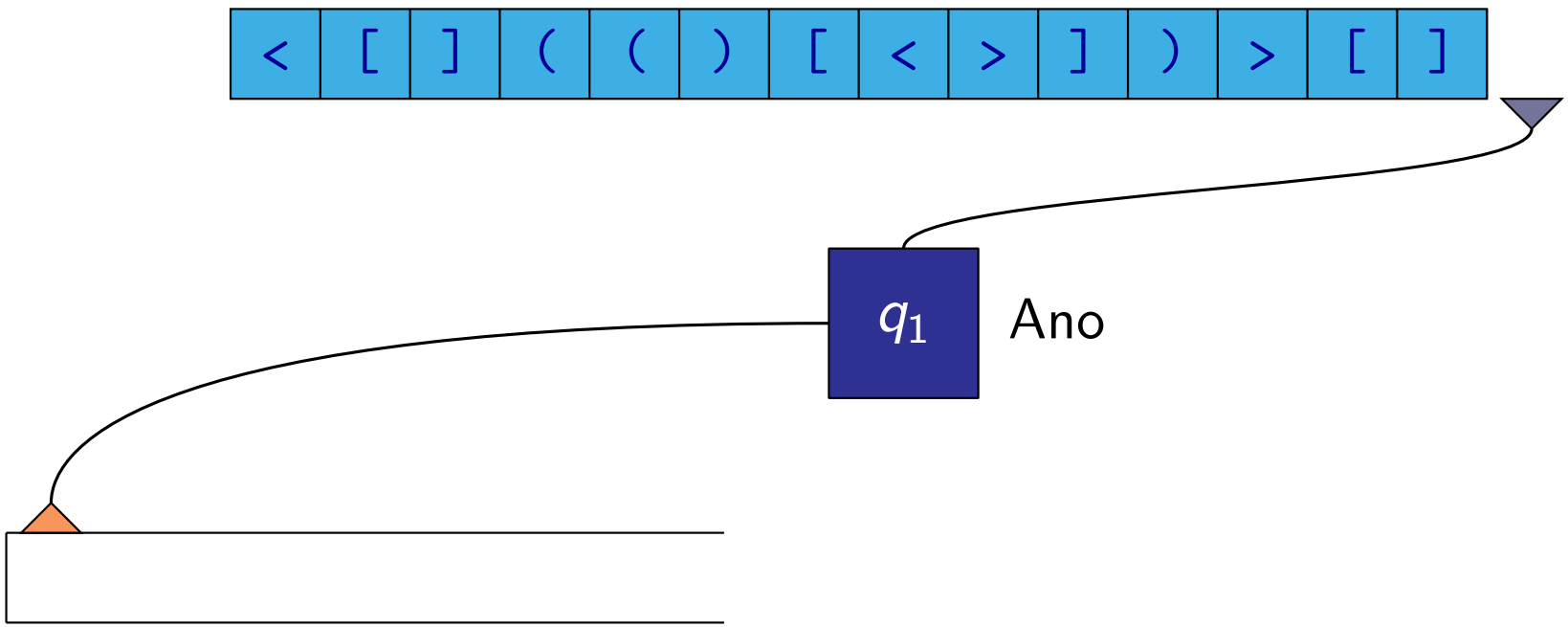
Zásobníkový automat

- Slovo `<[](()[<>])>[]` patří do jazyka.



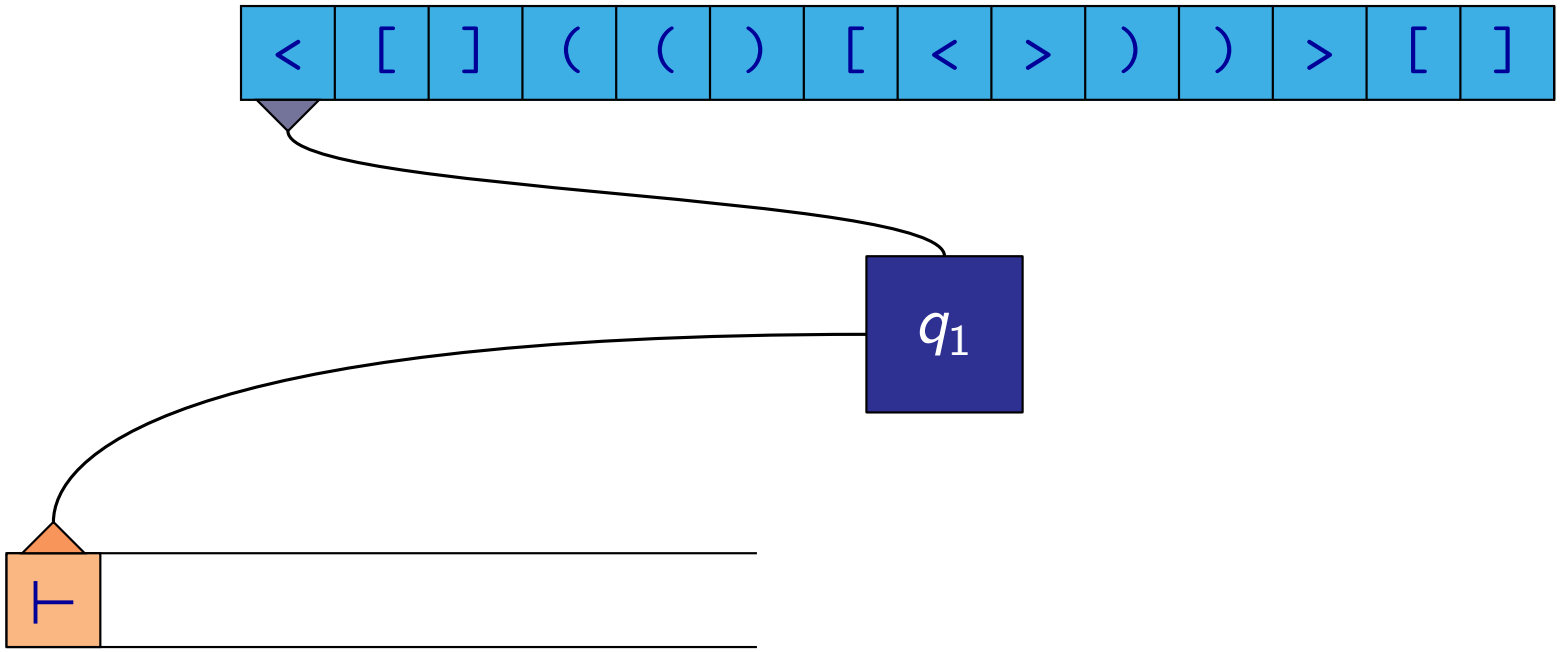
Zásobníkový automat

- Slovo `<[](()[<>])>[]` patří do jazyka.
- Automat přečetl celé slovo a skončil s prázdným zásobníkem, takže slovo přijal.



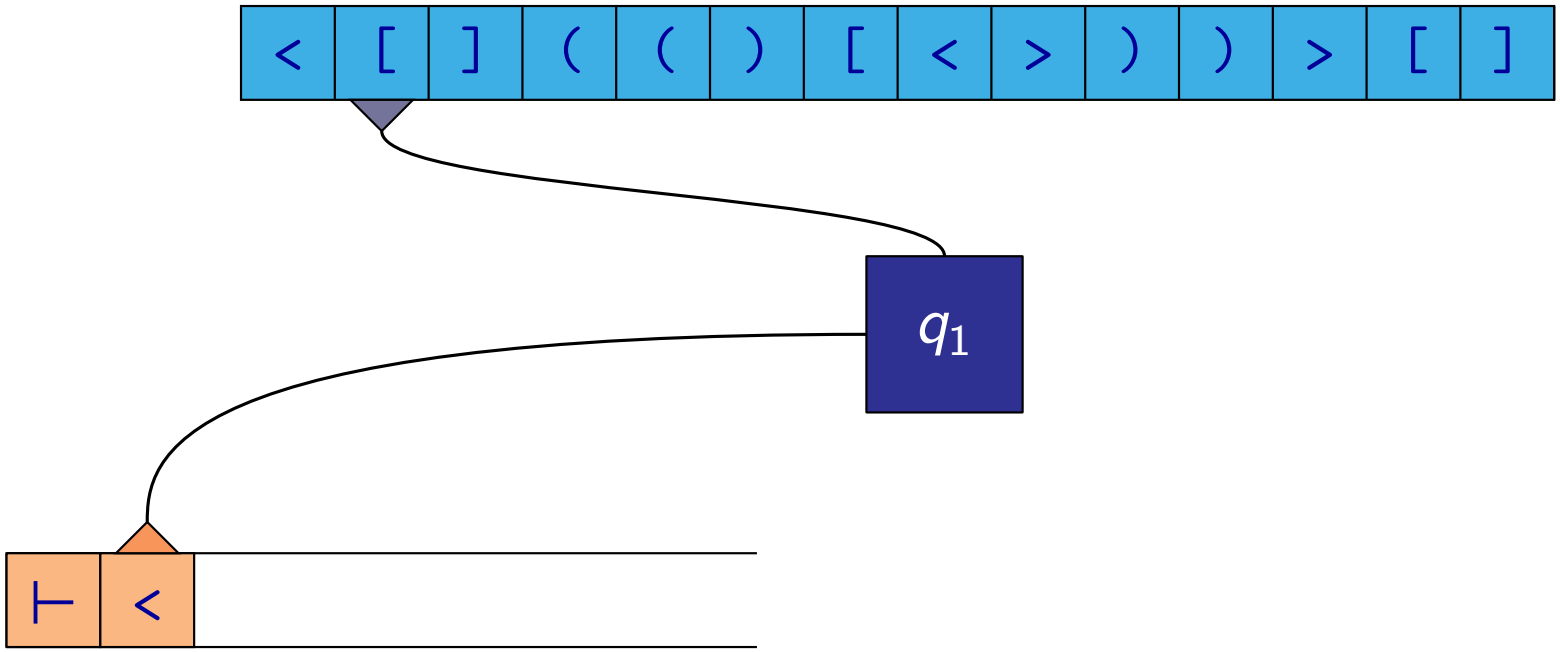
Zásobníkový automat

- Slovo <[](()[<>))>[] nepatří do jazyka.



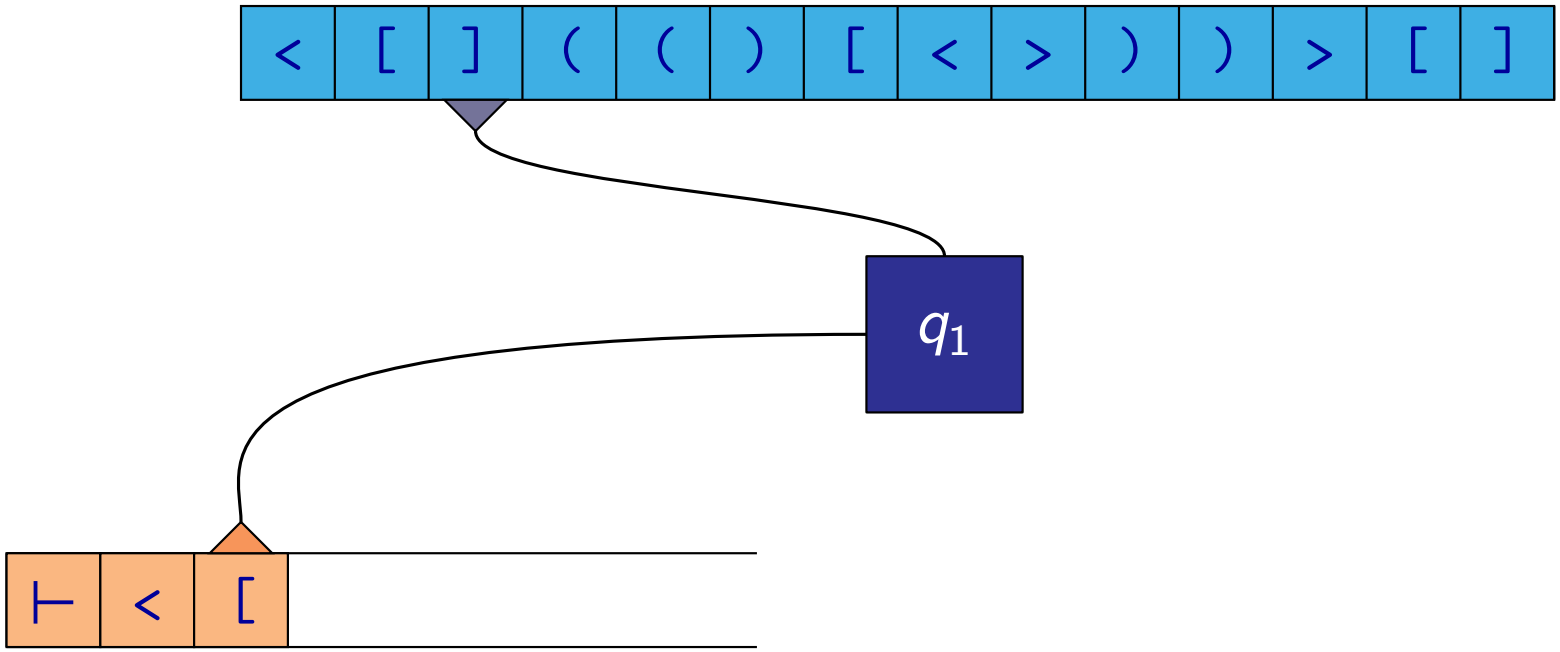
Zásobníkový automat

- Slovo `<[](()[<>))>[]` nepatří do jazyka.



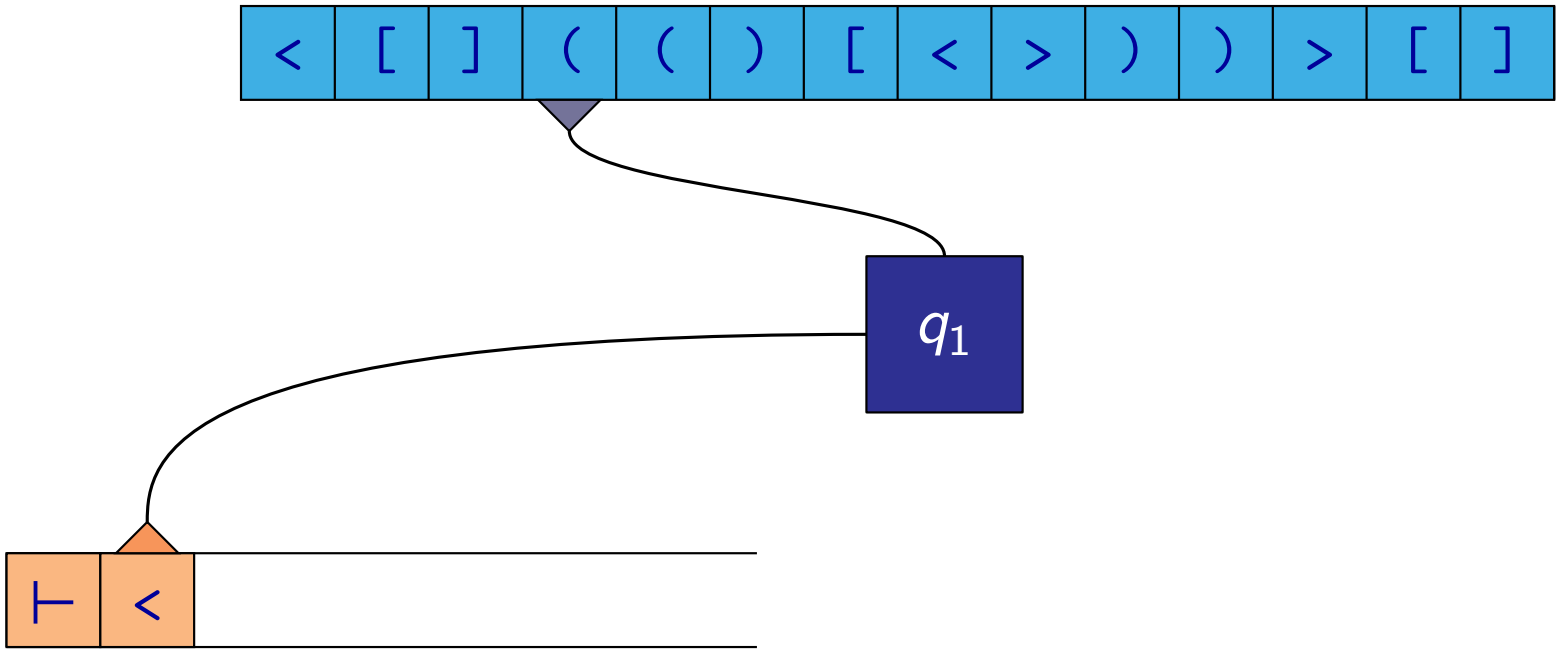
Zásobníkový automat

- Slovo <[](()[<>))>[] nepatří do jazyka.



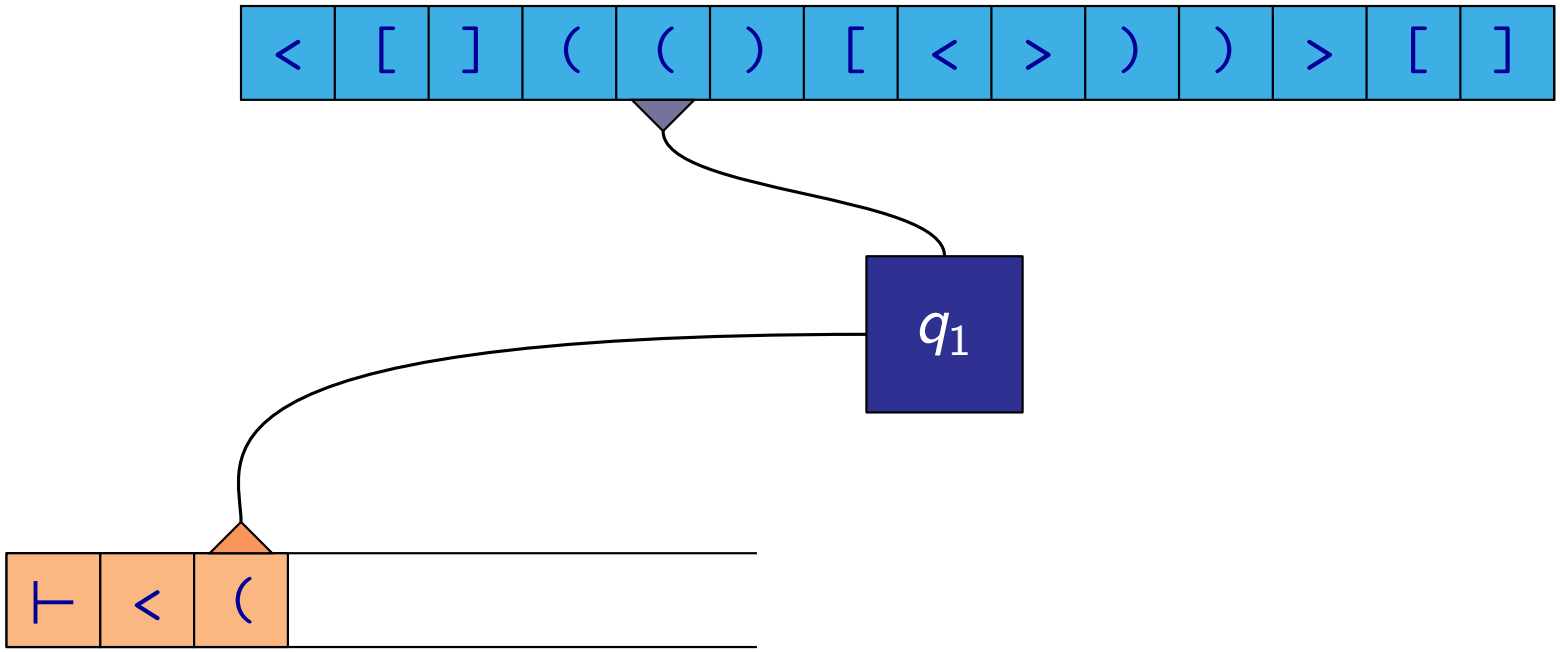
Zásobníkový automat

- Slovo `<[](()[<>))>[]` nepatří do jazyka.



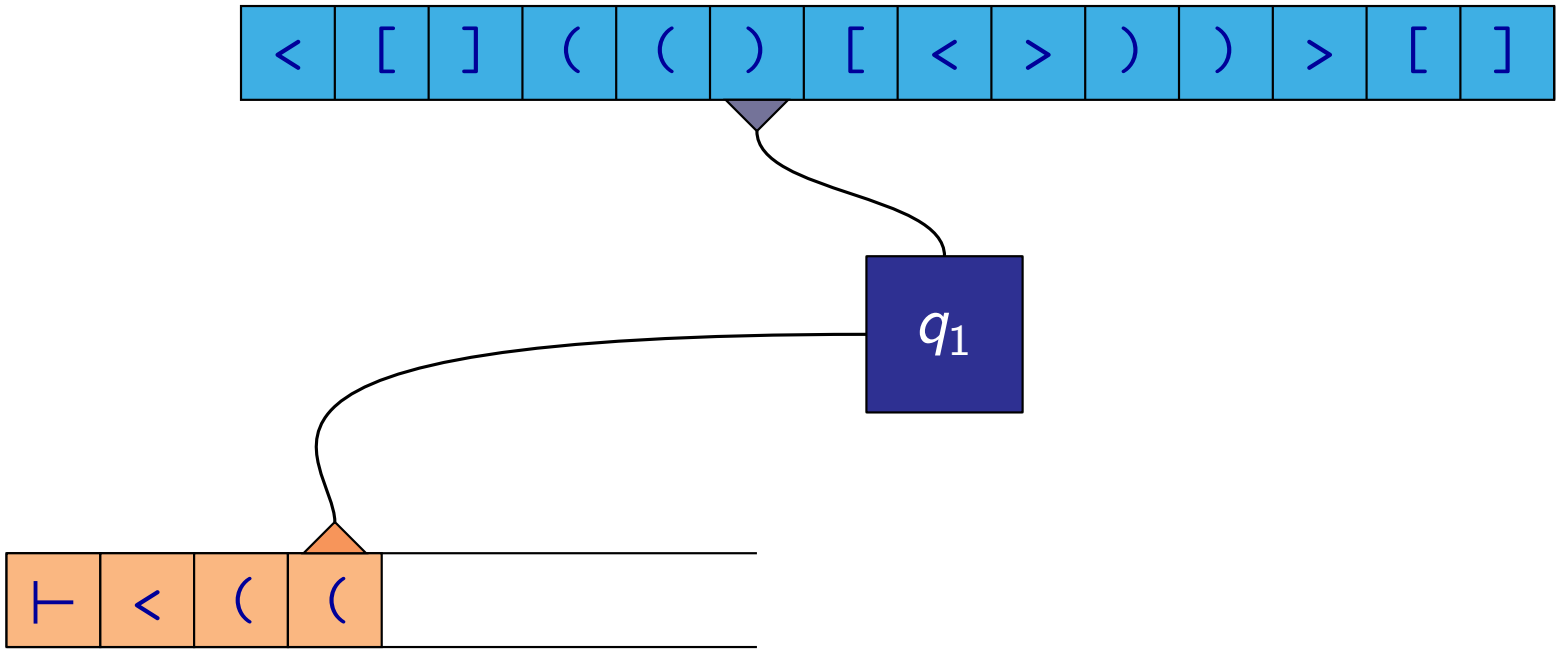
Zásobníkový automat

- Slovo <[](()[<>))>[] nepatří do jazyka.



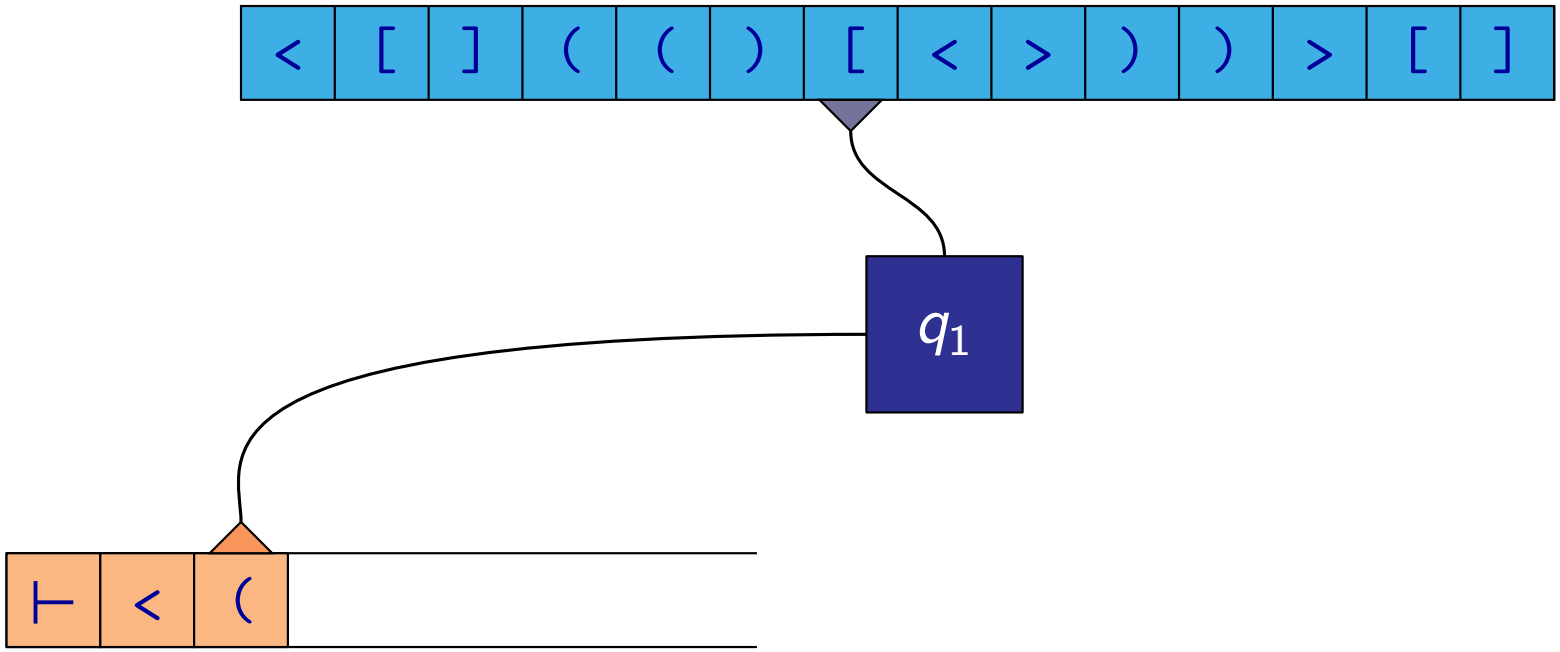
Zásobníkový automat

- Slovo <[](()[<>))>[] nepatří do jazyka.



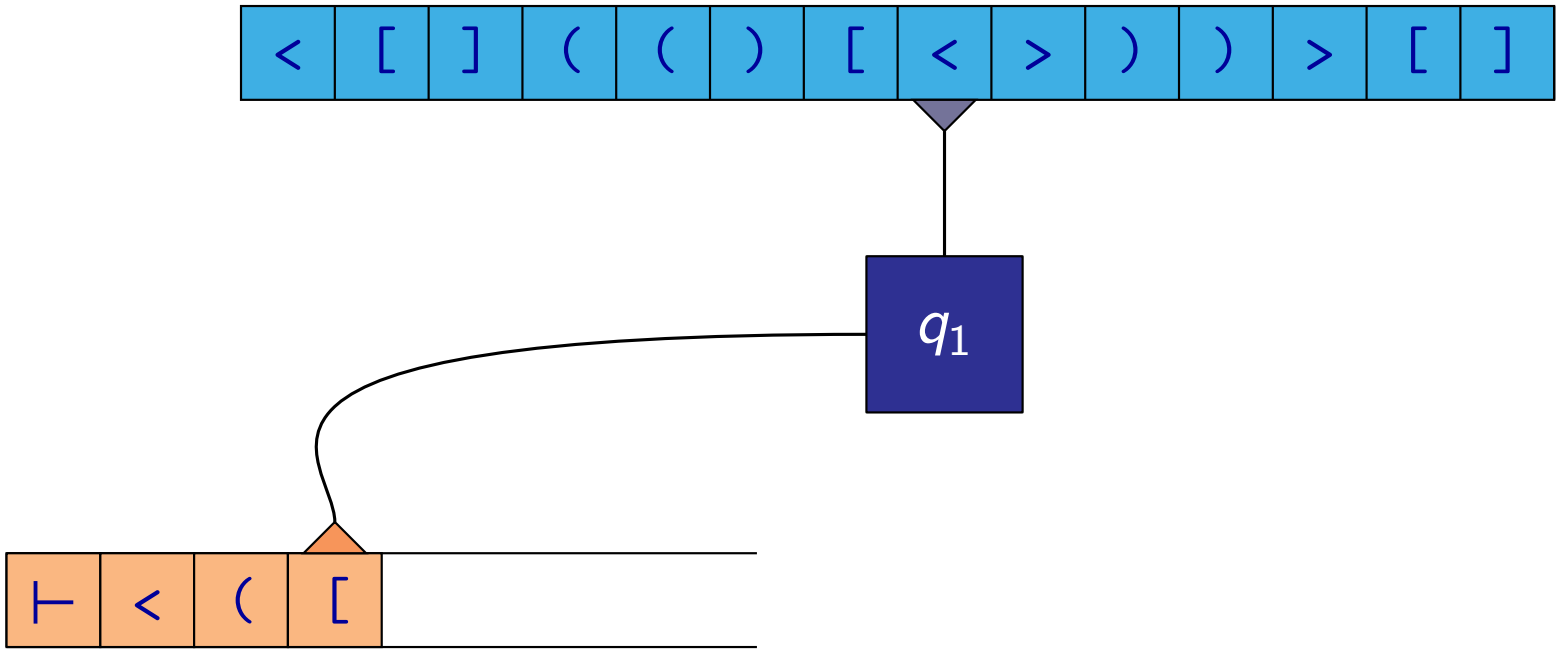
Zásobníkový automat

- Slovo <[](()[<>))>[] nepatří do jazyka.



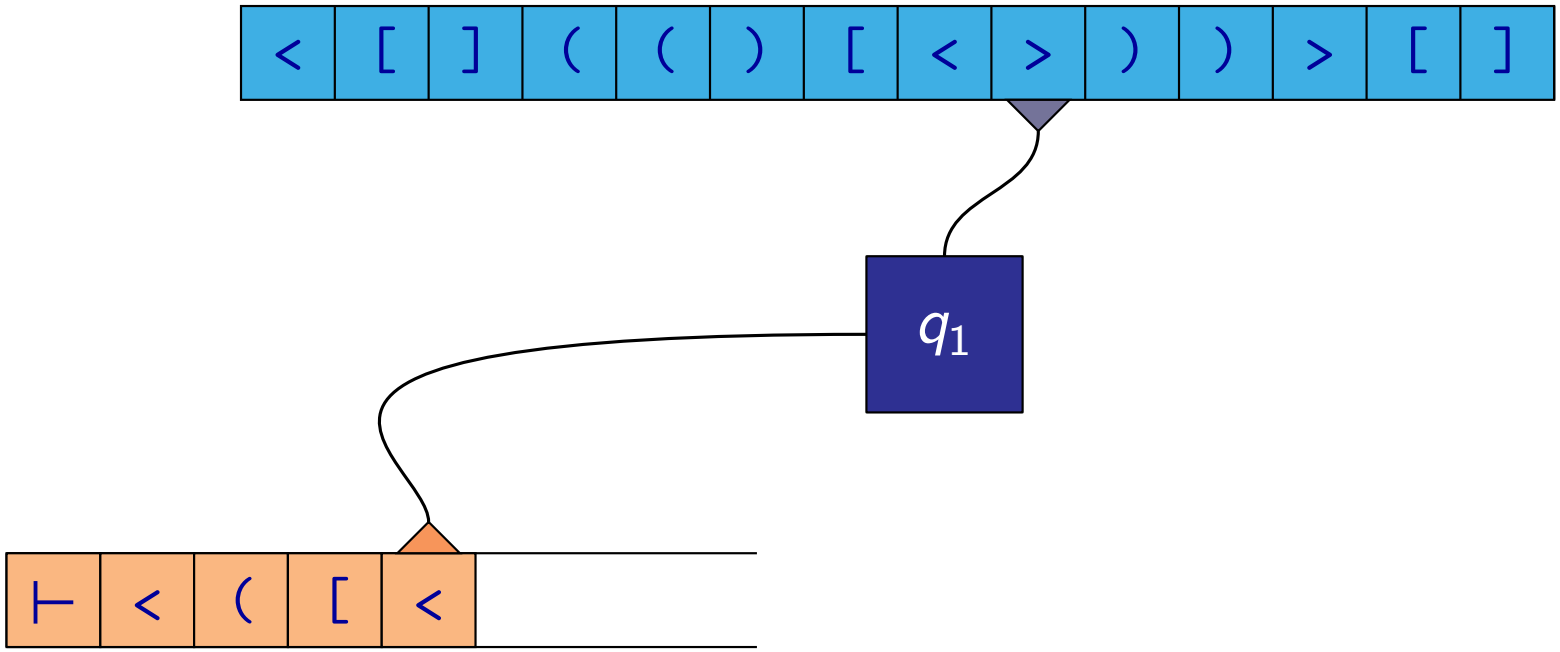
Zásobníkový automat

- Slovo <[](()[<>))>[] nepatří do jazyka.



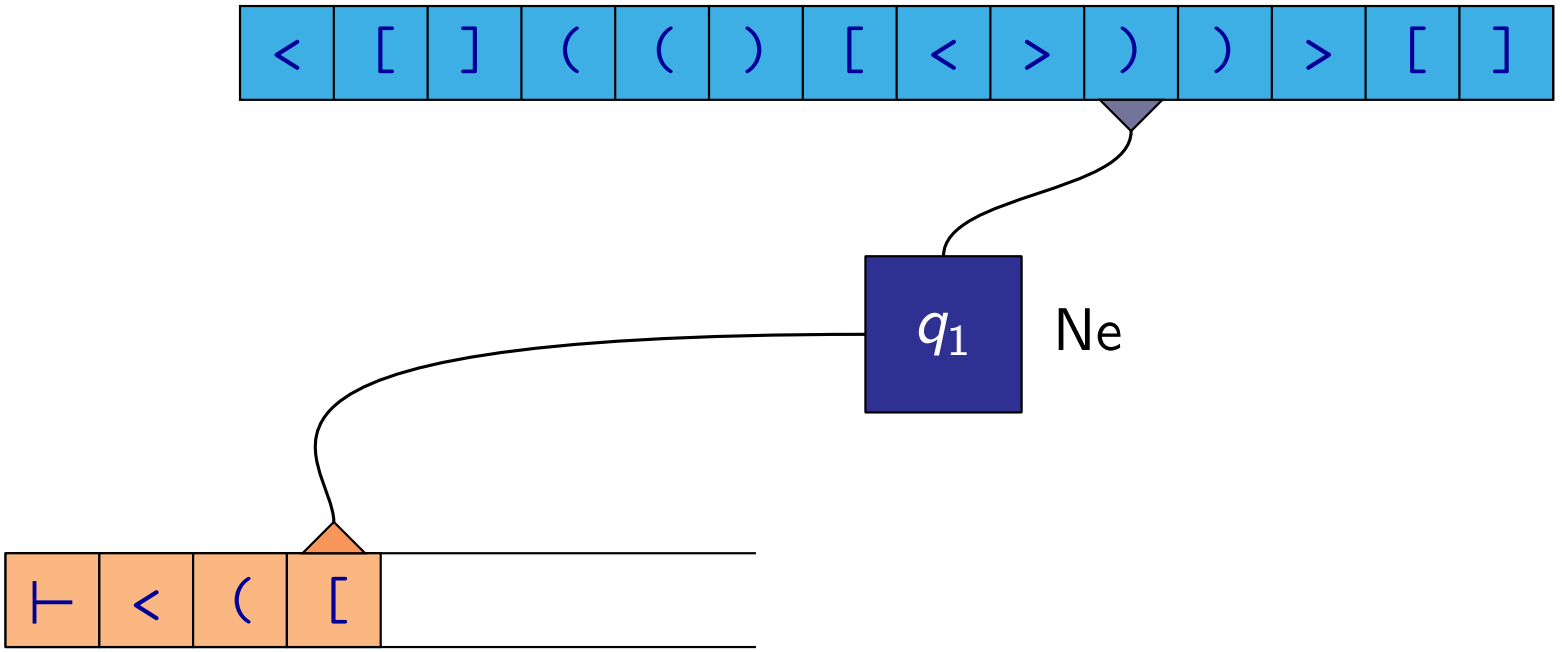
Zásobníkový automat

- Slovo <[](()[<>))>[] nepatří do jazyka.



Zásobníkový automat

- Slovo `<[](()[<>))>[]` nepatří do jazyka.
- Automat narazil na neodpovídající závorku, takže slovo nepřijal.



Zásobníkový automat

Příklad:

- Chtěli bychom rozpoznávat jazyk $L = \{a^n b^n \mid n \geq 1\}$

Opět se jedná o typický příklad neregulárního jazyka.

Zásobníkový automat

Příklad:

- Chtěli bychom rozpoznávat jazyk $L = \{a^n b^n \mid n \geq 1\}$

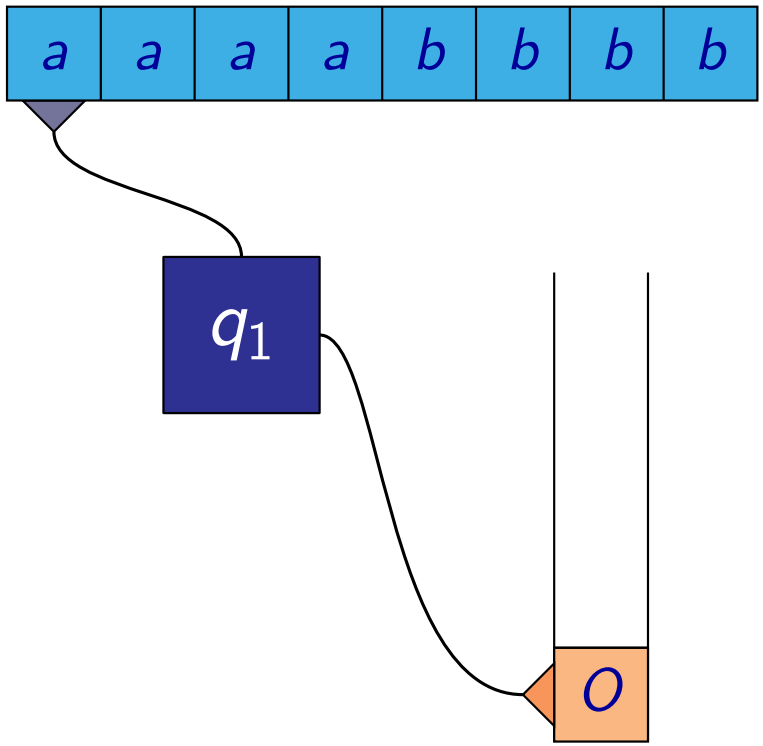
Opět se jedná o typický příklad neregulárního jazyka.

Zásobník můžeme používat jako čítač:

- Budeme do něj ukládat symboly jednoho druhu (nazvěme ho např. I).
- Počet těchto symbolů I na zásobníku bude reprezentovat hodnotu čítače.

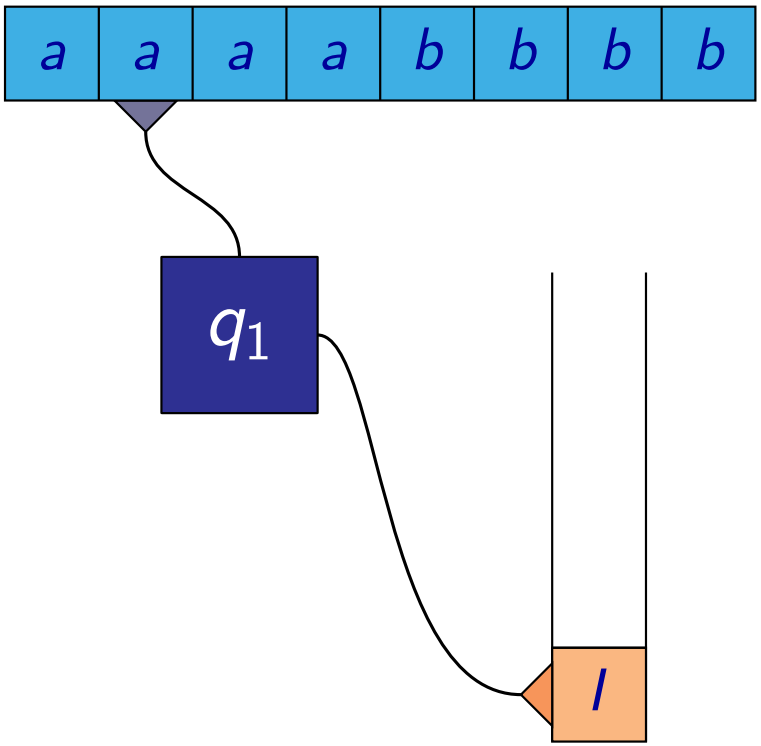
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



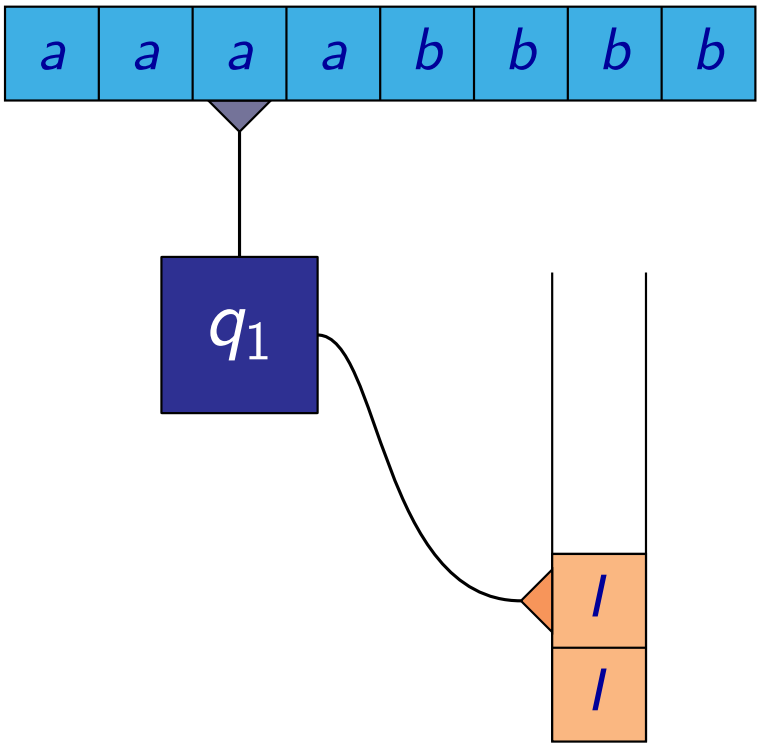
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



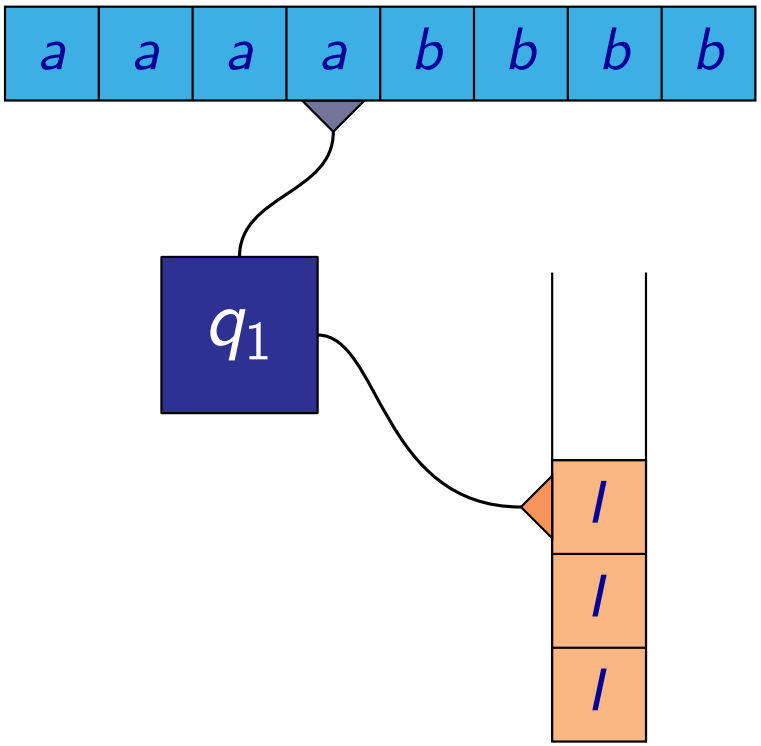
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



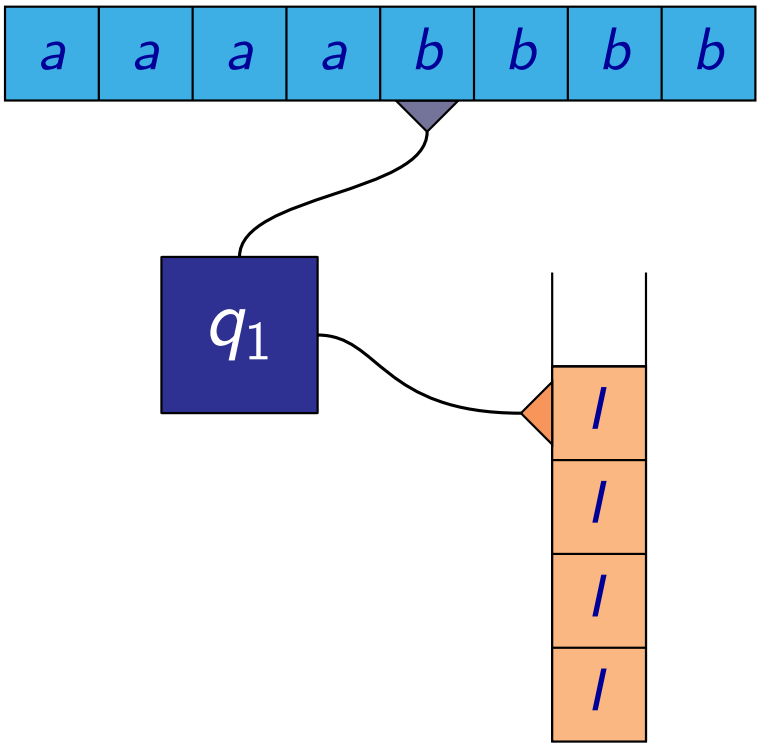
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



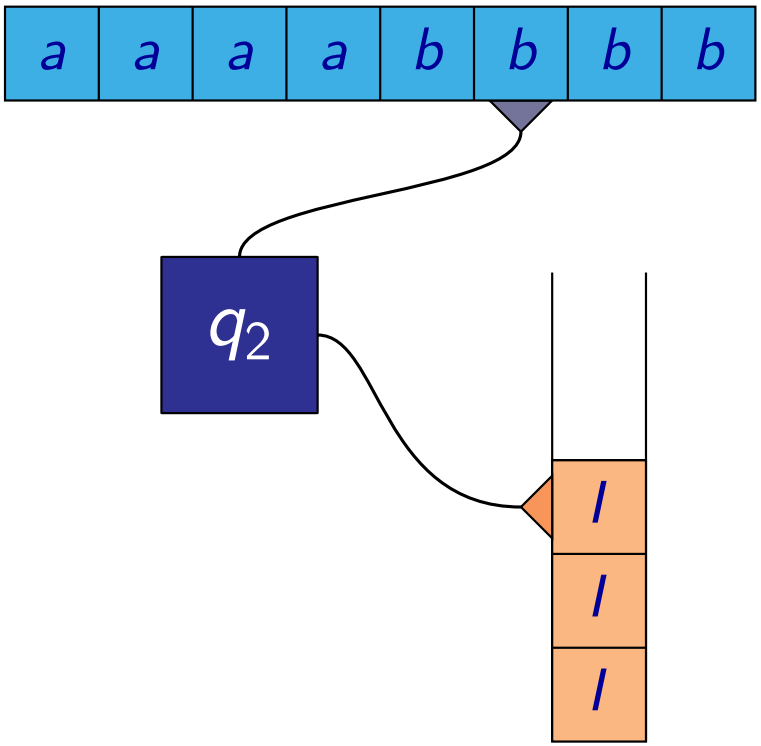
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



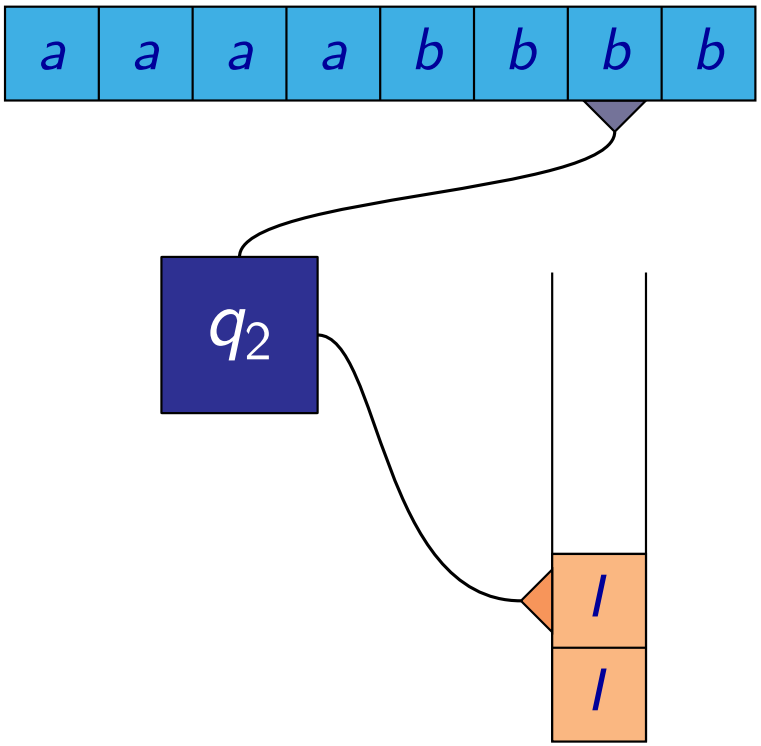
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



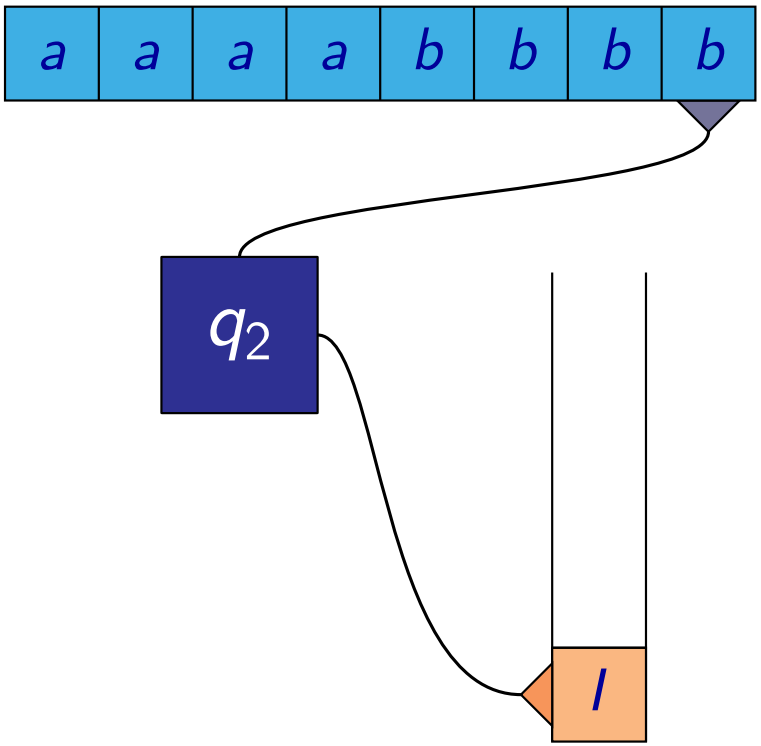
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



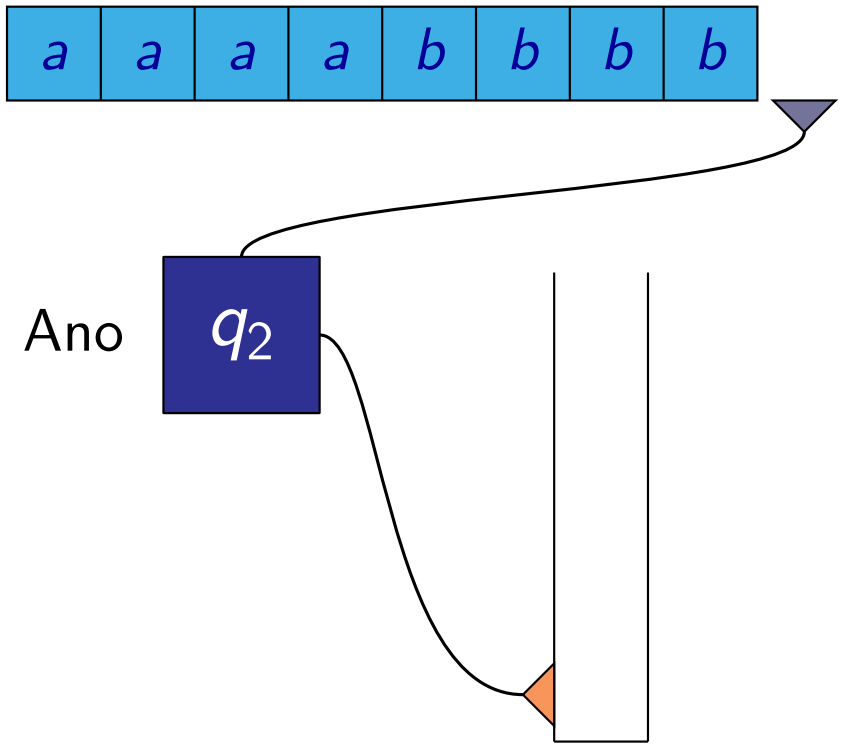
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



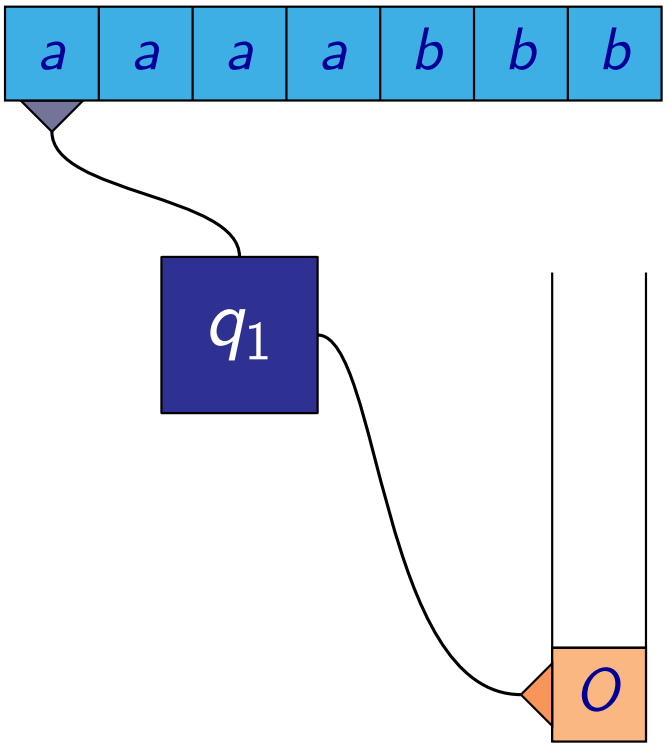
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^n b^n \mid n \geq 1\}$
- Automat přečetl celé slovo a skončil s prázdným zásobníkem, takže slovo přijal.



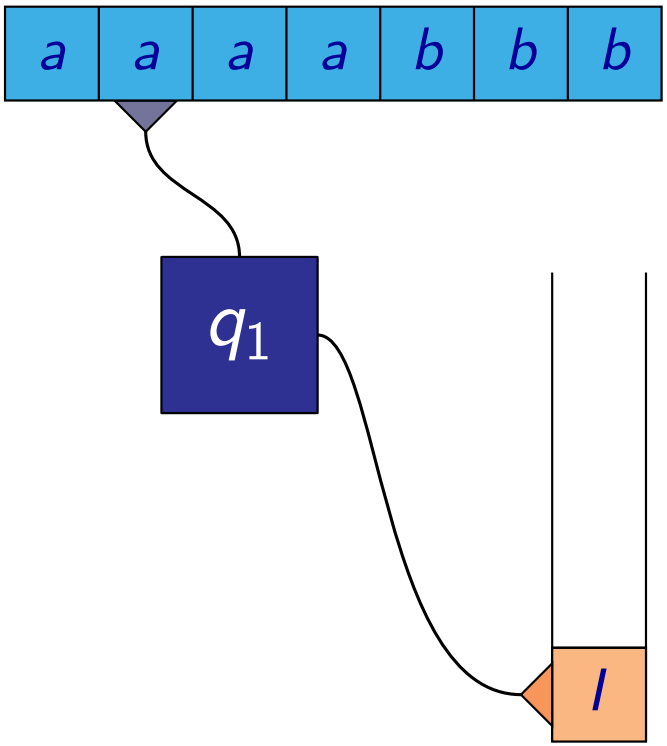
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



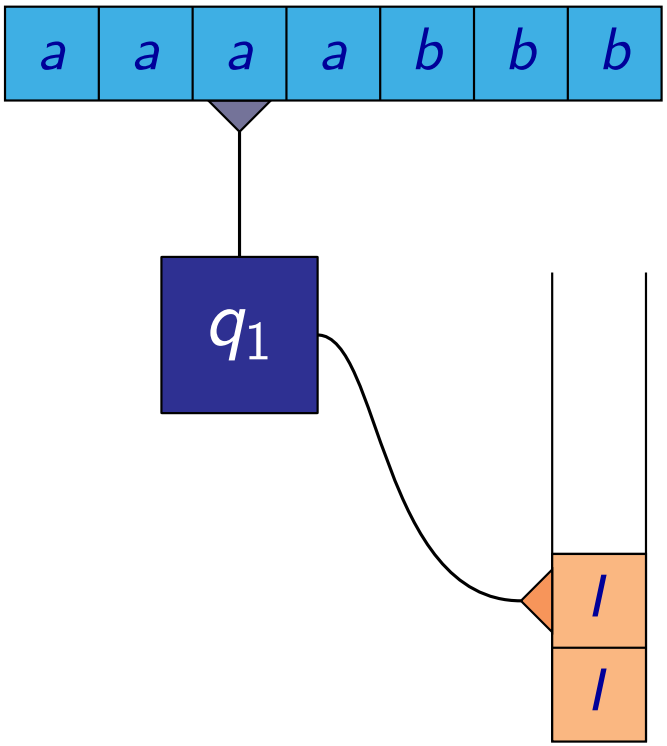
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



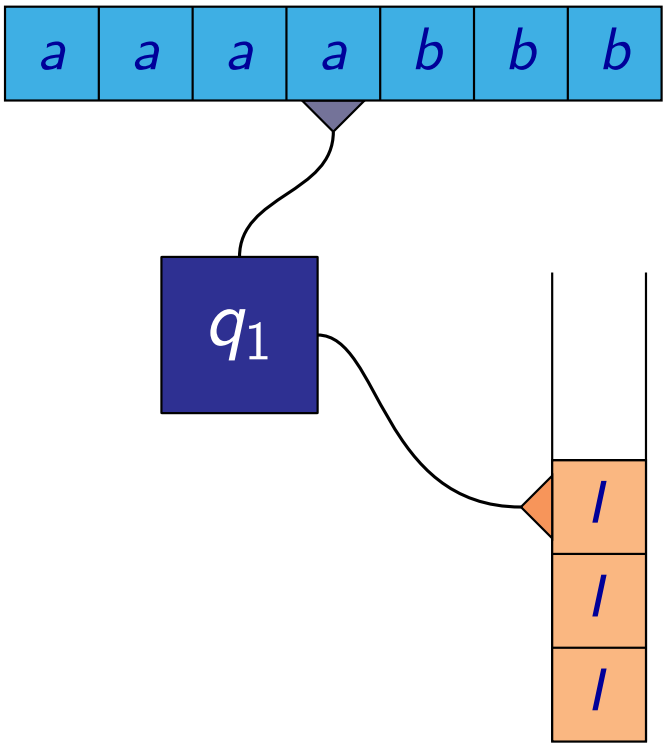
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



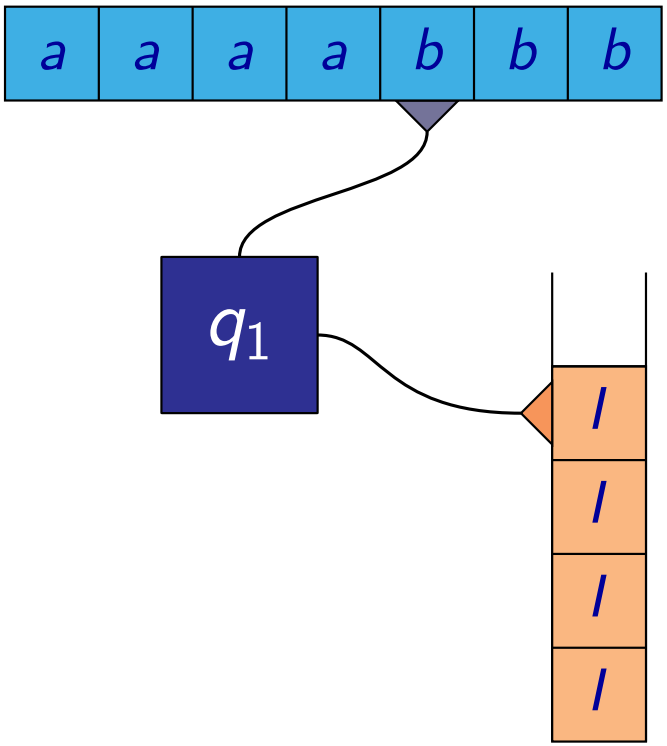
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



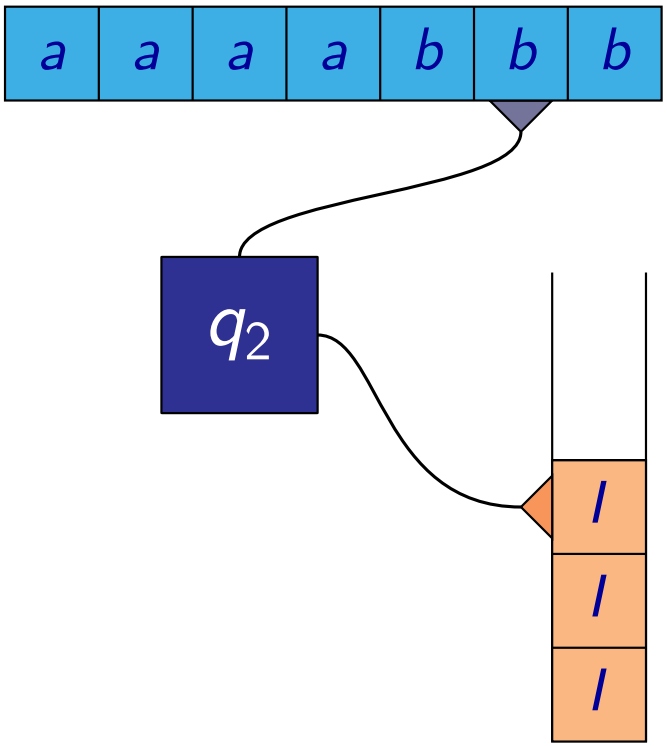
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



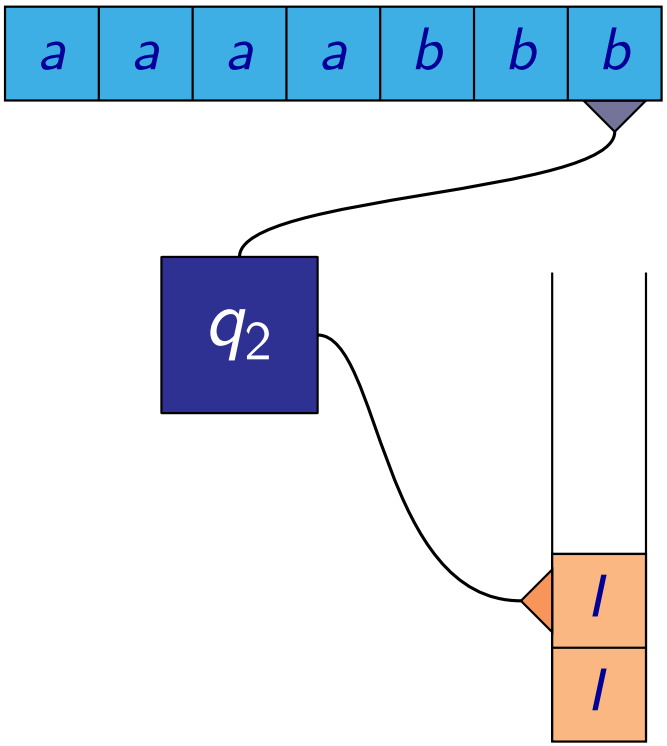
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



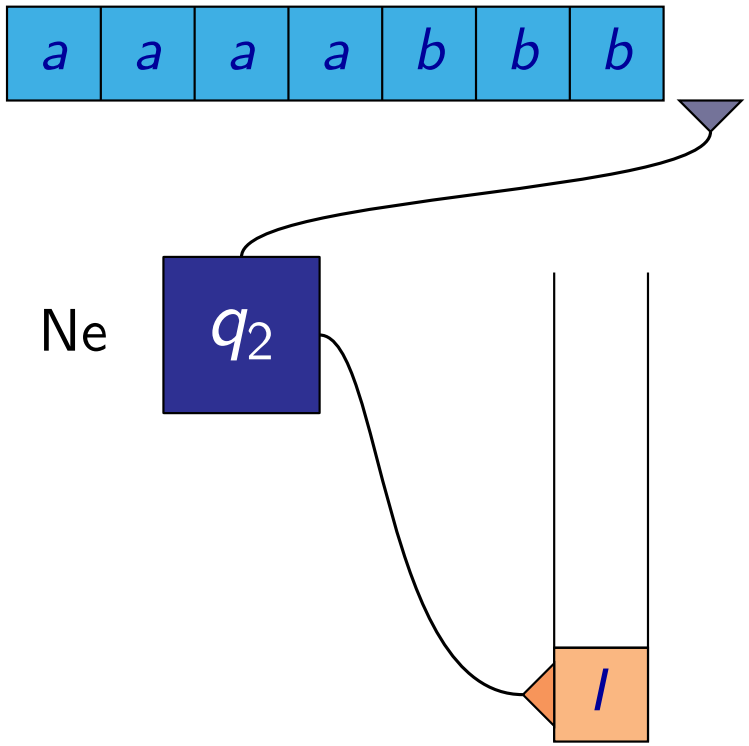
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



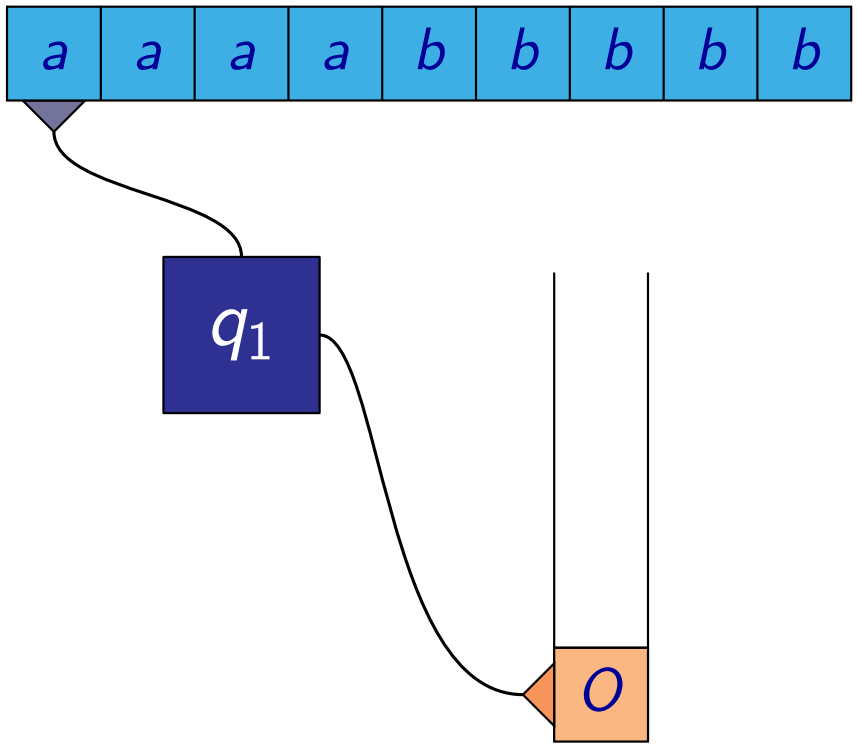
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$
- Automat přečetl celé slovo, ale nevyprázdnil zásobník, takže slovo nepřijal



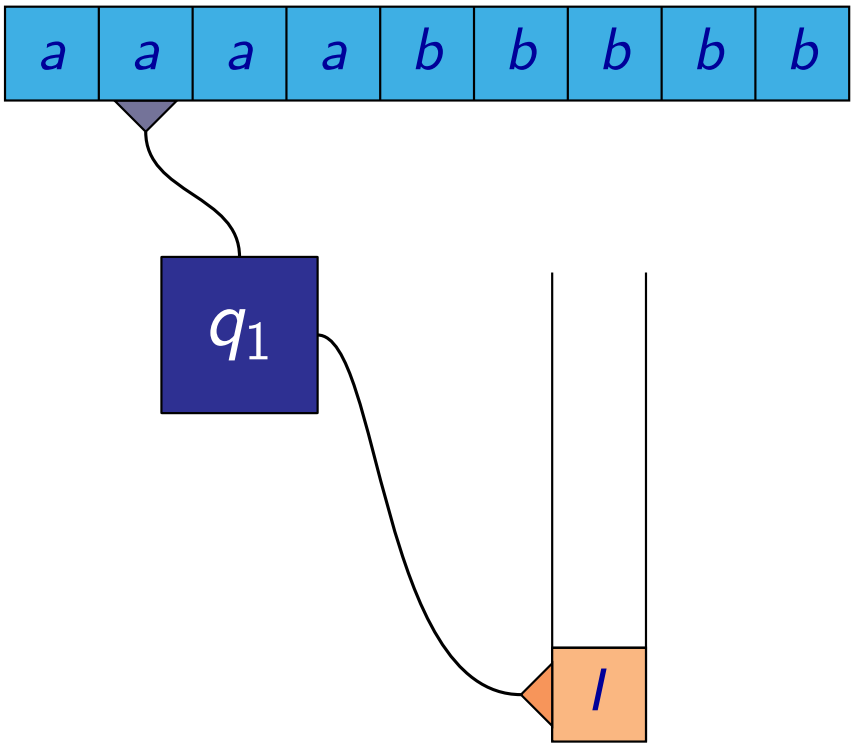
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



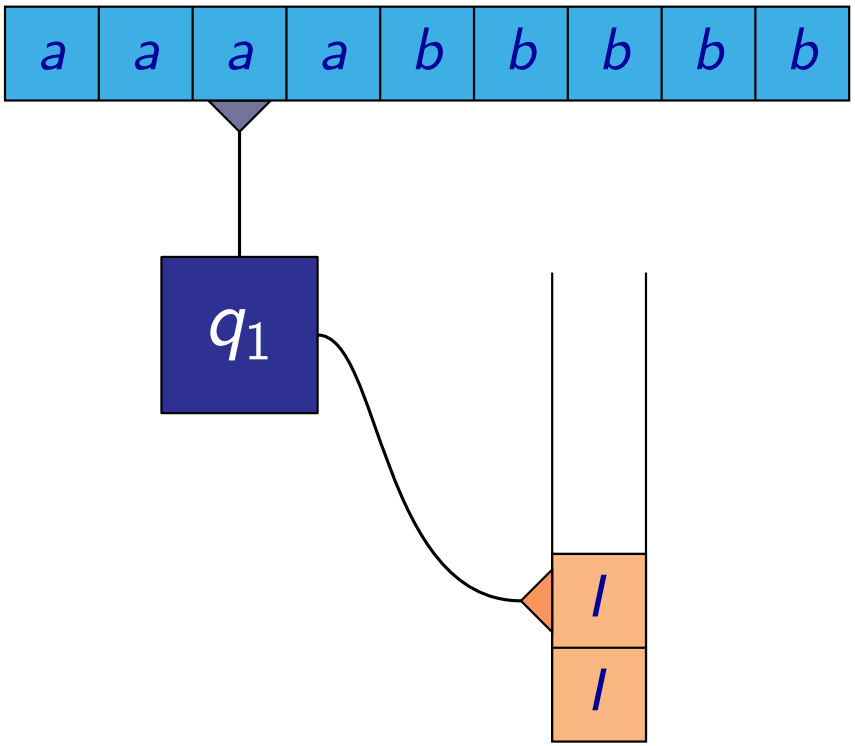
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



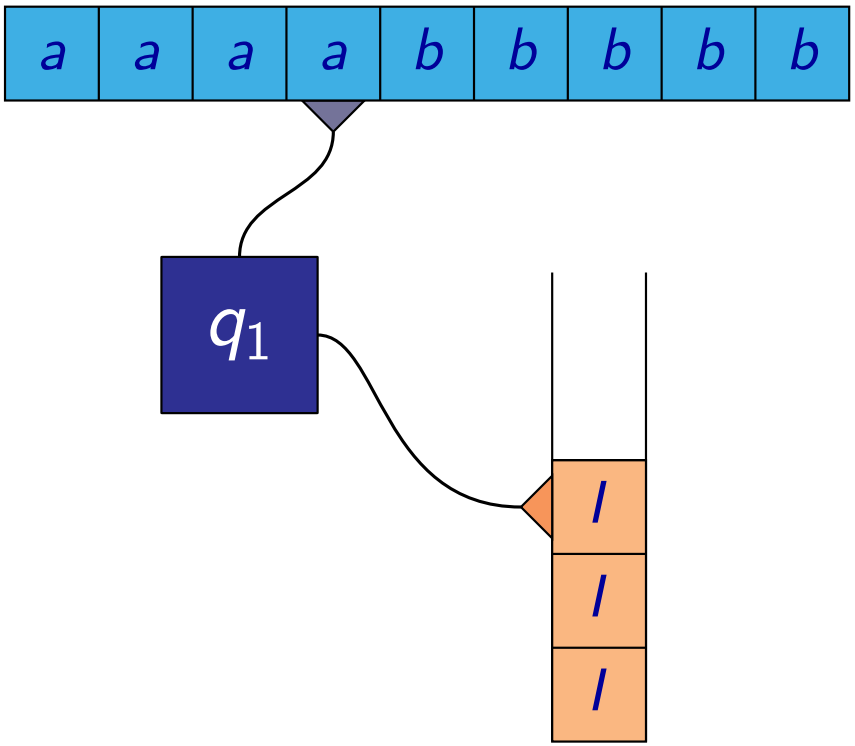
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



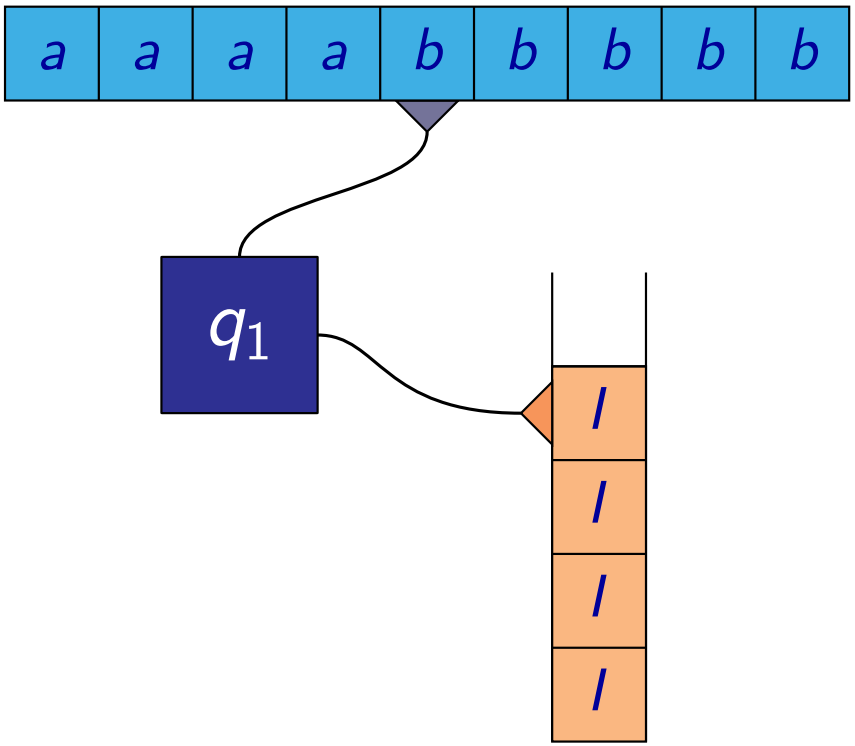
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



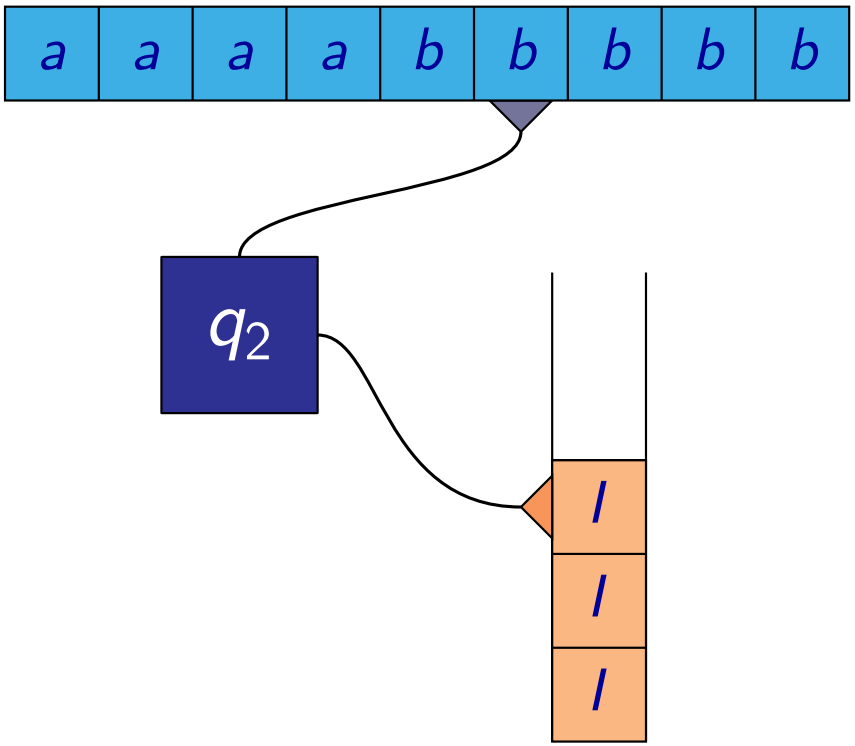
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



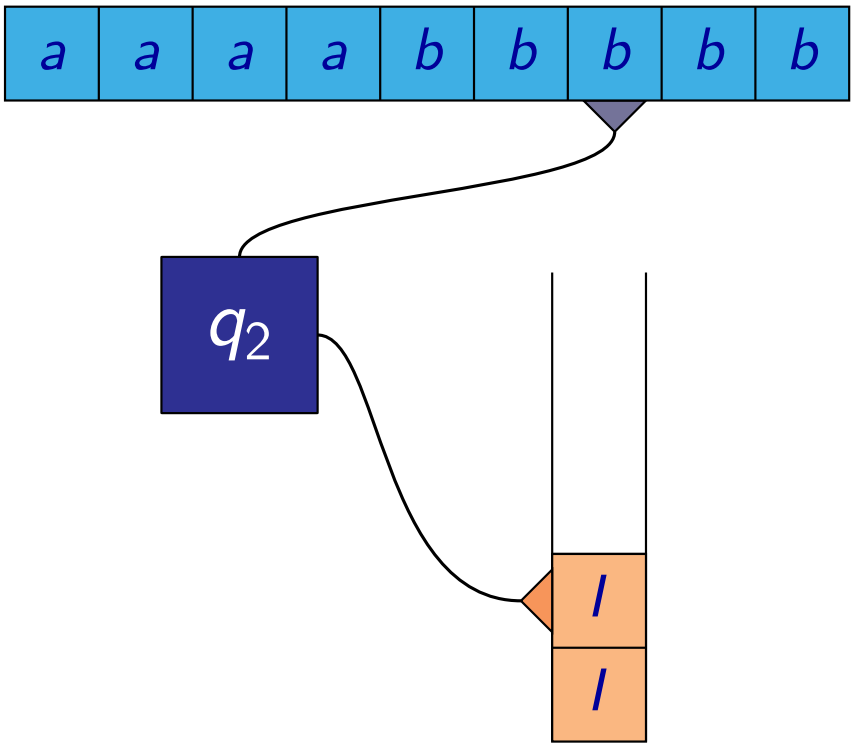
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



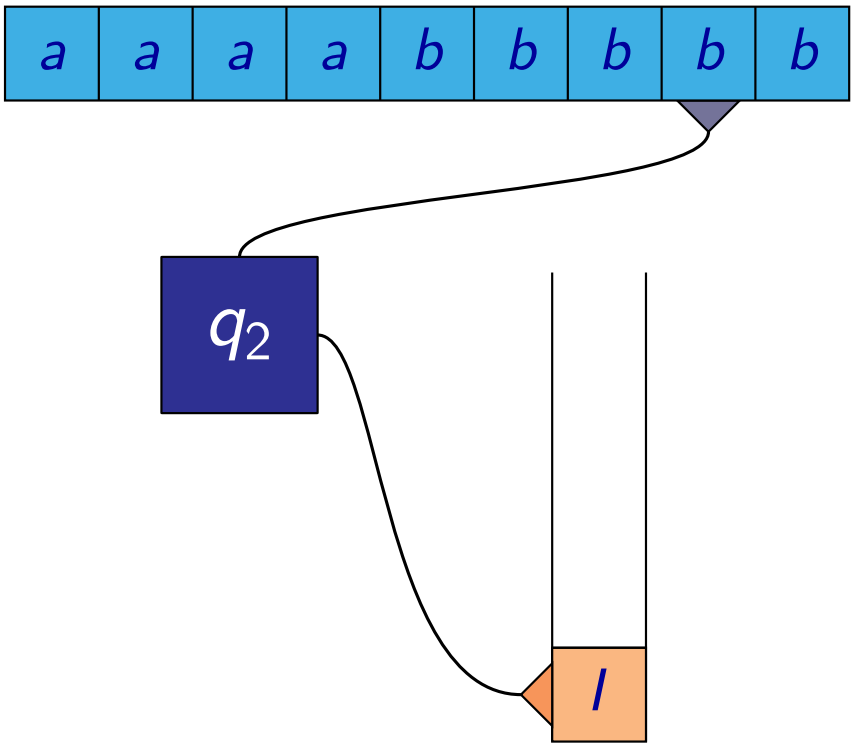
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



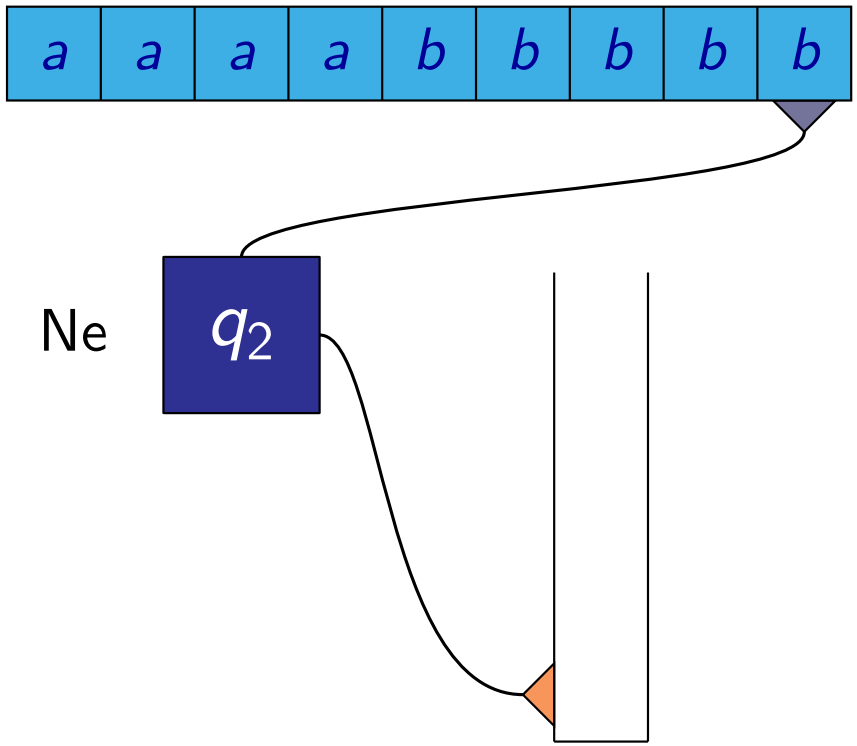
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



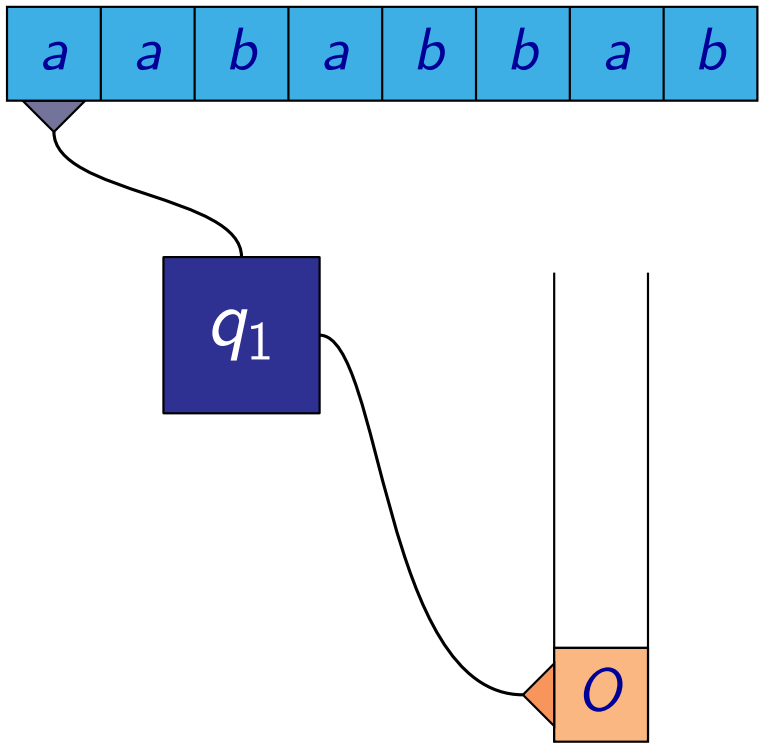
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$
- Automat čte *b*, má smazat symbol na zásobníku a tam žádný není, takže slovo nepřijal.



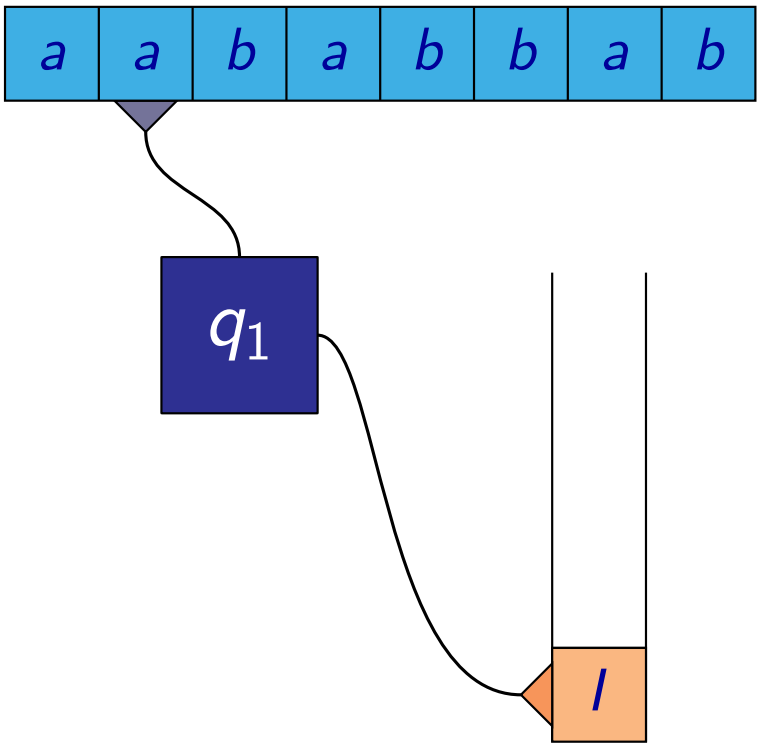
Zásobníkový automat

- Slovo *aababbab* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



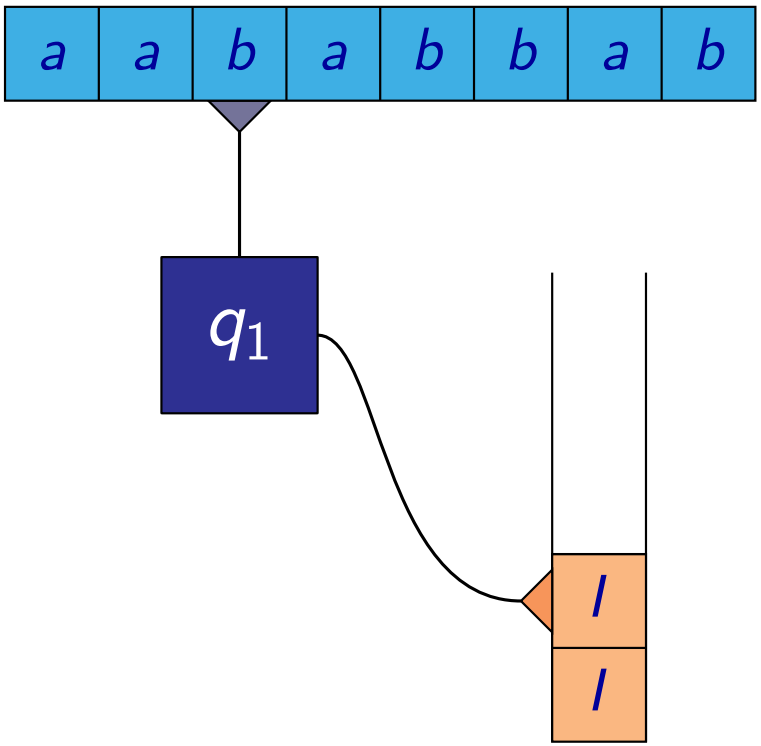
Zásobníkový automat

- Slovo *aababbab* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



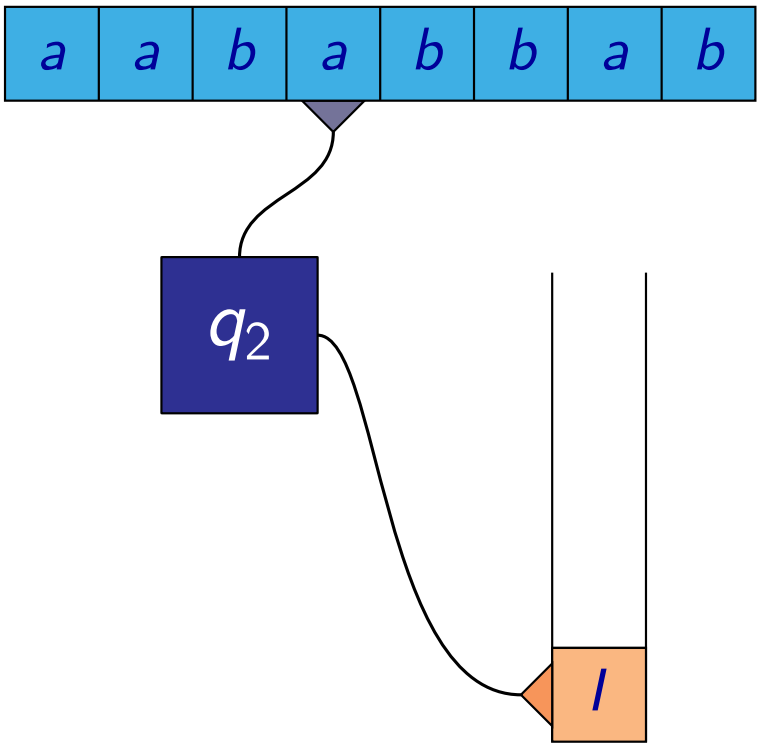
Zásobníkový automat

- Slovo *aababbab* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



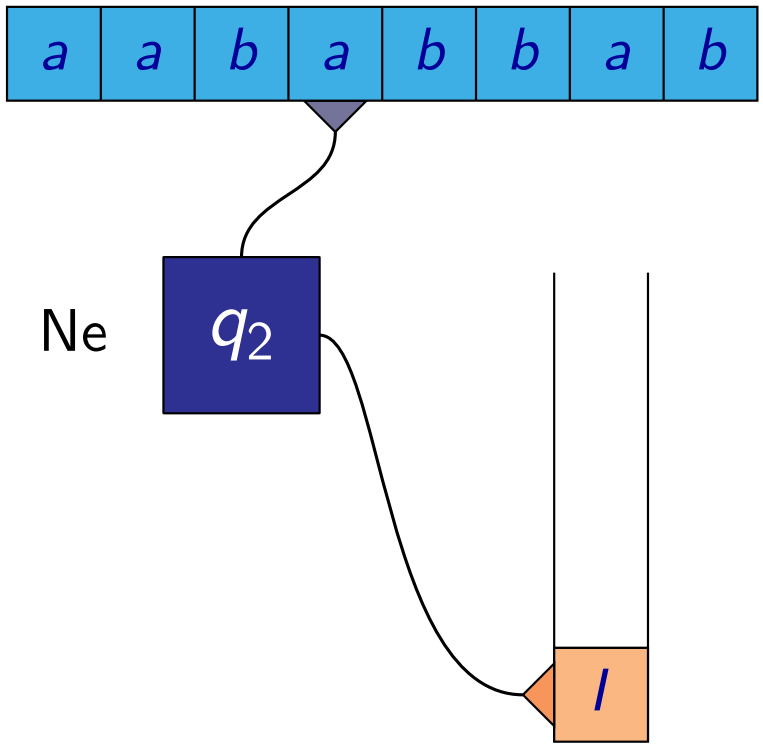
Zásobníkový automat

- Slovo *aababbab* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$



Zásobníkový automat

- Slovo *aababbab* nepatří do jazyka $L = \{a^n b^n \mid n \geq 1\}$
- Automat přečetl *a*, ale již byl ve stavu, kdy maže, takže slovo nepřijal.



Zásobníkový automat

- Zásobníkový automat může být nedeterministický a může mít ε-přechody.

Zásobníkový automat

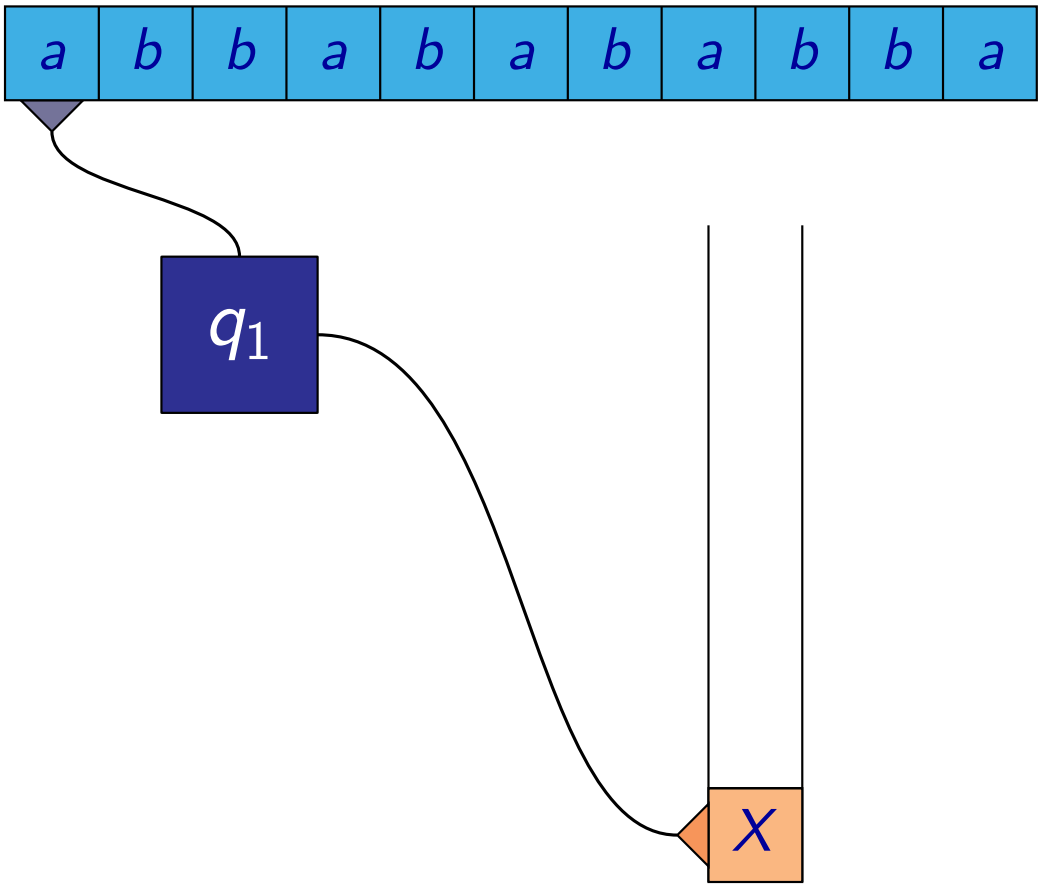
- Zásobníkový automat může být nedeterministický a může mít ε-přechody.

Příklad:

- Uvažujme jazyk $L = \{w \in \{a, b\}^* \mid w = w^R\}$.
- První polovinu slova můžeme uložit na zásobník.
- Při čtení druhé poloviny mažeme symboly ze zásobníku, pokud jsou stejné jako na vstupu.
- Pokud bude zásobník prázdný po přečtení celého slova, byla druhá polovina stejná jako první.
- Místo, kde se nachází „hranice“ mezi první a druhou polovinou slova může automat nedeterministicky uhodnout. Výpočty, při kterých bude hádat chybně, nepovedou k přijetí slova.

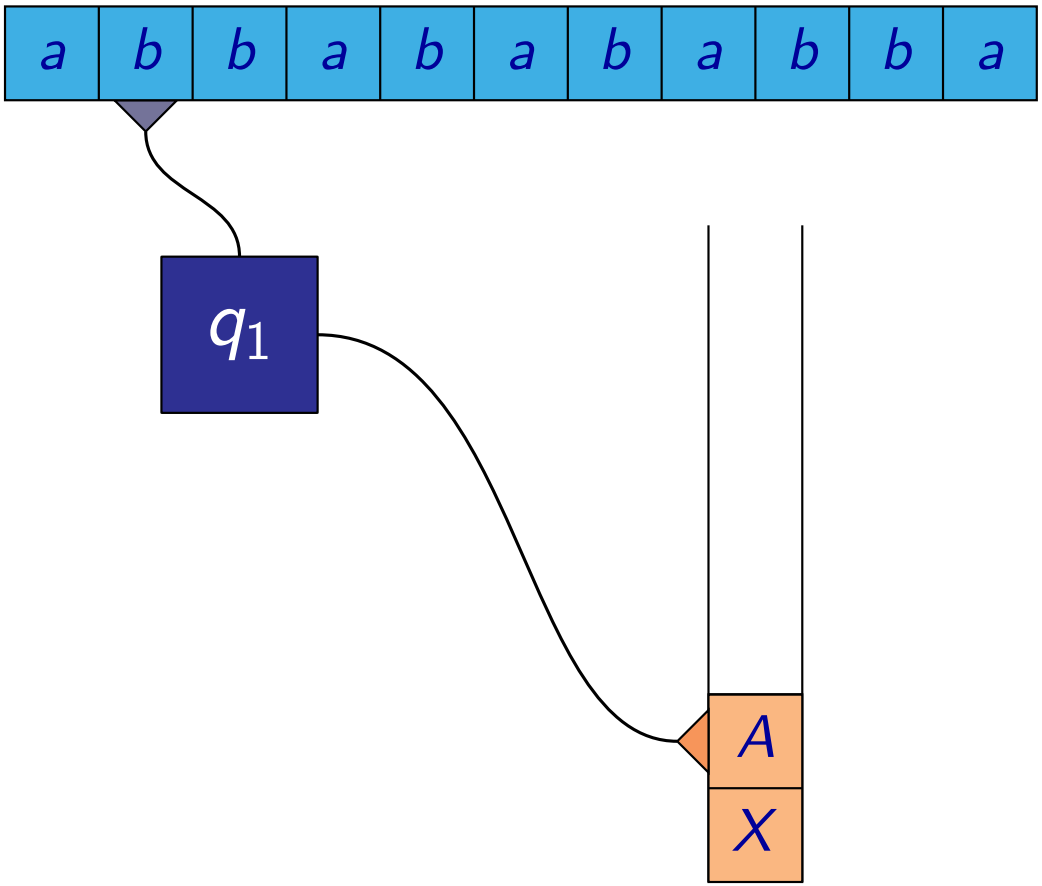
Zásobníkový automat

- Slovo *abbababababba* patří do jazyka $L = \{w \in \{a, b\}^* \mid w = w^R\}$



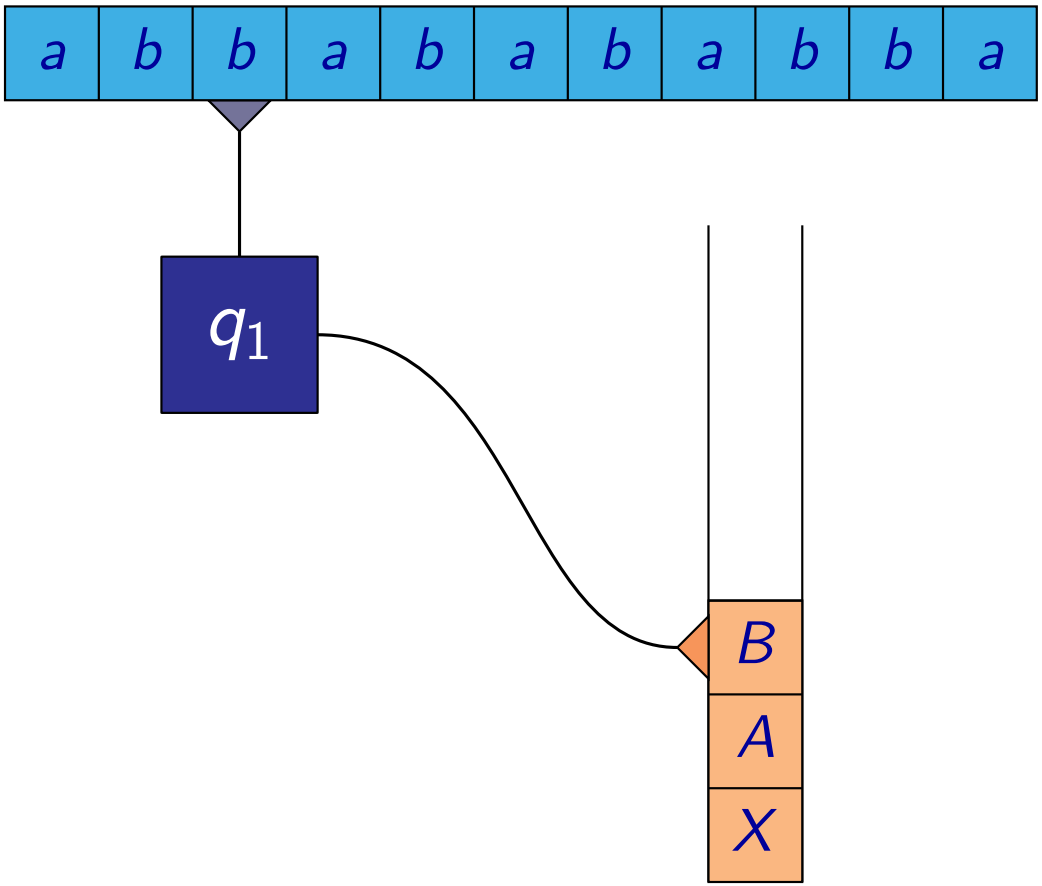
Zásobníkový automat

- Slovo *abbababbabba* patří do jazyka $L = \{w \in \{a, b\}^* \mid w = w^R\}$



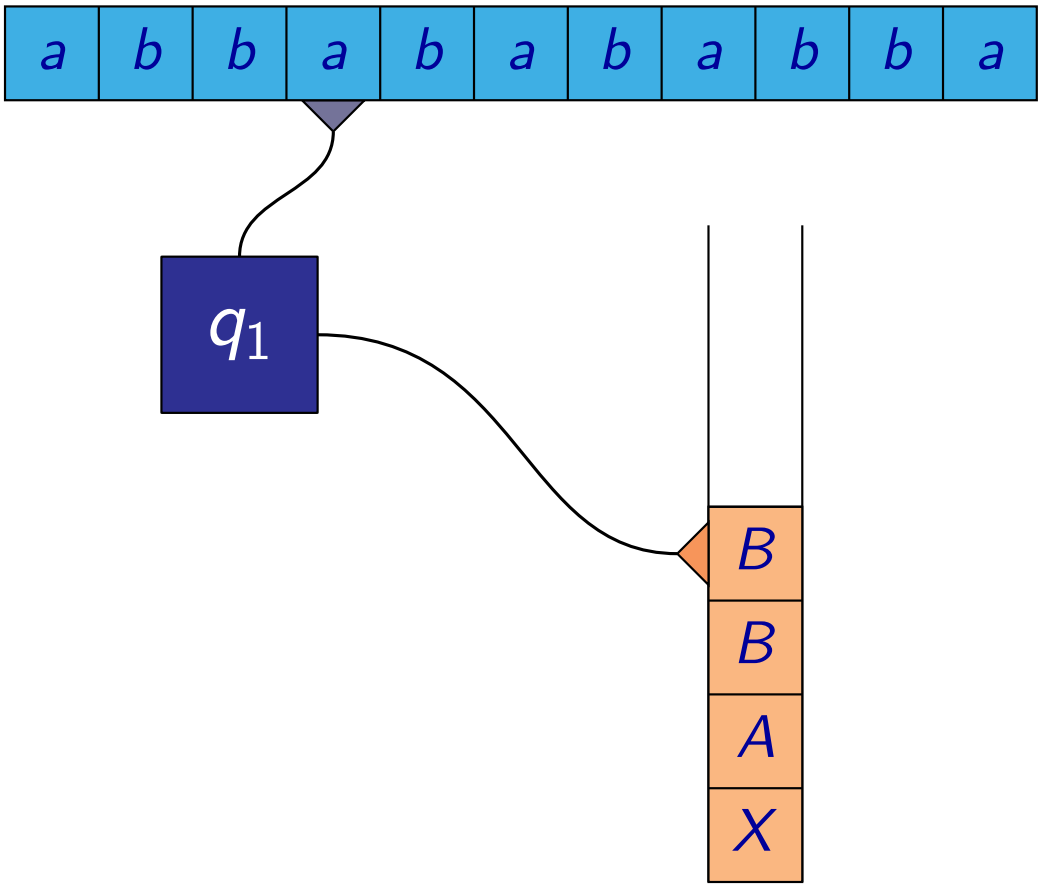
Zásobníkový automat

- Slovo *abbababababba* patří do jazyka $L = \{w \in \{a, b\}^* \mid w = w^R\}$



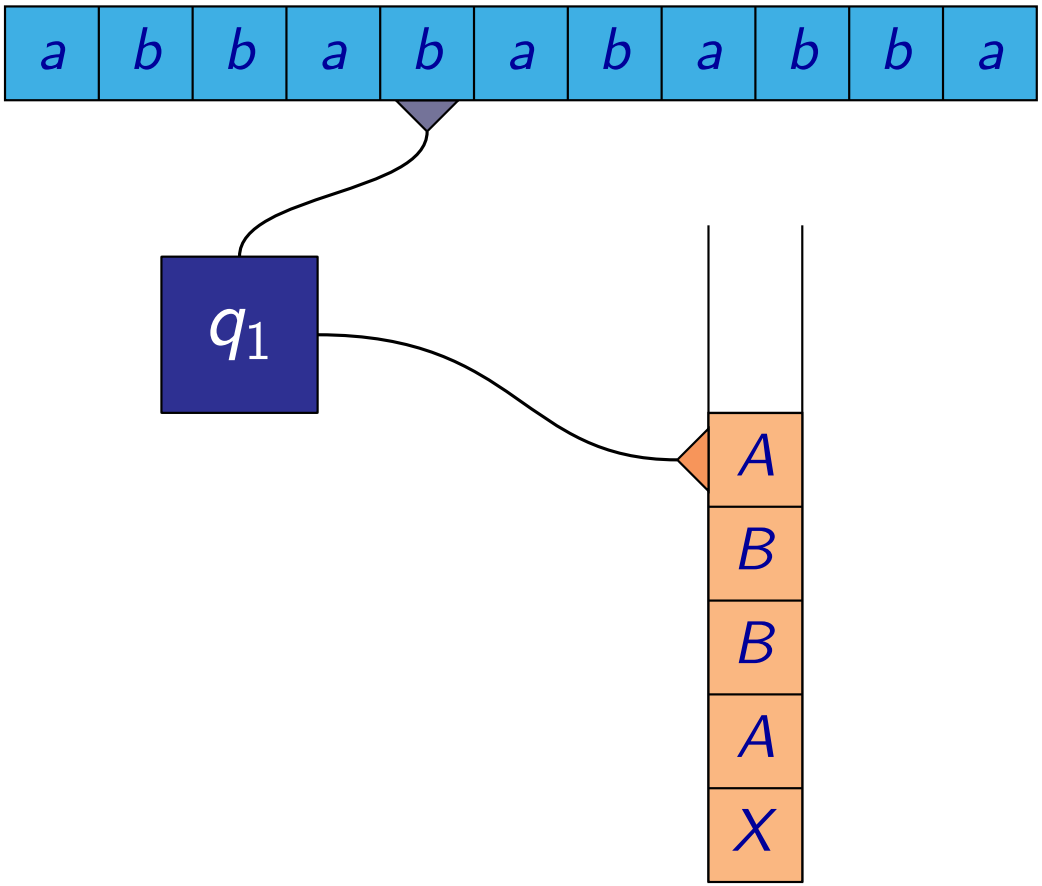
Zásobníkový automat

- Slovo *abbababababba* patří do jazyka $L = \{w \in \{a, b\}^* \mid w = w^R\}$



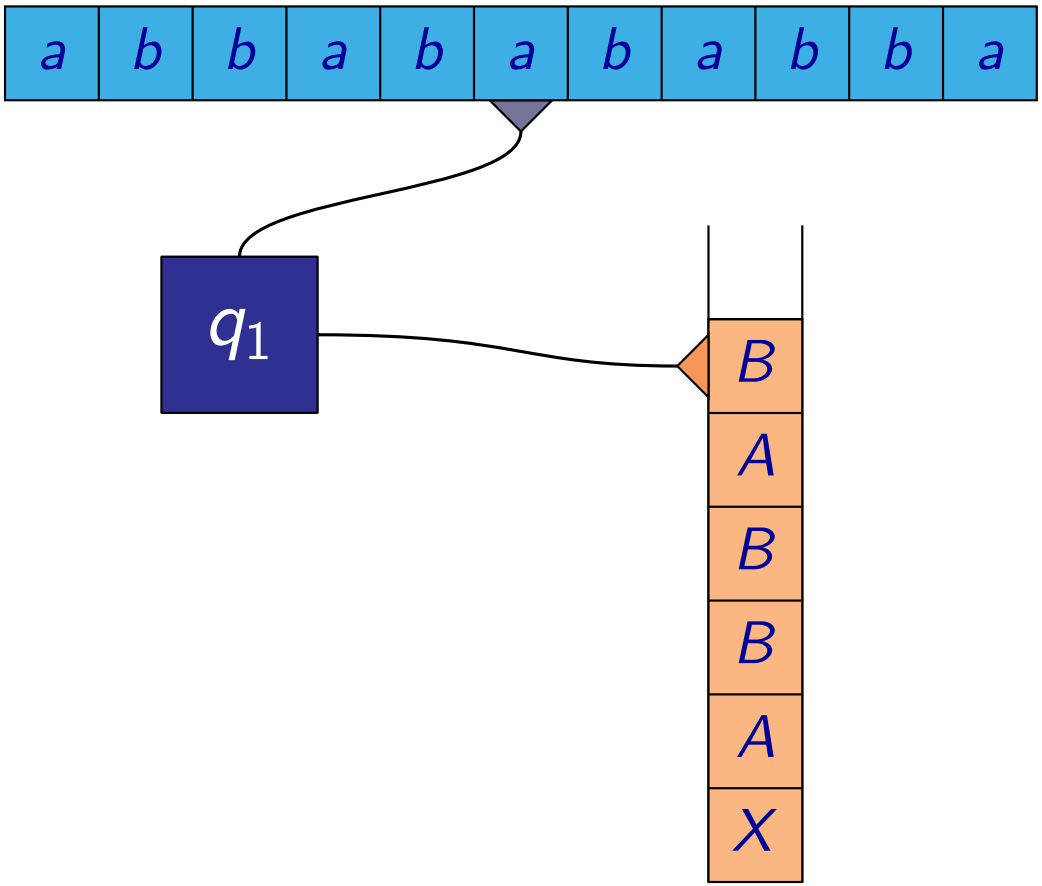
Zásobníkový automat

- Slovo *abbababababba* patří do jazyka $L = \{w \in \{a, b\}^* \mid w = w^R\}$



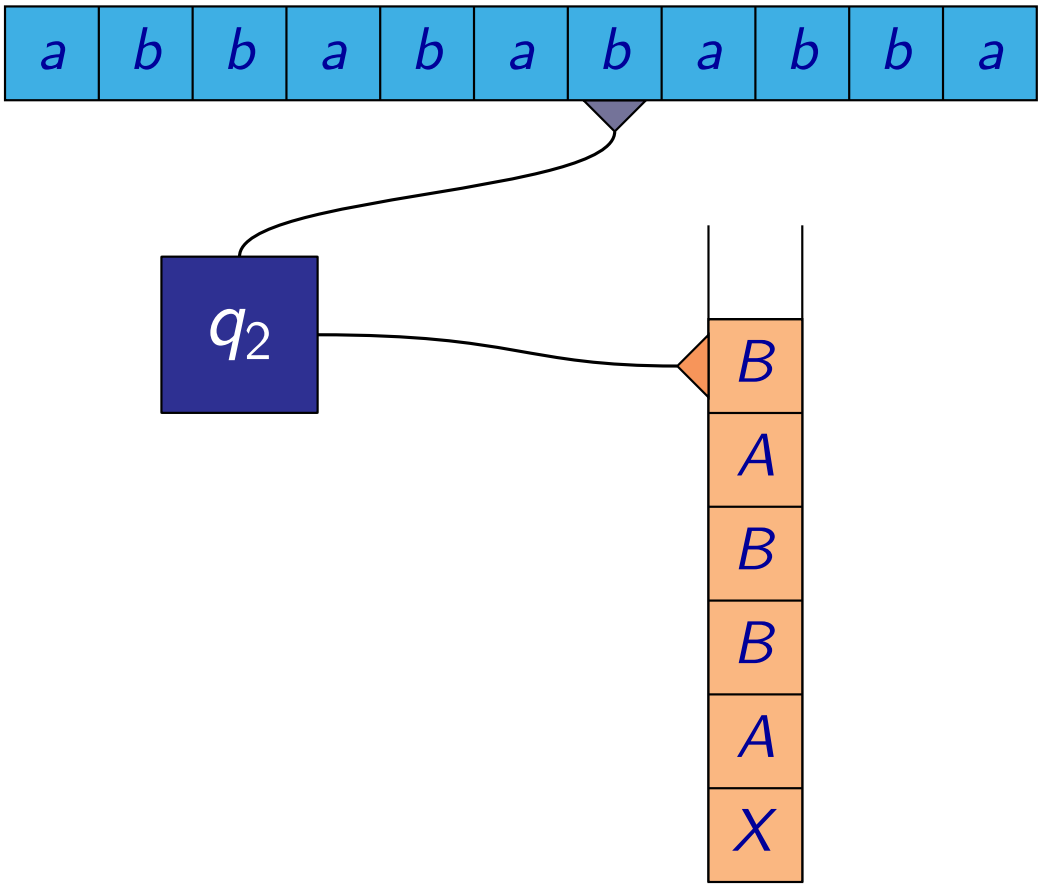
Zásobníkový automat

- Slovo *abbababababba* patří do jazyka $L = \{w \in \{a, b\}^* \mid w = w^R\}$



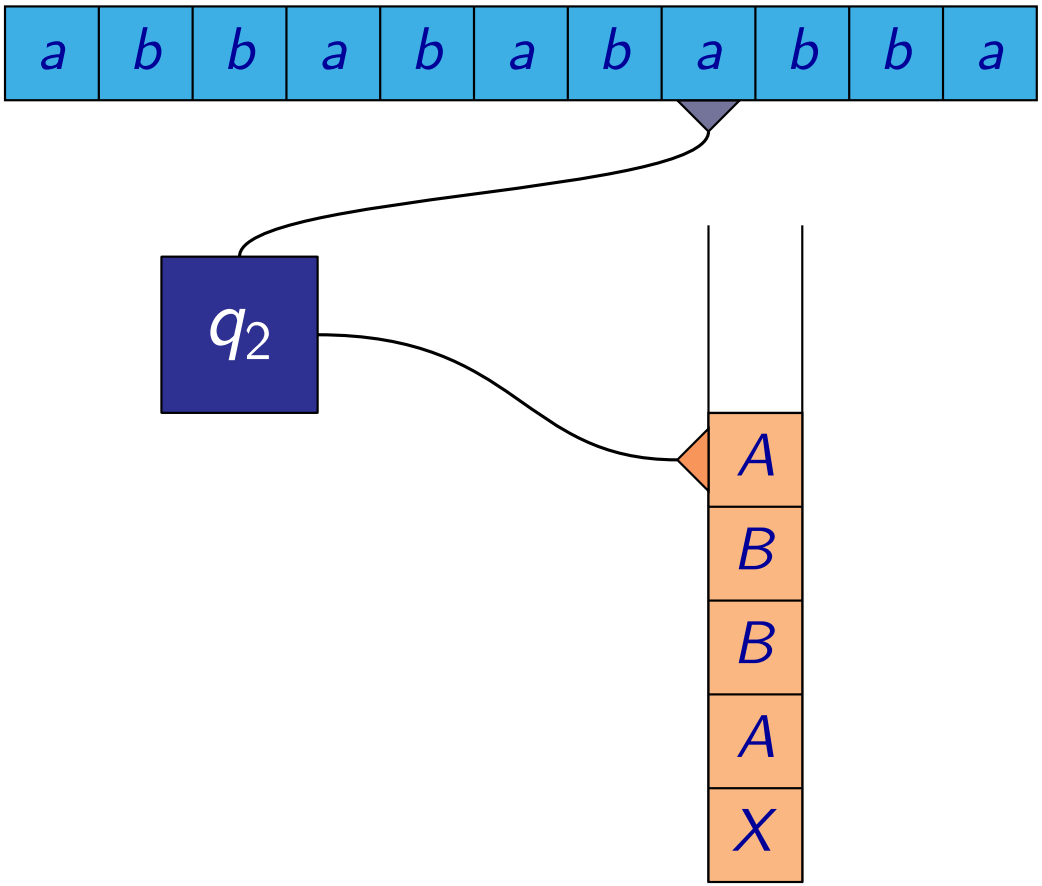
Zásobníkový automat

- Slovo *abbababababba* patří do jazyka $L = \{w \in \{a, b\}^* \mid w = w^R\}$



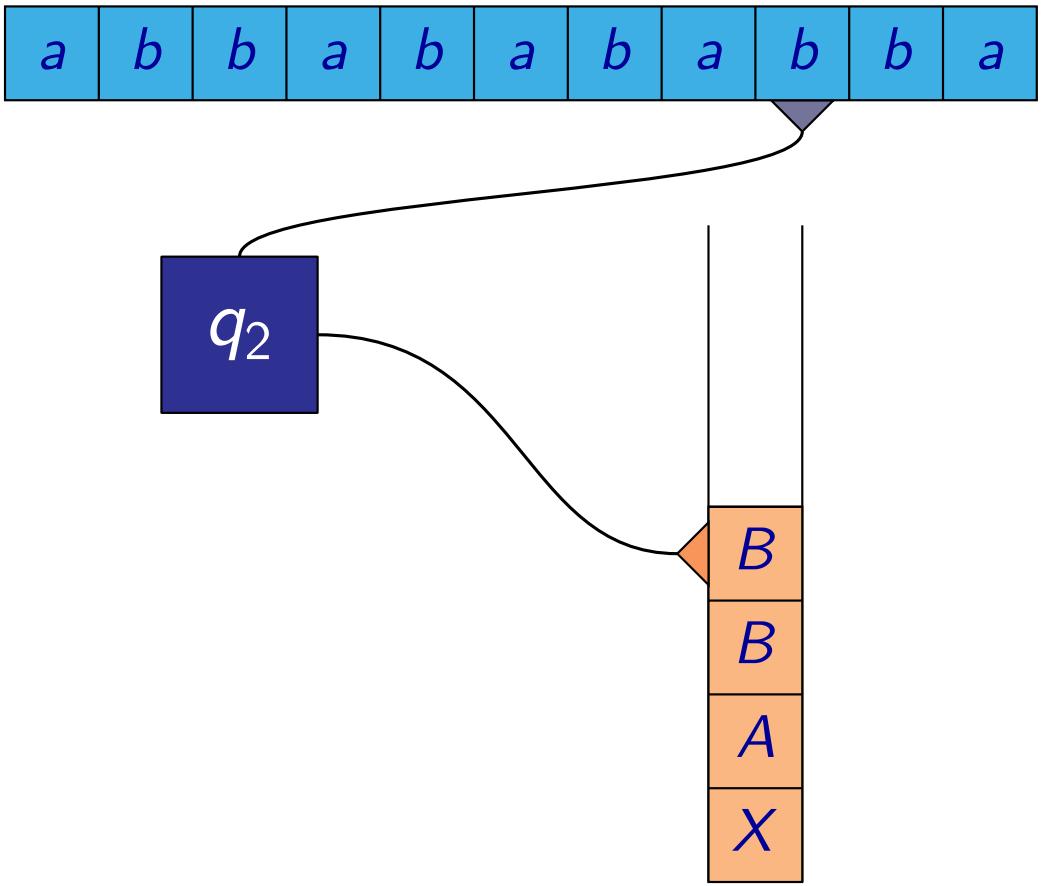
Zásobníkový automat

- Slovo *abbababababba* patří do jazyka $L = \{w \in \{a, b\}^* \mid w = w^R\}$



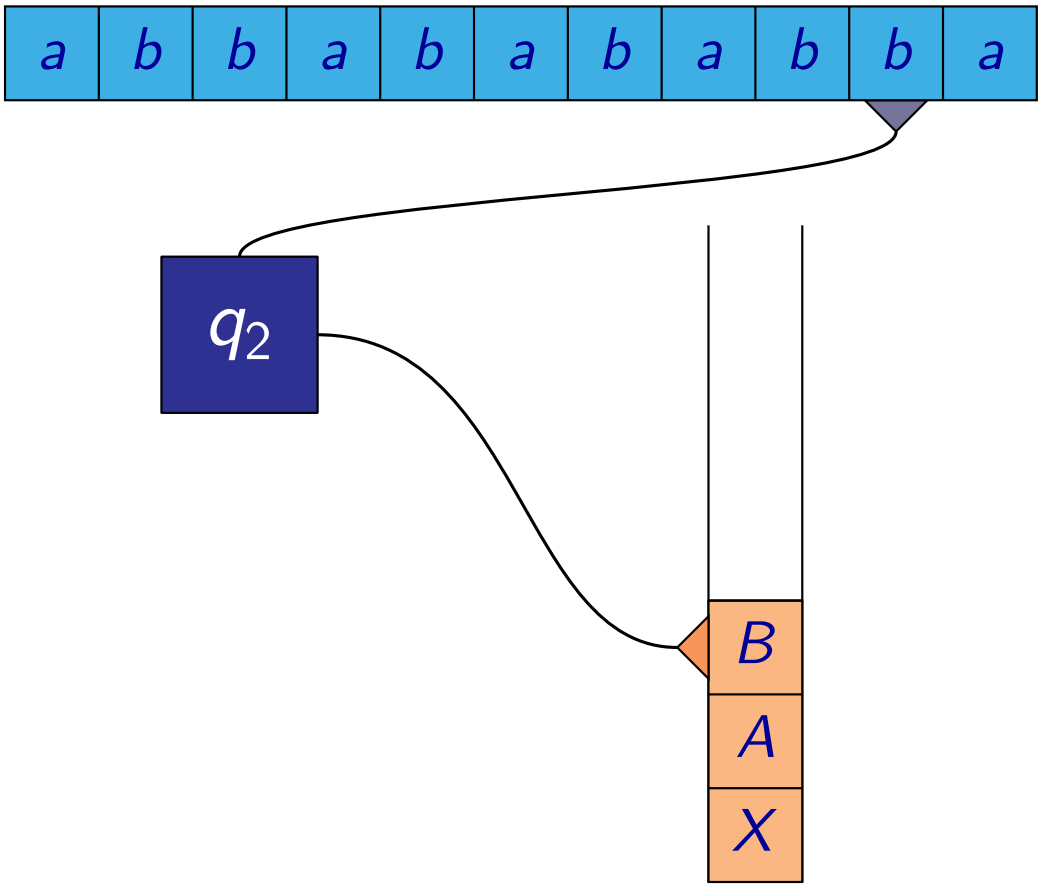
Zásobníkový automat

- Slovo *abbababababba* patří do jazyka $L = \{w \in \{a, b\}^* \mid w = w^R\}$



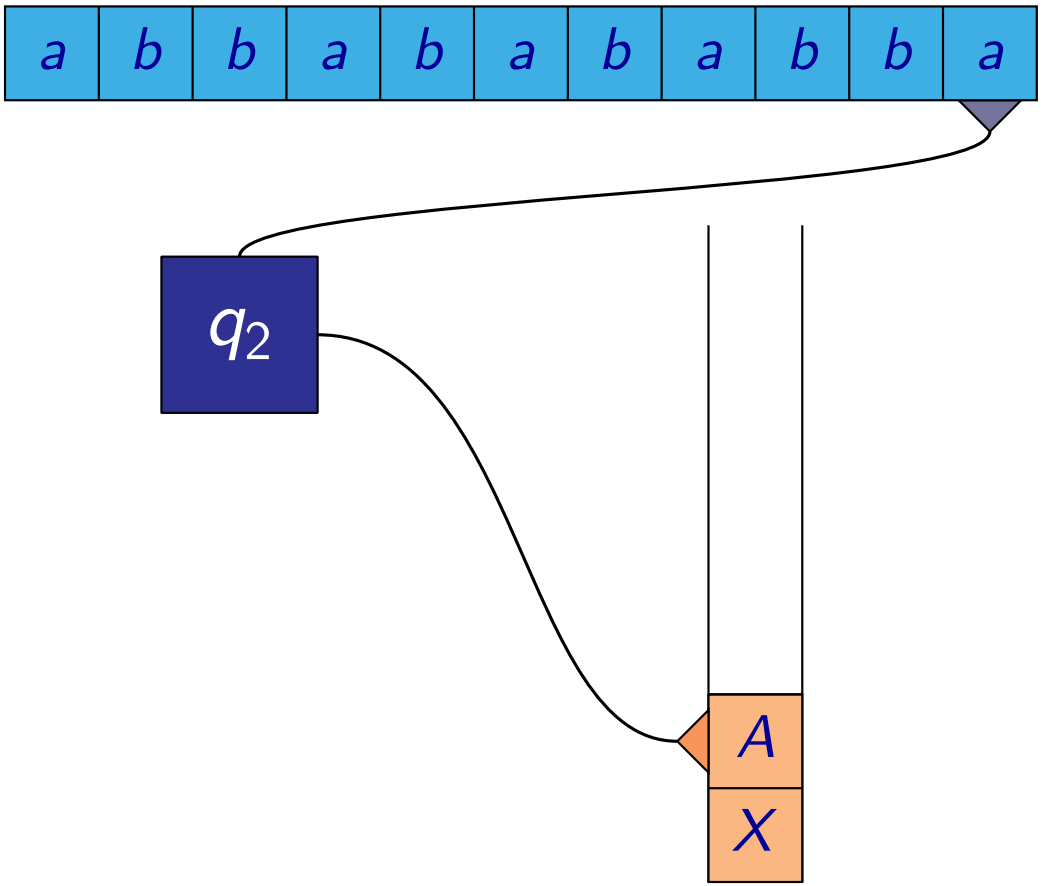
Zásobníkový automat

- Slovo *abbababababba* patří do jazyka $L = \{w \in \{a, b\}^* \mid w = w^R\}$



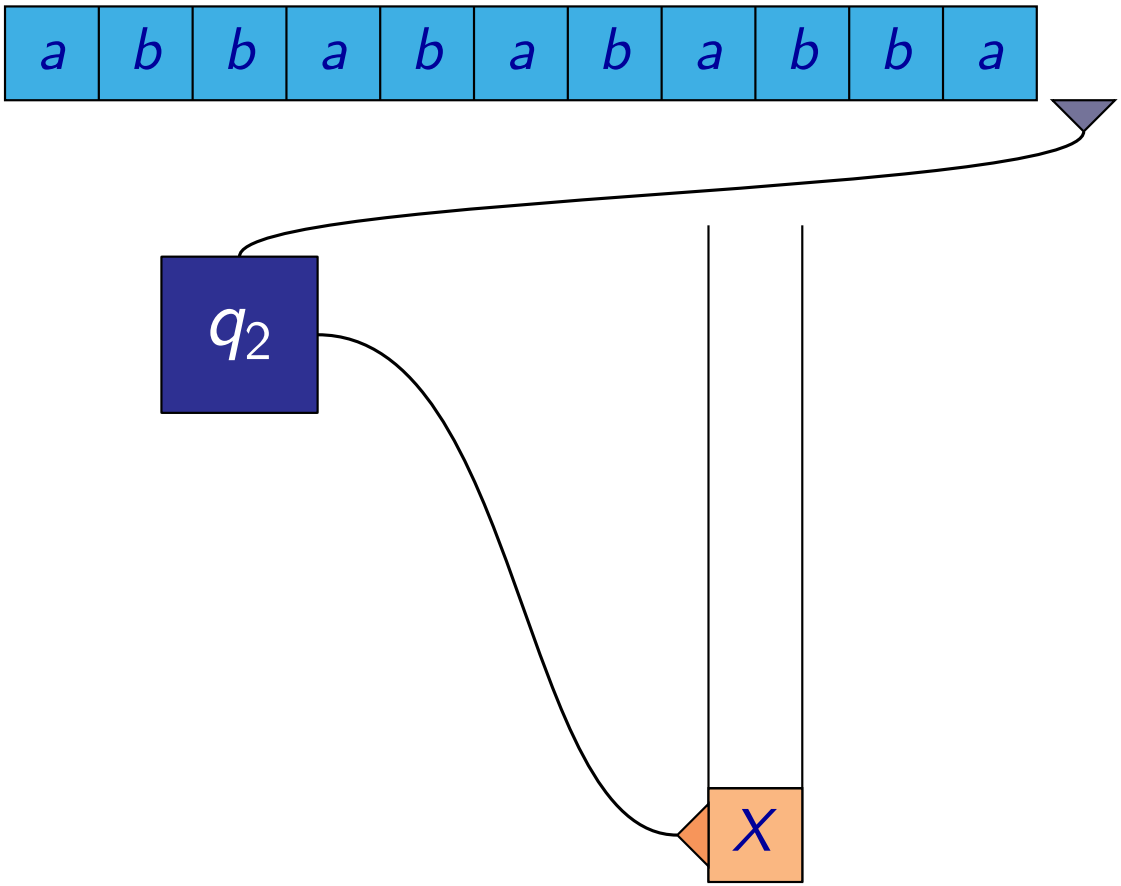
Zásobníkový automat

- Slovo *abbababbabba* patří do jazyka $L = \{w \in \{a, b\}^* \mid w = w^R\}$



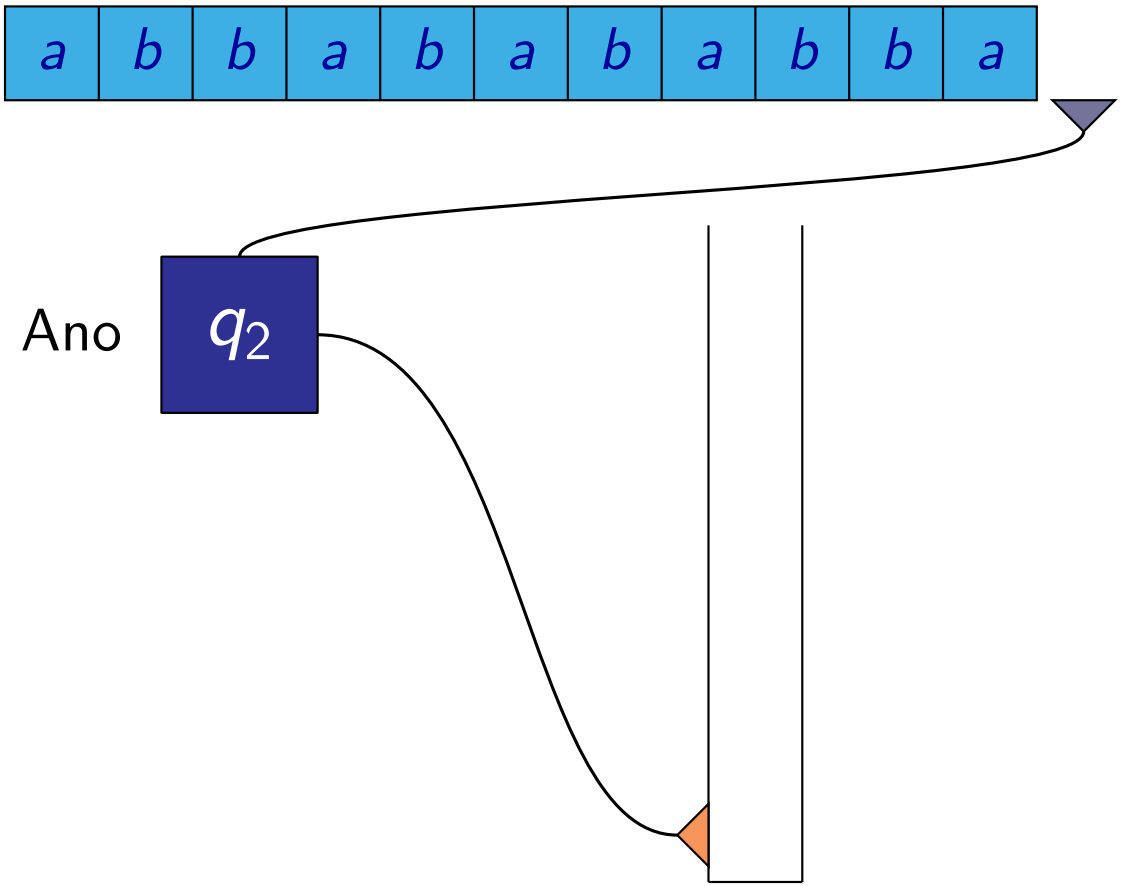
Zásobníkový automat

- Slovo *abbababababba* patří do jazyka $L = \{w \in \{a, b\}^* \mid w = w^R\}$



Zásobníkový automat

- Slovo *abbababababba* patří do jazyka $L = \{w \in \{a, b\}^* \mid w = w^R\}$



Zásobníkový automat

Definice

Zásobníkový automat (**ZA**) je uspořádaná šestice $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, X_0)$, kde

- Q je konečná neprázdná množina stavů
- Σ je konečná neprázdná množina zvaná vstupní abeceda
- Γ je konečná neprázdná množina zvaná zásobníková abeceda
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$ je (nedeterministická) přechodová funkce
- $q_0 \in Q$ je počáteční stav
- $X_0 \in \Gamma$ je počáteční zásobníkový symbol

Zásobníkový automat

Příklad: $L = \{ a^n b^n \mid n \geq 1 \}$

$\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, O)$, kde

- $Q = \{q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{O, I\}$
- $\delta(q_1, a, O) = \{(q_1, I)\}$ $\quad \delta(q_1, b, O) = \varnothing$
  $\delta(q_1, a, I) = \{(q_1, II)\}$ $\quad \delta(q_1, b, I) = \{(q_2, \varepsilon)\}$
  $\delta(q_2, a, I) = \varnothing$ $\quad \delta(q_2, b, I) = \{(q_2, \varepsilon)\}$
  $\delta(q_2, a, O) = \varnothing$ $\quad \delta(q_2, b, O) = \varnothing$

Poznámka: Často se uvádí jen ty hodnoty přechodové funkce, které přiřazují dané trojici něco jiného než prázdnou množinu.

Zásobníkový automat

Pro zápis přechodové funkce budeme též používat způsob zápisu, kdy se na přechodovou funkci díváme jako na sadu **pravidel**:

- Každému $q, q' \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $X \in \Gamma$ a $\alpha \in \Gamma^*$, kde
$$(q', \alpha) \in \delta(q, a, X)$$
odpovídá jedno pravidlo
$$qX \xrightarrow{a} q'\alpha\,.$$

Příklad: Pokud

$$\delta(q_5, b, C) = \{(q_3, ACC), (q_5, BB), (q_{13}, \varepsilon)\}$$

můžeme to reprezentovat jako tři pravidla:

$$q_5 C \xrightarrow{b} q_3 ACC \qquad q_5 C \xrightarrow{b} q_5 BB \qquad q_5 C \xrightarrow{b} q_{13}$$

Zásobníkový automat

Příklad: Dříve popsaný zásobníkový automat rozpoznávající jazyk $L = \{ a^n b^n \mid n \geq 1 \}$:

$\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, O)$, kde

- $Q = \{q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{O, I\}$
- $q_1 O \xrightarrow{a} q_1 I$
  $q_1 I \xrightarrow{a} q_1 II$
  $q_1 I \xrightarrow{b} q_2$
  $q_2 I \xrightarrow{b} q_2$

Zásobníkový automat

Příklad: $L = \{ w \in \{a, b\}^* \mid w = w^R \}$

$\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde

- $Q = \{q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{X, A, B\}$
- $\delta(q_1, a, X) = \{(q_1, AX), (q_2, X)\}$
  $\delta(q_1, a, A) = \{(q_1, AA), (q_2, A)\}$
  $\delta(q_1, a, B) = \{(q_1, AB), (q_2, B)\}$
  $\delta(q_1, \varepsilon, X) = \{(q_2, X)\}$
  $\delta(q_1, \varepsilon, A) = \{(q_2, A)\}$
  $\delta(q_1, \varepsilon, B) = \{(q_2, B)\}$
  $\delta(q_2, a, A) = \{(q_2, \varepsilon)\}$
  $\delta(q_2, a, B) = \varnothing$
  $\delta(q_2, a, X) = \varnothing$
  $\delta(q_1, b, X) = \{(q_1, BX), (q_2, X)\}$
  $\delta(q_1, b, A) = \{(q_1, BA), (q_2, A)\}$
  $\delta(q_1, b, B) = \{(q_1, BB), (q_2, B)\}$
  $\delta(q_2, \varepsilon, X) = \{(q_2, \varepsilon)\}$
  $\delta(q_2, \varepsilon, A) = \varnothing$
  $\delta(q_2, \varepsilon, B) = \varnothing$
  $\delta(q_2, b, A) = \varnothing$
  $\delta(q_2, b, B) = \{(q_2, \varepsilon)\}$
  $\delta(q_2, b, X) = \varnothing$

Zásobníkový automat

Příklad: $L = \{ w \in \{a, b\}^* \mid w = w^R \}$

$\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde

- $Q = \{q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{X, A, B\}$
- $q_1 X \xrightarrow{a} q_1 AX$
  $q_1 A \xrightarrow{a} q_1 AA$
  $q_1 B \xrightarrow{a} q_1 AB$
  $q_1 X \xrightarrow{a} q_2 X$
  $q_1 A \xrightarrow{a} q_2 A$
  $q_1 B \xrightarrow{a} q_2 B$

  $q_1 X \xrightarrow{b} q_1 BX$
  $q_1 A \xrightarrow{b} q_1 BA$
  $q_1 B \xrightarrow{b} q_1 BB$
  $q_1 X \xrightarrow{b} q_2 X$
  $q_1 A \xrightarrow{b} q_2 A$
  $q_1 B \xrightarrow{b} q_2 B$

  $q_2 X \xrightarrow{\varepsilon} q_2$
  $q_2 A \xrightarrow{a} q_2$
  $q_2 B \xrightarrow{b} q_2$
  $q_1 X \xrightarrow{\varepsilon} q_2 X$
  $q_1 A \xrightarrow{\varepsilon} q_2 A$
  $q_1 B \xrightarrow{\varepsilon} q_2 B$

Výpočet zásobníkového automatu

Vezměme si zásobníkový automat $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, X_0)$.

Konfigurace automatu $\mathcal{M}$:

- **Konfigurace ZA** je trojice

$$(q, w, \alpha)$$

  kde $q \in Q$, $w \in \Sigma^*$ a $\alpha \in \Gamma^*$.

- **Počáteční kofigurací** je kofigurace (q_0, w, X_0), kde $w \in \Sigma^*$.

Výpočet zásobníkového automatu

Kroky vykonané automatem $\mathcal{M}$:

- Binární relace $\longrightarrow$ na konfiguracích $\mathcal{M}$ reprezentuje možné kroky výpočtu, které může ZA $\mathcal{M}$ provést.

  To, že $\mathcal{M}$ může přejít jedním krokem z konfigurace (q, w, α) do konfigurace (q', w', α'), zapisujeme

$$(q, w, \alpha) \longrightarrow (q', w', \alpha').$$

- Tato relace $\longrightarrow$ je definována následovně:

$$(q, aw, X\beta) \longrightarrow (q', w, \alpha\beta) \iff (q', \alpha) \in \delta(q, a, X)$$

  kde $q, q' \in Q$, $a \in (\Sigma \cup \{\varepsilon\})$, $w \in \Sigma^*$, $X \in \Gamma$, $\alpha, \beta \in \Gamma^*$.

Výpočet zásobníkového automatu

Výpočty $\mathcal{M}$:

- Na konfiguracích $\mathcal{M}$ definujeme binární relaci $\longrightarrow^*$ jako reflexivní a tranzitivní uzávěr relace $\longrightarrow$, tj.,

$$(q, w, \alpha) \longrightarrow^* (q', w', \alpha')$$

jestliže existuje posloupnost konfigurací

$$(q_0, w_0, \alpha_0), (q_1, w_1, \alpha_1), \ldots, (q_n, w_n, \alpha_n)$$

taková, že

- $(q, w, \alpha) = (q_0, w_0, \alpha_0)$,
- $(q', w', \alpha') = (q_n, w_n, \alpha_n)$,
- $(q_i, w_i, \alpha_i) \longrightarrow (q_{i+1}, w_{i+1}, \alpha_{i+1})$ pro každé $i = 0, 1, \ldots, n-1$, tj.

$$(q_0, w_0, \alpha_0) \longrightarrow (q_1, w_1, \alpha_1) \longrightarrow \quad \cdots \quad \longrightarrow (q_n, w_n, \alpha_n)$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$$q_1 X \xrightarrow{a} q_1 AX \qquad q_1 X \xrightarrow{b} q_1 BX$$
$$q_1 A \xrightarrow{a} q_1 AA \qquad q_1 A \xrightarrow{b} q_1 BA$$
$$q_1 B \xrightarrow{a} q_1 AB \qquad q_1 B \xrightarrow{b} q_1 BB$$
$$q_1 X \xrightarrow{a} q_2 X \qquad q_1 X \xrightarrow{b} q_2 X$$
$$q_1 A \xrightarrow{a} q_2 A \qquad q_1 A \xrightarrow{b} q_2 A$$
$$q_1 B \xrightarrow{a} q_2 B \qquad q_1 B \xrightarrow{b} q_2 B$$
$$q_1 X \xrightarrow{\varepsilon} q_2 X$$
$$q_1 A \xrightarrow{\varepsilon} q_2 A$$
$$q_1 B \xrightarrow{\varepsilon} q_2 B$$
$$q_2 X \xrightarrow{\varepsilon} q_2$$
$$q_2 A \xrightarrow{a} q_2$$
$$q_2 B \xrightarrow{b} q_2$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$(q_1, abbababababba, X)$

$$q_1X \xrightarrow{a} q_1AX \qquad q_1X \xrightarrow{b} q_1BX$$
$$q_1A \xrightarrow{a} q_1AA \qquad q_1A \xrightarrow{b} q_1BA$$
$$q_1B \xrightarrow{a} q_1AB \qquad q_1B \xrightarrow{b} q_1BB$$
$$q_1X \xrightarrow{a} q_2X \qquad q_1X \xrightarrow{b} q_2X$$
$$q_1A \xrightarrow{a} q_2A \qquad q_1A \xrightarrow{b} q_2A$$
$$q_1B \xrightarrow{a} q_2B \qquad q_1B \xrightarrow{b} q_2B$$
$$q_1X \xrightarrow{\varepsilon} q_2X$$
$$q_1A \xrightarrow{\varepsilon} q_2A$$
$$q_1B \xrightarrow{\varepsilon} q_2B$$
$$q_2X \xrightarrow{\varepsilon} q_2$$
$$q_2A \xrightarrow{a} q_2$$
$$q_2B \xrightarrow{b} q_2$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$(q_1,\ abbababababba,\ X)$
$\longrightarrow\ (q_1,\ bbababababba,\ AX)$

$$q_1X \xrightarrow{a} q_1AX \qquad q_1X \xrightarrow{b} q_1BX$$
$$q_1A \xrightarrow{a} q_1AA \qquad q_1A \xrightarrow{b} q_1BA$$
$$q_1B \xrightarrow{a} q_1AB \qquad q_1B \xrightarrow{b} q_1BB$$
$$q_1X \xrightarrow{a} q_2X \qquad q_1X \xrightarrow{b} q_2X$$
$$q_1A \xrightarrow{a} q_2A \qquad q_1A \xrightarrow{b} q_2A$$
$$q_1B \xrightarrow{a} q_2B \qquad q_1B \xrightarrow{b} q_2B$$
$$q_1X \xrightarrow{\varepsilon} q_2X$$
$$q_1A \xrightarrow{\varepsilon} q_2A$$
$$q_1B \xrightarrow{\varepsilon} q_2B$$
$$q_2X \xrightarrow{\varepsilon} q_2$$
$$q_2A \xrightarrow{a} q_2$$
$$q_2B \xrightarrow{b} q_2$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$(q_1, abbababababba, X)$
$\longrightarrow (q_1, bbababababba, AX)$
$\longrightarrow (q_1, babababba, BAX)$

$q_1X \xrightarrow{a} q_1AX$
$q_1A \xrightarrow{a} q_1AA$
$q_1B \xrightarrow{a} q_1AB$
$q_1X \xrightarrow{a} q_2X$
$q_1A \xrightarrow{a} q_2A$
$q_1B \xrightarrow{a} q_2B$
$q_1X \xrightarrow{\varepsilon} q_2X$
$q_1A \xrightarrow{\varepsilon} q_2A$
$q_1B \xrightarrow{\varepsilon} q_2B$
$q_2X \xrightarrow{\varepsilon} q_2$
$q_2A \xrightarrow{a} q_2$
$q_2B \xrightarrow{b} q_2$

$q_1X \xrightarrow{b} q_1BX$
$q_1A \xrightarrow{b} q_1BA$
$q_1B \xrightarrow{b} q_1BB$
$q_1X \xrightarrow{b} q_2X$
$q_1A \xrightarrow{b} q_2A$
$q_1B \xrightarrow{b} q_2B$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$(q_1, abbababbba, X)$
$\longrightarrow (q_1, bbababbba, AX)$
$\longrightarrow (q_1, bababbba, BAX)$
$\longrightarrow (q_1, ababbba, BBAX)$

$q_1X \xrightarrow{a} q_1AX$
$q_1A \xrightarrow{a} q_1AA$
$q_1B \xrightarrow{a} q_1AB$
$q_1X \xrightarrow{a} q_2X$
$q_1A \xrightarrow{a} q_2A$
$q_1B \xrightarrow{a} q_2B$
$q_1X \xrightarrow{\varepsilon} q_2X$
$q_1A \xrightarrow{\varepsilon} q_2A$
$q_1B \xrightarrow{\varepsilon} q_2B$
$q_2X \xrightarrow{\varepsilon} q_2$
$q_2A \xrightarrow{a} q_2$
$q_2B \xrightarrow{b} q_2$

$q_1X \xrightarrow{b} q_1BX$
$q_1A \xrightarrow{b} q_1BA$
$q_1B \xrightarrow{b} q_1BB$
$q_1X \xrightarrow{b} q_2X$
$q_1A \xrightarrow{b} q_2A$
$q_1B \xrightarrow{b} q_2B$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$(q_1, abbababbba, X)$
$\longrightarrow (q_1, bbababbba, AX)$
$\longrightarrow (q_1, bababbba, BAX)$
$\longrightarrow (q_1, ababbba, BBAX)$
$\longrightarrow (q_1, babbba, ABBAX)$

$q_1X \xrightarrow{a} q_1AX$
$q_1A \xrightarrow{a} q_1AA$
$q_1B \xrightarrow{a} q_1AB$
$q_1X \xrightarrow{a} q_2X$
$q_1A \xrightarrow{a} q_2A$
$q_1B \xrightarrow{a} q_2B$
$q_1X \xrightarrow{\varepsilon} q_2X$
$q_1A \xrightarrow{\varepsilon} q_2A$
$q_1B \xrightarrow{\varepsilon} q_2B$
$q_2X \xrightarrow{\varepsilon} q_2$
$q_2A \xrightarrow{a} q_2$
$q_2B \xrightarrow{b} q_2$

$q_1X \xrightarrow{b} q_1BX$
$q_1A \xrightarrow{b} q_1BA$
$q_1B \xrightarrow{b} q_1BB$
$q_1X \xrightarrow{b} q_2X$
$q_1A \xrightarrow{b} q_2A$
$q_1B \xrightarrow{b} q_2B$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$(q_1, abbababbba, X)$
$\longrightarrow (q_1, bbababbba, AX)$
$\longrightarrow (q_1, bababbba, BAX)$
$\longrightarrow (q_1, ababbba, BBAX)$
$\longrightarrow (q_1, babbba, ABBAX)$
$\longrightarrow (q_1, abbba, BABBAX)$

$q_1X \xrightarrow{a} q_1AX$
$q_1A \xrightarrow{a} q_1AA$
$q_1B \xrightarrow{a} q_1AB$
$q_1X \xrightarrow{a} q_2X$
$q_1A \xrightarrow{a} q_2A$
$q_1B \xrightarrow{a} q_2B$
$q_1X \xrightarrow{\varepsilon} q_2X$
$q_1A \xrightarrow{\varepsilon} q_2A$
$q_1B \xrightarrow{\varepsilon} q_2B$
$q_2X \xrightarrow{\varepsilon} q_2$
$q_2A \xrightarrow{a} q_2$
$q_2B \xrightarrow{b} q_2$

$q_1X \xrightarrow{b} q_1BX$
$q_1A \xrightarrow{b} q_1BA$
$q_1B \xrightarrow{b} q_1BB$
$q_1X \xrightarrow{b} q_2X$
$q_1A \xrightarrow{b} q_2A$
$q_1B \xrightarrow{b} q_2B$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$(q_1, abbababbba, X)$
$\longrightarrow (q_1, bbababbba, AX)$
$\longrightarrow (q_1, bababbba, BAX)$
$\longrightarrow (q_1, ababbba, BBAX)$
$\longrightarrow (q_1, babbba, ABBAX)$
$\longrightarrow (q_1, ababba, BABBAX)$
$\longrightarrow (q_2, babba, BABBAX)$

$q_1X \xrightarrow{a} q_1AX$
$q_1A \xrightarrow{a} q_1AA$
$q_1B \xrightarrow{a} q_1AB$
$q_1X \xrightarrow{a} q_2X$
$q_1A \xrightarrow{a} q_2A$
$q_1B \xrightarrow{a} q_2B$
$q_1X \xrightarrow{\varepsilon} q_2X$
$q_1A \xrightarrow{\varepsilon} q_2A$
$q_1B \xrightarrow{\varepsilon} q_2B$
$q_2X \xrightarrow{\varepsilon} q_2$
$q_2A \xrightarrow{a} q_2$
$q_2B \xrightarrow{b} q_2$

$q_1X \xrightarrow{b} q_1BX$
$q_1A \xrightarrow{b} q_1BA$
$q_1B \xrightarrow{b} q_1BB$
$q_1X \xrightarrow{b} q_2X$
$q_1A \xrightarrow{b} q_2A$
$q_1B \xrightarrow{b} q_2B$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$(q_1, abbababbba, X)$
$\longrightarrow (q_1, bbababbba, AX)$
$\longrightarrow (q_1, bababbba, BAX)$
$\longrightarrow (q_1, ababbba, BBAX)$
$\longrightarrow (q_1, babbba, ABBAX)$
$\longrightarrow (q_1, ababba, BABBAX)$
$\longrightarrow (q_2, babba, BABBAX)$
$\longrightarrow (q_2, abba, ABBAX)$

$q_1X \xrightarrow{a} q_1AX$
$q_1A \xrightarrow{a} q_1AA$
$q_1B \xrightarrow{a} q_1AB$
$q_1X \xrightarrow{a} q_2X$
$q_1A \xrightarrow{a} q_2A$
$q_1B \xrightarrow{a} q_2B$
$q_1X \xrightarrow{\varepsilon} q_2X$
$q_1A \xrightarrow{\varepsilon} q_2A$
$q_1B \xrightarrow{\varepsilon} q_2B$
$q_2X \xrightarrow{\varepsilon} q_2$
$q_2A \xrightarrow{a} q_2$
$q_2B \xrightarrow{b} q_2$

$q_1X \xrightarrow{b} q_1BX$
$q_1A \xrightarrow{b} q_1BA$
$q_1B \xrightarrow{b} q_1BB$
$q_1X \xrightarrow{b} q_2X$
$q_1A \xrightarrow{b} q_2A$
$q_1B \xrightarrow{b} q_2B$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$(q_1, abbababbba, X)$
$\longrightarrow (q_1, bbababbba, AX)$
$\longrightarrow (q_1, bababbba, BAX)$
$\longrightarrow (q_1, ababbba, BBAX)$
$\longrightarrow (q_1, babbba, ABBAX)$
$\longrightarrow (q_1, abbba, BABBAX)$
$\longrightarrow (q_2, babba, BABBAX)$
$\longrightarrow (q_2, abba, ABBAX)$
$\longrightarrow (q_2, bba, BBAX)$

$q_1X \xrightarrow{a} q_1AX$
$q_1A \xrightarrow{a} q_1AA$
$q_1B \xrightarrow{a} q_1AB$
$q_1X \xrightarrow{a} q_2X$
$q_1A \xrightarrow{a} q_2A$
$q_1B \xrightarrow{a} q_2B$
$q_1X \xrightarrow{\varepsilon} q_2X$
$q_1A \xrightarrow{\varepsilon} q_2A$
$q_1B \xrightarrow{\varepsilon} q_2B$
$q_2X \xrightarrow{\varepsilon} q_2$
$q_2A \xrightarrow{a} q_2$
$q_2B \xrightarrow{b} q_2$

$q_1X \xrightarrow{b} q_1BX$
$q_1A \xrightarrow{b} q_1BA$
$q_1B \xrightarrow{b} q_1BB$
$q_1X \xrightarrow{b} q_2X$
$q_1A \xrightarrow{b} q_2A$
$q_1B \xrightarrow{b} q_2B$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$(q_1, abbababbba, X)$
$\longrightarrow (q_1, bbababbba, AX)$
$\longrightarrow (q_1, bababbba, BAX)$
$\longrightarrow (q_1, ababbba, BBAX)$
$\longrightarrow (q_1, babbba, ABBAX)$
$\longrightarrow (q_1, ababba, BABBAX)$
$\longrightarrow (q_2, babba, BABBAX)$
$\longrightarrow (q_2, abba, ABBAX)$
$\longrightarrow (q_2, bba, BBAX)$
$\longrightarrow (q_2, ba, BAX)$

$q_1X \xrightarrow{a} q_1AX$
$q_1A \xrightarrow{a} q_1AA$
$q_1B \xrightarrow{a} q_1AB$
$q_1X \xrightarrow{a} q_2X$
$q_1A \xrightarrow{a} q_2A$
$q_1B \xrightarrow{a} q_2B$
$q_1X \xrightarrow{\varepsilon} q_2X$
$q_1A \xrightarrow{\varepsilon} q_2A$
$q_1B \xrightarrow{\varepsilon} q_2B$
$q_2X \xrightarrow{\varepsilon} q_2$
$q_2A \xrightarrow{a} q_2$
$q_2B \xrightarrow{b} q_2$

$q_1X \xrightarrow{b} q_1BX$
$q_1A \xrightarrow{b} q_1BA$
$q_1B \xrightarrow{b} q_1BB$
$q_1X \xrightarrow{b} q_2X$
$q_1A \xrightarrow{b} q_2A$
$q_1B \xrightarrow{b} q_2B$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$(q_1, abbababbba, X)$
$\longrightarrow (q_1, bbababbba, AX)$
$\longrightarrow (q_1, bababbba, BAX)$
$\longrightarrow (q_1, ababbba, BBAX)$
$\longrightarrow (q_1, babbba, ABBAX)$
$\longrightarrow (q_1, ababba, BABBAX)$
$\longrightarrow (q_2, babba, BABBAX)$
$\longrightarrow (q_2, abba, ABBAX)$
$\longrightarrow (q_2, bba, BBAX)$
$\longrightarrow (q_2, ba, BAX)$
$\longrightarrow (q_2, a, AX)$

$q_1X \xrightarrow{a} q_1AX$
$q_1A \xrightarrow{a} q_1AA$
$q_1B \xrightarrow{a} q_1AB$
$q_1X \xrightarrow{a} q_2X$
$q_1A \xrightarrow{a} q_2A$
$q_1B \xrightarrow{a} q_2B$
$q_1X \xrightarrow{\varepsilon} q_2X$
$q_1A \xrightarrow{\varepsilon} q_2A$
$q_1B \xrightarrow{\varepsilon} q_2B$
$q_2X \xrightarrow{\varepsilon} q_2$
$q_2A \xrightarrow{a} q_2$
$q_2B \xrightarrow{b} q_2$

$q_1X \xrightarrow{b} q_1BX$
$q_1A \xrightarrow{b} q_1BA$
$q_1B \xrightarrow{b} q_1BB$
$q_1X \xrightarrow{b} q_2X$
$q_1A \xrightarrow{b} q_2A$
$q_1B \xrightarrow{b} q_2B$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$(q_1, abbababbba, X)$
$\longrightarrow (q_1, bbababbba, AX)$
$\longrightarrow (q_1, bababbba, BAX)$
$\longrightarrow (q_1, ababbba, BBAX)$
$\longrightarrow (q_1, babbba, ABBAX)$
$\longrightarrow (q_1, abbba, BABBAX)$
$\longrightarrow (q_2, babba, BABBAX)$
$\longrightarrow (q_2, abba, ABBAX)$
$\longrightarrow (q_2, bba, BBAX)$
$\longrightarrow (q_2, ba, BAX)$
$\longrightarrow (q_2, a, AX)$
$\longrightarrow (q_2, \varepsilon, X)$

$q_1X \xrightarrow{a} q_1AX$
$q_1A \xrightarrow{a} q_1AA$
$q_1B \xrightarrow{a} q_1AB$
$q_1X \xrightarrow{a} q_2X$
$q_1A \xrightarrow{a} q_2A$
$q_1B \xrightarrow{a} q_2B$
$q_1X \xrightarrow{\varepsilon} q_2X$
$q_1A \xrightarrow{\varepsilon} q_2A$
$q_1B \xrightarrow{\varepsilon} q_2B$
$q_2X \xrightarrow{\varepsilon} q_2$
$q_2A \xrightarrow{a} q_2$
$q_2B \xrightarrow{b} q_2$

$q_1X \xrightarrow{b} q_1BX$
$q_1A \xrightarrow{b} q_1BA$
$q_1B \xrightarrow{b} q_1BB$
$q_1X \xrightarrow{b} q_2X$
$q_1A \xrightarrow{b} q_2A$
$q_1B \xrightarrow{b} q_2B$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$(q_1, abbababbba, X)$
$\longrightarrow (q_1, bbababbba, AX)$
$\longrightarrow (q_1, bababbba, BAX)$
$\longrightarrow (q_1, ababbba, BBAX)$
$\longrightarrow (q_1, babbba, ABBAX)$
$\longrightarrow (q_1, abbba, BABBAX)$
$\longrightarrow (q_2, babba, BABBAX)$
$\longrightarrow (q_2, abba, ABBAX)$
$\longrightarrow (q_2, bba, BBAX)$
$\longrightarrow (q_2, ba, BAX)$
$\longrightarrow (q_2, a, AX)$
$\longrightarrow (q_2, \varepsilon, X)$
$\longrightarrow (q_2, \varepsilon, \varepsilon)$

$q_1X \xrightarrow{a} q_1AX$
$q_1A \xrightarrow{a} q_1AA$
$q_1B \xrightarrow{a} q_1AB$
$q_1X \xrightarrow{a} q_2X$
$q_1A \xrightarrow{a} q_2A$
$q_1B \xrightarrow{a} q_2B$
$q_1X \xrightarrow{\varepsilon} q_2X$
$q_1A \xrightarrow{\varepsilon} q_2A$
$q_1B \xrightarrow{\varepsilon} q_2B$
$q_2X \xrightarrow{\varepsilon} q_2$
$q_2A \xrightarrow{a} q_2$
$q_2B \xrightarrow{b} q_2$

$q_1X \xrightarrow{b} q_1BX$
$q_1A \xrightarrow{b} q_1BA$
$q_1B \xrightarrow{b} q_1BB$
$q_1X \xrightarrow{b} q_2X$
$q_1A \xrightarrow{b} q_2A$
$q_1B \xrightarrow{b} q_2B$

Výpočet zásobníkového automatu

V předchozí definici byla množina konfigurací definována jako

$$Conf = Q \times \Sigma^* \times \Gamma^*$$

a relace $\longrightarrow$ byla podmnožinou množiny $Conf \times Conf$.

Výpočet zásobníkového automatu

Alternativně bychom mohli definovat konfigurace tak, že by nezahrnovaly vstupní slovo:

$$Conf = Q \times \Gamma^*$$

Relaci $\longrightarrow$ bychom pak definovali jako podmnožinu množiny $Conf \times (\Sigma \cup \{\varepsilon\}) \times Conf$, přičemž zápis

$$q\alpha \xrightarrow{a} q'\alpha'$$

by označoval, že přečtením symbolu a (nebo nepřečtením ničeho, pokud $a = \varepsilon$) může přejít daný zásobníkový automat z konfigurace (q, α) do konfigurace (q', α'), tj.

$$qX\beta \xrightarrow{a} q'\gamma\beta \quad \Longleftrightarrow \quad (q', \gamma) \in \delta(q, a, X)$$

kde $q, q' \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $X \in \Gamma$ a $\beta, \gamma \in \Gamma^*$.

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$$q_1 X \xrightarrow{a} q_1 AX \qquad q_1 X \xrightarrow{b} q_1 BX$$
$$q_1 A \xrightarrow{a} q_1 AA \qquad q_1 A \xrightarrow{b} q_1 BA$$
$$q_1 B \xrightarrow{a} q_1 AB \qquad q_1 B \xrightarrow{b} q_1 BB$$
$$q_1 X \xrightarrow{a} q_2 X \qquad q_1 X \xrightarrow{b} q_2 X$$
$$q_1 A \xrightarrow{a} q_2 A \qquad q_1 A \xrightarrow{b} q_2 A$$
$$q_1 B \xrightarrow{a} q_2 B \qquad q_1 B \xrightarrow{b} q_2 B$$
$$q_1 X \xrightarrow{\varepsilon} q_2 X$$
$$q_1 A \xrightarrow{\varepsilon} q_2 A$$
$$q_1 B \xrightarrow{\varepsilon} q_2 B$$
$$q_2 X \xrightarrow{\varepsilon} q_2$$
$$q_2 A \xrightarrow{a} q_2$$
$$q_2 B \xrightarrow{b} q_2$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

q_1X

$$q_1X \xrightarrow{a} q_1AX \qquad q_1X \xrightarrow{b} q_1BX$$
$$q_1A \xrightarrow{a} q_1AA \qquad q_1A \xrightarrow{b} q_1BA$$
$$q_1B \xrightarrow{a} q_1AB \qquad q_1B \xrightarrow{b} q_1BB$$
$$q_1X \xrightarrow{a} q_2X \qquad q_1X \xrightarrow{b} q_2X$$
$$q_1A \xrightarrow{a} q_2A \qquad q_1A \xrightarrow{b} q_2A$$
$$q_1B \xrightarrow{a} q_2B \qquad q_1B \xrightarrow{b} q_2B$$
$$q_1X \xrightarrow{\varepsilon} q_2X$$
$$q_1A \xrightarrow{\varepsilon} q_2A$$
$$q_1B \xrightarrow{\varepsilon} q_2B$$
$$q_2X \xrightarrow{\varepsilon} q_2$$
$$q_2A \xrightarrow{a} q_2$$
$$q_2B \xrightarrow{b} q_2$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$$q_1 X \xrightarrow{a} q_1 AX$$

$$q_1 X \xrightarrow{a} q_1 AX$$
$$q_1 A \xrightarrow{a} q_1 AA$$
$$q_1 B \xrightarrow{a} q_1 AB$$
$$q_1 X \xrightarrow{a} q_2 X$$
$$q_1 A \xrightarrow{a} q_2 A$$
$$q_1 B \xrightarrow{a} q_2 B$$
$$q_1 X \xrightarrow{\varepsilon} q_2 X$$
$$q_1 A \xrightarrow{\varepsilon} q_2 A$$
$$q_1 B \xrightarrow{\varepsilon} q_2 B$$
$$q_2 X \xrightarrow{\varepsilon} q_2$$
$$q_2 A \xrightarrow{a} q_2$$
$$q_2 B \xrightarrow{b} q_2$$

$$q_1 X \xrightarrow{b} q_1 BX$$
$$q_1 A \xrightarrow{b} q_1 BA$$
$$q_1 B \xrightarrow{b} q_1 BB$$
$$q_1 X \xrightarrow{b} q_2 X$$
$$q_1 A \xrightarrow{b} q_2 A$$
$$q_1 B \xrightarrow{b} q_2 B$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$$q_1 X \xrightarrow{a} q_1 AX$$
$$\xrightarrow{b} q_1 BAX$$

$$q_1 X \xrightarrow{a} q_1 AX$$
$$q_1 A \xrightarrow{a} q_1 AA$$
$$q_1 B \xrightarrow{a} q_1 AB$$
$$q_1 X \xrightarrow{a} q_2 X$$
$$q_1 A \xrightarrow{a} q_2 A$$
$$q_1 B \xrightarrow{a} q_2 B$$
$$q_1 X \xrightarrow{\varepsilon} q_2 X$$
$$q_1 A \xrightarrow{\varepsilon} q_2 A$$
$$q_1 B \xrightarrow{\varepsilon} q_2 B$$
$$q_2 X \xrightarrow{\varepsilon} q_2$$
$$q_2 A \xrightarrow{a} q_2$$
$$q_2 B \xrightarrow{b} q_2$$

$$q_1 X \xrightarrow{b} q_1 BX$$
$$q_1 A \xrightarrow{b} q_1 BA$$
$$q_1 B \xrightarrow{b} q_1 BB$$
$$q_1 X \xrightarrow{b} q_2 X$$
$$q_1 A \xrightarrow{b} q_2 A$$
$$q_1 B \xrightarrow{b} q_2 B$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
&\xrightarrow{b} q_1BAX \\
&\xrightarrow{b} q_1BBAX
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
q_1A &\xrightarrow{a} q_1AA \\
q_1B &\xrightarrow{a} q_1AB \\
q_1X &\xrightarrow{a} q_2X \\
q_1A &\xrightarrow{a} q_2A \\
q_1B &\xrightarrow{a} q_2B \\
q_1X &\xrightarrow{\varepsilon} q_2X \\
q_1A &\xrightarrow{\varepsilon} q_2A \\
q_1B &\xrightarrow{\varepsilon} q_2B \\
q_2X &\xrightarrow{\varepsilon} q_2 \\
q_2A &\xrightarrow{a} q_2 \\
q_2B &\xrightarrow{b} q_2
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{b} q_1BX \\
q_1A &\xrightarrow{b} q_1BA \\
q_1B &\xrightarrow{b} q_1BB \\
q_1X &\xrightarrow{b} q_2X \\
q_1A &\xrightarrow{b} q_2A \\
q_1B &\xrightarrow{b} q_2B
\end{aligned}
$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
&\xrightarrow{b} q_1BAX \\
&\xrightarrow{b} q_1BBAX \\
&\xrightarrow{a} q_1ABBAX
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
q_1A &\xrightarrow{a} q_1AA \\
q_1B &\xrightarrow{a} q_1AB \\
q_1X &\xrightarrow{a} q_2X \\
q_1A &\xrightarrow{a} q_2A \\
q_1B &\xrightarrow{a} q_2B \\
q_1X &\xrightarrow{\varepsilon} q_2X \\
q_1A &\xrightarrow{\varepsilon} q_2A \\
q_1B &\xrightarrow{\varepsilon} q_2B \\
q_2X &\xrightarrow{\varepsilon} q_2 \\
q_2A &\xrightarrow{a} q_2 \\
q_2B &\xrightarrow{b} q_2
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{b} q_1BX \\
q_1A &\xrightarrow{b} q_1BA \\
q_1B &\xrightarrow{b} q_1BB \\
q_1X &\xrightarrow{b} q_2X \\
q_1A &\xrightarrow{b} q_2A \\
q_1B &\xrightarrow{b} q_2B
\end{aligned}
$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
&\xrightarrow{b} q_1BAX \\
&\xrightarrow{b} q_1BBAX \\
&\xrightarrow{a} q_1ABBAX \\
&\xrightarrow{b} q_1BABBAX
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
q_1A &\xrightarrow{a} q_1AA \\
q_1B &\xrightarrow{a} q_1AB \\
q_1X &\xrightarrow{a} q_2X \\
q_1A &\xrightarrow{a} q_2A \\
q_1B &\xrightarrow{a} q_2B \\
q_1X &\xrightarrow{\varepsilon} q_2X \\
q_1A &\xrightarrow{\varepsilon} q_2A \\
q_1B &\xrightarrow{\varepsilon} q_2B \\
q_2X &\xrightarrow{\varepsilon} q_2 \\
q_2A &\xrightarrow{a} q_2 \\
q_2B &\xrightarrow{b} q_2
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{b} q_1BX \\
q_1A &\xrightarrow{b} q_1BA \\
q_1B &\xrightarrow{b} q_1BB \\
q_1X &\xrightarrow{b} q_2X \\
q_1A &\xrightarrow{b} q_2A \\
q_1B &\xrightarrow{b} q_2B
\end{aligned}
$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
&\xrightarrow{b} q_1BAX \\
&\xrightarrow{b} q_1BBAX \\
&\xrightarrow{a} q_1ABBAX \\
&\xrightarrow{b} q_1BABBAX \\
&\xrightarrow{a} q_2BABBAX
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
q_1A &\xrightarrow{a} q_1AA \\
q_1B &\xrightarrow{a} q_1AB \\
q_1X &\xrightarrow{a} q_2X \\
q_1A &\xrightarrow{a} q_2A \\
q_1B &\xrightarrow{a} q_2B \\
q_1X &\xrightarrow{\varepsilon} q_2X \\
q_1A &\xrightarrow{\varepsilon} q_2A \\
q_1B &\xrightarrow{\varepsilon} q_2B \\
q_2X &\xrightarrow{\varepsilon} q_2 \\
q_2A &\xrightarrow{a} q_2 \\
q_2B &\xrightarrow{b} q_2
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{b} q_1BX \\
q_1A &\xrightarrow{b} q_1BA \\
q_1B &\xrightarrow{b} q_1BB \\
q_1X &\xrightarrow{b} q_2X \\
q_1A &\xrightarrow{b} q_2A \\
q_1B &\xrightarrow{b} q_2B
\end{aligned}
$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
&\xrightarrow{b} q_1BAX \\
&\xrightarrow{b} q_1BBAX \\
&\xrightarrow{a} q_1ABBAX \\
&\xrightarrow{b} q_1BABBAX \\
&\xrightarrow{a} q_2BABBAX \\
&\xrightarrow{b} q_2ABBAX
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
q_1A &\xrightarrow{a} q_1AA \\
q_1B &\xrightarrow{a} q_1AB \\
q_1X &\xrightarrow{a} q_2X \\
q_1A &\xrightarrow{a} q_2A \\
q_1B &\xrightarrow{a} q_2B \\
q_1X &\xrightarrow{\varepsilon} q_2X \\
q_1A &\xrightarrow{\varepsilon} q_2A \\
q_1B &\xrightarrow{\varepsilon} q_2B \\
q_2X &\xrightarrow{\varepsilon} q_2 \\
q_2A &\xrightarrow{a} q_2 \\
q_2B &\xrightarrow{b} q_2
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{b} q_1BX \\
q_1A &\xrightarrow{b} q_1BA \\
q_1B &\xrightarrow{b} q_1BB \\
q_1X &\xrightarrow{b} q_2X \\
q_1A &\xrightarrow{b} q_2A \\
q_1B &\xrightarrow{b} q_2B
\end{aligned}
$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
&\xrightarrow{b} q_1BAX \\
&\xrightarrow{b} q_1BBAX \\
&\xrightarrow{a} q_1ABBAX \\
&\xrightarrow{b} q_1BABBAX \\
&\xrightarrow{a} q_2BABBAX \\
&\xrightarrow{b} q_2ABBAX \\
&\xrightarrow{a} q_2BBAX
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
q_1A &\xrightarrow{a} q_1AA \\
q_1B &\xrightarrow{a} q_1AB \\
q_1X &\xrightarrow{a} q_2X \\
q_1A &\xrightarrow{a} q_2A \\
q_1B &\xrightarrow{a} q_2B \\
q_1X &\xrightarrow{\varepsilon} q_2X \\
q_1A &\xrightarrow{\varepsilon} q_2A \\
q_1B &\xrightarrow{\varepsilon} q_2B \\
q_2X &\xrightarrow{\varepsilon} q_2 \\
q_2A &\xrightarrow{a} q_2 \\
q_2B &\xrightarrow{b} q_2
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{b} q_1BX \\
q_1A &\xrightarrow{b} q_1BA \\
q_1B &\xrightarrow{b} q_1BB \\
q_1X &\xrightarrow{b} q_2X \\
q_1A &\xrightarrow{b} q_2A \\
q_1B &\xrightarrow{b} q_2B
\end{aligned}
$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
&\xrightarrow{b} q_1BAX \\
&\xrightarrow{b} q_1BBAX \\
&\xrightarrow{a} q_1ABBAX \\
&\xrightarrow{b} q_1BABBAX \\
&\xrightarrow{a} q_2BABBAX \\
&\xrightarrow{b} q_2ABBAX \\
&\xrightarrow{a} q_2BBAX \\
&\xrightarrow{b} q_2BAX
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
q_1A &\xrightarrow{a} q_1AA \\
q_1B &\xrightarrow{a} q_1AB \\
q_1X &\xrightarrow{a} q_2X \\
q_1A &\xrightarrow{a} q_2A \\
q_1B &\xrightarrow{a} q_2B \\
q_1X &\xrightarrow{\varepsilon} q_2X \\
q_1A &\xrightarrow{\varepsilon} q_2A \\
q_1B &\xrightarrow{\varepsilon} q_2B \\
q_2X &\xrightarrow{\varepsilon} q_2 \\
q_2A &\xrightarrow{a} q_2 \\
q_2B &\xrightarrow{b} q_2
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{b} q_1BX \\
q_1A &\xrightarrow{b} q_1BA \\
q_1B &\xrightarrow{b} q_1BB \\
q_1X &\xrightarrow{b} q_2X \\
q_1A &\xrightarrow{b} q_2A \\
q_1B &\xrightarrow{b} q_2B
\end{aligned}
$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
&\xrightarrow{b} q_1BAX \\
&\xrightarrow{b} q_1BBAX \\
&\xrightarrow{a} q_1ABBAX \\
&\xrightarrow{b} q_1BABBAX \\
&\xrightarrow{a} q_2BABBAX \\
&\xrightarrow{b} q_2ABBAX \\
&\xrightarrow{a} q_2BBAX \\
&\xrightarrow{b} q_2BAX \\
&\xrightarrow{b} q_2AX
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
q_1A &\xrightarrow{a} q_1AA \\
q_1B &\xrightarrow{a} q_1AB \\
q_1X &\xrightarrow{a} q_2X \\
q_1A &\xrightarrow{a} q_2A \\
q_1B &\xrightarrow{a} q_2B \\
q_1X &\xrightarrow{\varepsilon} q_2X \\
q_1A &\xrightarrow{\varepsilon} q_2A \\
q_1B &\xrightarrow{\varepsilon} q_2B \\
q_2X &\xrightarrow{\varepsilon} q_2 \\
q_2A &\xrightarrow{a} q_2 \\
q_2B &\xrightarrow{b} q_2
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{b} q_1BX \\
q_1A &\xrightarrow{b} q_1BA \\
q_1B &\xrightarrow{b} q_1BB \\
q_1X &\xrightarrow{b} q_2X \\
q_1A &\xrightarrow{b} q_2A \\
q_1B &\xrightarrow{b} q_2B
\end{aligned}
$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
&\xrightarrow{b} q_1BAX \\
&\xrightarrow{b} q_1BBAX \\
&\xrightarrow{a} q_1ABBAX \\
&\xrightarrow{b} q_1BABBAX \\
&\xrightarrow{a} q_2BABBAX \\
&\xrightarrow{b} q_2ABBAX \\
&\xrightarrow{a} q_2BBAX \\
&\xrightarrow{b} q_2BAX \\
&\xrightarrow{b} q_2AX \\
&\xrightarrow{a} q_2X
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
q_1A &\xrightarrow{a} q_1AA \\
q_1B &\xrightarrow{a} q_1AB \\
q_1X &\xrightarrow{a} q_2X \\
q_1A &\xrightarrow{a} q_2A \\
q_1B &\xrightarrow{a} q_2B \\
q_1X &\xrightarrow{\varepsilon} q_2X \\
q_1A &\xrightarrow{\varepsilon} q_2A \\
q_1B &\xrightarrow{\varepsilon} q_2B \\
q_2X &\xrightarrow{\varepsilon} q_2 \\
q_2A &\xrightarrow{a} q_2 \\
q_2B &\xrightarrow{b} q_2
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{b} q_1BX \\
q_1A &\xrightarrow{b} q_1BA \\
q_1B &\xrightarrow{b} q_1BB \\
q_1X &\xrightarrow{b} q_2X \\
q_1A &\xrightarrow{b} q_2A \\
q_1B &\xrightarrow{b} q_2B
\end{aligned}
$$

Výpočet zásobníkového automatu

Příklad: $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_1, X)$, kde $Q = \{q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{X, A, B\}$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
&\xrightarrow{b} q_1BAX \\
&\xrightarrow{b} q_1BBAX \\
&\xrightarrow{a} q_1ABBAX \\
&\xrightarrow{b} q_1BABBAX \\
&\xrightarrow{a} q_2BABBAX \\
&\xrightarrow{b} q_2ABBAX \\
&\xrightarrow{a} q_2BBAX \\
&\xrightarrow{b} q_2BAX \\
&\xrightarrow{b} q_2AX \\
&\xrightarrow{a} q_2X \\
&\xrightarrow{\varepsilon} q_2
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{a} q_1AX \\
q_1A &\xrightarrow{a} q_1AA \\
q_1B &\xrightarrow{a} q_1AB \\
q_1X &\xrightarrow{a} q_2X \\
q_1A &\xrightarrow{a} q_2A \\
q_1B &\xrightarrow{a} q_2B \\
q_1X &\xrightarrow{\varepsilon} q_2X \\
q_1A &\xrightarrow{\varepsilon} q_2A \\
q_1B &\xrightarrow{\varepsilon} q_2B \\
q_2X &\xrightarrow{\varepsilon} q_2 \\
q_2A &\xrightarrow{a} q_2 \\
q_2B &\xrightarrow{b} q_2
\end{aligned}
$$

$$
\begin{aligned}
q_1X &\xrightarrow{b} q_1BX \\
q_1A &\xrightarrow{b} q_1BA \\
q_1B &\xrightarrow{b} q_1BB \\
q_1X &\xrightarrow{b} q_2X \\
q_1A &\xrightarrow{b} q_2A \\
q_1B &\xrightarrow{b} q_2B
\end{aligned}
$$

Zásobníkový automat — přijímání slov

Používají se dvě různé definice toho, kdy automat přijímá dané slovo:

- Jestliže zásobníkový automat $\mathcal{M}$ přijímá **prázdným zásobníkem**, příjme slovo w tehdy, jestliže existuje výpočet automatu $\mathcal{M}$ nad slovem w takový, že automat přečte celé slovo w a po jeho přečtení má prázdný zásobník.

- Jestliže zásobníkový automat $\mathcal{M}$ přijímá pomocí **přijímajících stavů**, příjme slovo w tehdy, jestliže existuje výpočet automatu $\mathcal{M}$ nad slovem w takový, že automat přečte celé slovo w a po jeho přečtení je řídící jednotka automatu $\mathcal{M}$ v některém z přijímajících stavů z množiny F.

Zásobníkový automat — přijímání slov

- Slovo $w \in \Sigma^*$ je **přijímáno** ZA $\mathcal{M}$ **prázdným zásobníkem** právě tehdy, když

$$(q_0, w, X_0) \longrightarrow^* (q, \varepsilon, \varepsilon)$$

pro nějaké $q \in Q$.

Definice

Jazyk $\mathcal{L}(\mathcal{M})$ přijímaný ZA $\mathcal{M}$ **prázdným zásobníkem** je definován jako množina všech slov přijímaných ZA $\mathcal{M}$ prázdným zásobníkem, tj.

$$\mathcal{L}(\mathcal{M}) = \{ w \in \Sigma^* \mid (\exists q \in Q)((q_0, w, X_0) \longrightarrow^* (q, \varepsilon, \varepsilon)) \}.$$

Zásobníkový automat — přijímání slov

Rozšiřme definici ZA $\mathcal{M}$ o množinu **přijímajících stavů** F (kde $F \subseteq Q$).

- Slovo $w \in \Sigma^*$ je **přijímáno** ZA $\mathcal{M}$ **přijímajícím stavem** právě tehdy, když

$$(q_0, w, X_0) \longrightarrow^* (q, \varepsilon, \alpha)$$

pro nějaké $q \in F$ a $\alpha \in \Gamma^*$.

Definice

Jazyk $\mathcal{L}(\mathcal{M})$ **přijímaný** ZA $\mathcal{M}$ **přijímajícím stavem** je definován jako

$$\mathcal{L}(\mathcal{M}) = \{ w \in \Sigma^* \mid (\exists q \in F)(\exists \alpha \in \Gamma^*)((q_0, w, X_0) \longrightarrow^* (q, \varepsilon, \alpha)) \}.$$

Druhy zásobníkových automatů

V případě **nedeterministických** zásobníkových automatů není z hlediska jazyků, jaké jsou schopny tyto automaty rozpoznávat, rozdíl mezi rozpoznáváním prázdným zásobníkem a rozpoznáváním přijímajícím stavem.

Snadno sestrojíme:

- K danému (nedeterministickému) zásobníkovému automatu rozpoznávajícímu nějaký jazyk L prázdným zásobníkem ekvivalentní (nedeterministický) zásobníkový automat rozpoznávající jazyk L pomocí přijímajících stavů.
- K danému (nedeterministickému) zásobníkovému automatu rozpoznávajícímu nějaký jazyk L pomocí přijímajících stavů ekvivalentní (nedeterministický) zásobníkový automat rozpoznávající jazyk L prázdným zásobníkem.

Deterministické zásobníkové automaty

Zásobníkový automat $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, X_0)$ je **deterministický**, jestliže:

- Pro každé $q \in Q$, $a \in (\Sigma \cup \{\varepsilon\})$ a $X \in \Gamma$ platí:
$$|\delta(q, a, X)| \leq 1$$
- Pro každé $q \in Q$ a $X \in \Gamma$ platí nejvýše jedna z následujících dvou možností:
  - Existuje pravidlo $qX \xrightarrow{\varepsilon} q'\alpha$ pro nějaké $q' \in Q$ a $\alpha \in \Gamma^*$.
  - Existuje pravidlo $qX \xrightarrow{a} q'\alpha$ pro nějaké $a \in \Sigma$, $q' \in Q$ a $\alpha \in \Gamma^*$.

Deterministické zásobníkové automaty

Všimněme si, že **deterministické** zásobníkové automaty přijímající prázdným zásobníkem jsou schopny rozpoznávat jen **bezprefixové** jazyky, tj. jazyky L, kde:

- pokud $w \in L$, pak neexistuje žádné slovo $w' \in L$ takové, že w je vlastním prefixem slova w'.

Poznámka: Místo jazyka $L \subseteq \Sigma^*$, který může a nemusí být bezprefixový, můžeme vzít bezprefixový jazyk

$$L' = L \cdot \{\dashv\}$$

nad abecedou $\Sigma \cup \{\dashv\}$, kde $\dashv \notin \Sigma$ je speciální „zarážka“ označující konec slova.
Tj. místo zjišťování, zda $w \in L$, kde $w \in \Sigma^*$, můžeme zjišťovat, zda $(w \dashv) \in L'$.

Deterministické zásobníkové automaty

- Ke každému deterministickému zásobníkovému automatu přijímajícímu prázdným zásobníkem je možné snadno sestrojit ekvivalentní deterministický zásobníkový automat přijímající pomocí přijímajících stavů.

- Ke každému deterministickému zásobníkovému automatu přijímajícímu jazyk L (kde $L \subseteq \Sigma^*$) pomocí přijímajících stavů je možné snadno sestrojit deterministický zásobníkový automat přimající prázdným zásobníkem jazyk $L \cdot \{\dashv\}$, kde $\dashv \notin \Sigma$.

Ekvivalence bezkontextových gramatik a zás. automatů

Věta

Ke každé bezkontextové gramatice $\mathcal{G}$ lze sestrojit nedeterministický zásobníkový automat $\mathcal{M}$ přijímající prázdným zásobníkem takový, že $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{G})$.

Důkaz: Pro BG $\mathcal{G} = (\Pi, \Sigma, S, P)$ vytvoříme $\mathcal{M} = (\{q_0\}, \Sigma, \Gamma, \delta, q_0, S)$, kde

- $\Gamma = \Pi \cup \Sigma$
- Pro každé pravidlo $(X \to \alpha) \in P$ z bezkontextové gramatiky $\mathcal{G}$ (kde $X \in \Pi$ a $\alpha \in (\Pi \cup \Sigma)^*$) přidáme do přechodové funkce δ zásobníkového automatu $\mathcal{M}$ odpovídající pravidlo

$$q_0 X \xrightarrow{\varepsilon} q_0 \alpha \,.$$

- Pro každý symbol $a \in \Sigma$ přidáme do přechodové funkce δ zásobníkového automatu $\mathcal{M}$ pravidlo

$$q_0 a \xrightarrow{a} q_0 \,.$$

Ekvivalence bezkontextových gramatik a zás. automatů

Příklad: Uvažujme bezkontextovou gramatiku $\mathcal{G} = (\Pi, \Sigma, S, P)$, kde

- $\Pi = \{S, E, T, F\}$
- $\Sigma = \{\, \mathtt{a}, +, *, (,), \dashv \}$
- Množina P obsahuje následující pravidla:

$$
\begin{aligned}
S &\to E \dashv \\
E &\to T \mid E{+}T \\
T &\to F \mid T{*}F \\
F &\to \mathtt{a} \mid (E)
\end{aligned}
$$

Ekvivalence bezkontextových gramatik a zás. automatů

K dané gramatice $\mathcal{G} = (\Pi, \Sigma, S, P)$ s pravidly

$$\begin{aligned} S &\to E \dashv \\ E &\to T \mid E{+}T \\ T &\to F \mid T{*}F \\ F &\to \mathtt{a} \mid (E) \end{aligned}$$

sestrojíme zásobníkový automat $\mathcal{M} = (\{q_0\}, \Sigma, \Gamma, \delta, q_0, S)$, kde

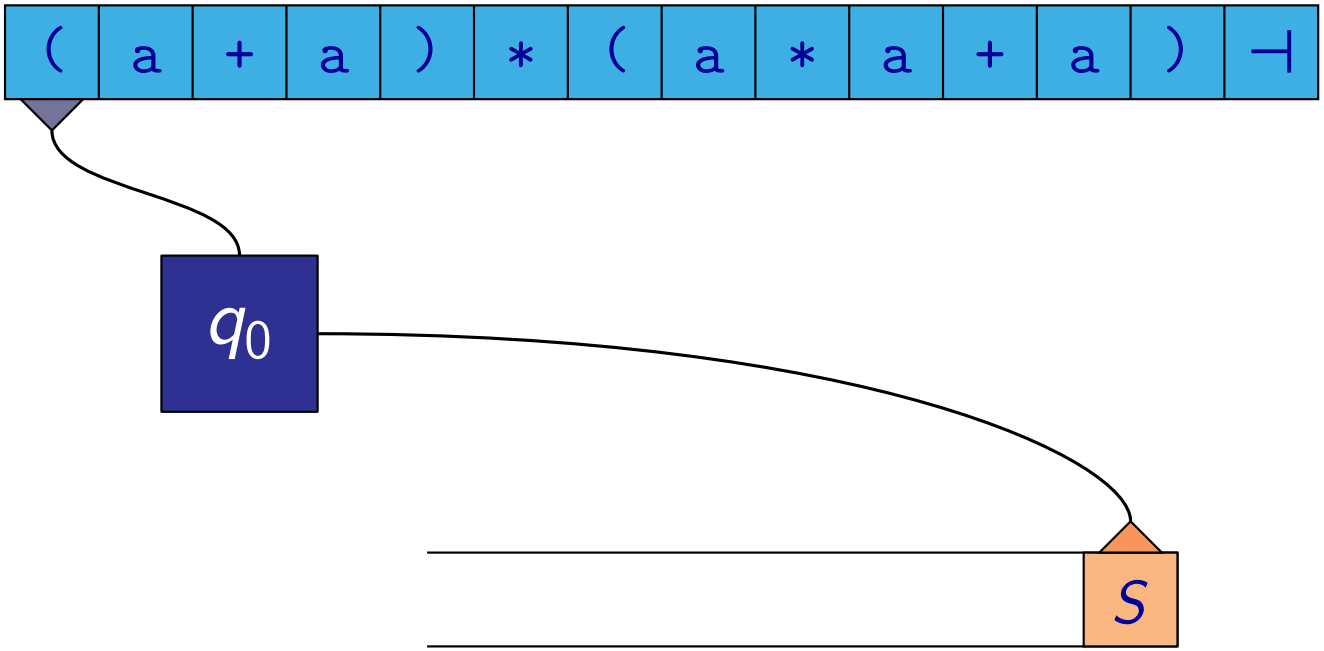
- $\Sigma = \{\, \mathtt{a}, +, *, (,), \dashv \,\}$
- $\Gamma = \{\, S, E, T, F, \mathtt{a}, +, *, (,), \dashv \,\}$
- Přechodová funkce δ obsahuje následující pravidla:

$$q_0 S \xrightarrow{\varepsilon} q_0 E \dashv \qquad q_0 E \xrightarrow{\varepsilon} q_0 T \qquad q_0 E \xrightarrow{\varepsilon} q_0 E{+}T \qquad q_0 T \xrightarrow{\varepsilon} q_0 F \qquad q_0 T \xrightarrow{\varepsilon} q_0 T{*}F$$

$$q_0 F \xrightarrow{\varepsilon} q_0 \mathtt{a} \qquad q_0 F \xrightarrow{\varepsilon} q_0 (E)$$

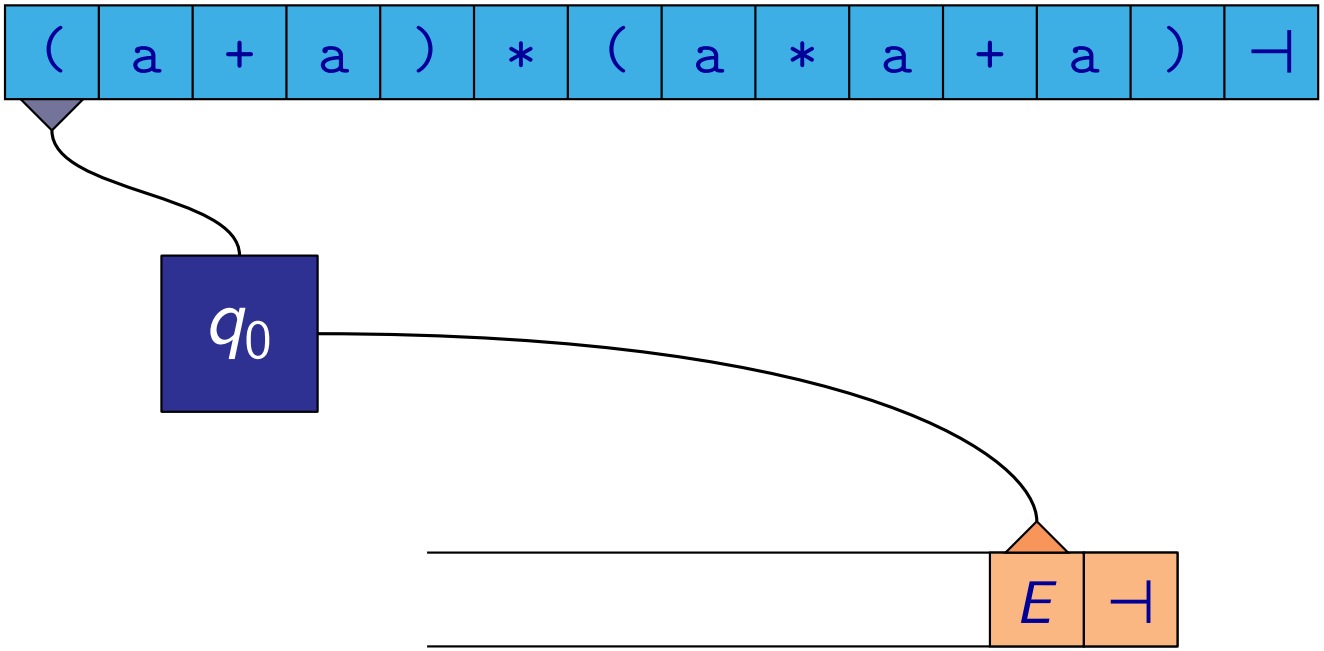
$$q_0 \mathtt{a} \xrightarrow{\mathtt{a}} q_0 \qquad q_0 + \xrightarrow{+} q_0 \qquad q_0 * \xrightarrow{*} q_0 \qquad q_0 (\xrightarrow{(} q_0 \qquad q_0) \xrightarrow{)} q_0 \qquad q_0 \dashv \xrightarrow{\dashv} q_0$$

Ekvivalence bezkontextových gramatik a zás. automatů



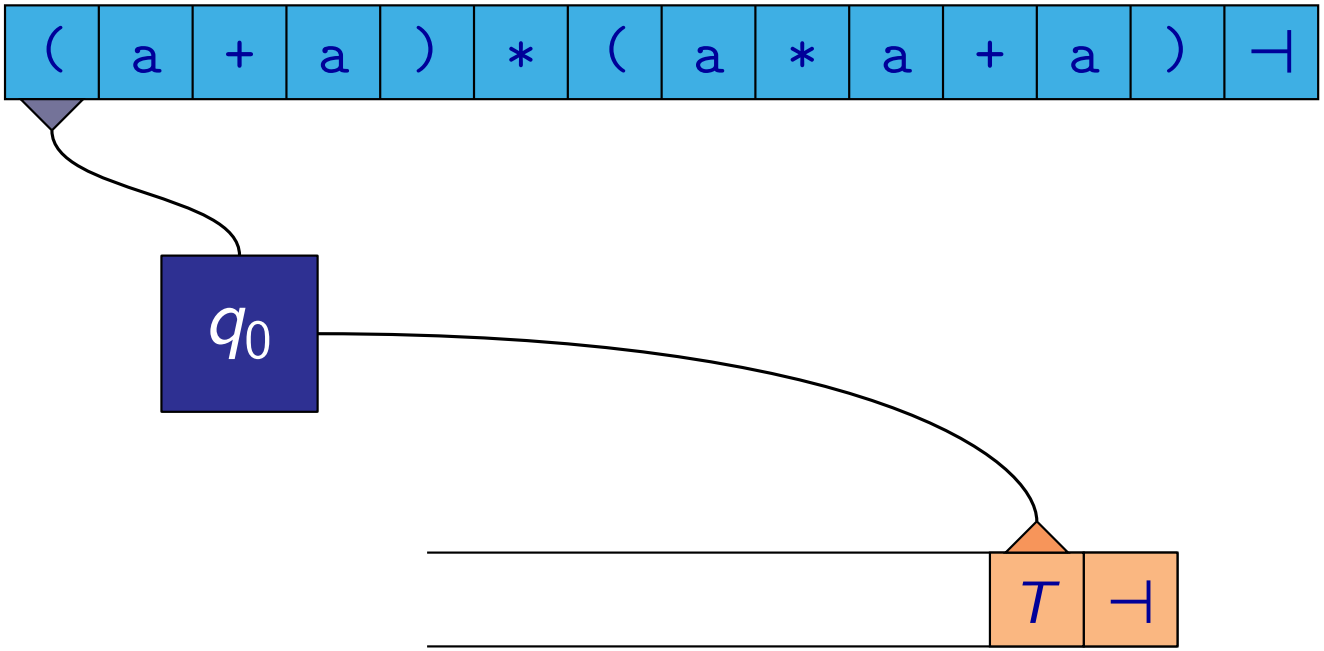
$\underline{S}$

Ekvivalence bezkontextových gramatik a zás. automatů



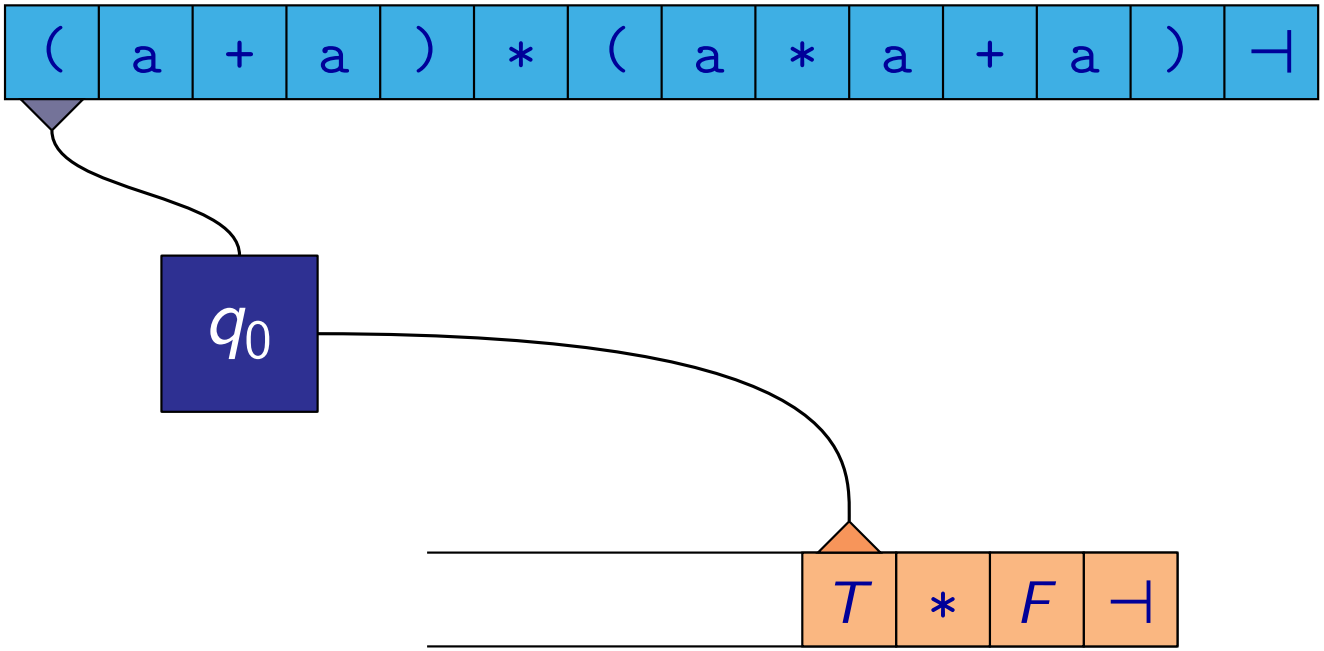
$\underline{S} \Rightarrow \underline{E} \dashv$

Ekvivalence bezkontextových gramatik a zás. automatů



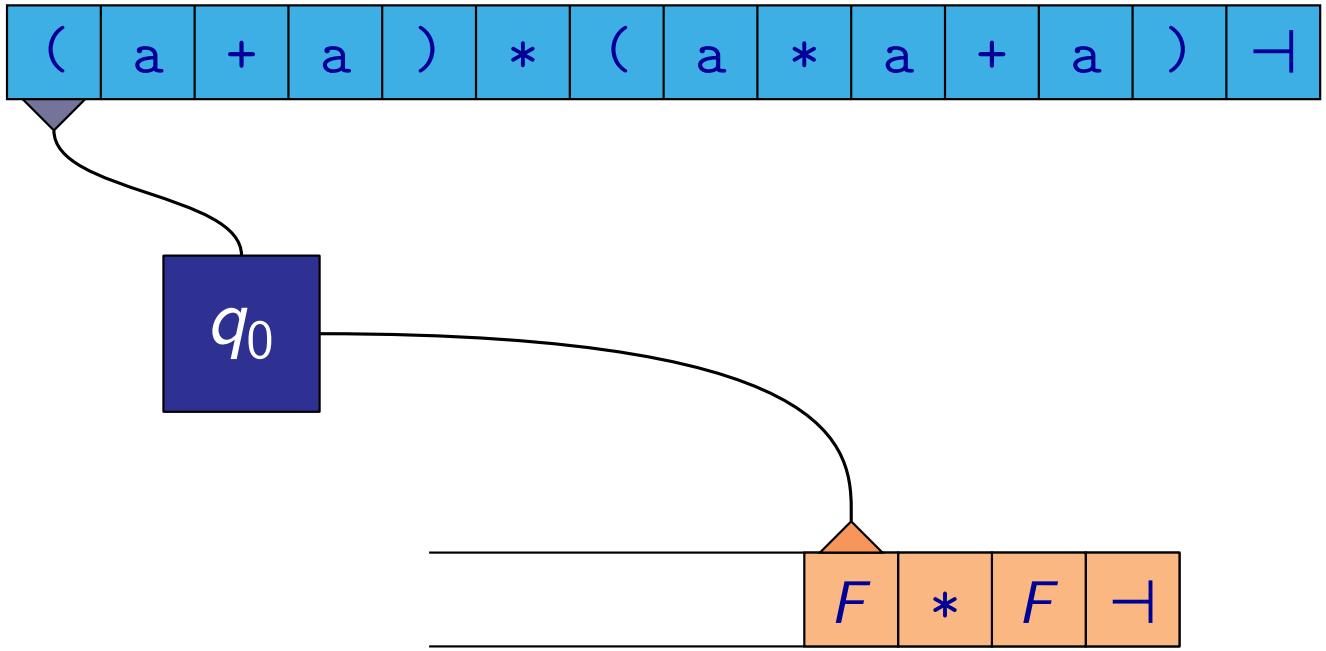
$$\underline{S} \Rightarrow \underline{E}\dashv \Rightarrow \underline{T}\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



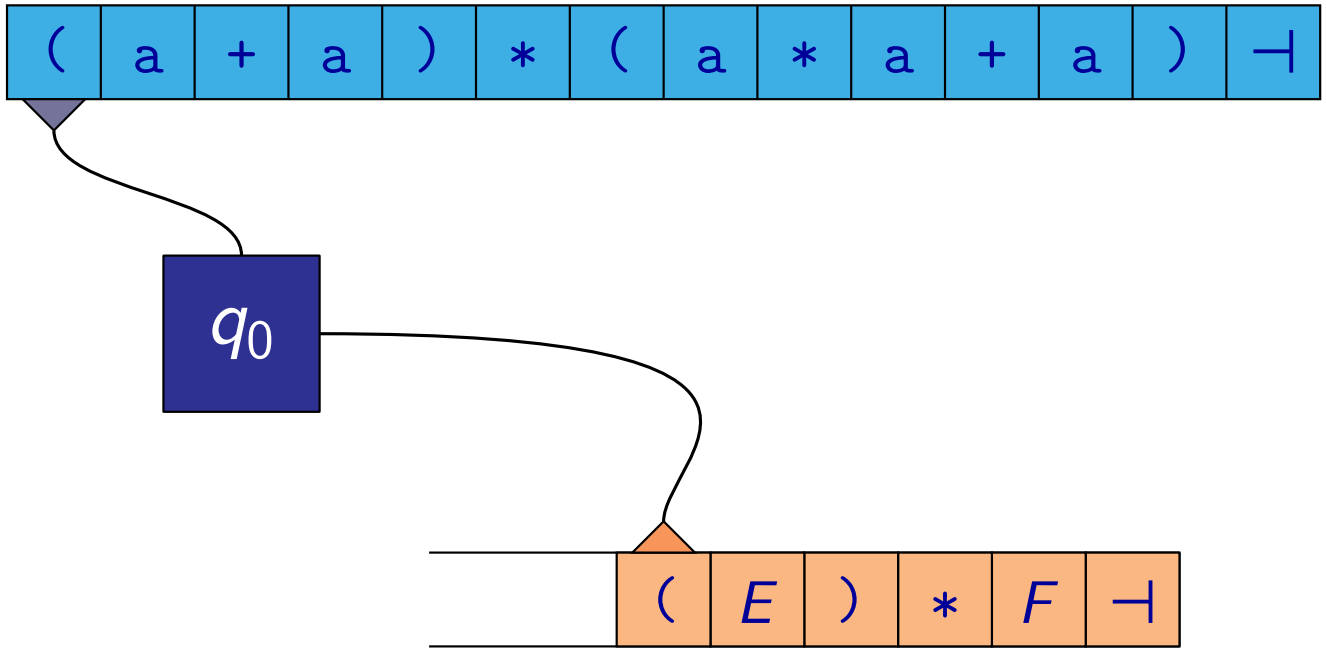
$$\underline{S} \Rightarrow \underline{E}\dashv \Rightarrow \underline{T}\dashv \Rightarrow \underline{T}*F\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



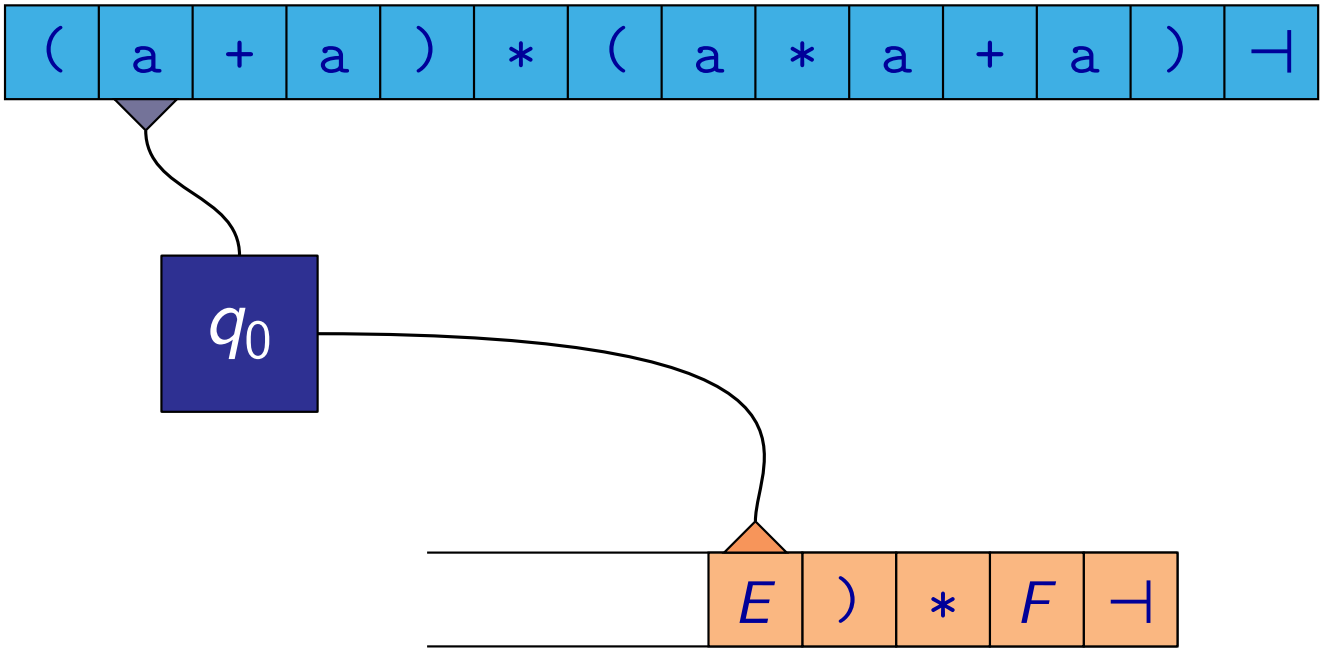
$$\underline{S} \Rightarrow \underline{E}\dashv \Rightarrow \underline{T}\dashv \Rightarrow \underline{T}*F\dashv \Rightarrow \underline{F}*F\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



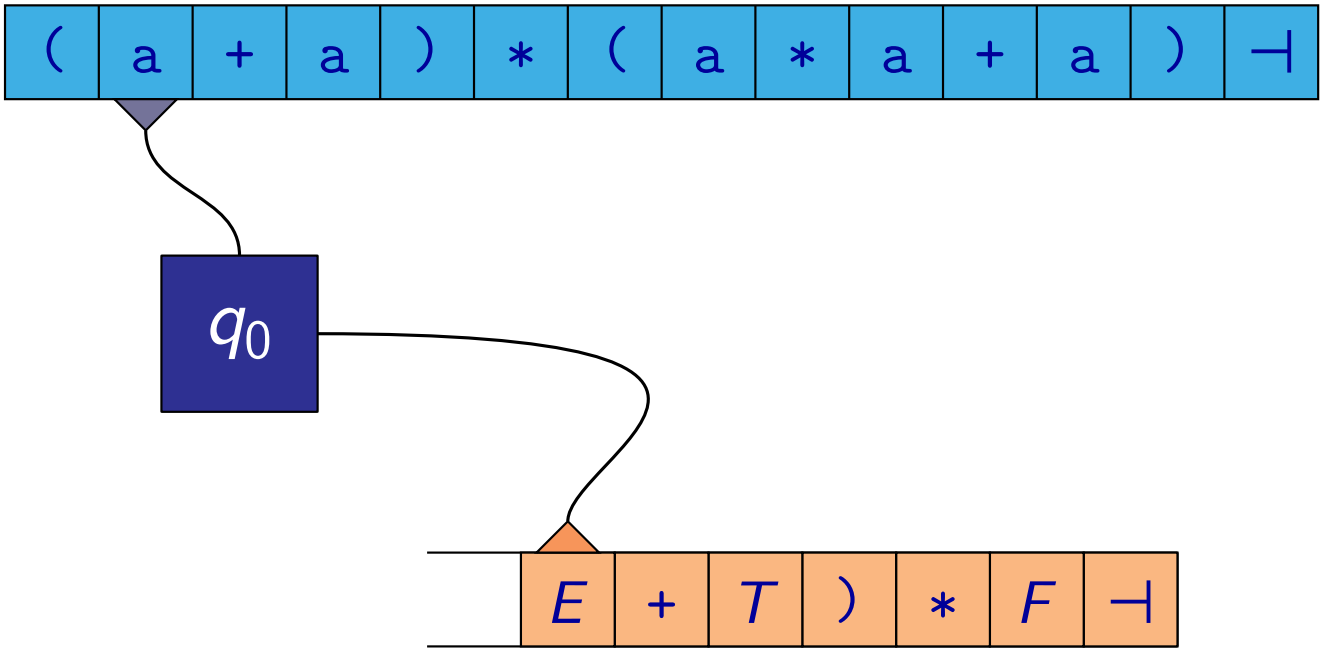
$$\underline{S} \Rightarrow \underline{E}\dashv \Rightarrow \underline{T}\dashv \Rightarrow \underline{T}*F\dashv \Rightarrow \underline{F}*F\dashv \Rightarrow (\underline{E})*F\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



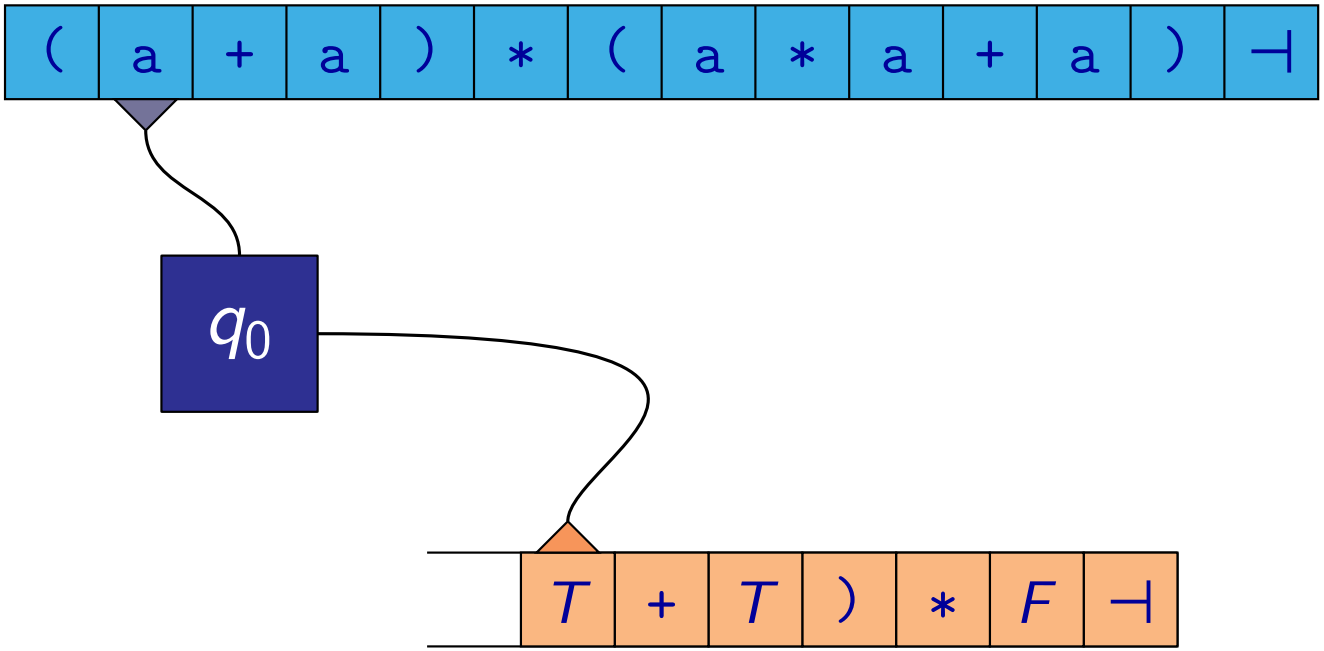
$$\underline{S} \Rightarrow \underline{E}\dashv \Rightarrow \underline{T}\dashv \Rightarrow \underline{T}*F\dashv \Rightarrow \underline{F}*F\dashv \Rightarrow (\underline{E})*F\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



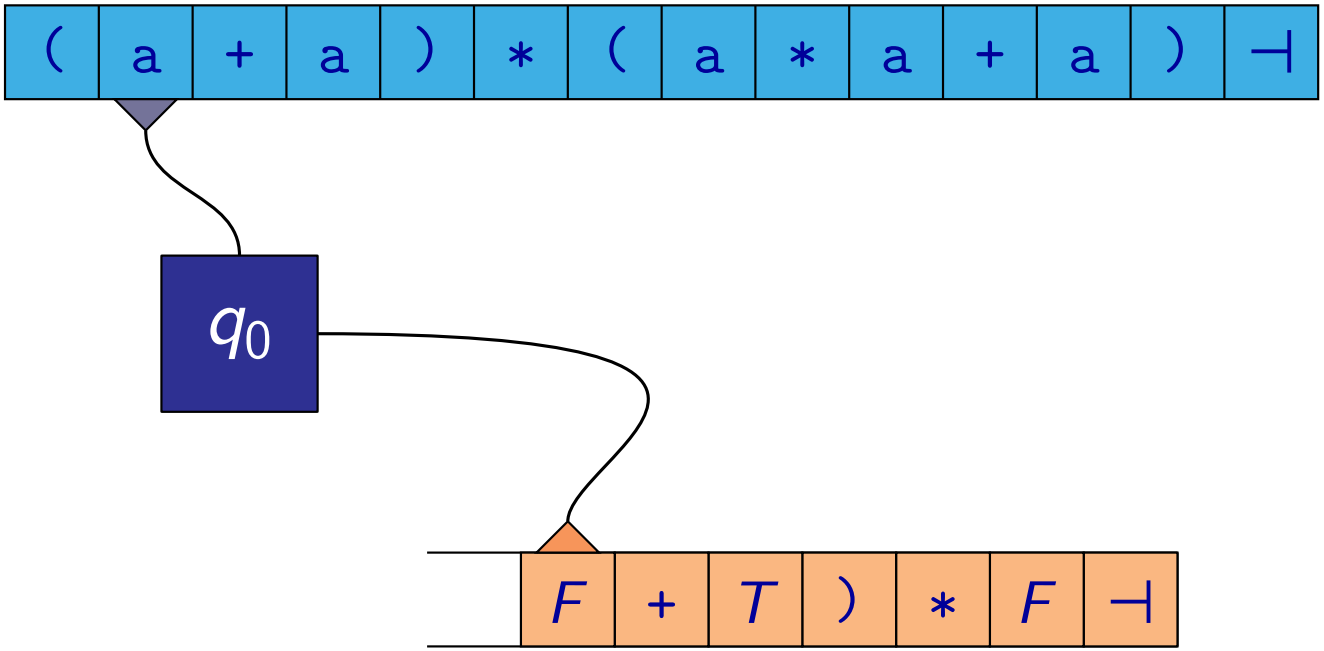
$$\cdots \Rightarrow \underline{T}\dashv \Rightarrow \underline{T}*F\dashv \Rightarrow \underline{F}*F\dashv \Rightarrow (\underline{E})*F\dashv \Rightarrow (\underline{E}+T)*F\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



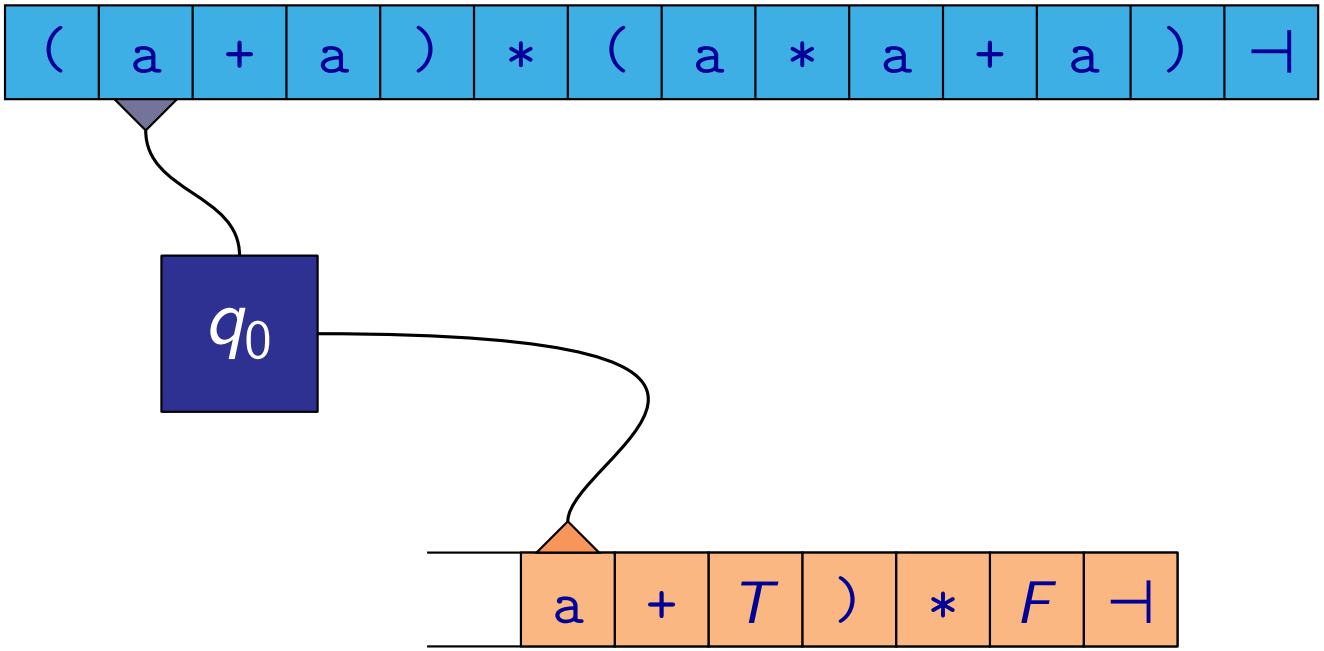
$$\cdots \Rightarrow \underline{F}*F\dashv \Rightarrow (\underline{E})*F\dashv \Rightarrow (\underline{E}+T)*F\dashv \Rightarrow (\underline{T}+T)*F\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



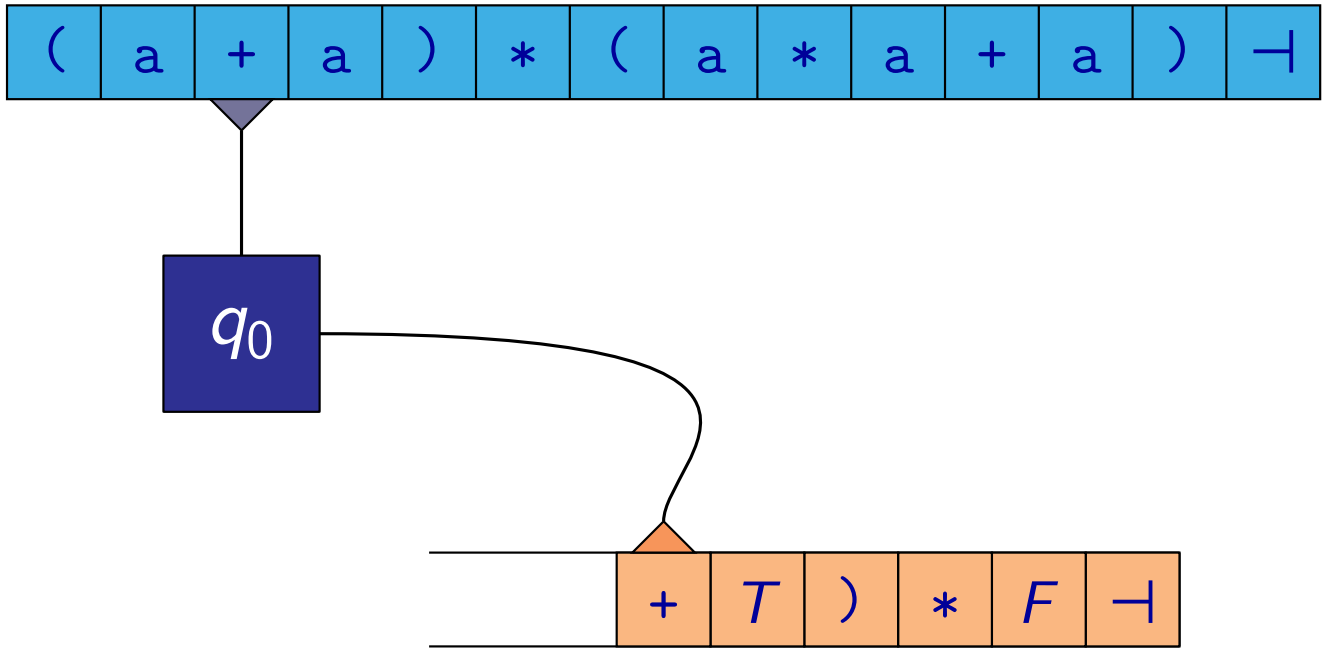
$$\cdots \Rightarrow (\underline{E})*F\dashv \Rightarrow (\underline{E}+T)*F\dashv \Rightarrow (\underline{T}+T)*F\dashv \Rightarrow (\underline{F}+T)*F\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



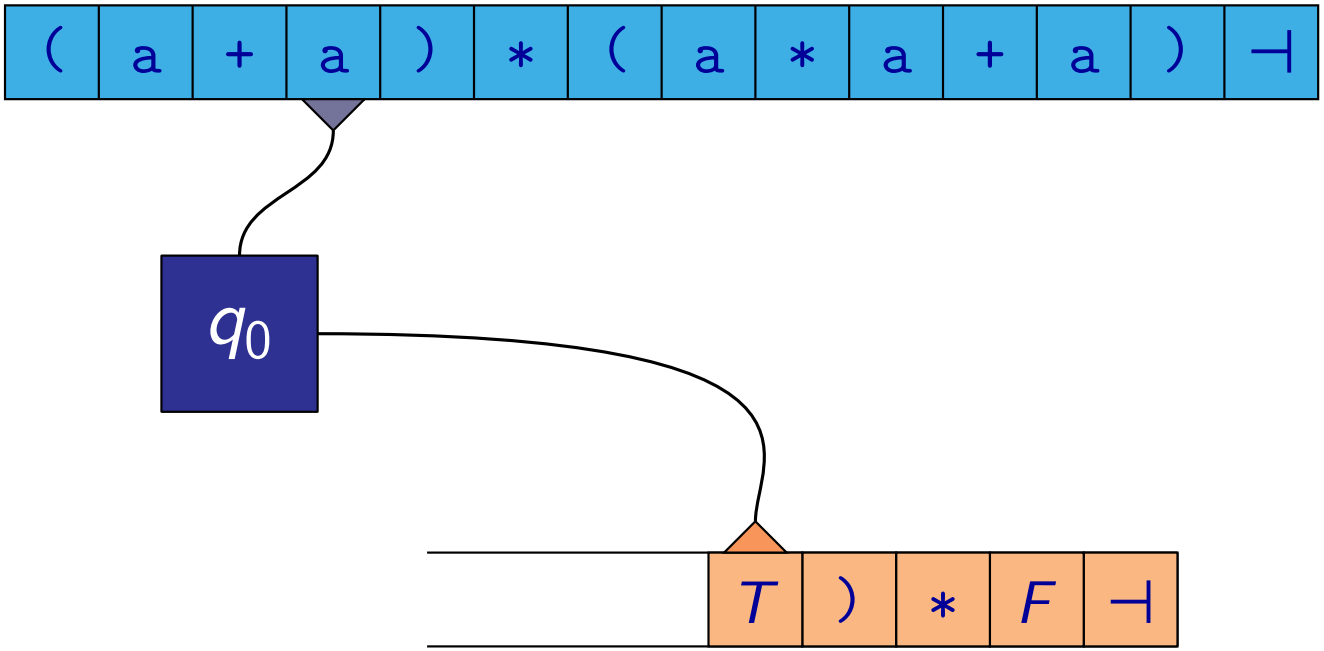
$$\cdots \Rightarrow (\underline{E}+T)*F\dashv \Rightarrow (\underline{T}+T)*F\dashv \Rightarrow (\underline{F}+T)*F\dashv \Rightarrow (\mathtt{a}+\underline{T})*F\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



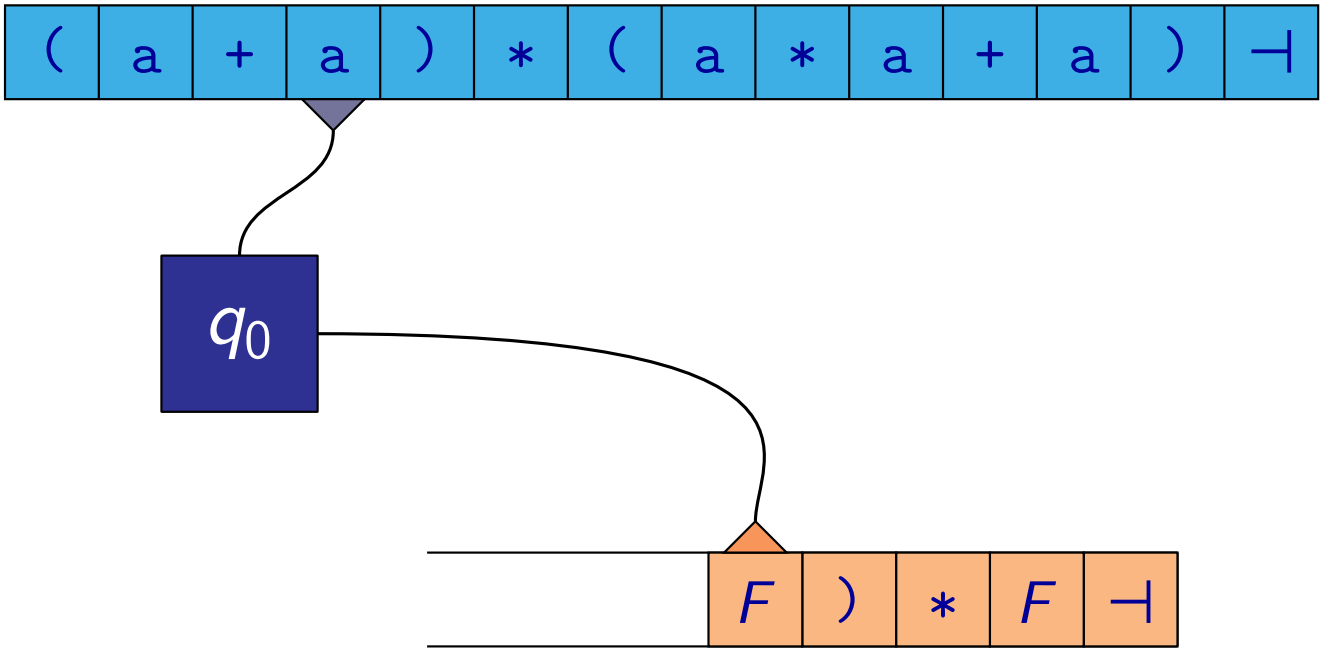
$$\cdots \Rightarrow (\underline{E}+T)*F\dashv \Rightarrow (\underline{T}+T)*F\dashv \Rightarrow (\underline{F}+T)*F\dashv \Rightarrow (\mathtt{a}+\underline{T})*F\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



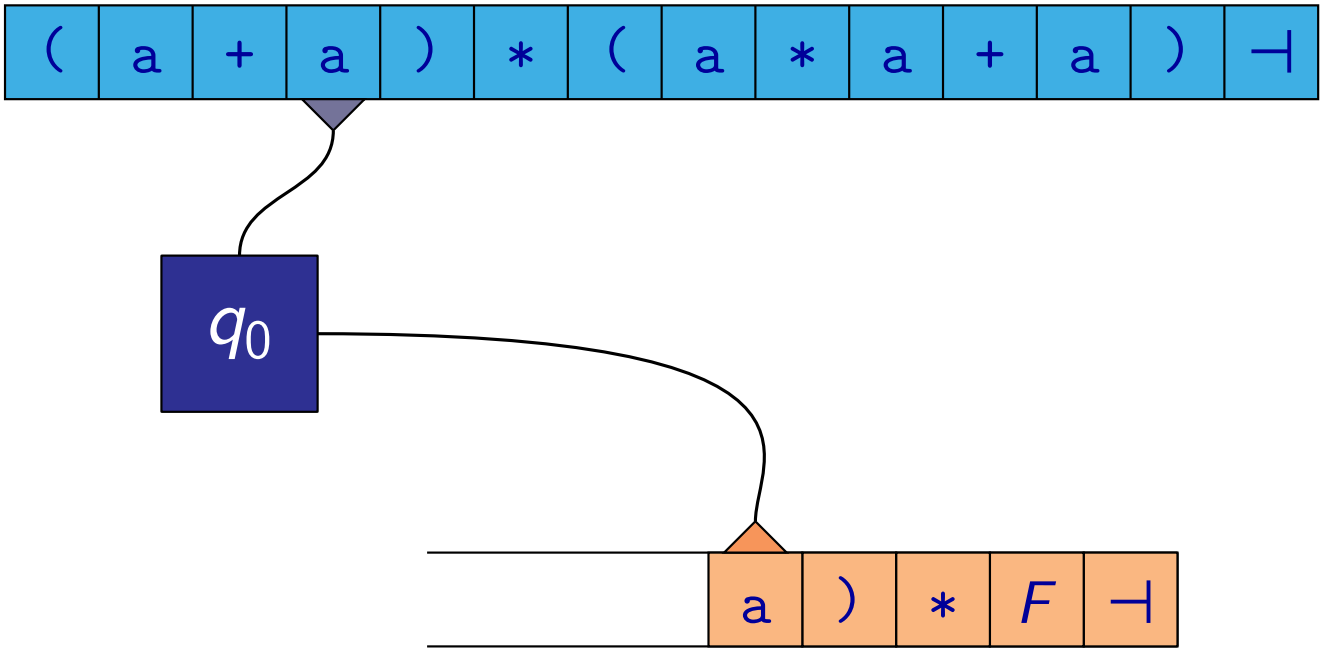
$$\cdots \Rightarrow (\underline{E}+T)*F\dashv \Rightarrow (\underline{T}+T)*F\dashv \Rightarrow (\underline{F}+T)*F\dashv \Rightarrow (\mathtt{a}+\underline{T})*F\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



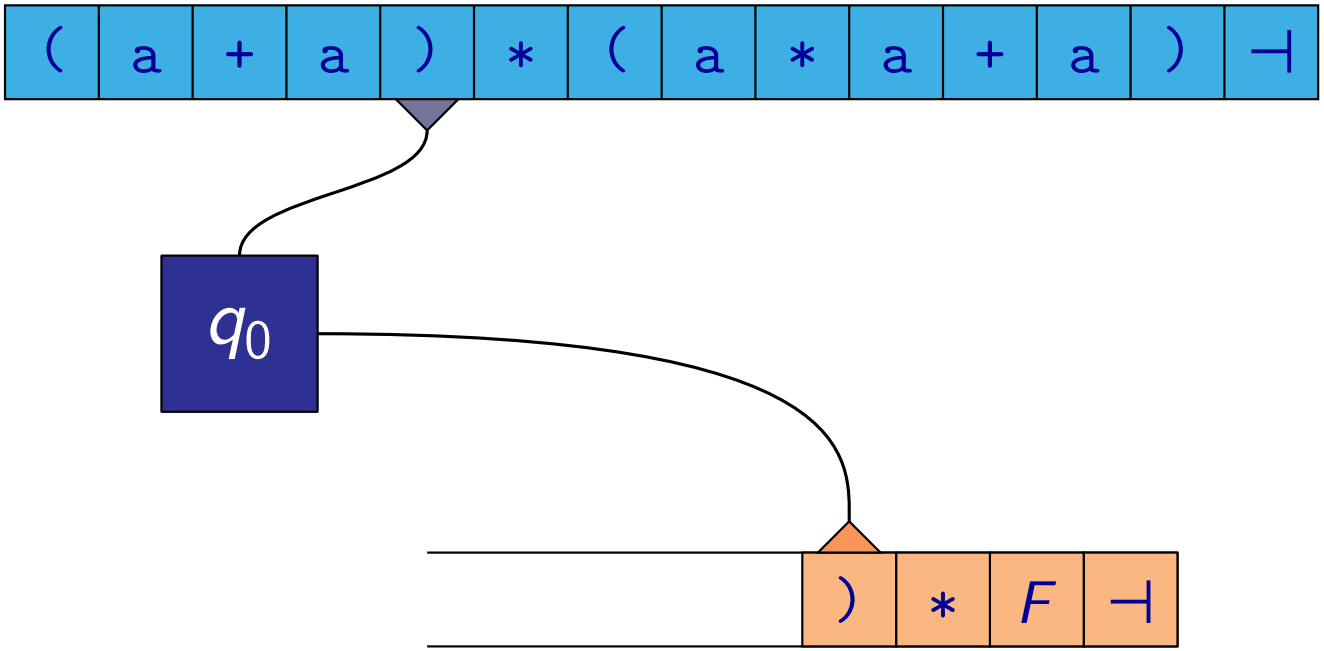
$$\cdots \Rightarrow (\underline{T}+T)*F\dashv \Rightarrow (\underline{F}+T)*F\dashv \Rightarrow (a+\underline{T})*F\dashv \Rightarrow (a+\underline{F})*F\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



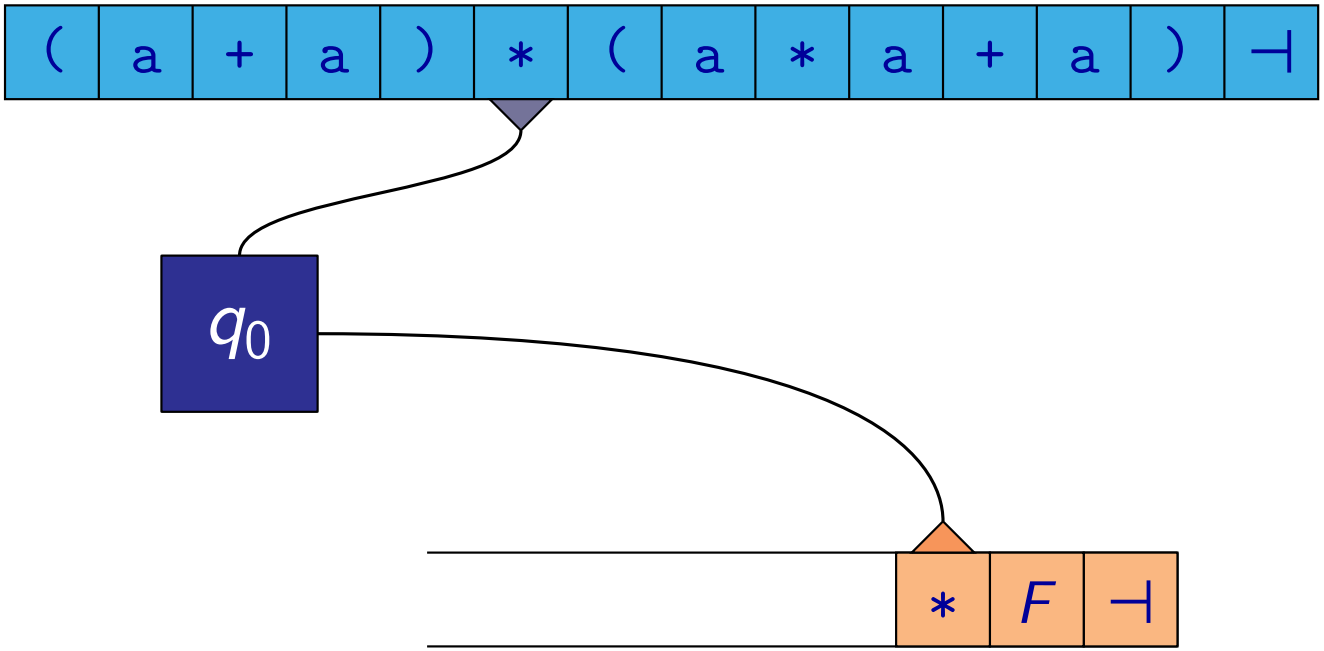
$$\cdots \Rightarrow (\underline{F}+T)*F\dashv \Rightarrow (\mathtt{a}+\underline{T})*F\dashv \Rightarrow (\mathtt{a}+\underline{F})*F\dashv \Rightarrow (\mathtt{a}+\mathtt{a})*\underline{F}\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



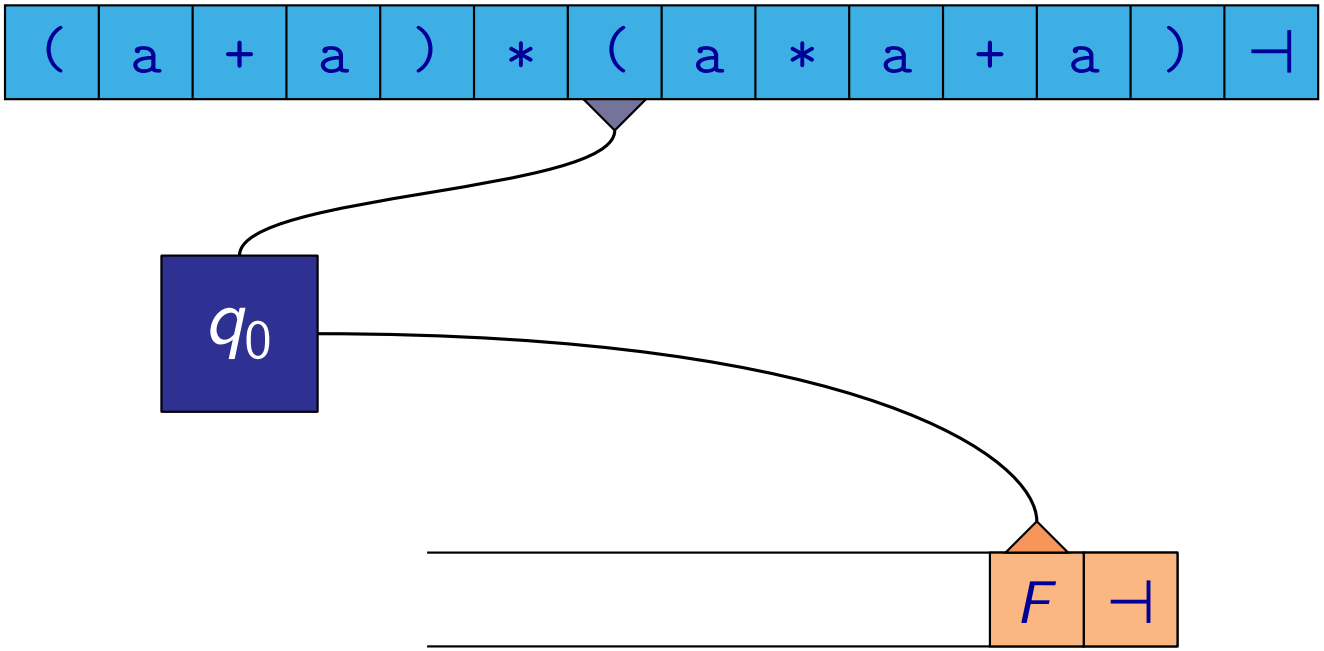
$$\cdots \Rightarrow (\underline{F}+T)*F\dashv \Rightarrow (\mathtt{a}+\underline{T})*F\dashv \Rightarrow (\mathtt{a}+\underline{F})*F\dashv \Rightarrow (\mathtt{a}+\mathtt{a})*\underline{F}\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



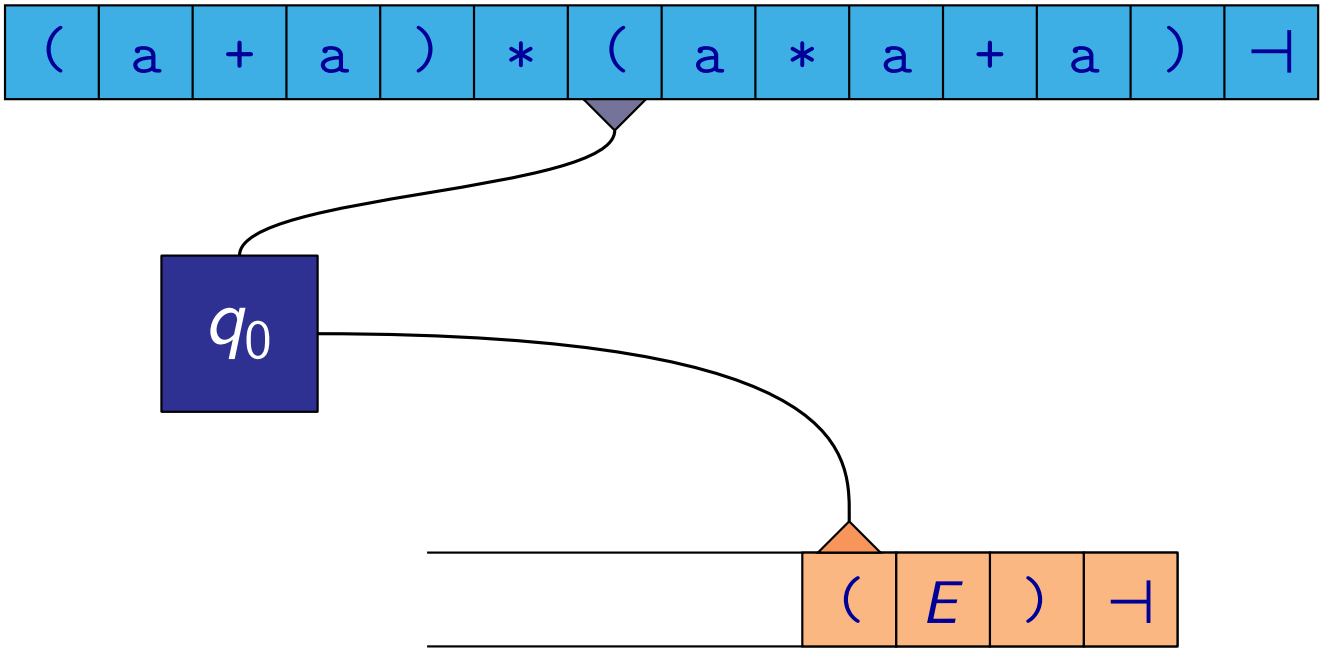
$$\cdots \Rightarrow (\underline{F}+T)*F\dashv \Rightarrow (\mathtt{a}+\underline{T})*F\dashv \Rightarrow (\mathtt{a}+\underline{F})*F\dashv \Rightarrow (\mathtt{a}+\mathtt{a})*\underline{F}\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



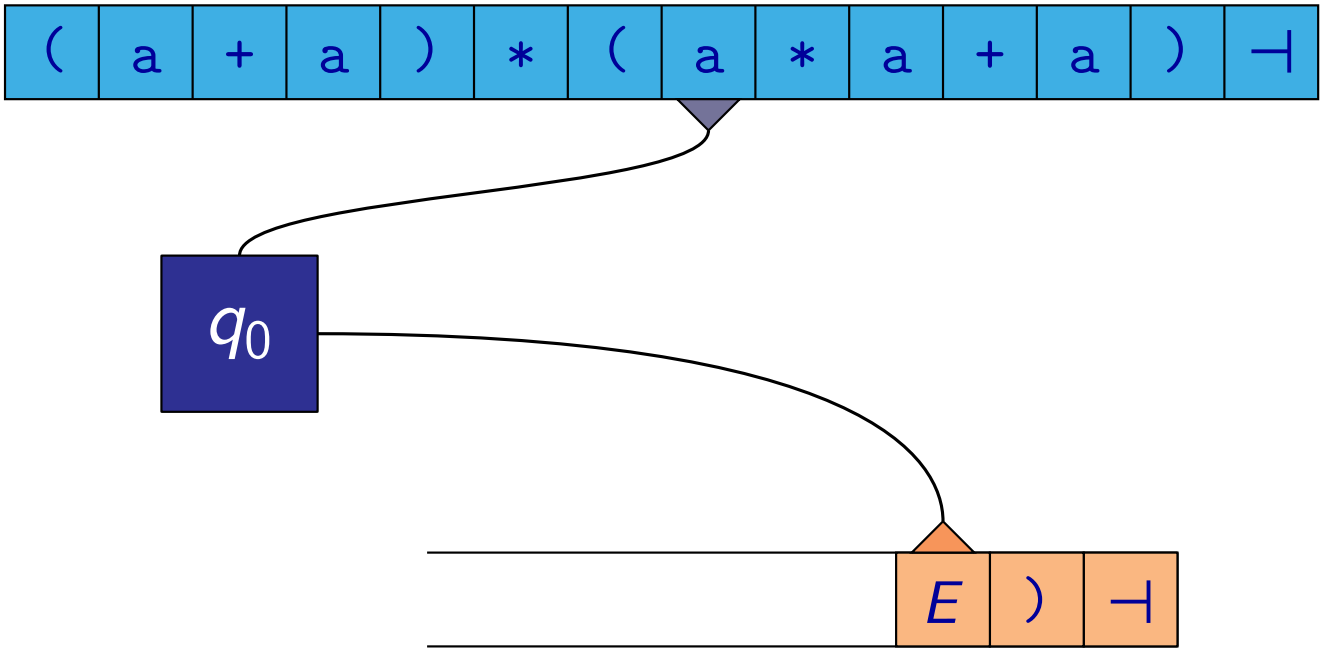
$$\cdots \Rightarrow (\underline{F}+T)*F\dashv \Rightarrow (\mathrm{a}+\underline{T})*F\dashv \Rightarrow (\mathrm{a}+\underline{F})*F\dashv \Rightarrow (\mathrm{a}+\mathrm{a})*\underline{F}\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



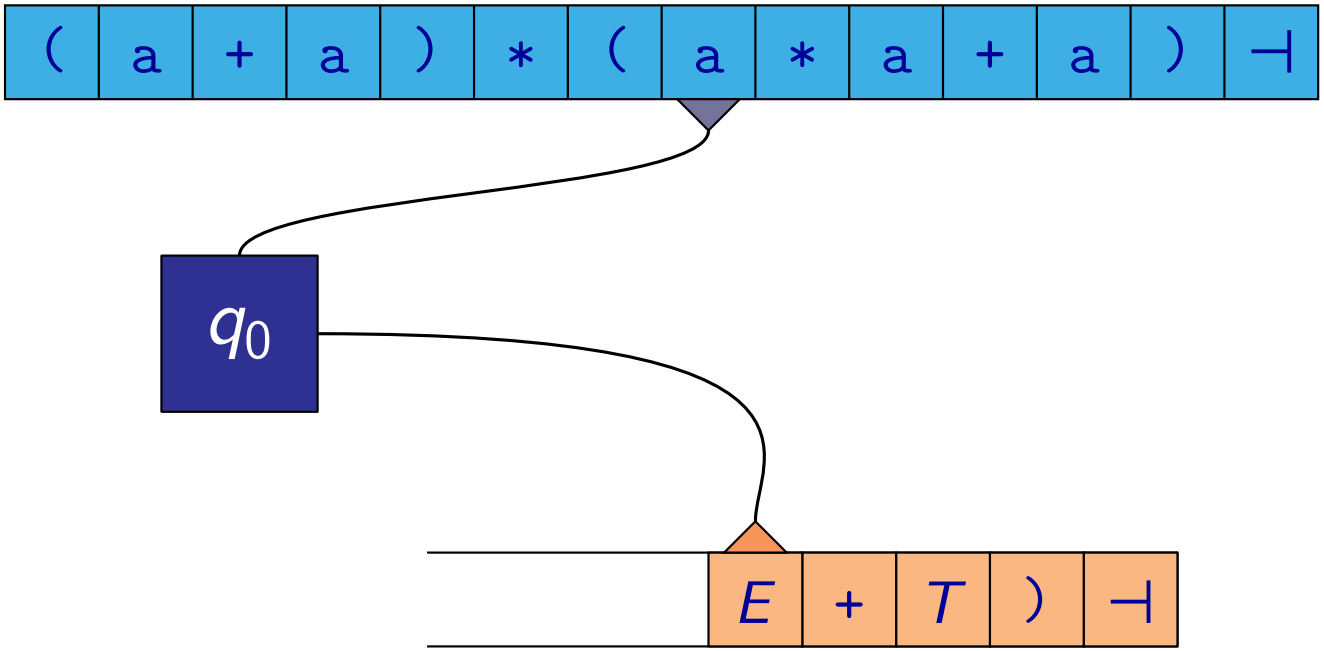
$$\cdots \Rightarrow (\mathtt{a}+\underline{T})*F\dashv \Rightarrow (\mathtt{a}+\underline{F})*F\dashv \Rightarrow (\mathtt{a}+\mathtt{a})*\underline{F}\dashv \Rightarrow (\mathtt{a}+\mathtt{a})*(\underline{E})\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



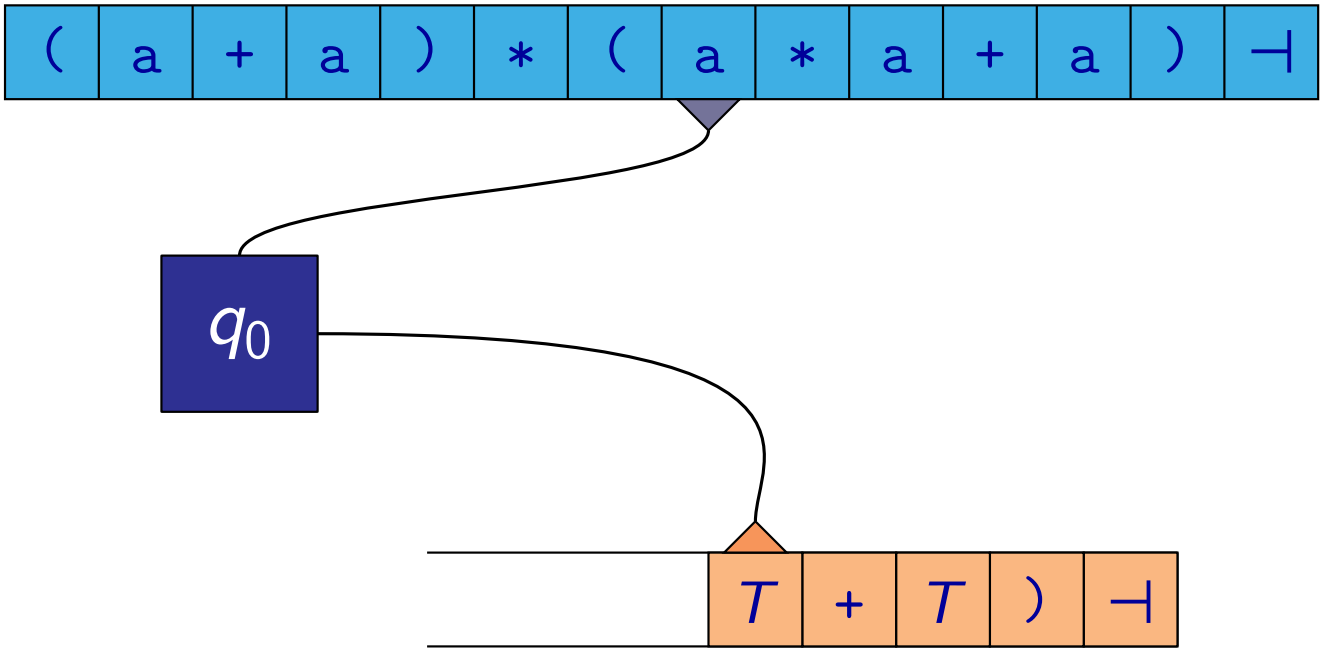
$$\cdots \Rightarrow (\mathtt{a}+\underline{T})*F\dashv \Rightarrow (\mathtt{a}+\underline{F})*F\dashv \Rightarrow (\mathtt{a}+\mathtt{a})*\underline{F}\dashv \Rightarrow (\mathtt{a}+\mathtt{a})*(\underline{E})\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



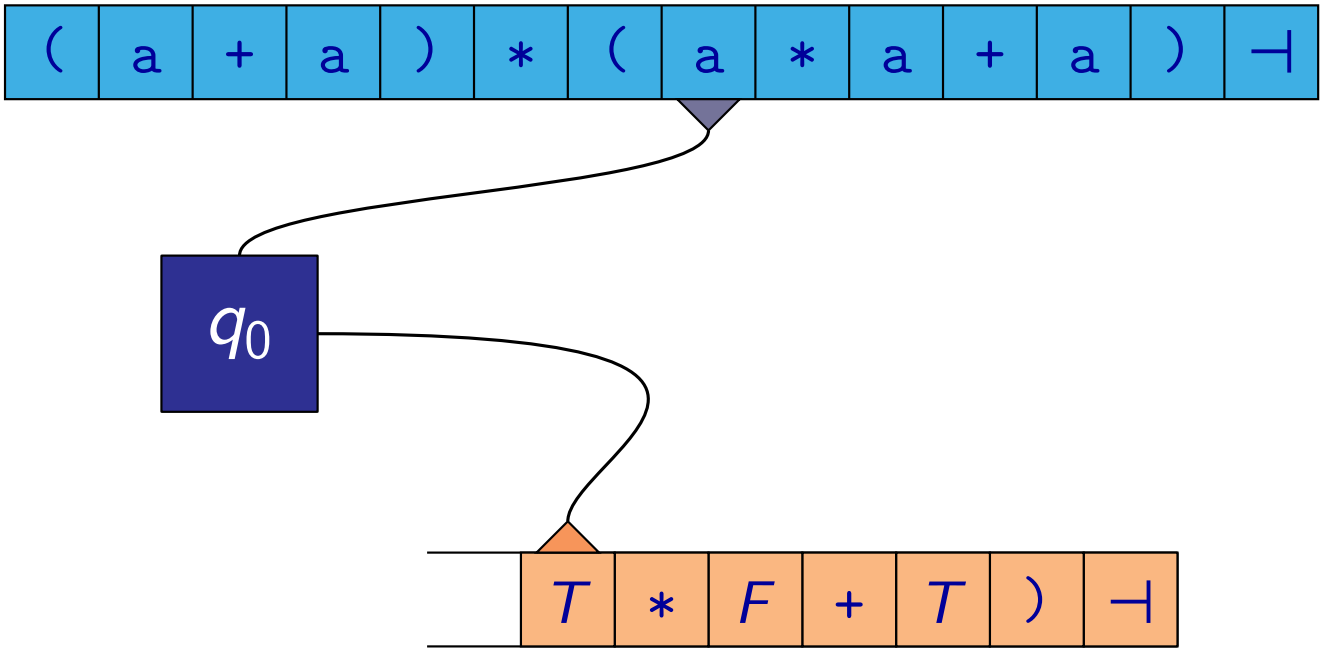
$$\cdots \Rightarrow (a+a)*\underline{F}\dashv \Rightarrow (a+a)*(\underline{E})\dashv \Rightarrow (a+a)*(\underline{E}+T)\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



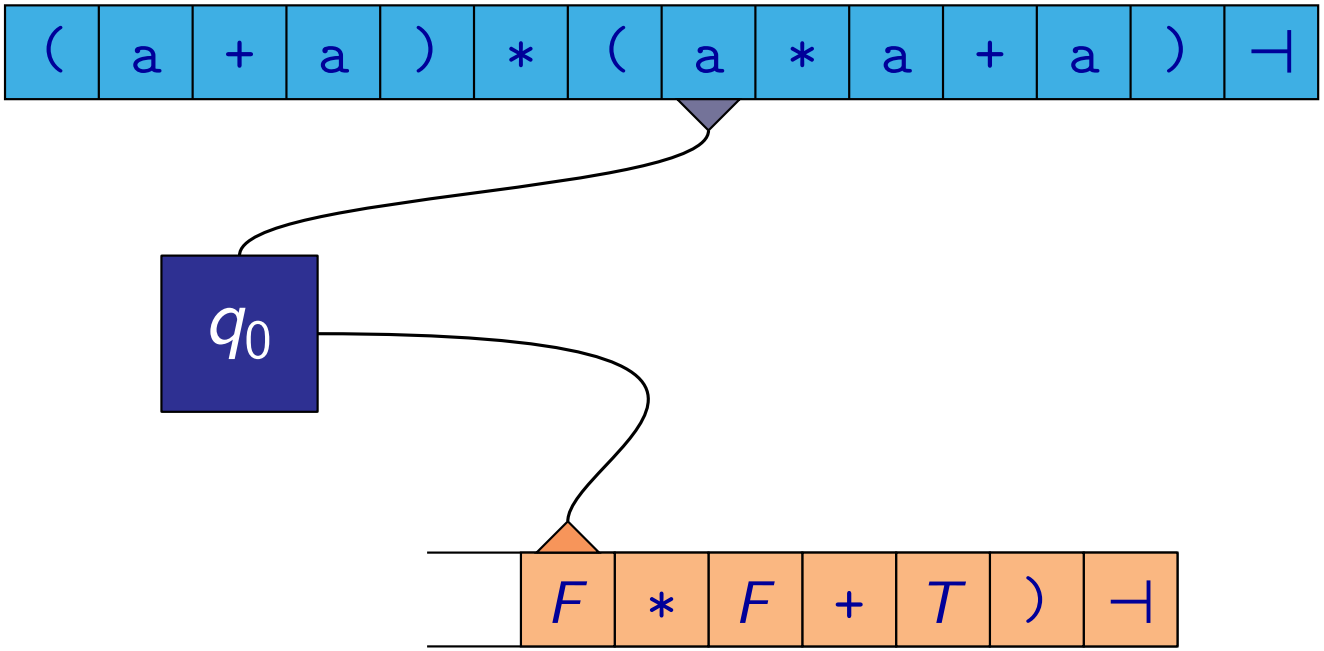
$$\cdots \Rightarrow \texttt{(a+a)*(}\underline{E}\texttt{)}\dashv \Rightarrow \texttt{(a+a)*(}\underline{E}\texttt{+}T\texttt{)}\dashv \Rightarrow \texttt{(a+a)*(}\underline{T}\texttt{+}T\texttt{)}\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



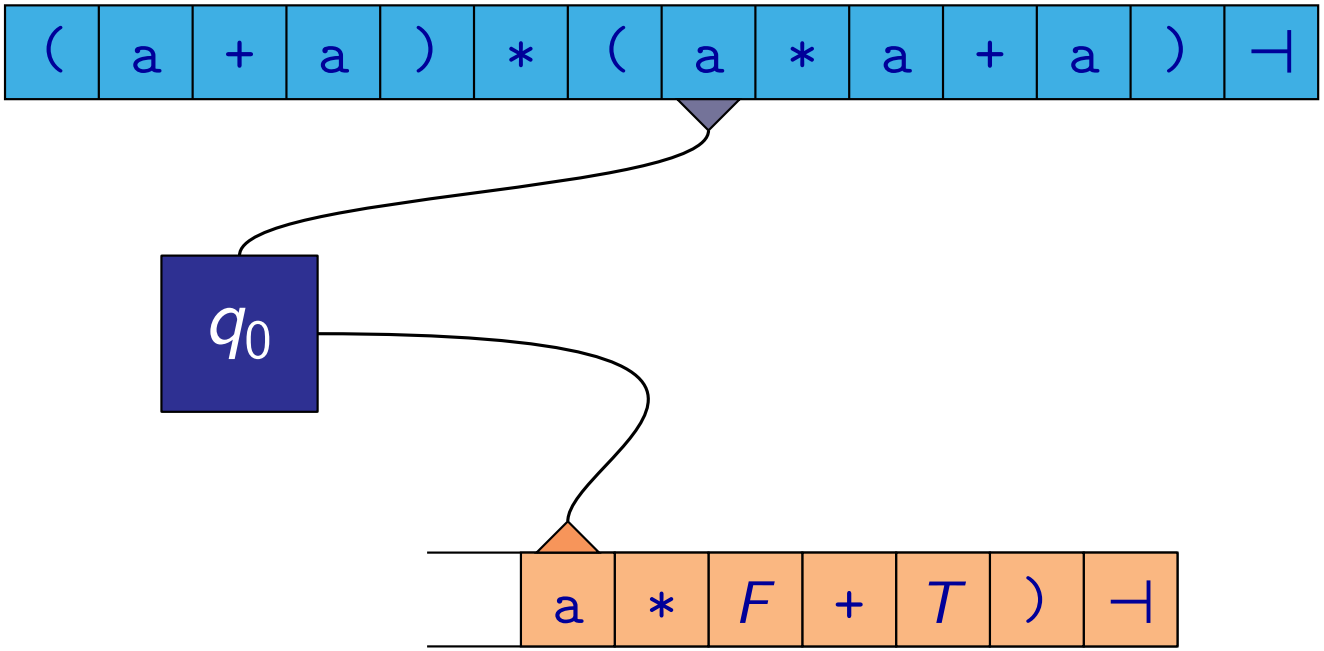
$\cdots \Rightarrow$ (a+a)*($\underline{E}$+T)$\dashv \Rightarrow$ (a+a)*($\underline{T}$+T)$\dashv \Rightarrow$ (a+a)*($\underline{T}$*F+T)$\dashv$

Ekvivalence bezkontextových gramatik a zás. automatů



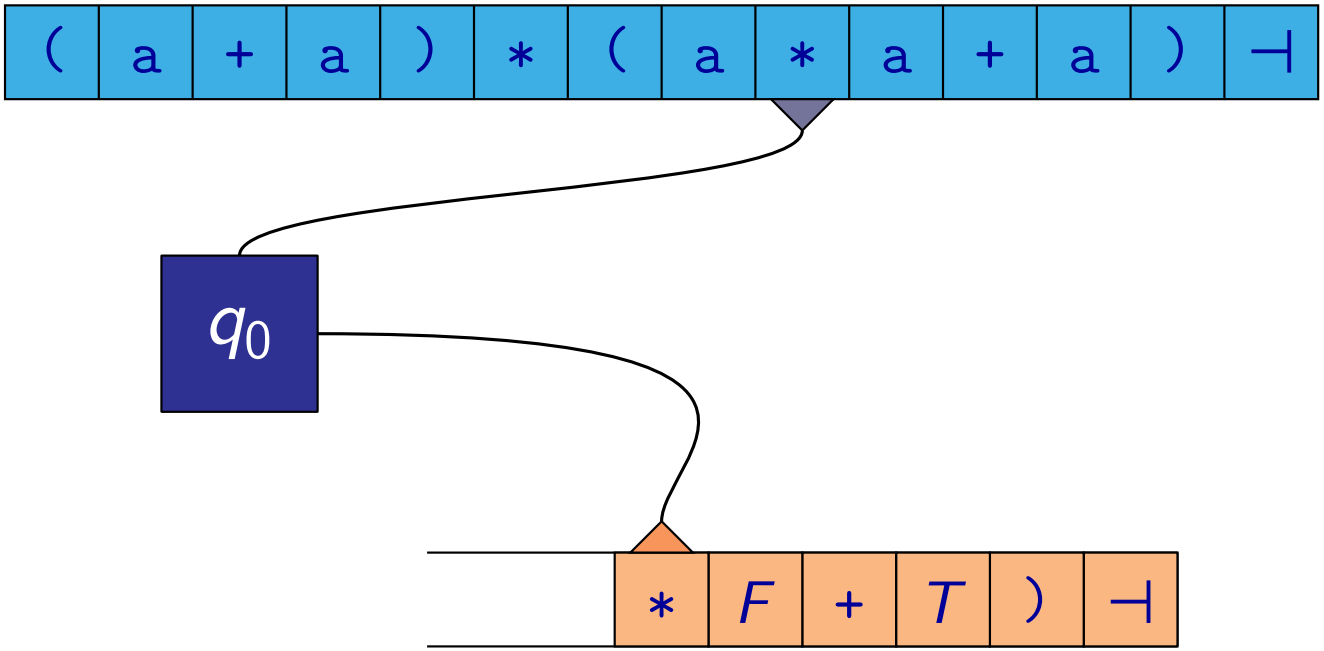
$$\cdots \Rightarrow (\mathtt{a}+\mathtt{a})*(\underline{T}*F+T)\dashv \Rightarrow (\mathtt{a}+\mathtt{a})*(\underline{F}*F+T)\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



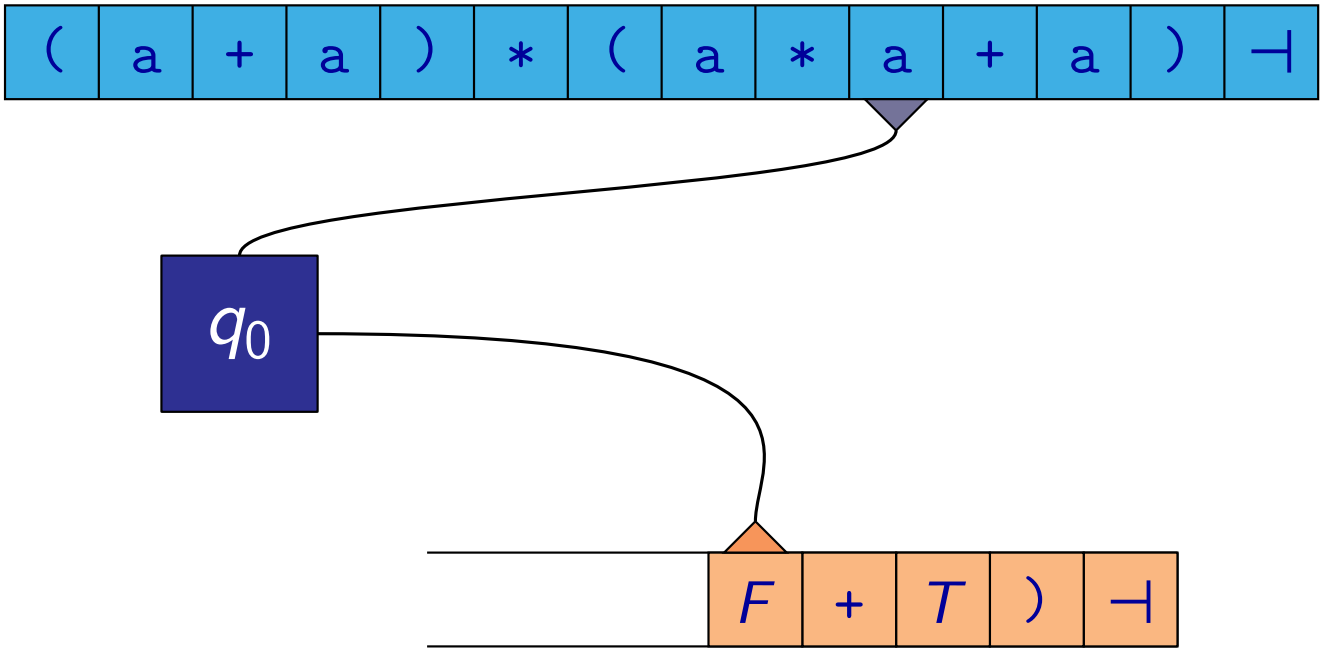
$$\cdots \Rightarrow (\mathtt{a}+\mathtt{a})*(\underline{F}*F+T)\dashv \Rightarrow (\mathtt{a}+\mathtt{a})*(\mathtt{a}*\underline{F}+T)\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



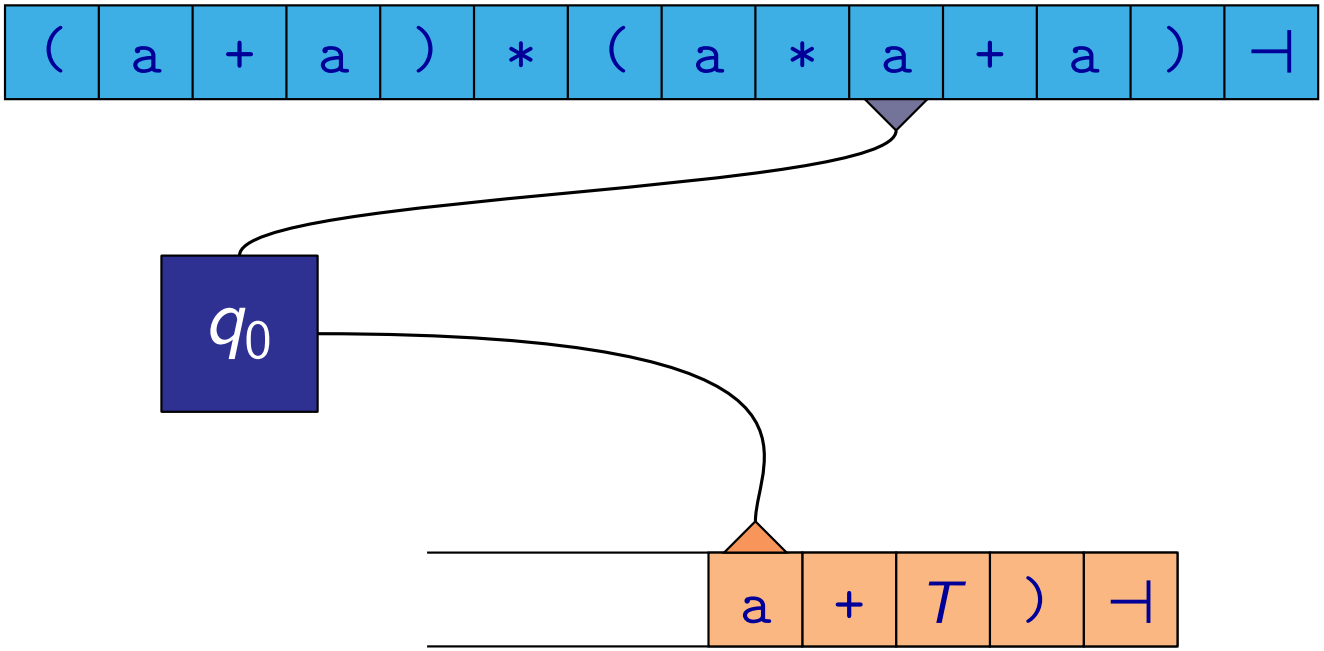
$$\cdots \Rightarrow \texttt{(a+a)*(}\underline{F}\texttt{*}F\texttt{+}T\texttt{)}\dashv \Rightarrow \texttt{(a+a)*(a*}\underline{F}\texttt{+}T\texttt{)}\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



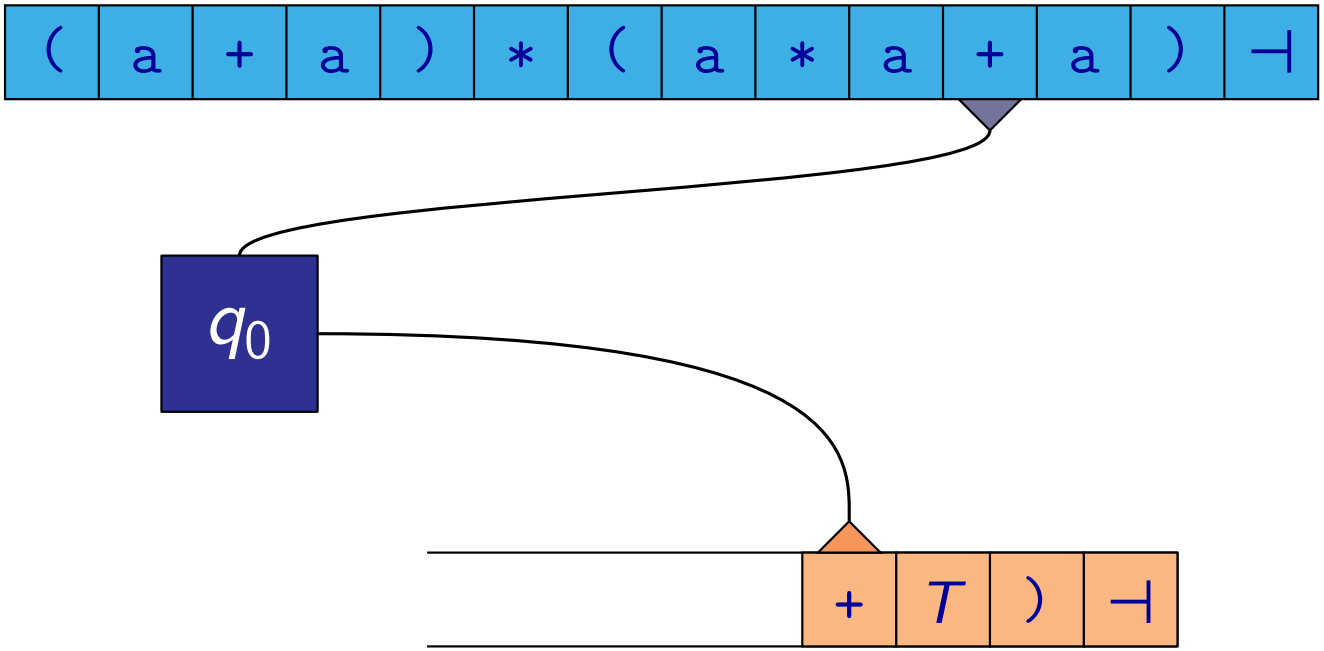
$$\cdots \Rightarrow (a+a)*(\underline{F}*F+T)\dashv \Rightarrow (a+a)*(a*\underline{F}+T)\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



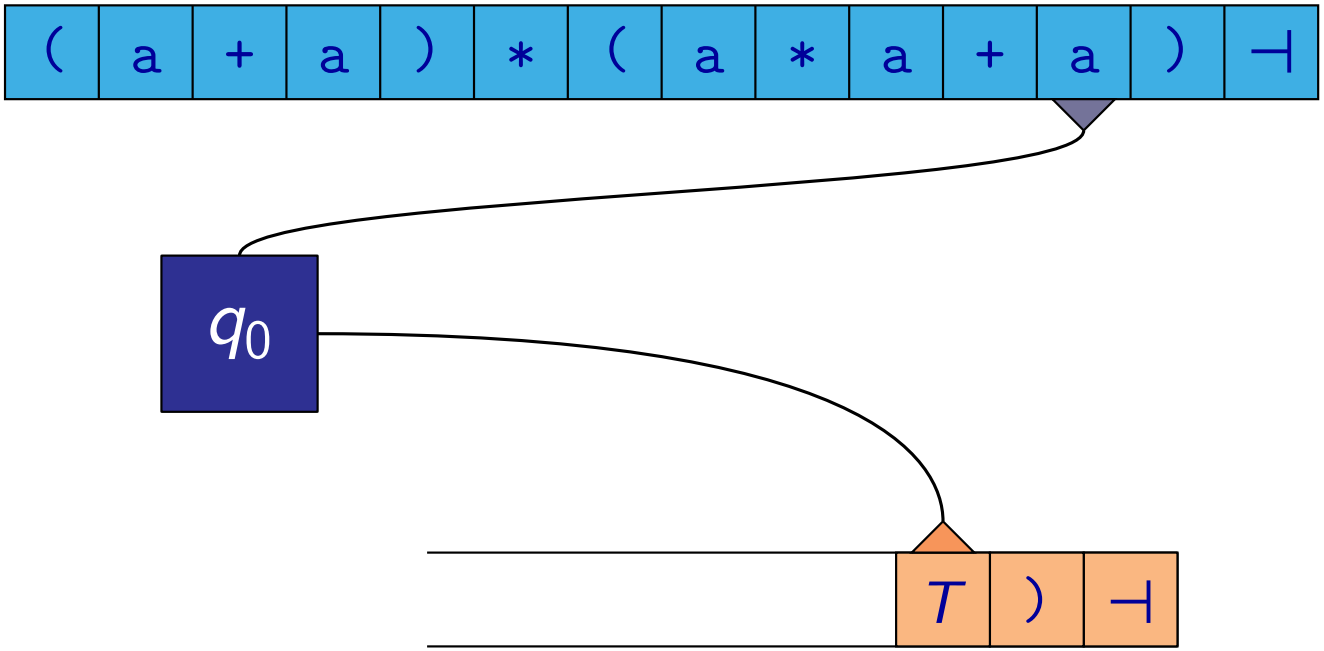
$$\cdots \Rightarrow (a+a)*(a*\underline{F}+T)\dashv \Rightarrow (a+a)*(a*a+\underline{T})\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



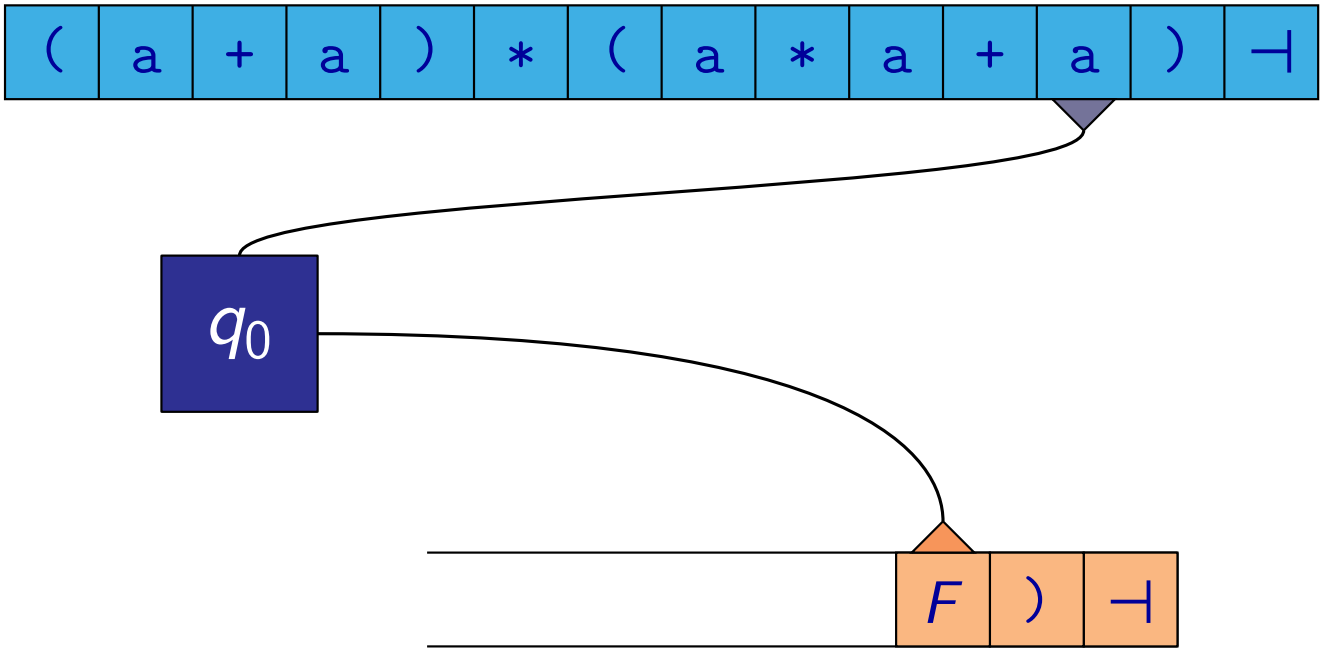
$$\cdots \Rightarrow (a+a)*(a*\underline{F}+T)\dashv \Rightarrow (a+a)*(a*a+\underline{T})\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



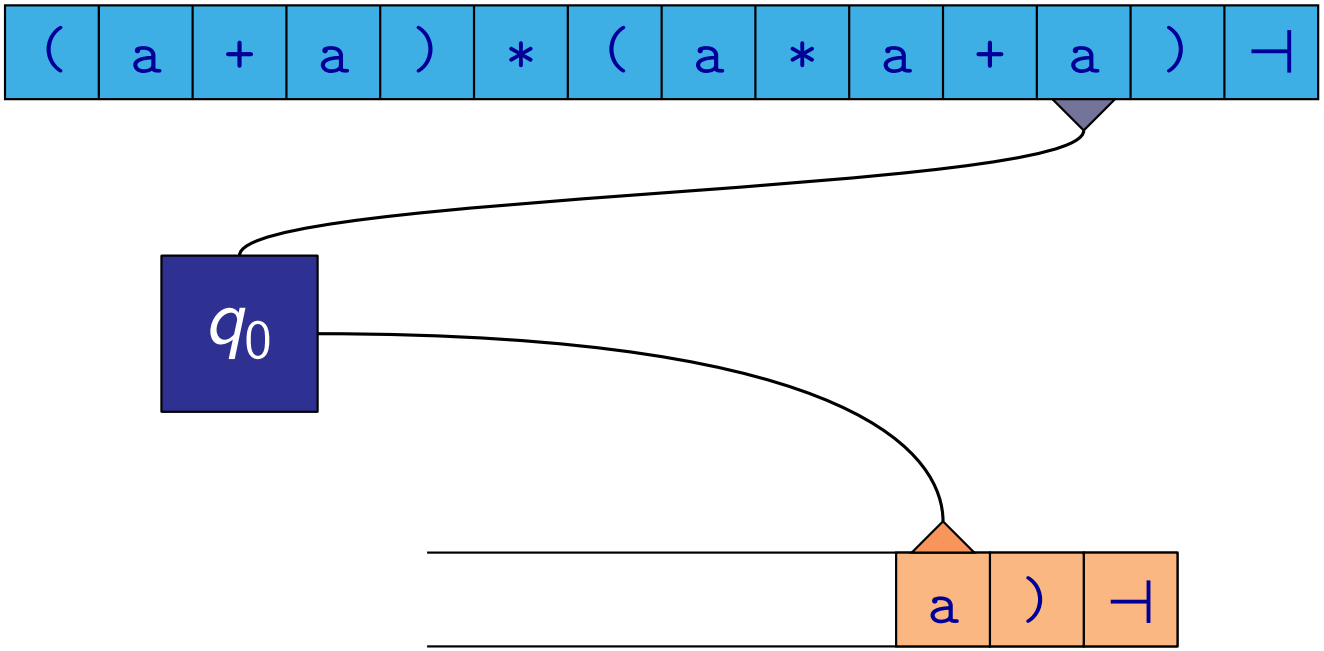
$$\cdots \Rightarrow (\mathtt{a}+\mathtt{a})*(\mathtt{a}*\underline{F}+T)\dashv \Rightarrow (\mathtt{a}+\mathtt{a})*(\mathtt{a}*\mathtt{a}+\underline{T})\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



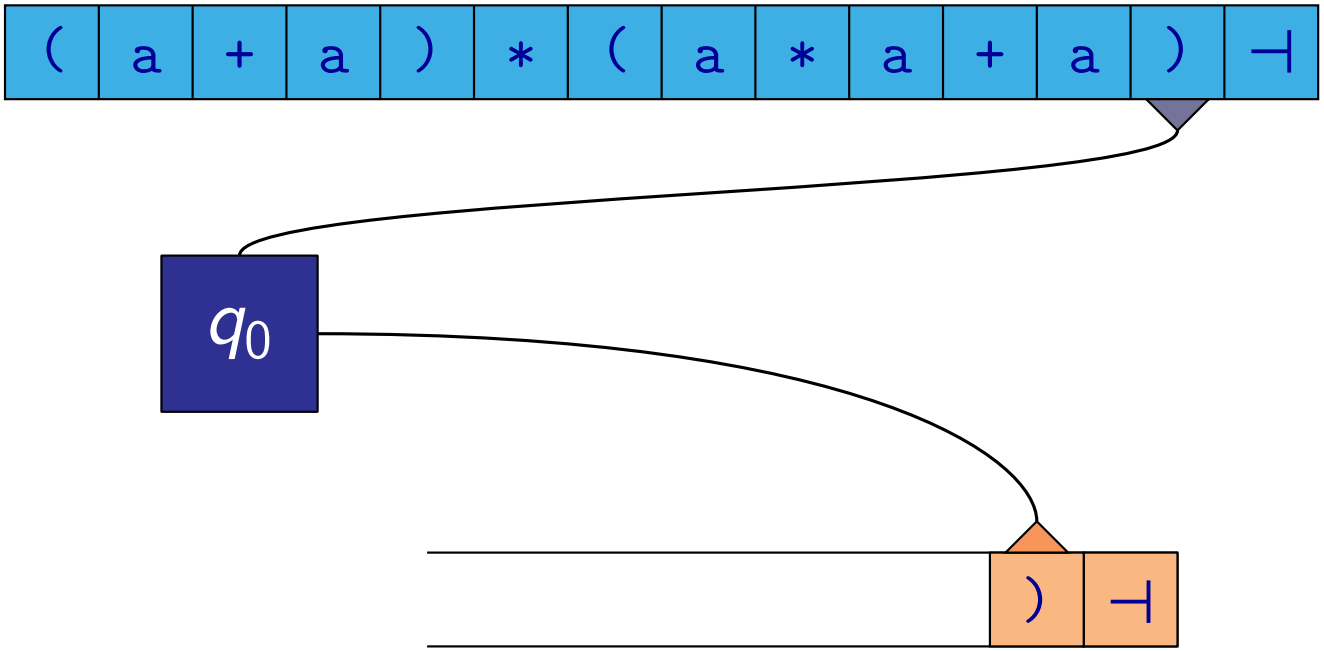
$$\cdots \Rightarrow \texttt{(a+a)*(a*a+}\underline{T}\texttt{)}\dashv \Rightarrow \texttt{(a+a)*(a*a+}\underline{F}\texttt{)}\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



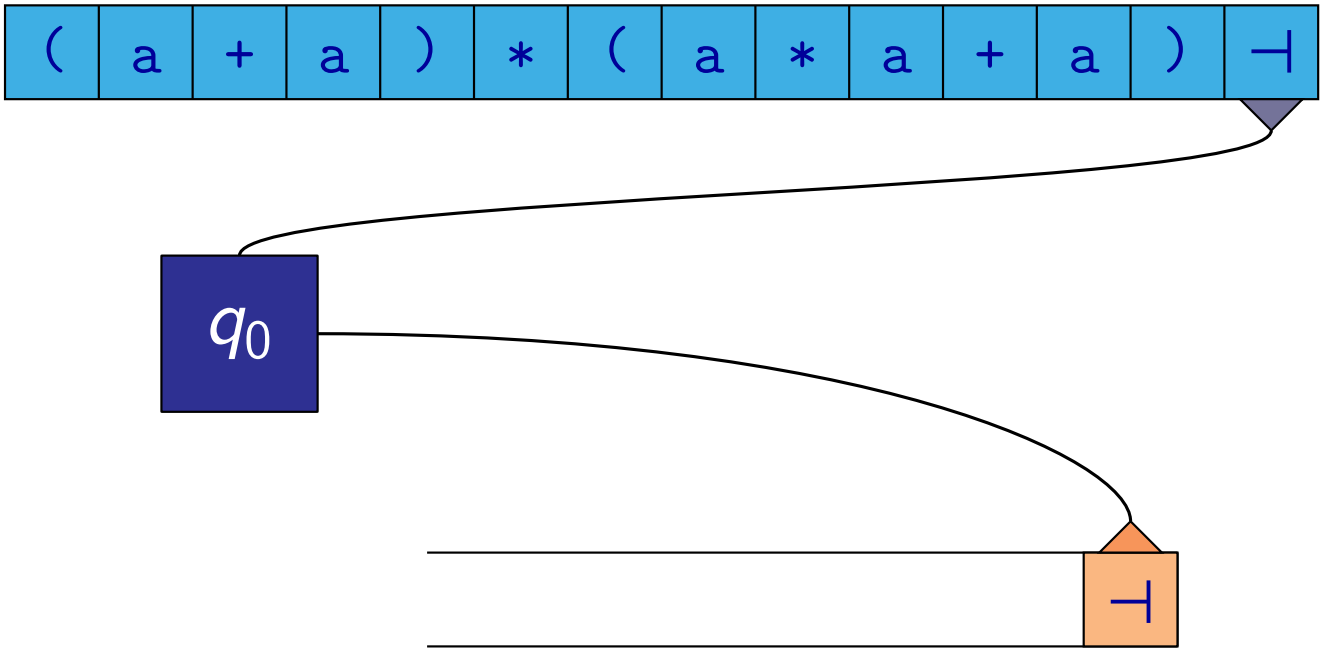
$$\cdots \Rightarrow (a+a)*(a*a+\underline{F})\dashv \Rightarrow (a+a)*(a*a+a)\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



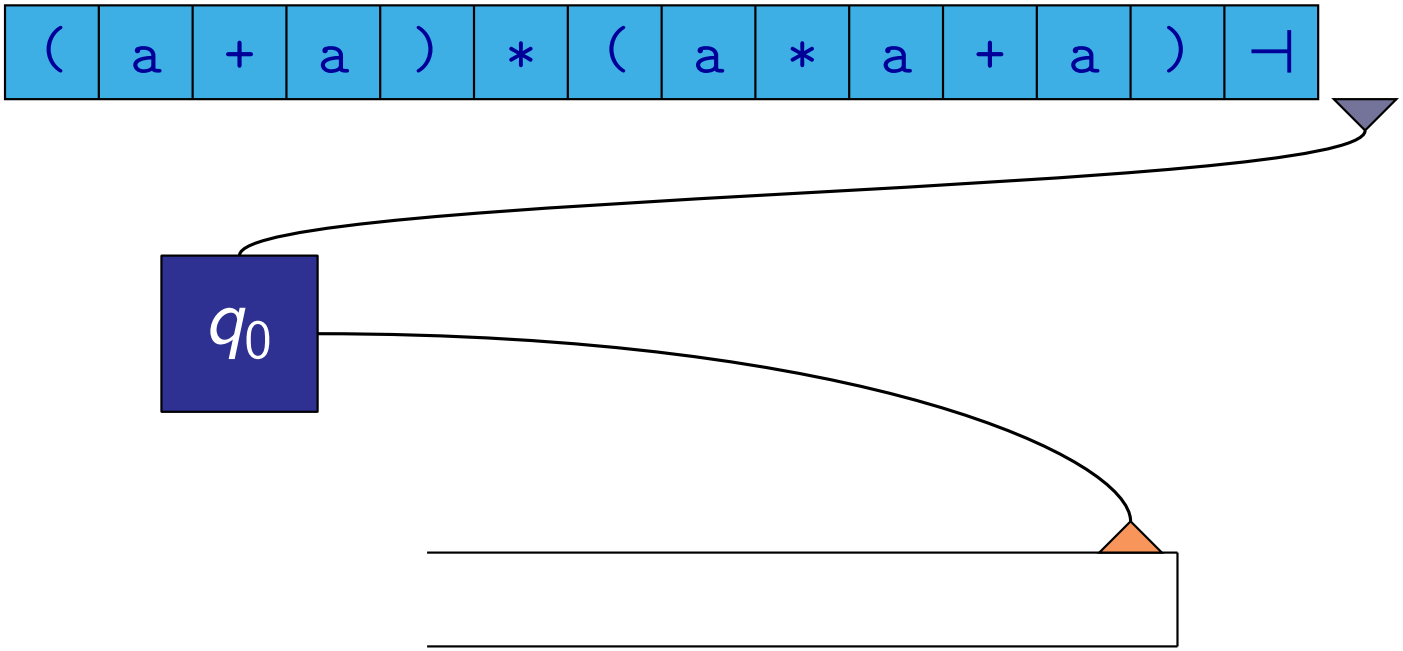
$$\cdots \Rightarrow (a+a)*(a*a+\underline{F})\dashv \Rightarrow (a+a)*(a*a+a)\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



$$\cdots \Rightarrow (a+a)*(a*a+\underline{F})\dashv \Rightarrow (a+a)*(a*a+a)\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů



$$\cdots \Rightarrow (a+a)*(a*a+\underline{F})\dashv \Rightarrow (a+a)*(a*a+a)\dashv$$

Ekvivalence bezkontextových gramatik a zás. automatů

Z předchozího příkladu je vidět, že zásobníkový automat $\mathcal{M}$ během výpočtu v zásadě provádí **levou derivaci** v gramatice $\mathcal{G}$.

Snadno se ukáže, že:

- Každé levé derivaci v gramatice $\mathcal{G}$ odpovídá nějaký výpočet automatu $\mathcal{M}$.
- Každému výpočtu automatu $\mathcal{M}$ odpovídá nějaká levá derivace v gramatice $\mathcal{G}$.

Poznámka: Výše uvedený postup odpovídá syntaktické analýze **shora dolů**.

Ekvivalence bezkontextových gramatik a zás. automatů

Alternativně lze při syntaktické analýze postupovat též **zdola nahoru**.

Tomu odpovídá následující konstrukce nedeterministického zásobníkového automatu $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, X_0)$ k dané gramatice $\mathcal{G} = (\Pi, \Sigma, S, P)$, kde:

- $\Gamma = \Pi \cup \Sigma \cup \{\vdash\}$, kde $\vdash \notin (\Pi \cup \Sigma)$
- $X_0 = \vdash$
- Q obsahuje stavy odpovídající všem sufixům pravých stran pravidel z P a dále speciální stav $\langle S \rangle$ (kde $S \in \Pi$ je počáteční neterminál gramatiky $\mathcal{G}$) a speciální stav q_{acc}.

  Stav odpovídající suffixu α (kde $\alpha \in (\Pi \cup \Sigma)^*$) budeme označovat zápisem $\langle \alpha \rangle$.

  Speciálním případem je stav odpovídající sufixu ε. Tento stav budeme označovat $\langle\rangle$.
- $q_0 = \langle\rangle$

Ekvivalence bezkontextových gramatik a zás. automatů

- Pro každý vstupní symbol $a \in \Sigma$ a každý zásobníkový symbol $W \in \Gamma$ přidáme do δ následující pravidlo:

$$\langle\rangle W \xrightarrow{a} \langle\rangle aW$$

- Pro každé pravidlo $X \to Y_1 Y_2 \cdots Y_n$ z gramatiky $\mathcal{G}$ (kde $X \in \Pi$, $n \geq 0$ a $Y_i \in (\Pi \cup \Sigma)$ pro $1 \leq i \leq n$) přidáme do přechodové funkce δ automatu $\mathcal{M}$ následující sadu pravidel:

$$\begin{aligned}
\langle\rangle Y_n &\xrightarrow{\varepsilon} \langle Y_n\rangle \\
\langle Y_n\rangle Y_{n-1} &\xrightarrow{\varepsilon} \langle Y_{n-1} Y_n\rangle \\
\langle Y_{n-1} Y_n\rangle Y_{n-2} &\xrightarrow{\varepsilon} \langle Y_{n-2} Y_{n-1} Y_n\rangle \\
&\vdots \\
\langle Y_2 Y_3 \ldots Y_n\rangle Y_1 &\xrightarrow{\varepsilon} \langle Y_1 Y_2 Y_3 \cdots Y_n\rangle
\end{aligned}$$

a dále pro každé $W \in \Gamma$ pravidla

$$\langle Y_1 Y_2 \cdots Y_n\rangle W \xrightarrow{\varepsilon} \langle\rangle XW$$

Ekvivalence bezkontextových gramatik a zás. automatů

- Pokud například bude gramatika $\mathcal{G}$ obsahovat pravidlo

$$B \rightarrow CaADb$$

bude přechodová funkce δ automatu $\mathcal{M}$ obsahovat pravidla

$$\langle\rangle b \xrightarrow{\varepsilon} \langle b\rangle$$
$$\langle b\rangle D \xrightarrow{\varepsilon} \langle Db\rangle$$
$$\langle Db\rangle A \xrightarrow{\varepsilon} \langle ADb\rangle$$
$$\langle ADb\rangle a \xrightarrow{\varepsilon} \langle aADb\rangle$$
$$\langle aADb\rangle C \xrightarrow{\varepsilon} \langle CaADb\rangle$$

a dále pro každé $W \in \Gamma$ pravidlo

$$\langle CaADb\rangle W \xrightarrow{\varepsilon} \langle\rangle BW$$

Ekvivalence bezkontextových gramatik a zás. automatů

- Speciálně pro ε-pravidla z gramatiky $\mathcal{G}$ budou přidaná pravidla vypadat následovně: ε-pravidlu

$$X \to \varepsilon$$

z gramatiky $\mathcal{G}$, kde $X \in \Pi$, budou odpovídat pravidla v δ tvaru

$$\langle\rangle W \xrightarrow{\varepsilon} \langle\rangle XW$$

kde $W \in \Gamma$.

- Nakonec přidáme do δ dvě speciální pravidla (kde $S \in \Pi$ je počáteční neterminál gramatiky $\mathcal{G}$):

$$\langle\rangle S \xrightarrow{\varepsilon} \langle S\rangle \qquad\qquad \langle S\rangle \vdash \xrightarrow{\varepsilon} q_{acc}$$

Ekvivalence bezkontextových gramatik a zás. automatů

Příklad: Vezměme si opět stejnou gramatiku $\mathcal{G}$ jako v předchozím příkladě:

$$\begin{aligned} S &\to E \dashv \\ E &\to T \mid E{+}T \\ T &\to F \mid T{*}F \\ F &\to \mathtt{a} \mid (E) \end{aligned}$$

K ní sestrojíme zásobníkový automat $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, X_0)$, kde

- $\Sigma = \{ \mathtt{a}, +, *, (,), \dashv \}$
- $\Gamma = \{ S, E, T, F, \mathtt{a}, +, *, (,), \dashv, \vdash \}$
- $Q = \{ \langle\rangle, \langle\dashv\rangle, \langle E \dashv\rangle, \langle T\rangle, \langle{+}T\rangle, \langle E{+}T\rangle, \langle F\rangle, \langle{*}F\rangle, \langle T{*}F\rangle,$
  $\langle\mathtt{a}\rangle, \langle)\rangle, \langle E)\rangle, \langle(E)\rangle, \langle S\rangle, q_{acc} \}$
- $q_0 = \langle\rangle$
- $X_0 = \vdash$

Ekvivalence bezkontextových gramatik a zás. automatů

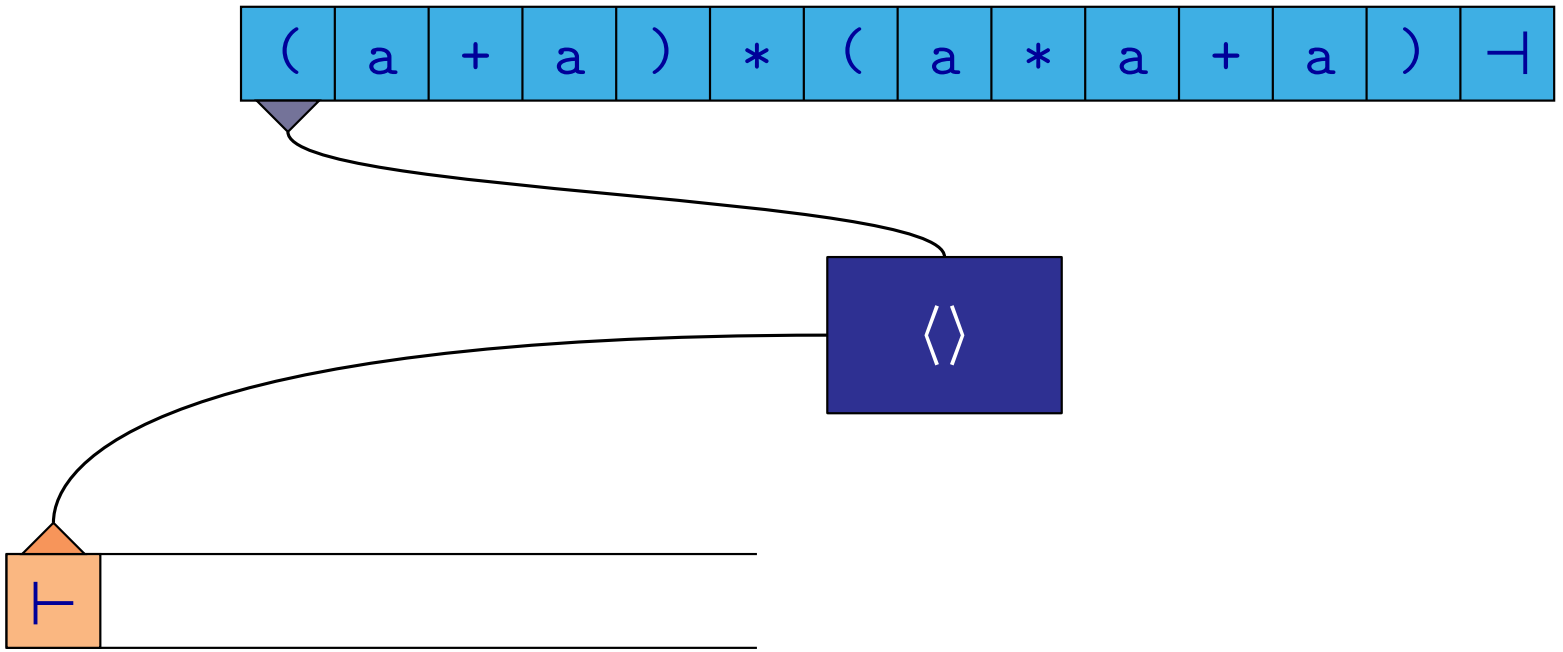
Pro každé $X \in \Gamma$ přidáme do δ následující pravidla:

$$\langle\rangle X \xrightarrow{\mathrm{a}} \langle\rangle \mathrm{a} X$$
$$\langle\rangle X \xrightarrow{+} \langle\rangle + X$$
$$\langle\rangle X \xrightarrow{*} \langle\rangle * X$$
$$\langle\rangle X \xrightarrow{(} \langle\rangle (X$$
$$\langle\rangle X \xrightarrow{)} \langle\rangle) X$$
$$\langle\rangle X \xrightarrow{\dashv} \langle\rangle \dashv X$$

$$\langle\rangle S \xrightarrow{\varepsilon} \langle S \rangle$$
$$\langle S \rangle \vdash \xrightarrow{\varepsilon} q_{acc}$$

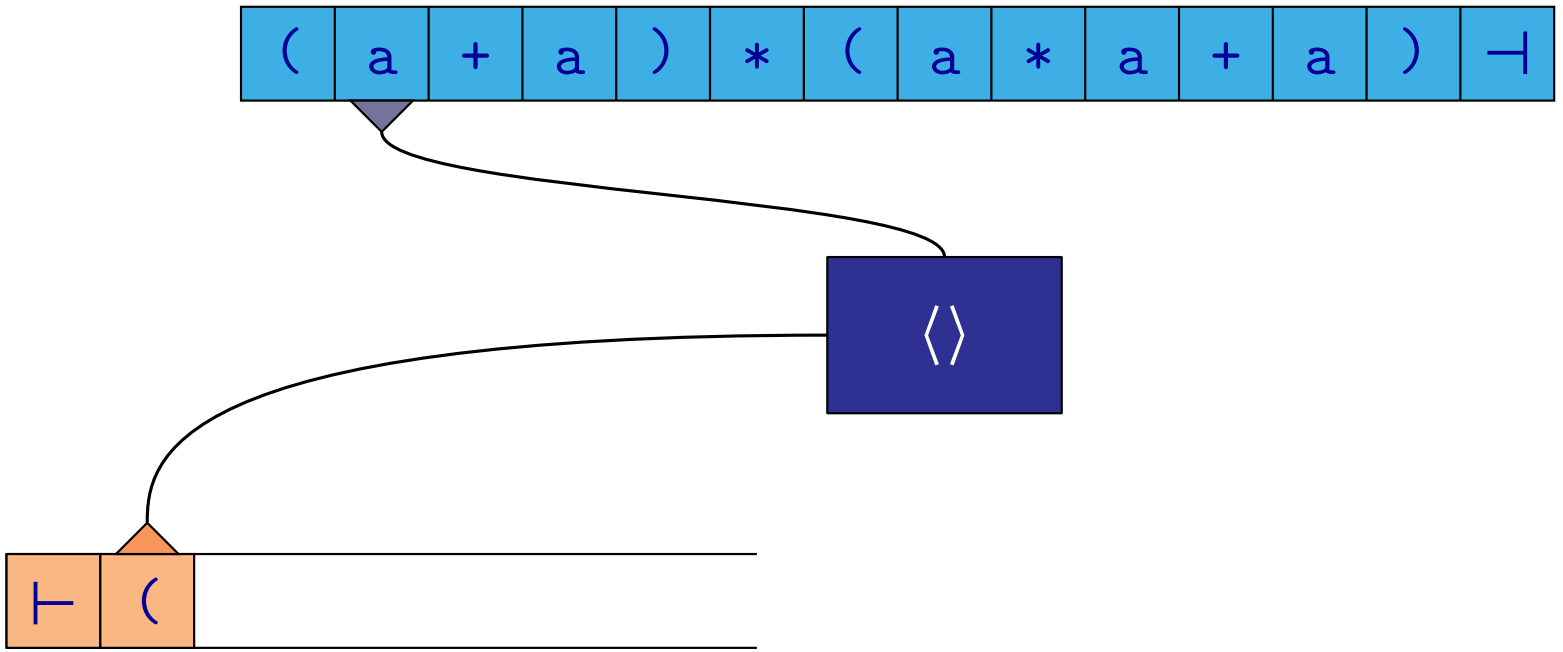
$$\langle\rangle \dashv \xrightarrow{\varepsilon} \langle \dashv \rangle$$
$$\langle \dashv \rangle E \xrightarrow{\varepsilon} \langle E \dashv \rangle \qquad \langle E \dashv \rangle X \xrightarrow{\varepsilon} \langle\rangle S X$$
$$\langle\rangle T \xrightarrow{\varepsilon} \langle T \rangle \qquad \langle T \rangle X \xrightarrow{\varepsilon} \langle\rangle E X$$
$$\langle T \rangle + \xrightarrow{\varepsilon} \langle + T \rangle$$
$$\langle + T \rangle E \xrightarrow{\varepsilon} \langle E + T \rangle \qquad \langle E + T \rangle X \xrightarrow{\varepsilon} \langle\rangle E X$$
$$\langle\rangle F \xrightarrow{\varepsilon} \langle F \rangle \qquad \langle F \rangle X \xrightarrow{\varepsilon} \langle\rangle T X$$
$$\langle F \rangle * \xrightarrow{\varepsilon} \langle * F \rangle$$
$$\langle * F \rangle T \xrightarrow{\varepsilon} \langle T * F \rangle \qquad \langle T * F \rangle X \xrightarrow{\varepsilon} \langle\rangle T X$$
$$\langle\rangle \mathrm{a} \xrightarrow{\varepsilon} \langle \mathrm{a} \rangle \qquad \langle \mathrm{a} \rangle X \xrightarrow{\varepsilon} \langle\rangle F X$$
$$\langle\rangle) \xrightarrow{\varepsilon} \langle) \rangle$$
$$\langle) \rangle E \xrightarrow{\varepsilon} \langle E) \rangle$$
$$\langle E) \rangle (\xrightarrow{\varepsilon} \langle (E) \rangle \qquad \langle (E) \rangle X \xrightarrow{\varepsilon} \langle\rangle F X$$

Ekvivalence bezkontextových gramatik a zás. automatů



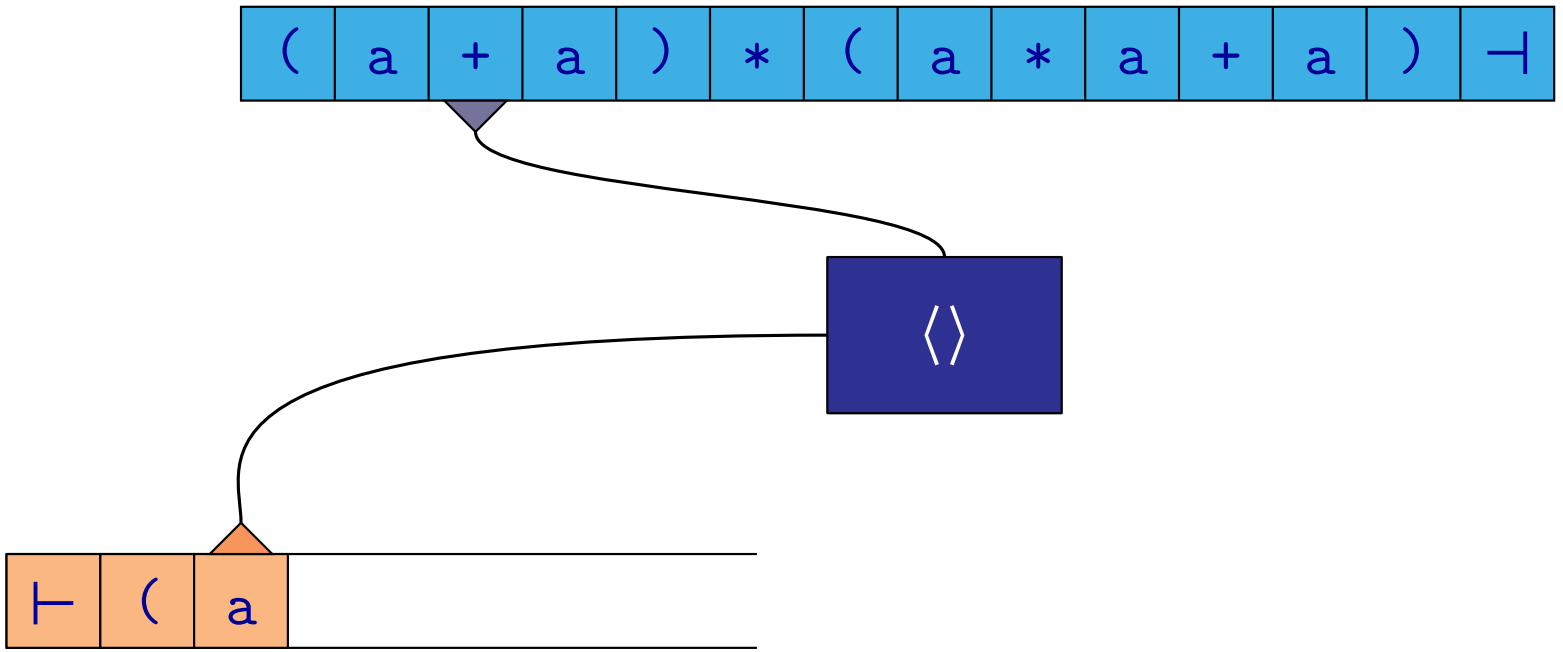
(a+a)*(a*a+a) ⊣

Ekvivalence bezkontextových gramatik a zás. automatů



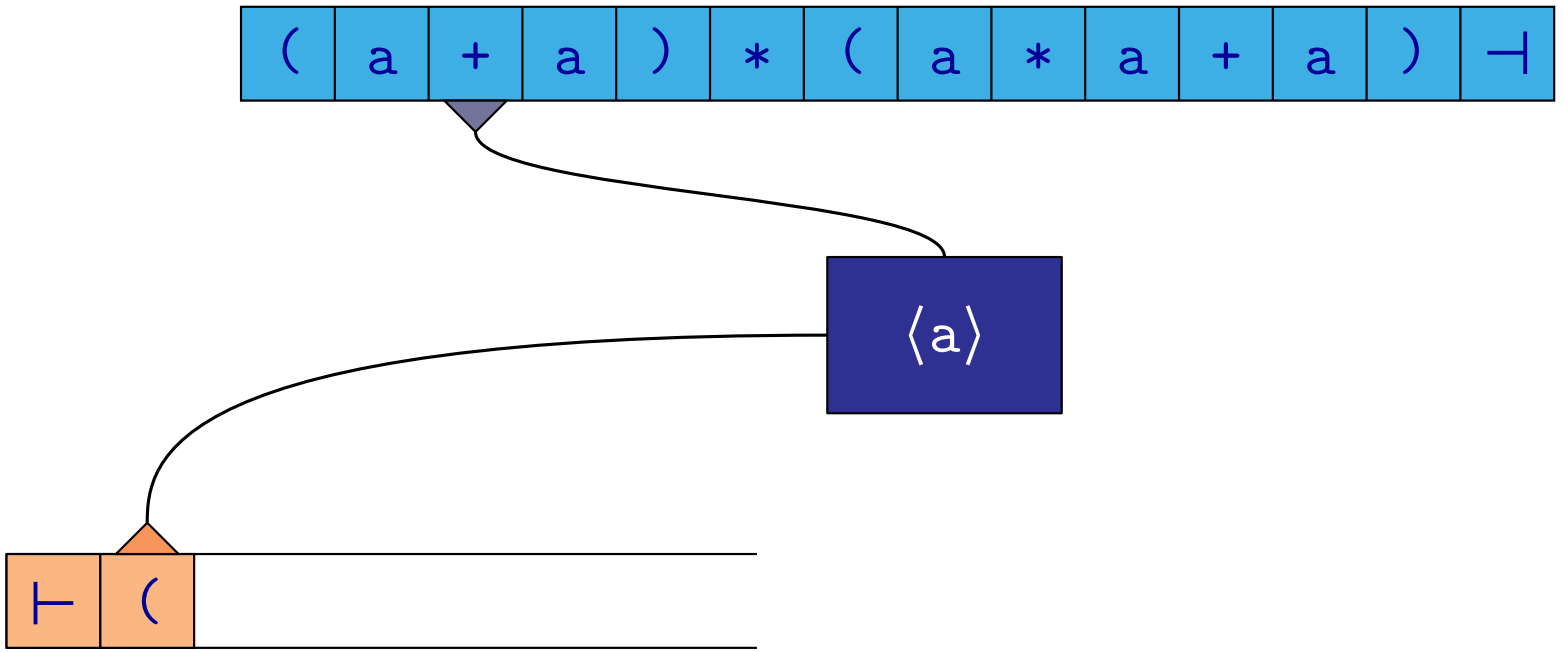
(a+a)*(a*a+a) ⊣

Ekvivalence bezkontextových gramatik a zás. automatů



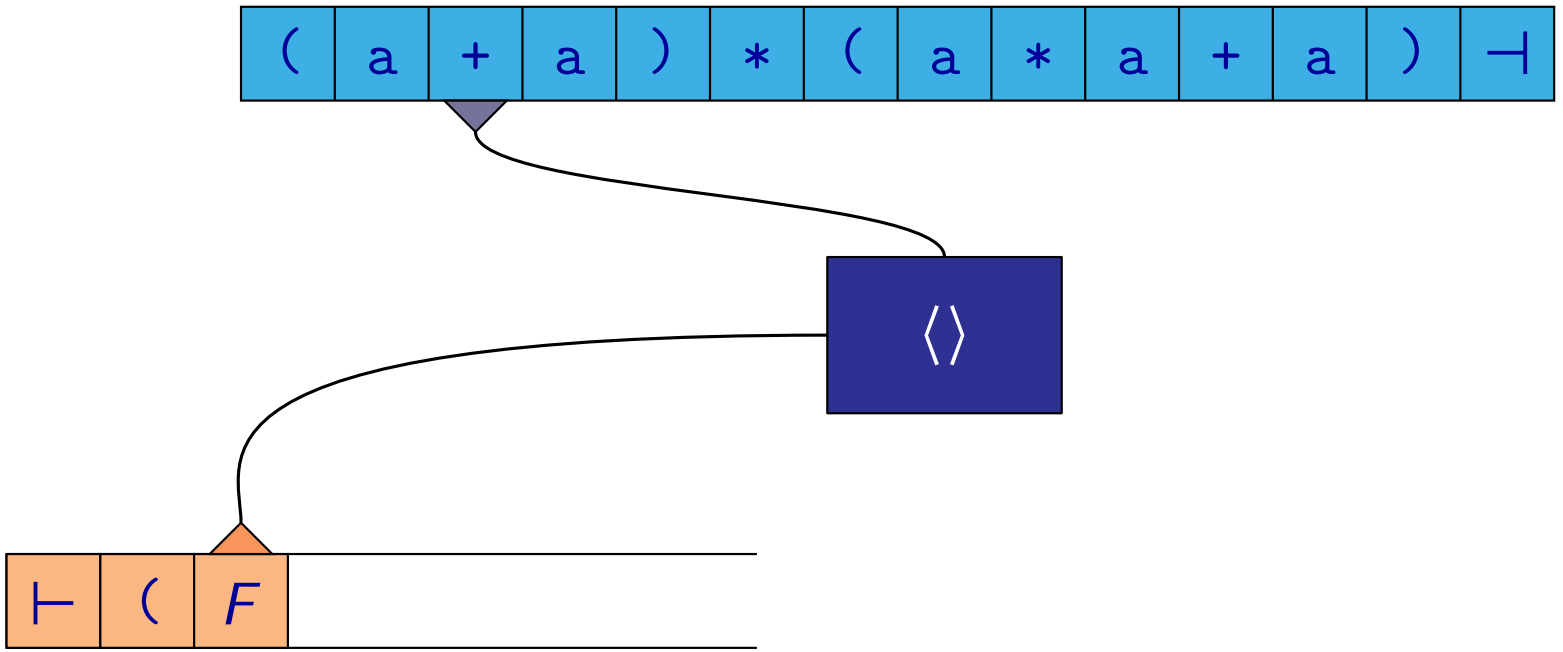
(a+a)*(a*a+a) ⊣

Ekvivalence bezkontextových gramatik a zás. automatů



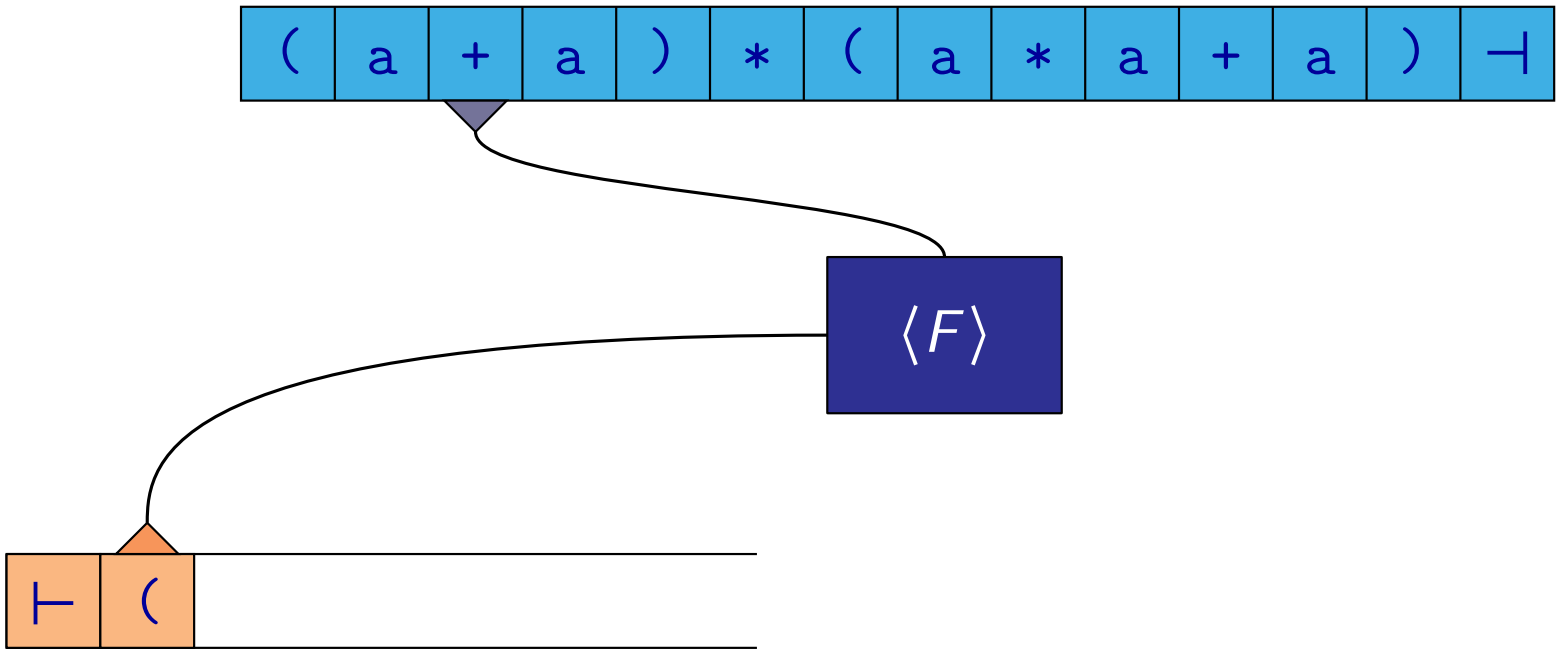
(a+a)*(a*a+a) ⊣

Ekvivalence bezkontextových gramatik a zás. automatů



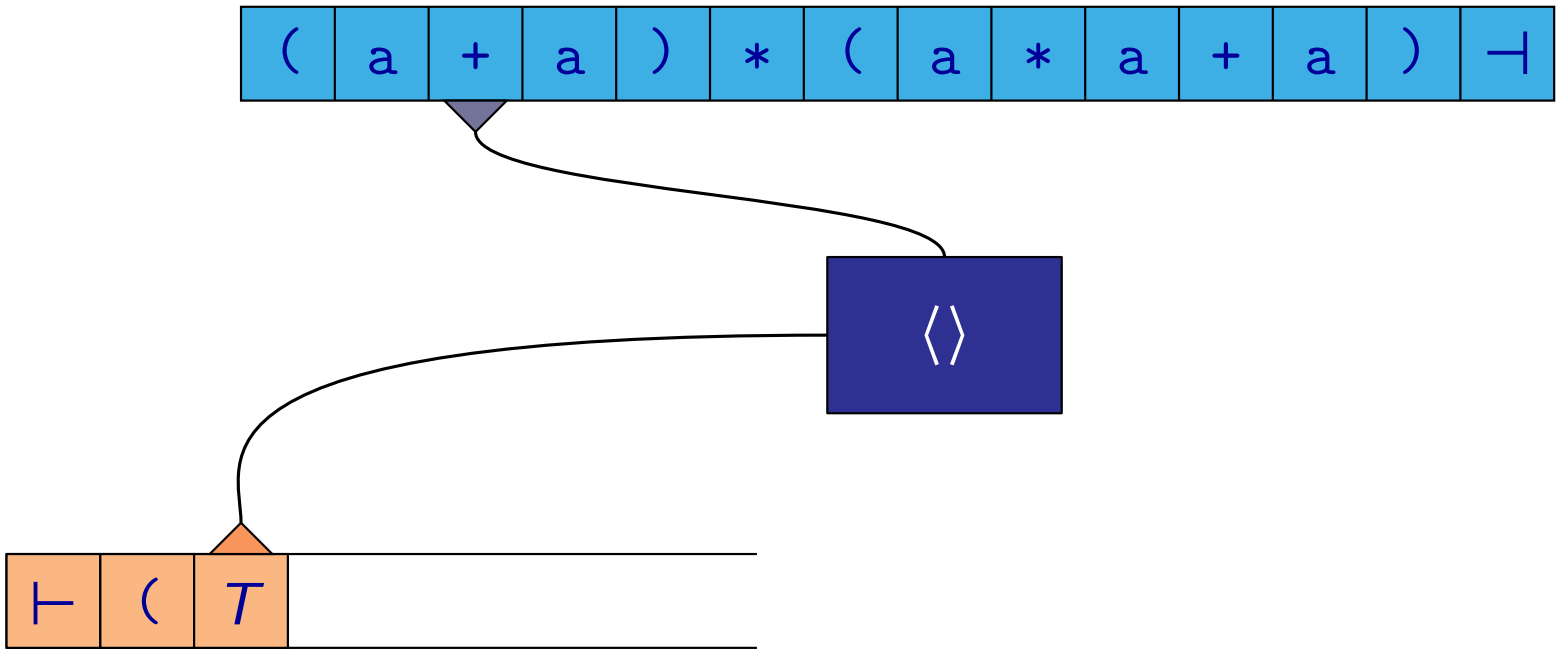
$(\underline{F}+a)*(a*a+a)\dashv \Rightarrow (a+a)*(a*a+a)\dashv$

Ekvivalence bezkontextových gramatik a zás. automatů



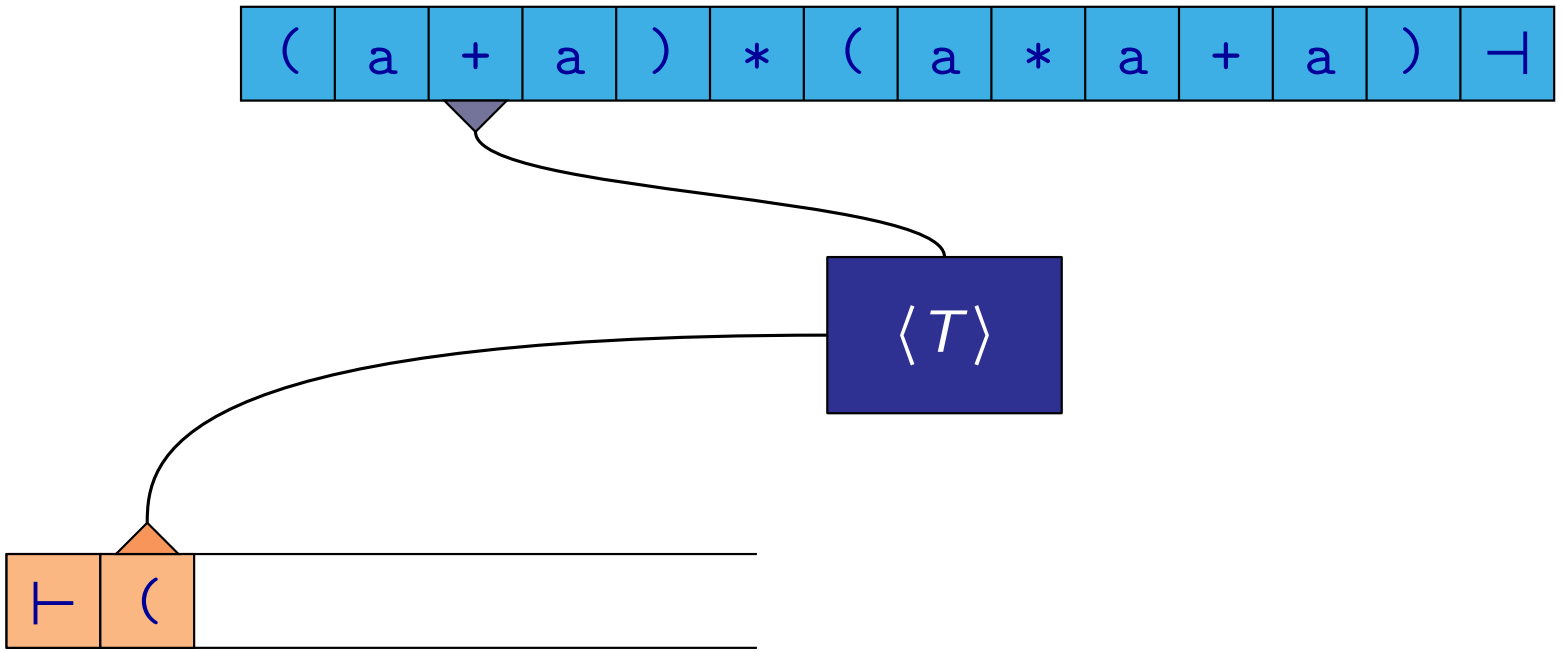
$(\underline{F}+a)*(a*a+a)\dashv \Rightarrow (a+a)*(a*a+a)\dashv$

Ekvivalence bezkontextových gramatik a zás. automatů



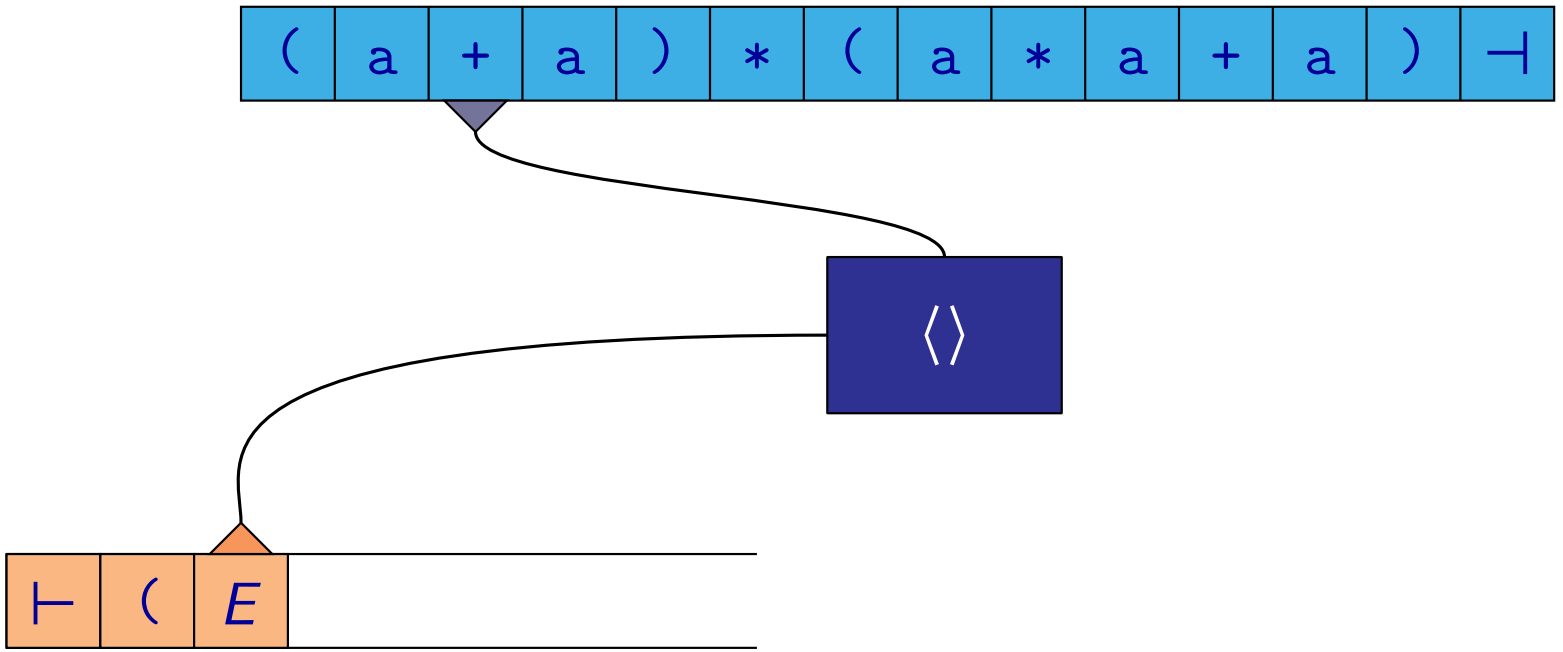
$(\underline{T}+a)*(a*a+a)\dashv \Rightarrow (\underline{F}+a)*(a*a+a)\dashv \Rightarrow (a+a)*(a*a+a)\dashv$

Ekvivalence bezkontextových gramatik a zás. automatů



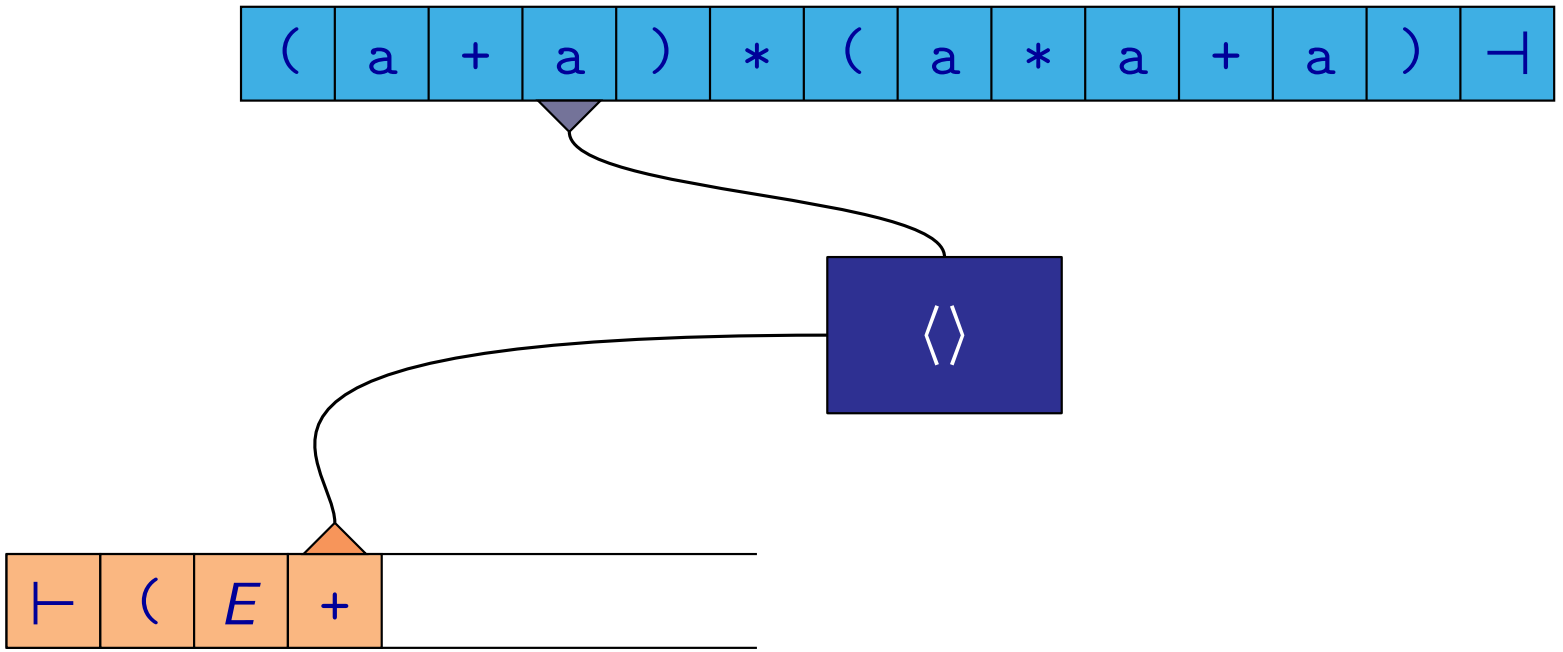
$(\underline{T}+a)*(a*a+a)\dashv \Rightarrow (\underline{F}+a)*(a*a+a)\dashv \Rightarrow (a+a)*(a*a+a)\dashv$

Ekvivalence bezkontextových gramatik a zás. automatů



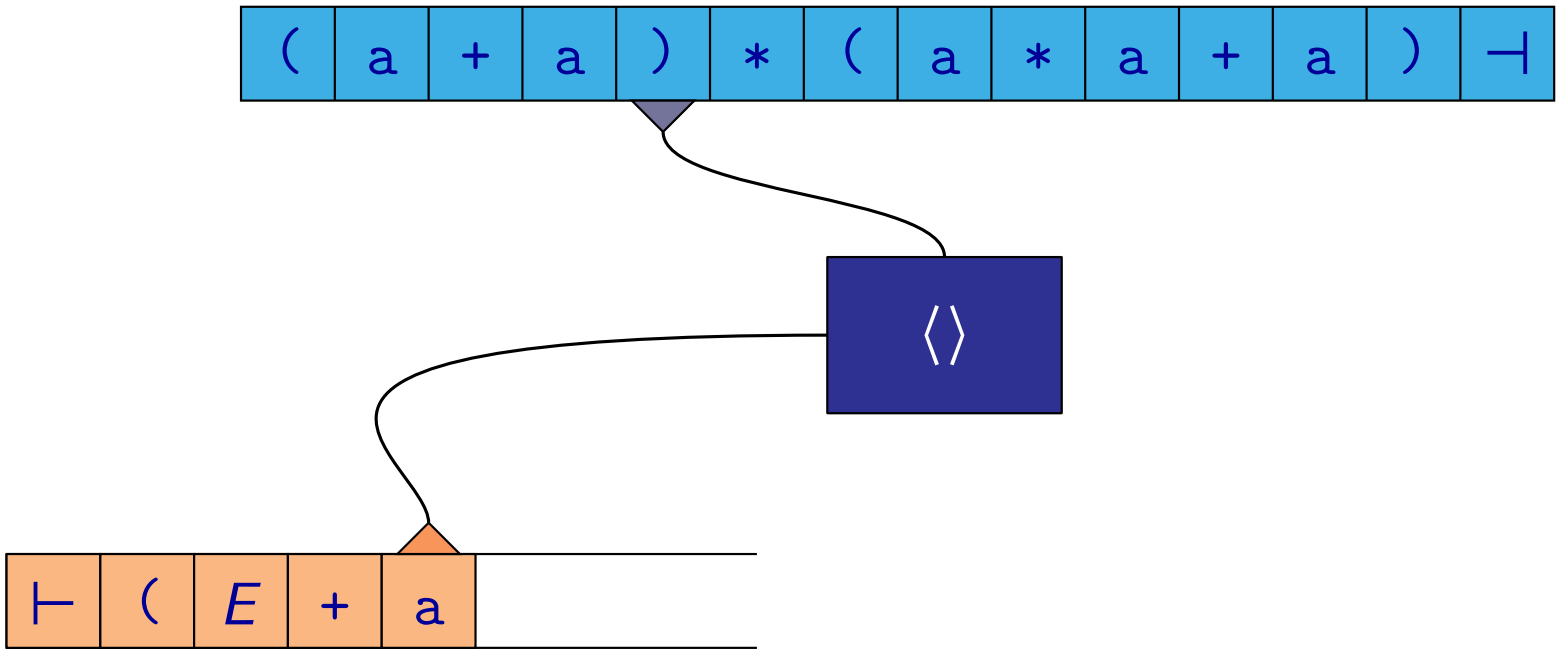
$(\underline{E}+a)*(a*a+a)\dashv \Rightarrow (\underline{T}+a)*(a*a+a)\dashv \Rightarrow (\underline{F}+a)*(a*a+a)\dashv \Rightarrow$
$\cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



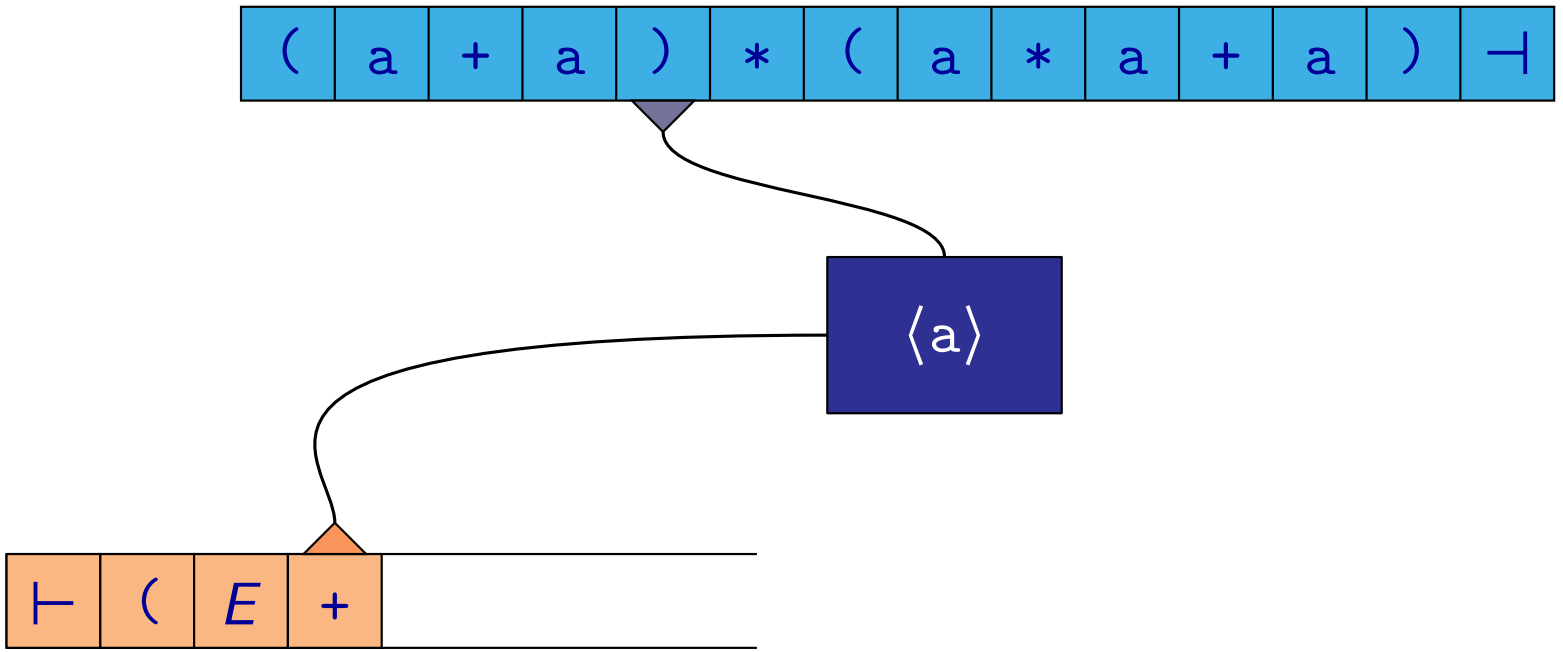
$(\underline{E}+a)*(a*a+a)\dashv \Rightarrow (\underline{T}+a)*(a*a+a)\dashv \Rightarrow (\underline{F}+a)*(a*a+a)\dashv \Rightarrow$
$\cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



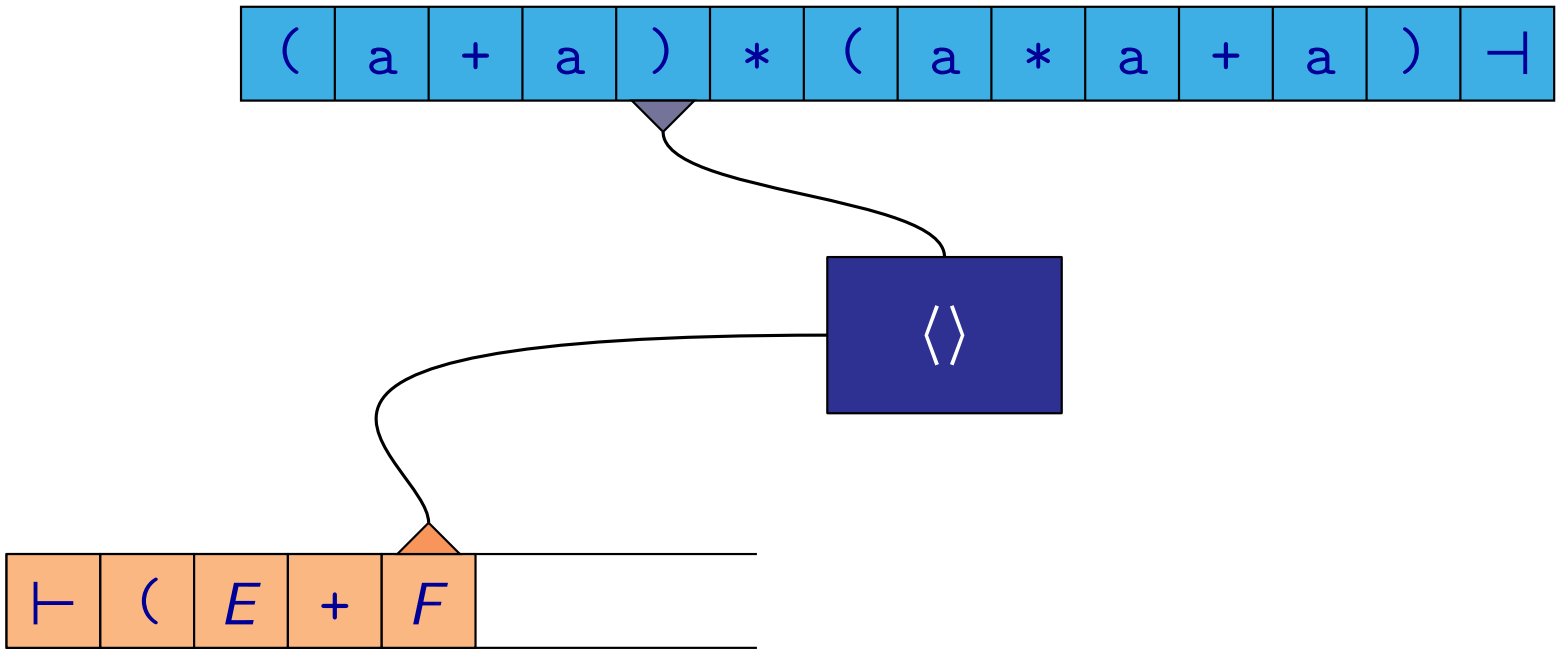
$(\underline{E}+a)*(a*a+a)\dashv \Rightarrow (\underline{T}+a)*(a*a+a)\dashv \Rightarrow (\underline{F}+a)*(a*a+a)\dashv \Rightarrow$
$\cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



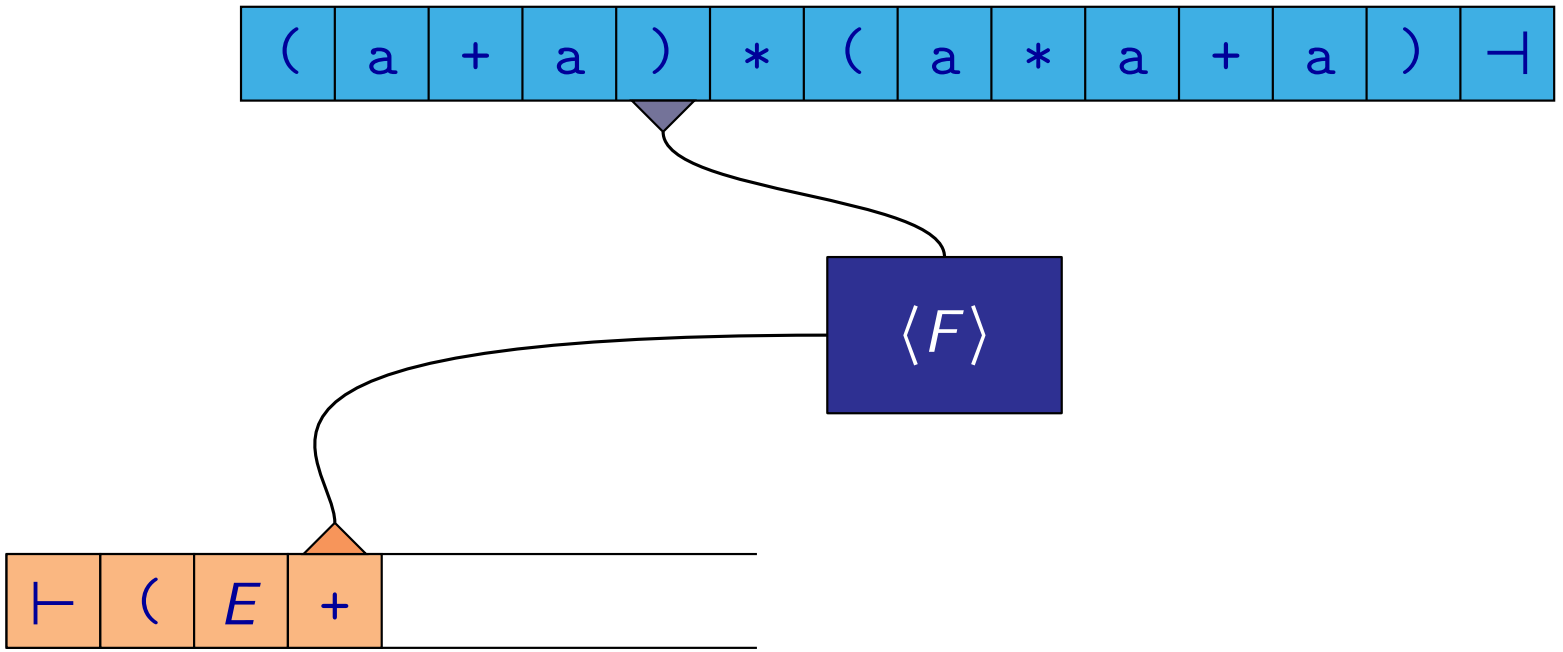
$(\underline{E}+a)*(a*a+a)\dashv \Rightarrow (\underline{T}+a)*(a*a+a)\dashv \Rightarrow (\underline{F}+a)*(a*a+a)\dashv \Rightarrow$
$\cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



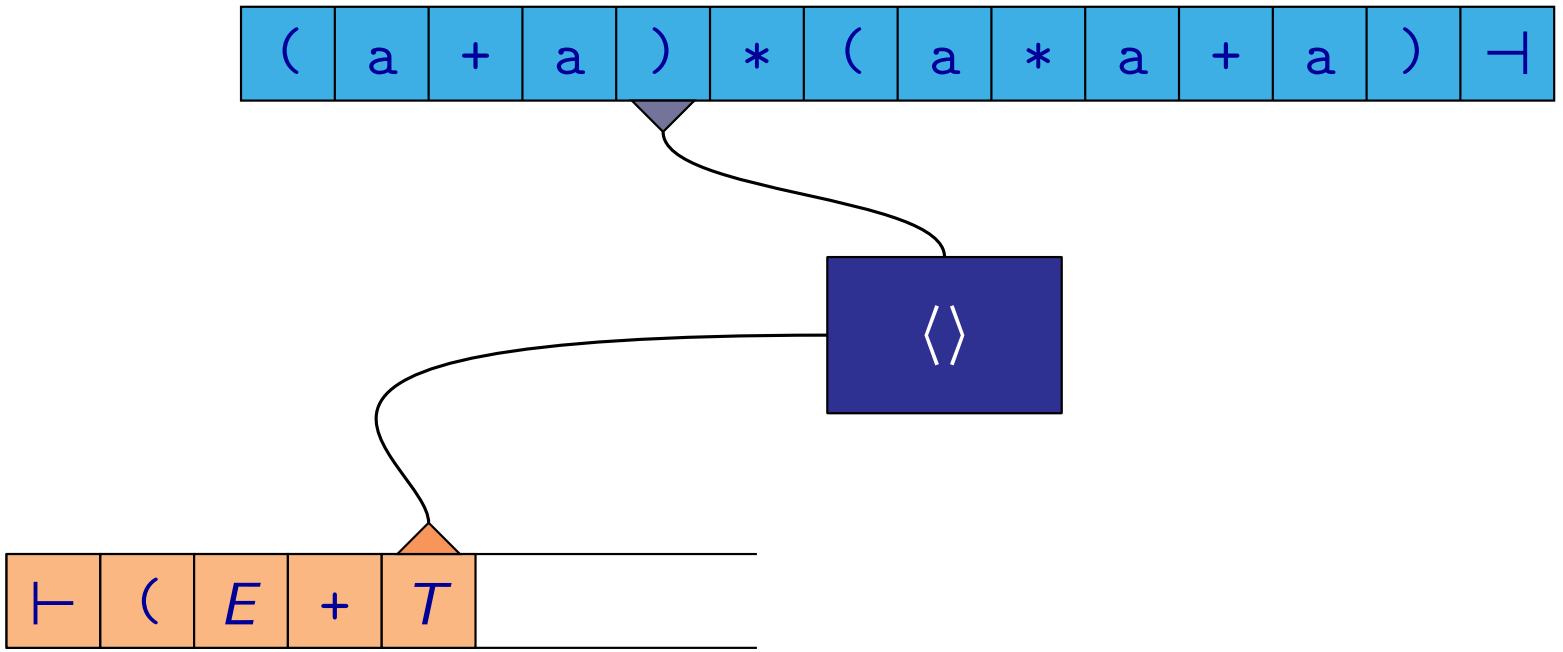
$(E+\underline{F})*(a*a+a)\dashv \Rightarrow (\underline{E}+a)*(a*a+a)\dashv \Rightarrow (\underline{T}+a)*(a*a+a)\dashv \Rightarrow$
$\cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



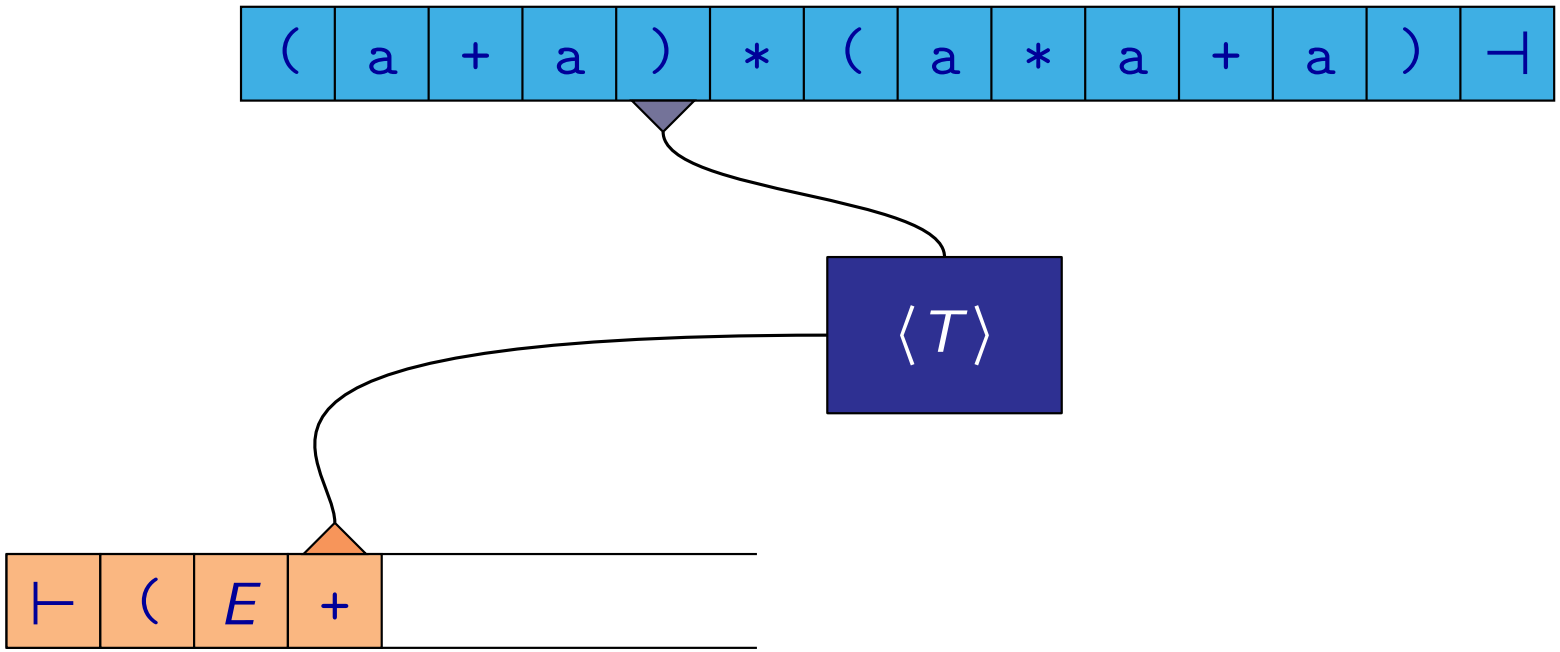
$(E+\underline{F})*(a*a+a)\dashv \Rightarrow (\underline{E}+a)*(a*a+a)\dashv \Rightarrow (\underline{T}+a)*(a*a+a)\dashv \Rightarrow$
$\cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



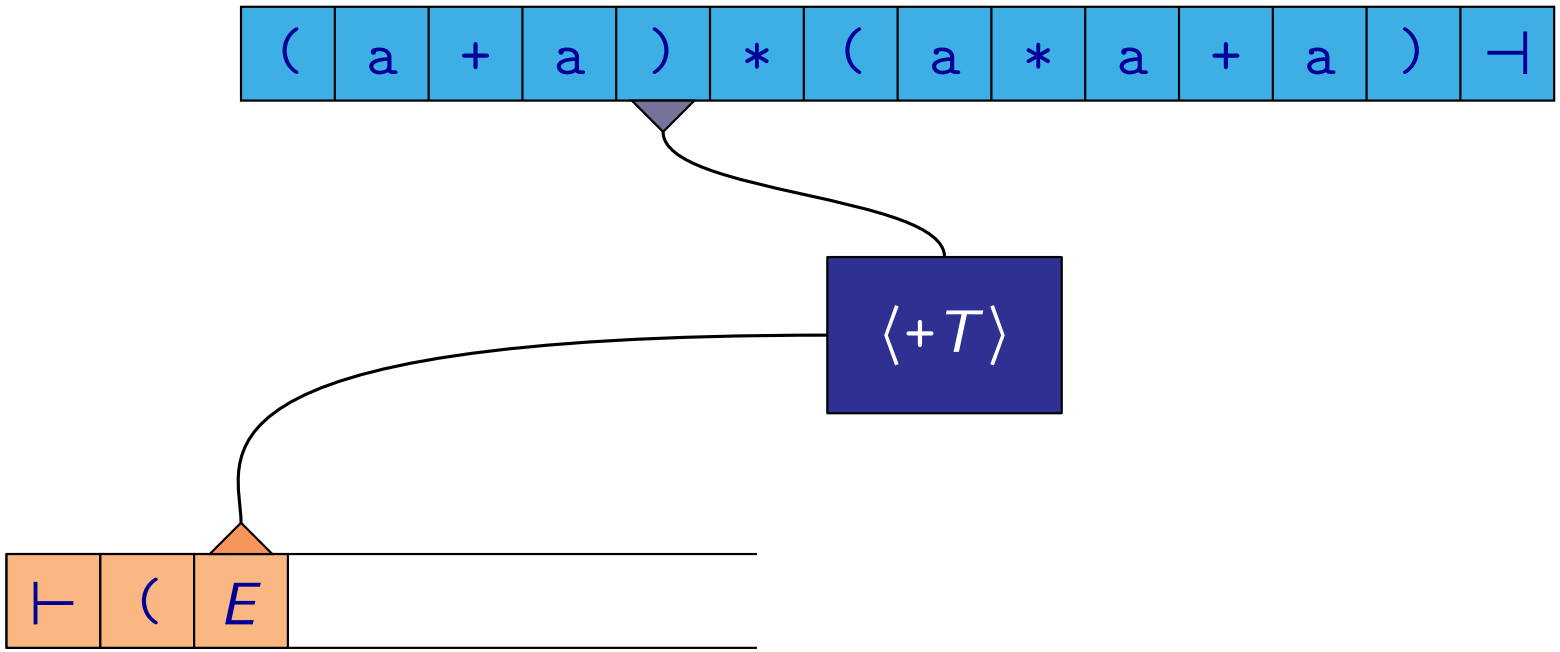
$(E+\underline{T})*(a*a+a)\dashv \Rightarrow (E+\underline{F})*(a*a+a)\dashv \Rightarrow (\underline{E}+a)*(a*a+a)\dashv \Rightarrow$
$\cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



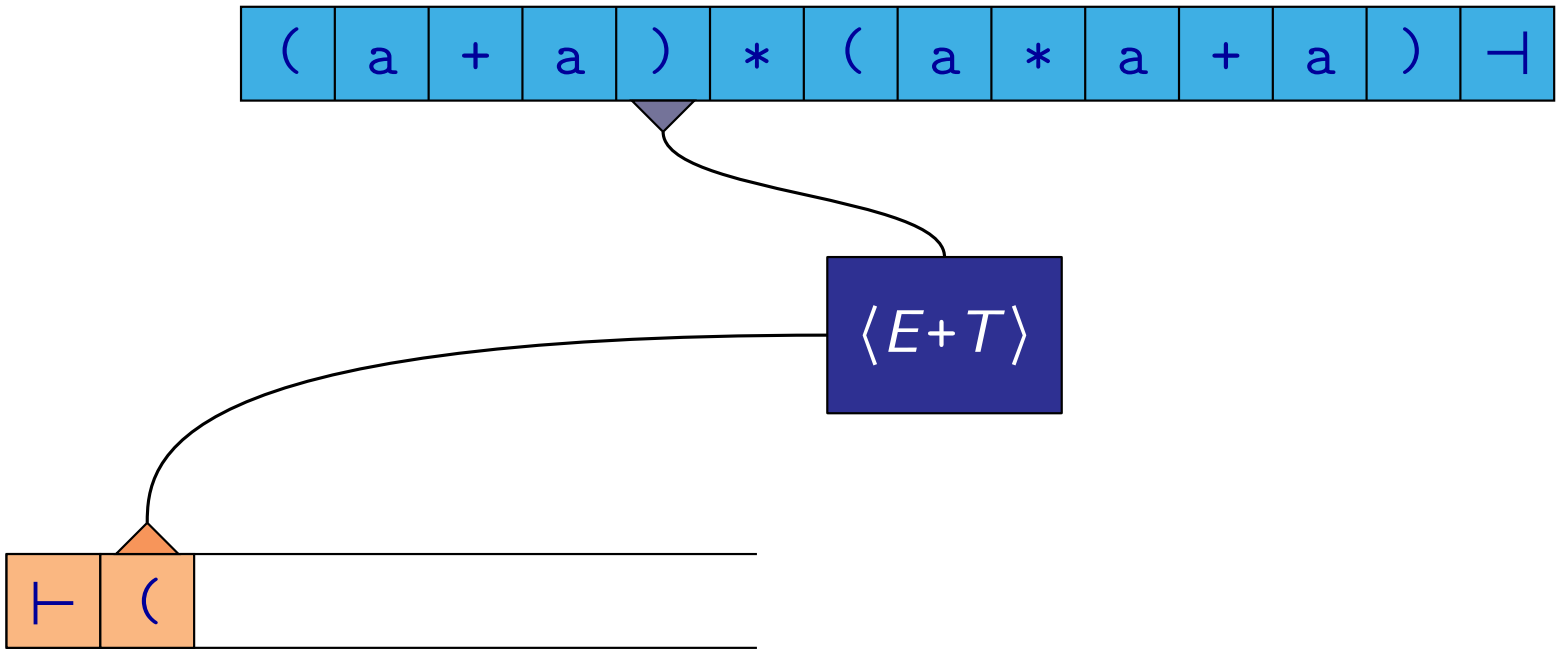
$(E+\underline{T})*(a*a+a)\dashv \Rightarrow (E+\underline{F})*(a*a+a)\dashv \Rightarrow (\underline{E}+a)*(a*a+a)\dashv \Rightarrow$
$\cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



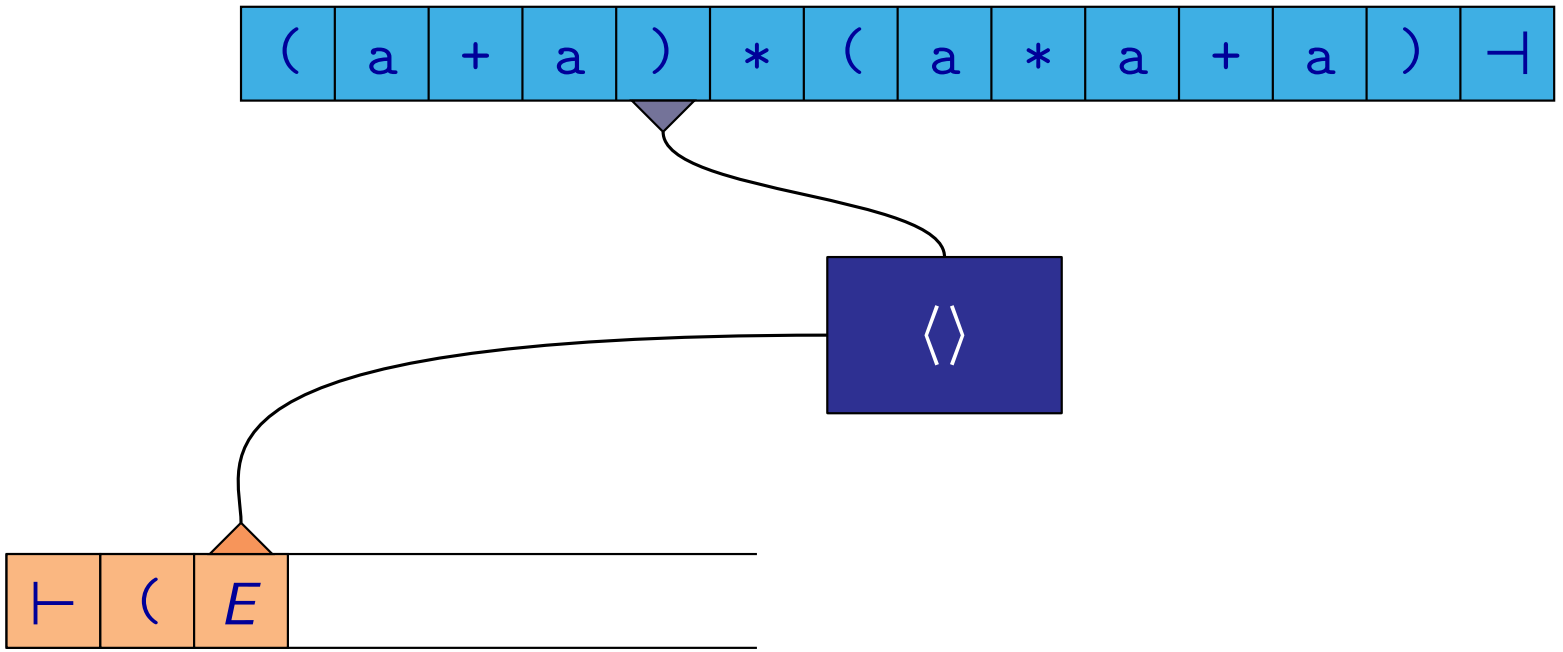
$(E+\underline{T})*(a*a+a)\dashv \Rightarrow (E+\underline{F})*(a*a+a)\dashv \Rightarrow (\underline{E}+a)*(a*a+a)\dashv \Rightarrow$
$\cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



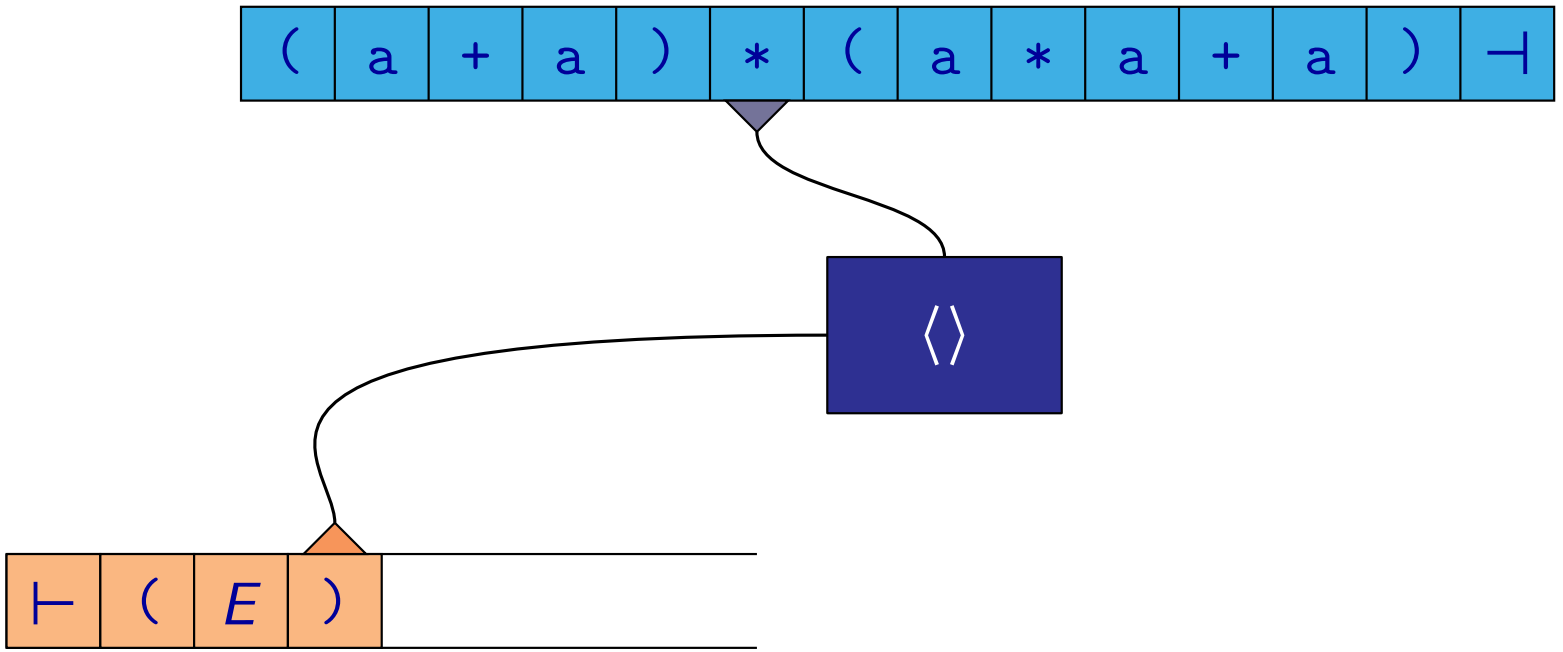
$(E+\underline{T})*(a*a+a)\dashv \Rightarrow (E+\underline{F})*(a*a+a)\dashv \Rightarrow (\underline{E}+a)*(a*a+a)\dashv \Rightarrow$
$\cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



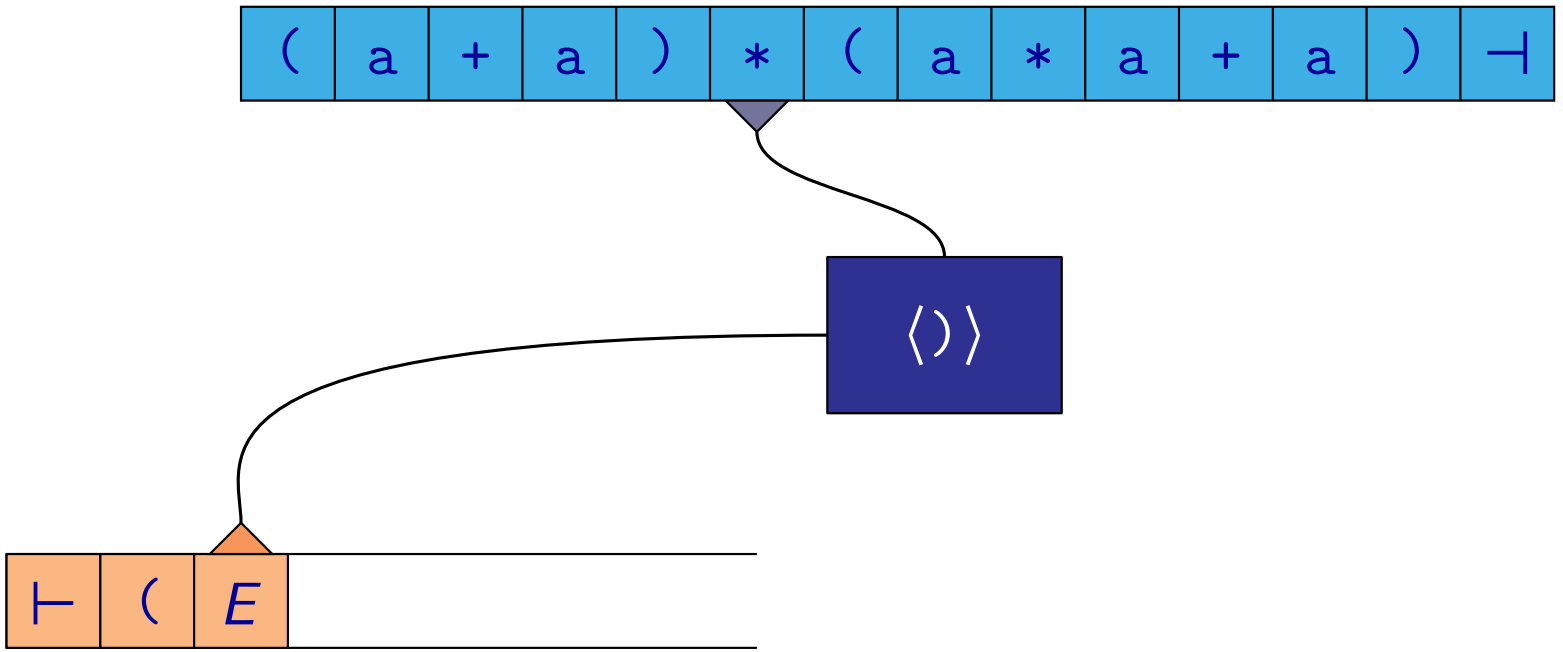
$(\underline{E})*(a*a+a)\dashv \Rightarrow (E+\underline{T})*(a*a+a)\dashv \Rightarrow (E+\underline{F})*(a*a+a)\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



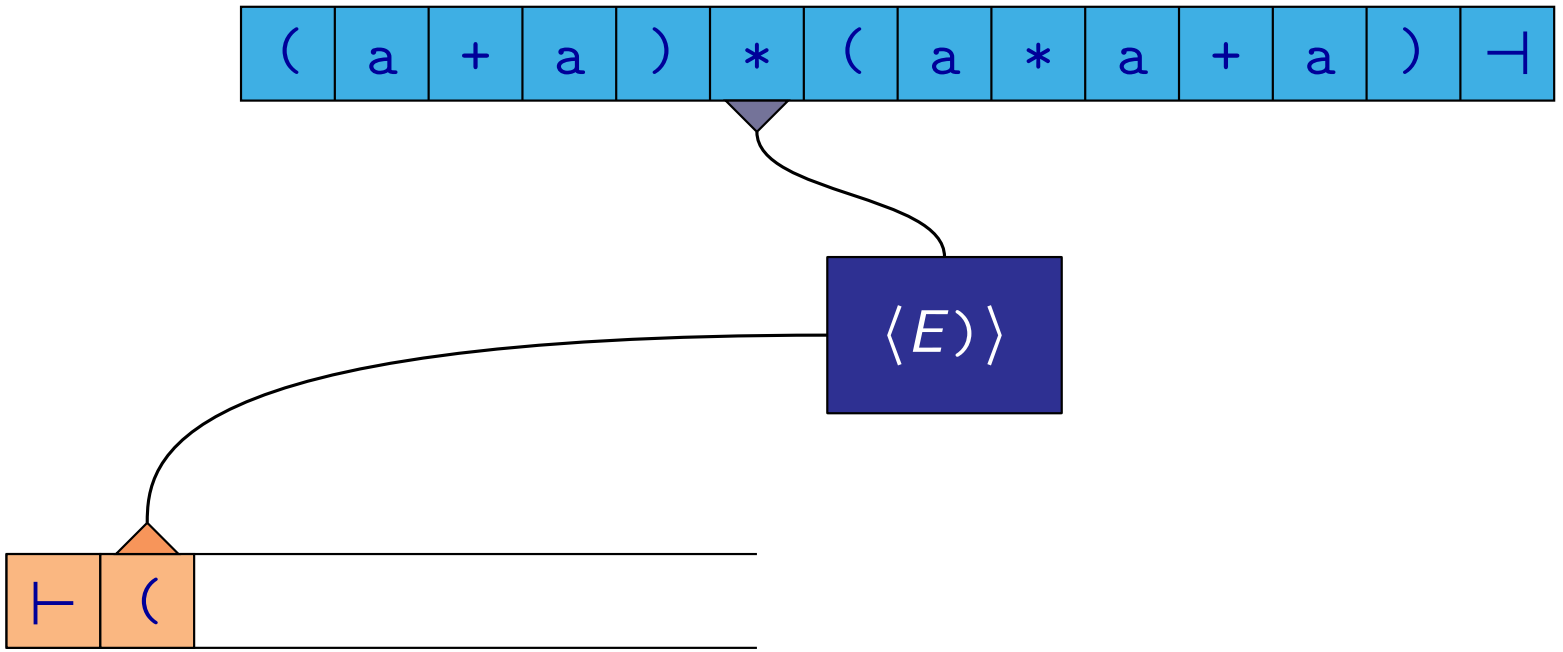
$(\underline{E})*(a*a+a)\dashv \Rightarrow (E+\underline{T})*(a*a+a)\dashv \Rightarrow (E+\underline{F})*(a*a+a)\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



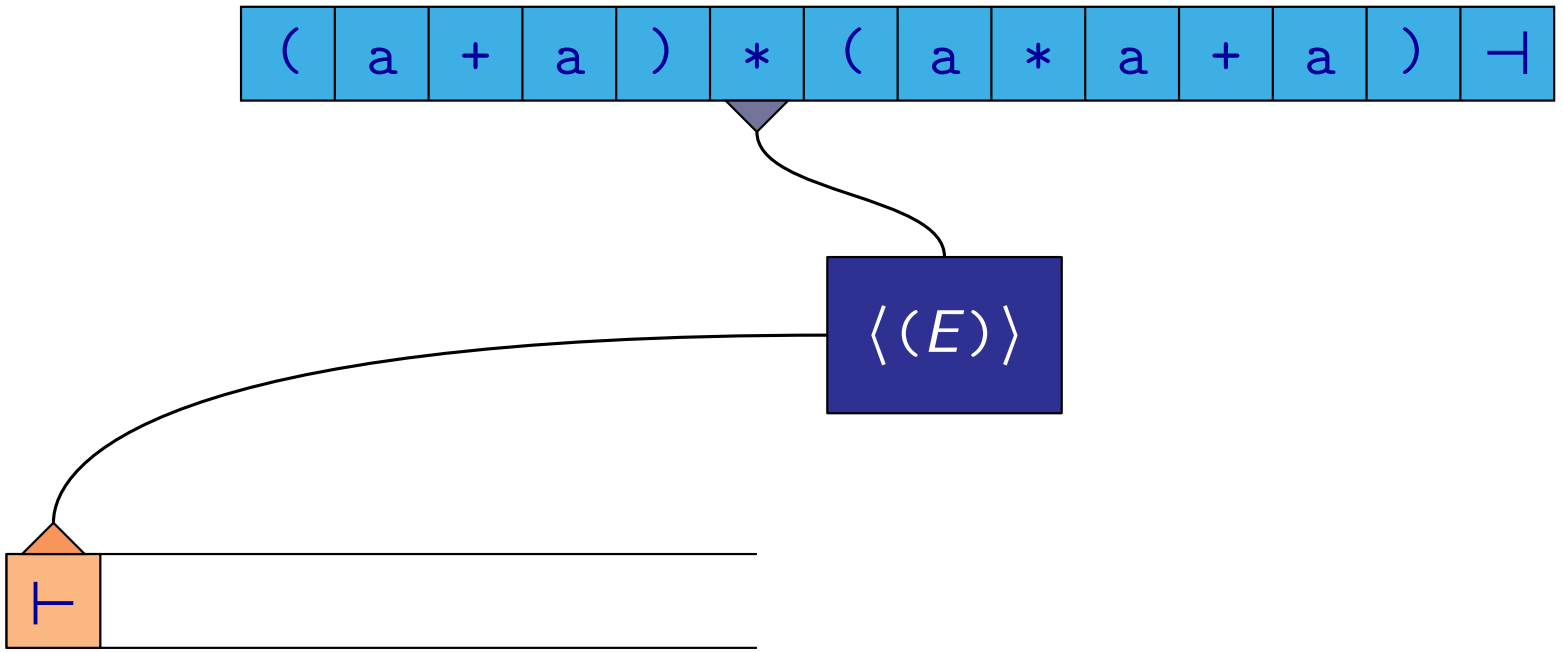
$(\underline{E})*(a*a+a)\dashv \Rightarrow (E+\underline{T})*(a*a+a)\dashv \Rightarrow (E+\underline{F})*(a*a+a)\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



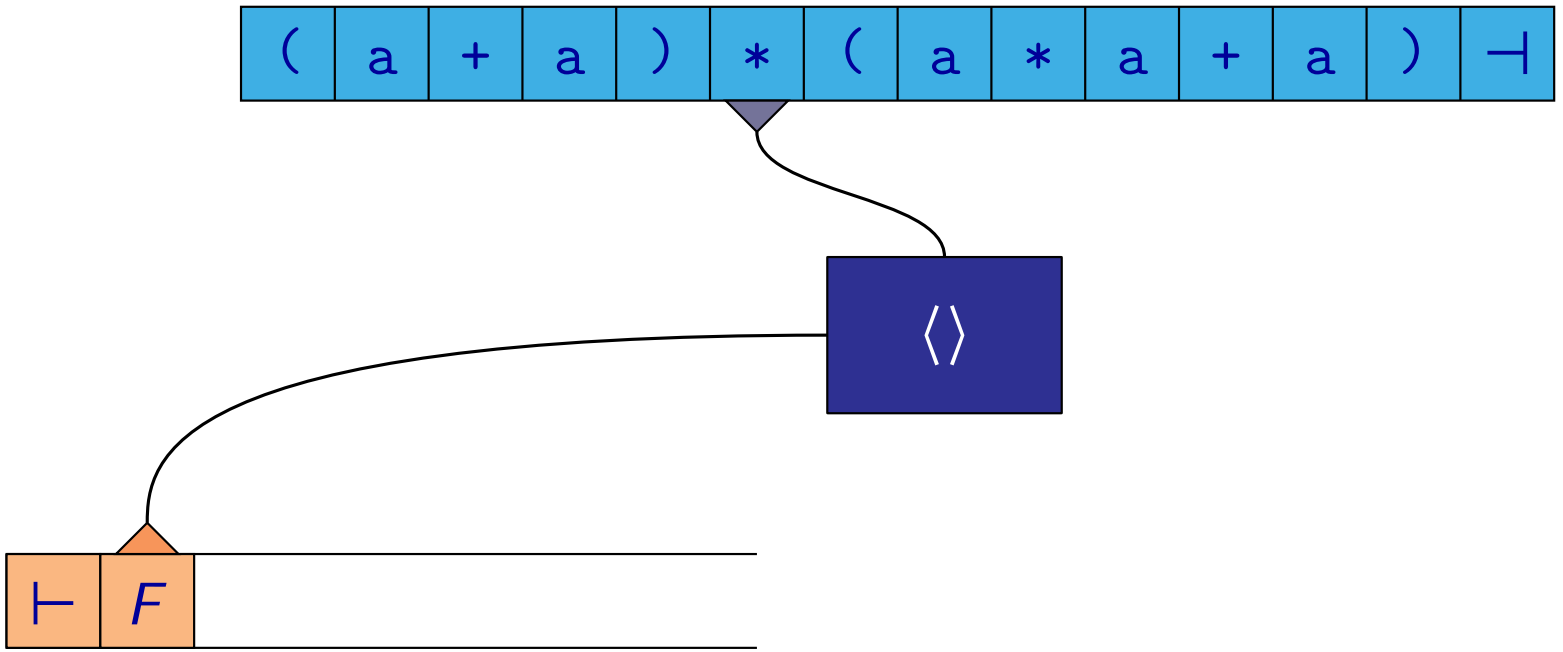
$(\underline{E})*(a*a+a)\dashv \Rightarrow (E+\underline{T})*(a*a+a)\dashv \Rightarrow (E+\underline{F})*(a*a+a)\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



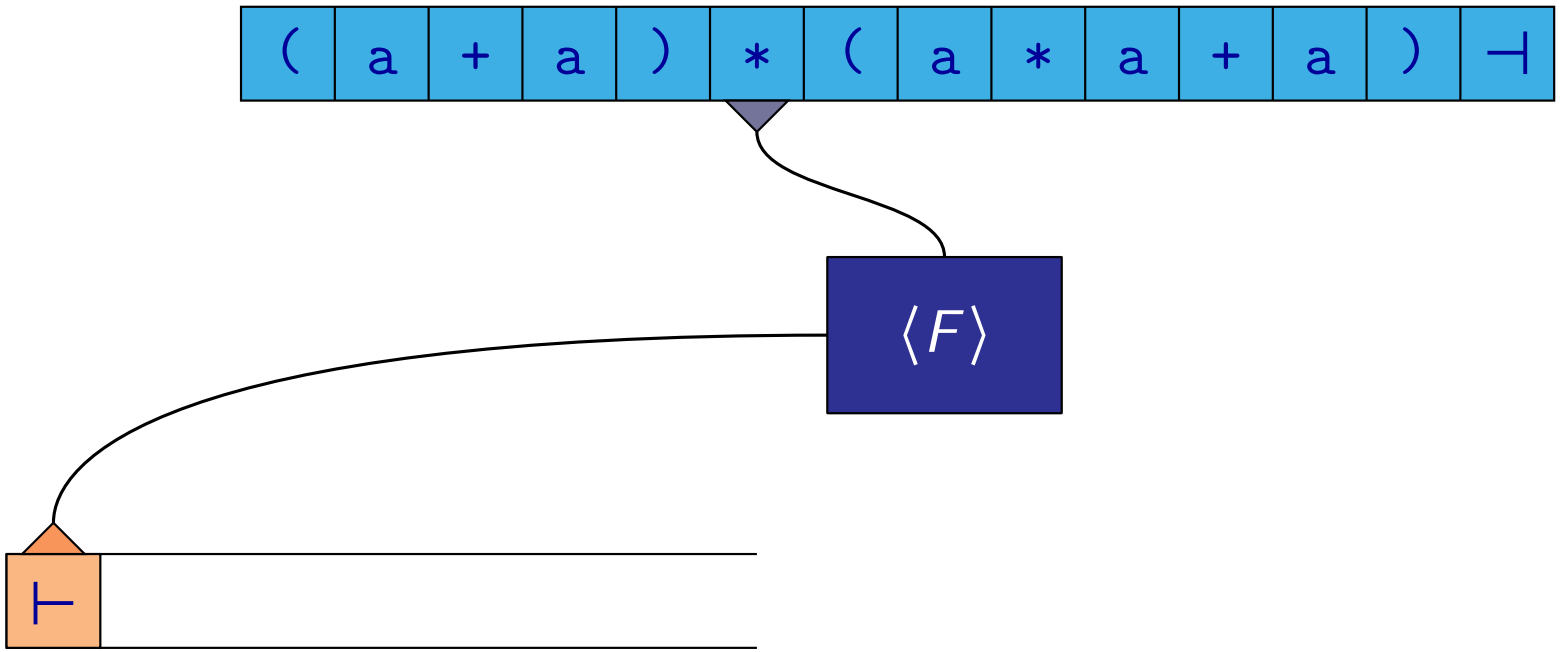
$(\underline{E})*(a*a+a)\dashv \Rightarrow (E+\underline{T})*(a*a+a)\dashv \Rightarrow (E+\underline{F})*(a*a+a)\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



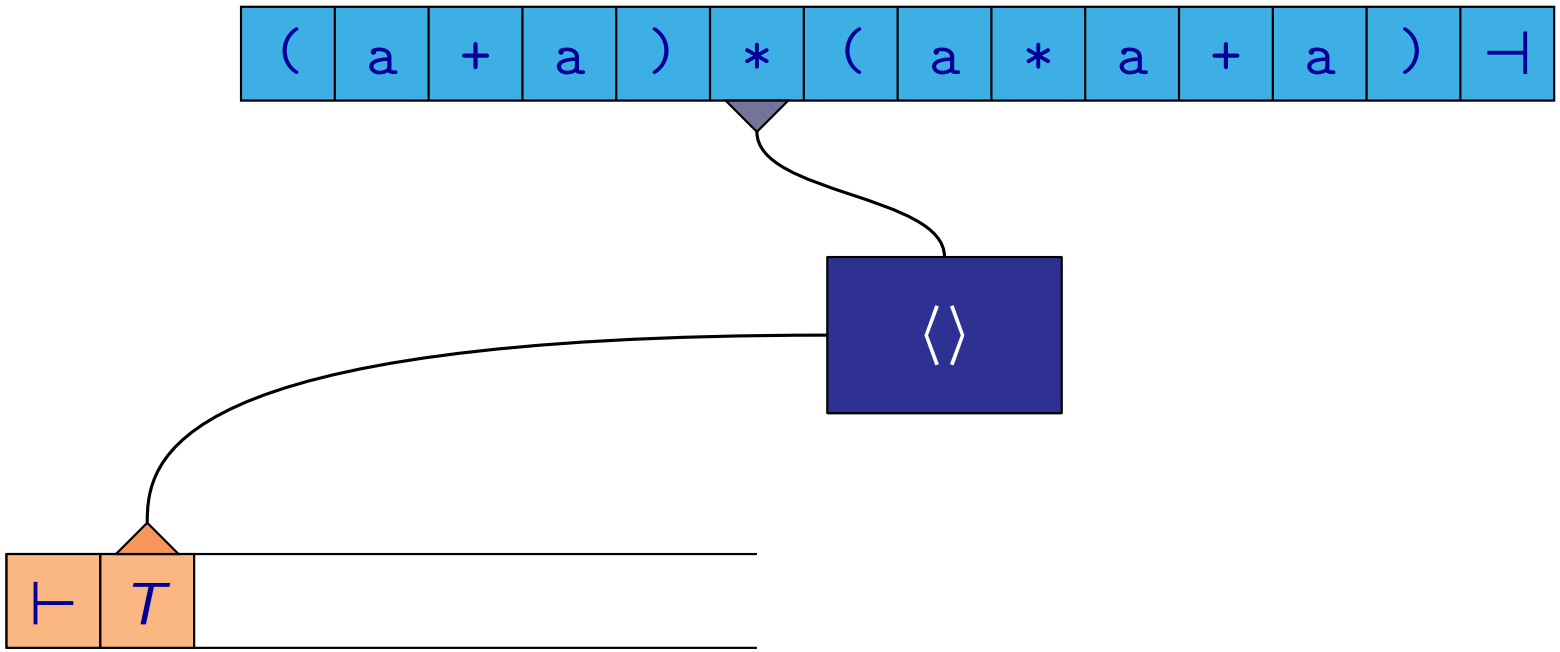
$\underline{F}*(a*a+a)\dashv \Rightarrow (\underline{E})*(a*a+a)\dashv \Rightarrow (E+\underline{T})*(a*a+a)\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



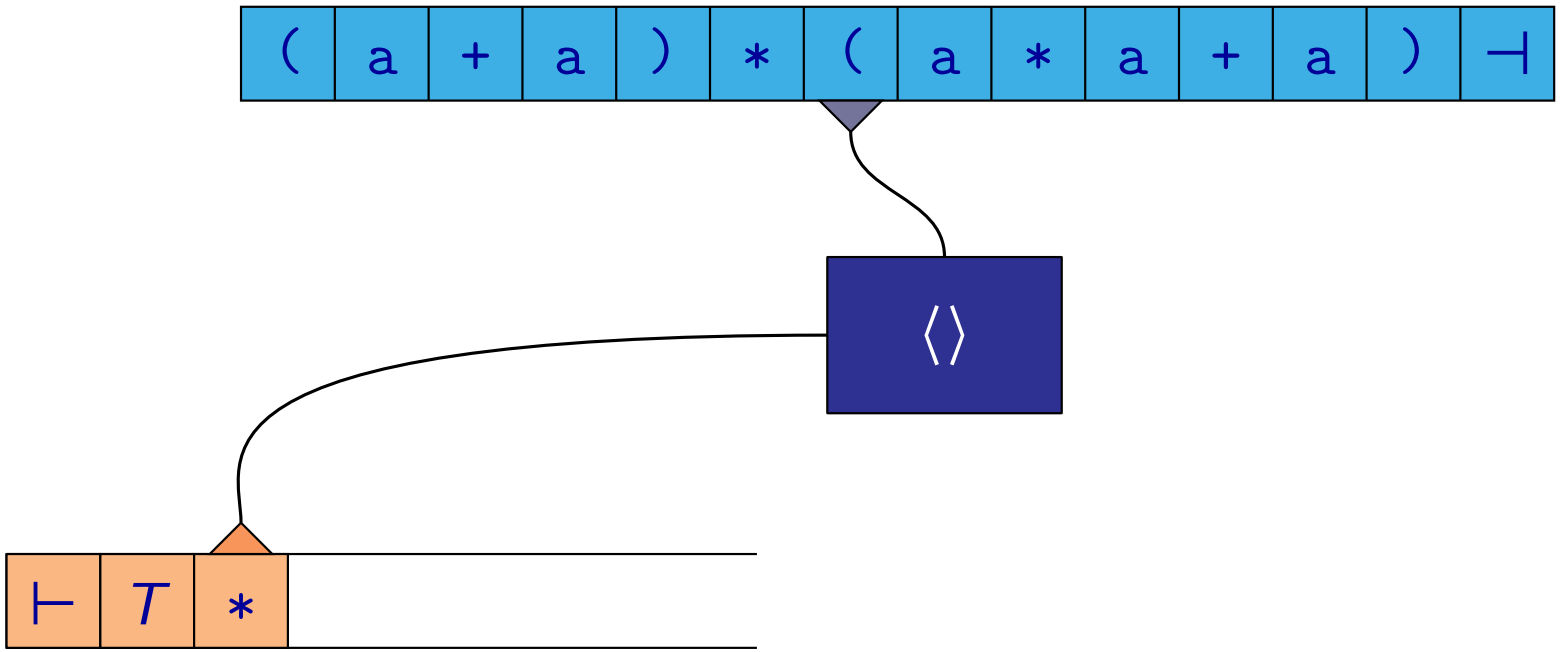
$\underline{F}$*(a*a+a)⊣ ⇒ (<u>E</u>)*(a*a+a)⊣ ⇒ (E+<u>T</u>)*(a*a+a)⊣ ⇒ ⋯

Ekvivalence bezkontextových gramatik a zás. automatů



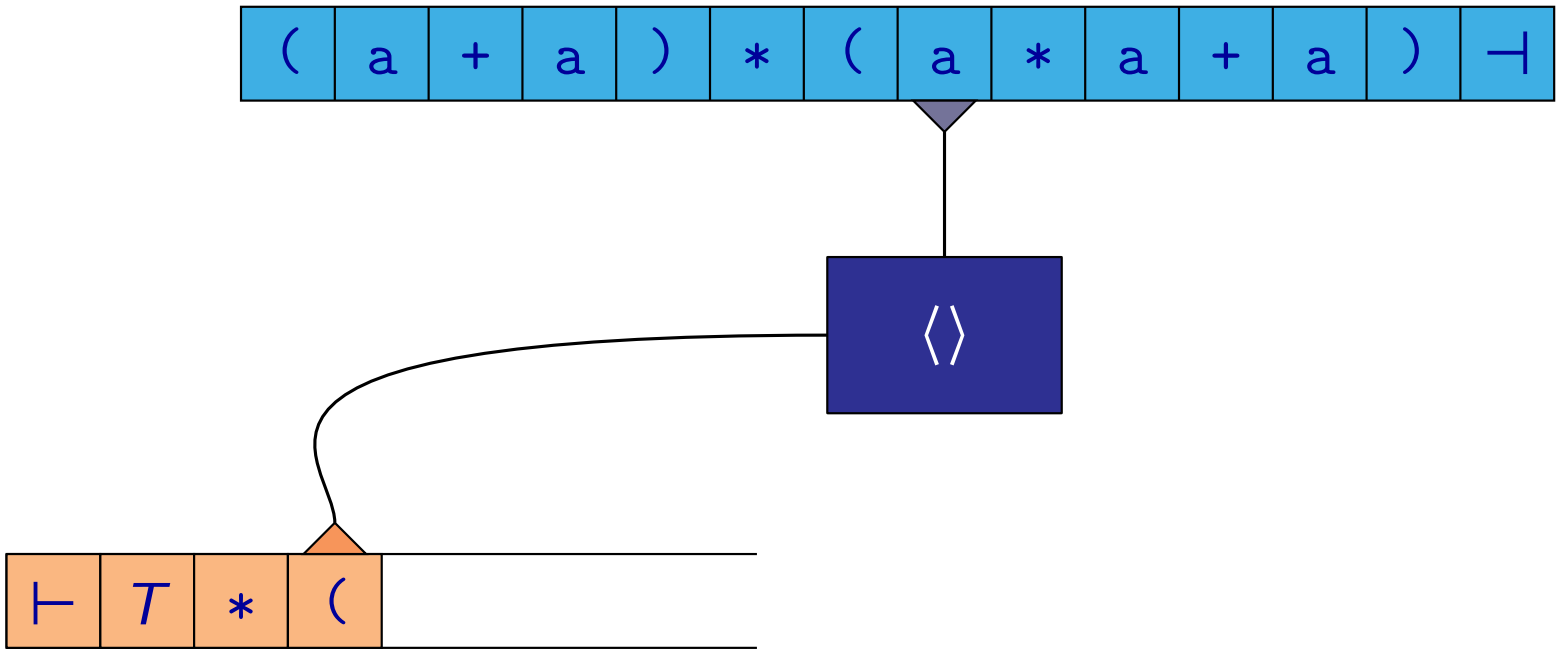
$\underline{T}$*(a*a+a)⊣ ⇒ $\underline{F}$*(a*a+a)⊣ ⇒ (E)*(a*a+a)⊣ ⇒ ⋯

Ekvivalence bezkontextových gramatik a zás. automatů



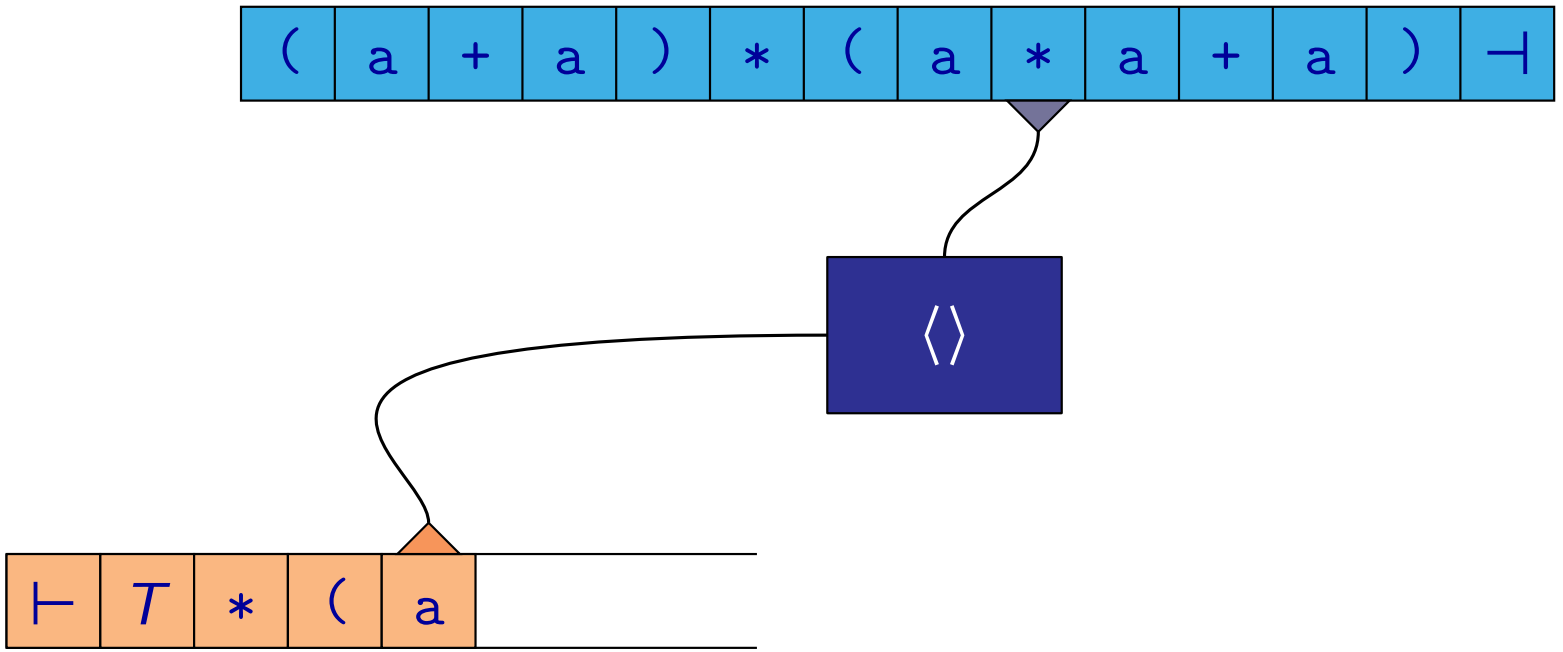
$\underline{T}$*(a*a+a)⊣ ⇒ $\underline{F}$*(a*a+a)⊣ ⇒ (E)*(a*a+a)⊣ ⇒ ⋯

Ekvivalence bezkontextových gramatik a zás. automatů



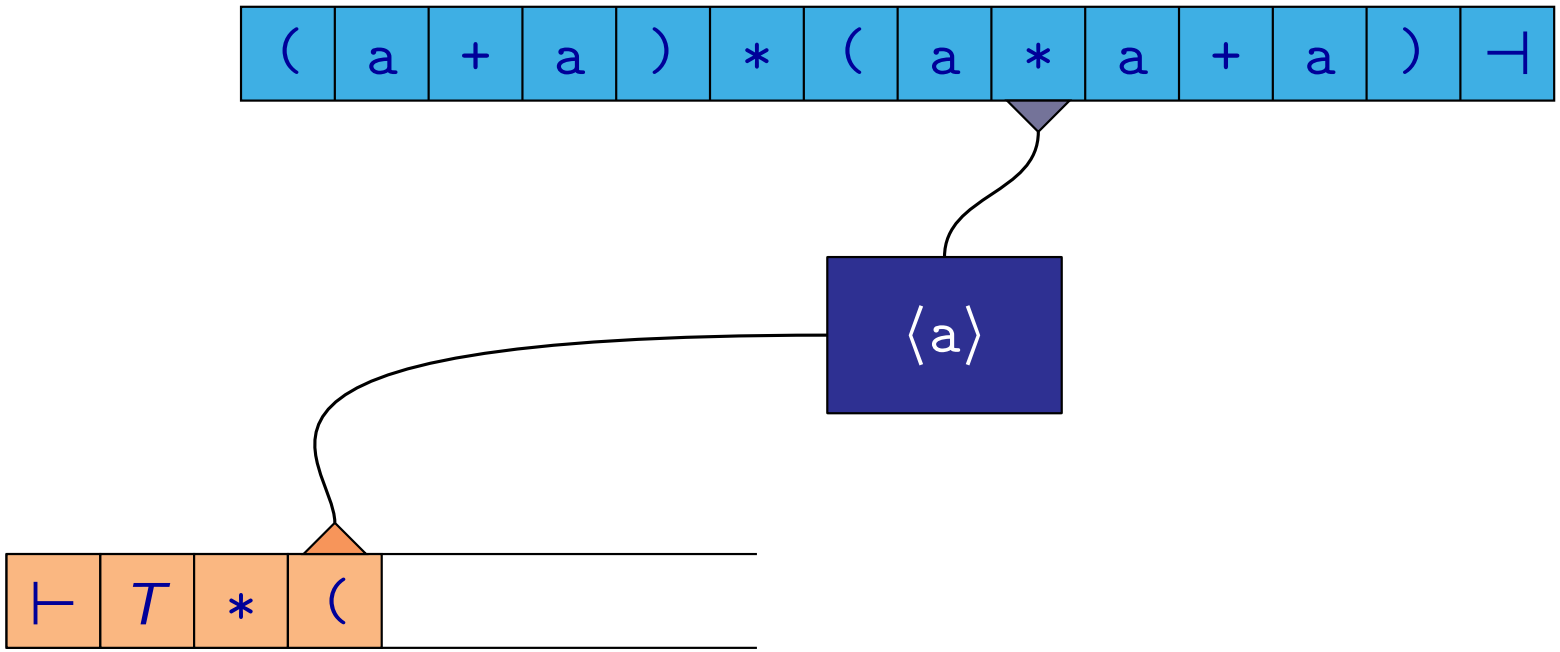
$\underline{T}$*(a*a+a)⊣ ⇒ $\underline{F}$*(a*a+a)⊣ ⇒ (E)*(a*a+a)⊣ ⇒ ⋯

Ekvivalence bezkontextových gramatik a zás. automatů



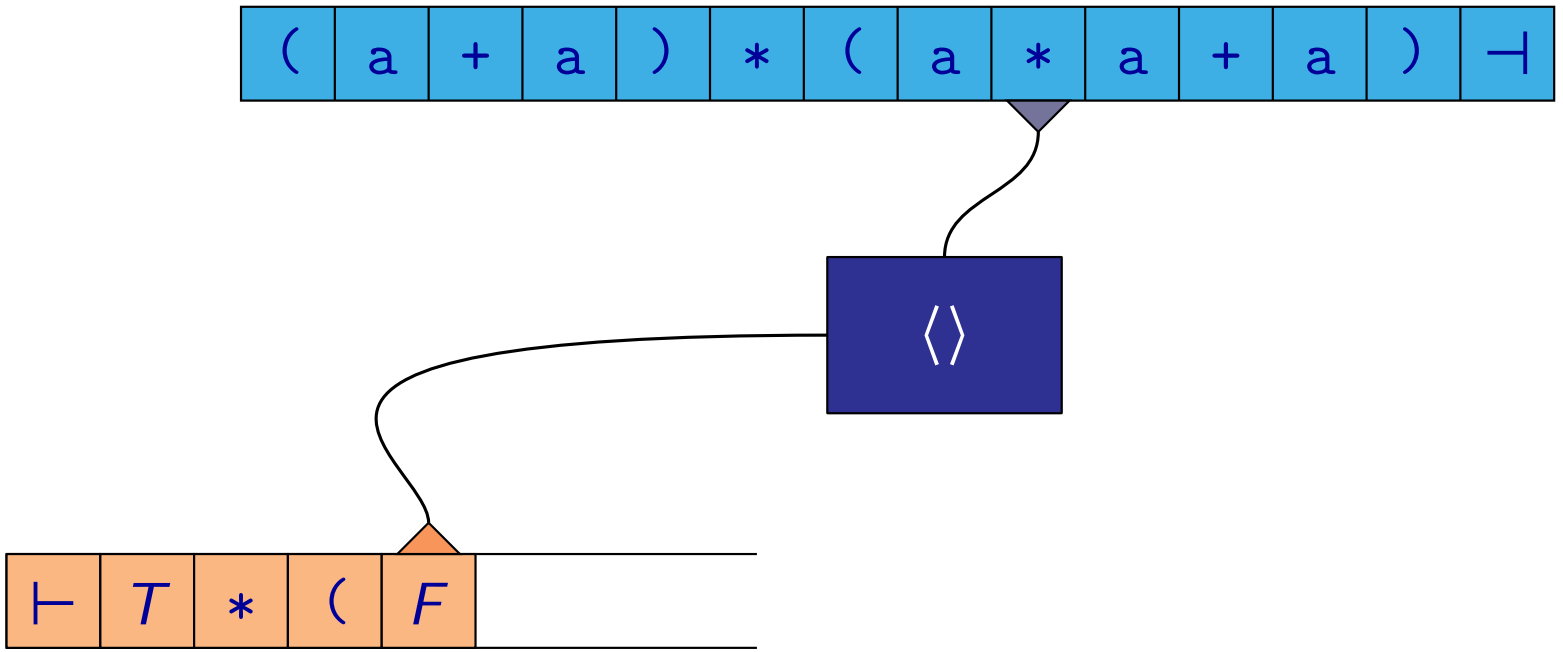
$\underline{T}$*(a*a+a)⊣ ⇒ $\underline{F}$*(a*a+a)⊣ ⇒ (E)*(a*a+a)⊣ ⇒ ⋯

Ekvivalence bezkontextových gramatik a zás. automatů



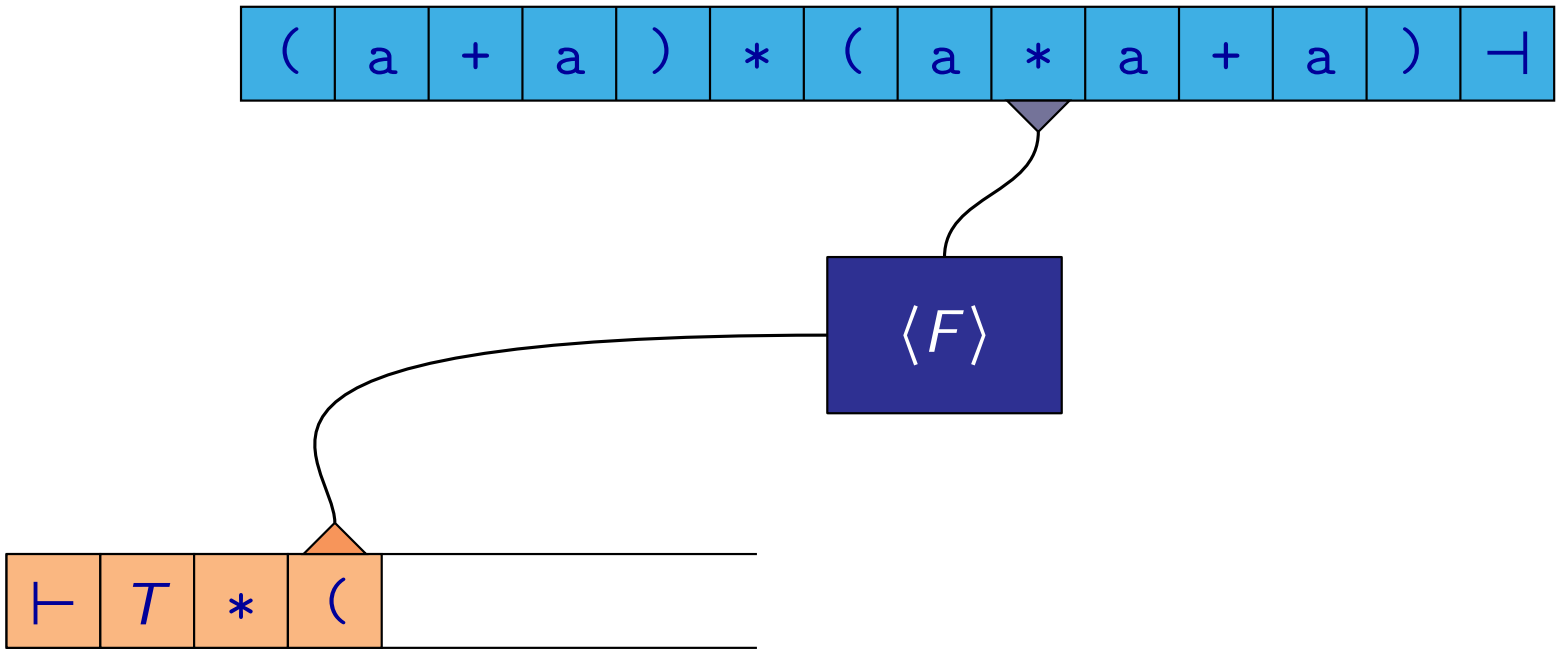
$\underline{T}$*(a*a+a)⊣ ⇒ $\underline{F}$*(a*a+a)⊣ ⇒ (E)*(a*a+a)⊣ ⇒ ⋯

Ekvivalence bezkontextových gramatik a zás. automatů



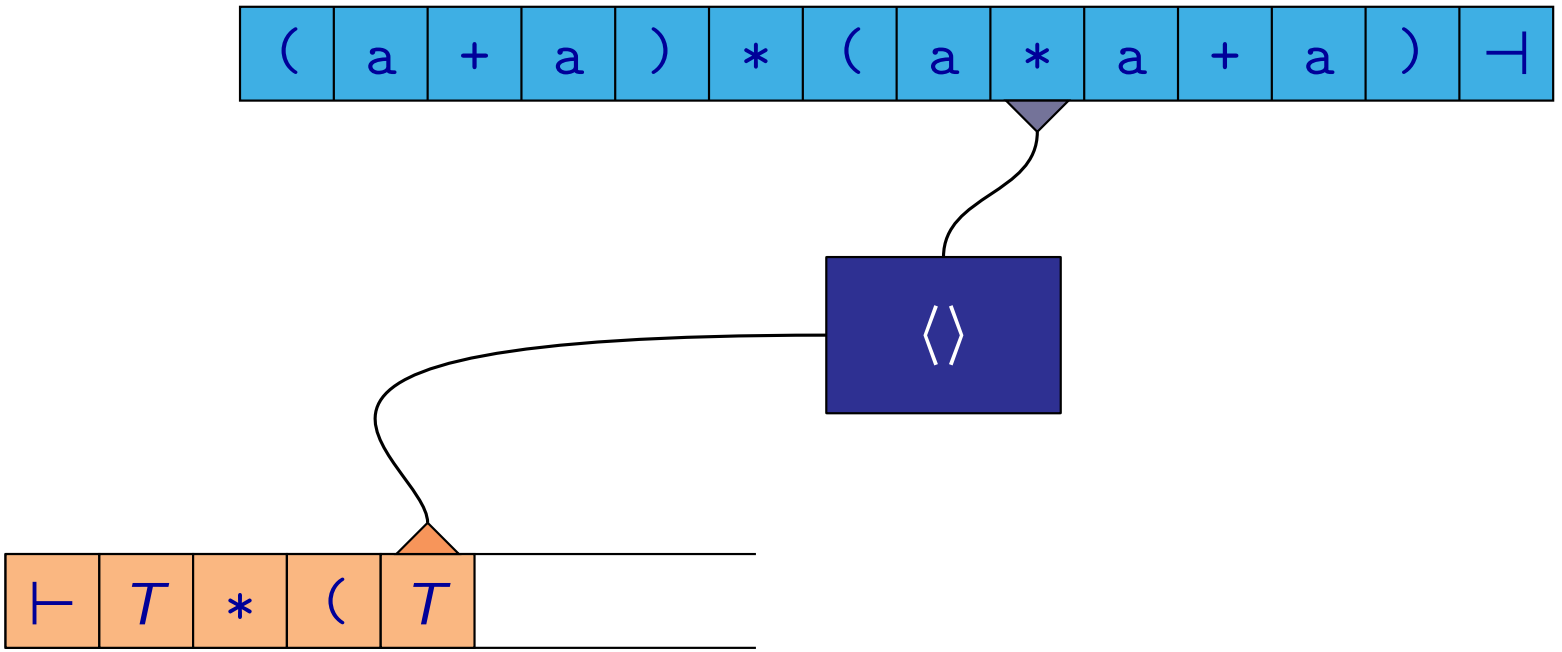
$T*(\underline{F}*a+a)\dashv \Rightarrow \underline{T}*(a*a+a)\dashv \Rightarrow \underline{F}*(a*a+a)\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



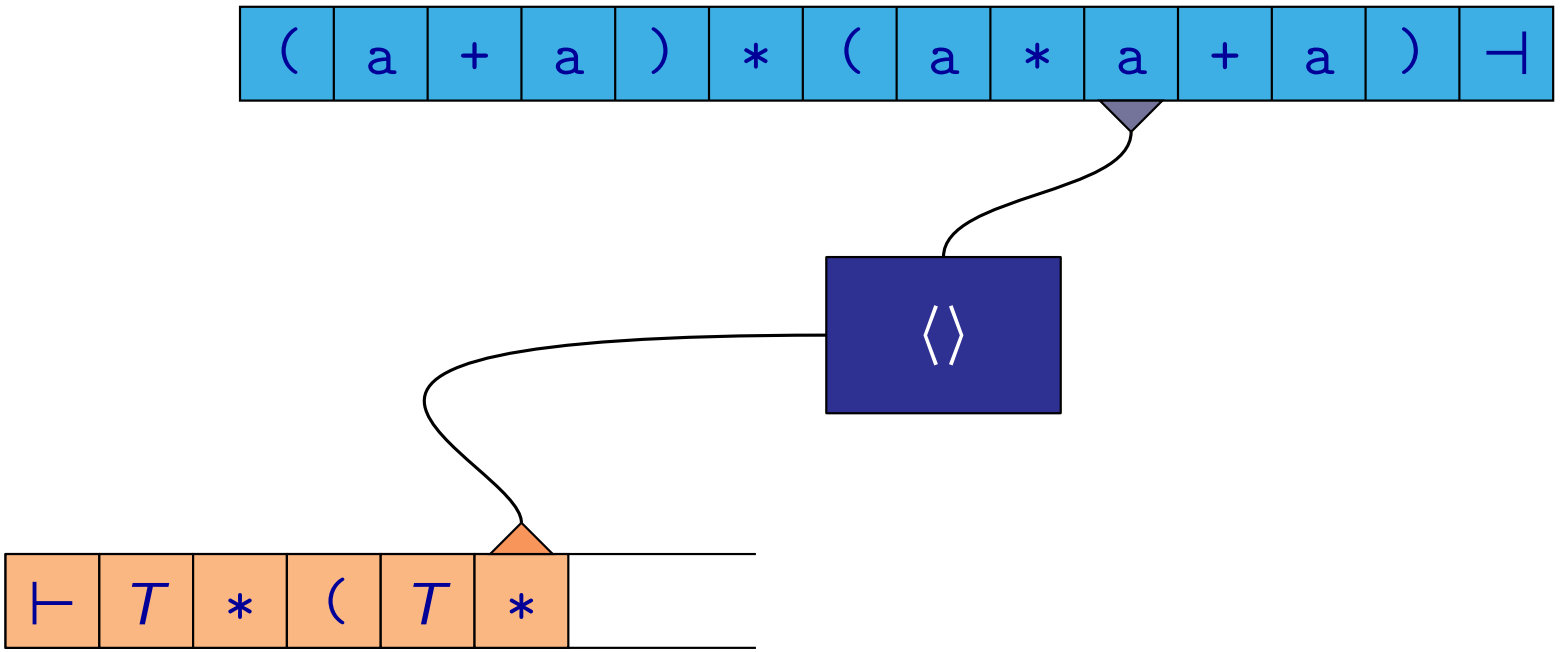
$T*(\underline{F}*a+a)\dashv \Rightarrow \underline{T}*(a*a+a)\dashv \Rightarrow \underline{F}*(a*a+a)\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



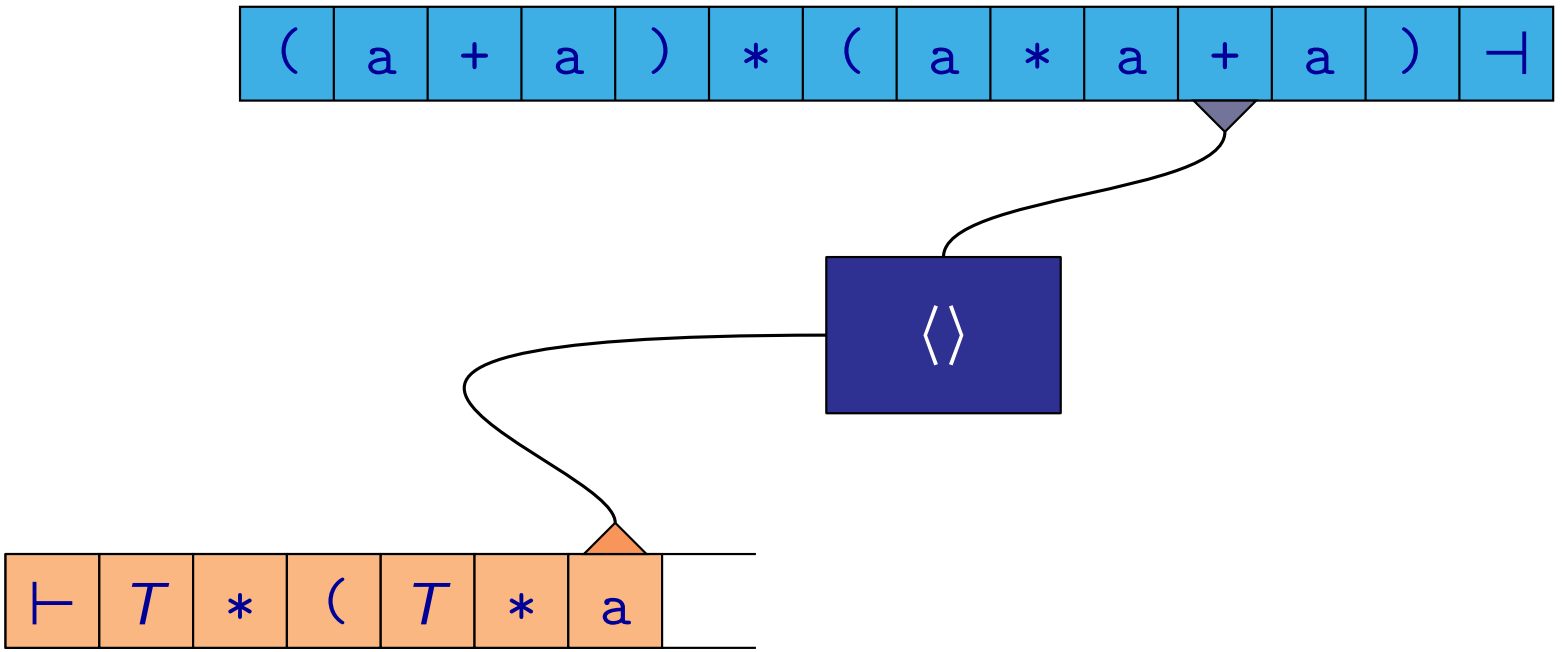
$T*(\underline{T}*a+a)\dashv \Rightarrow T*(\underline{F}*a+a)\dashv \Rightarrow \underline{T}*(a*a+a)\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



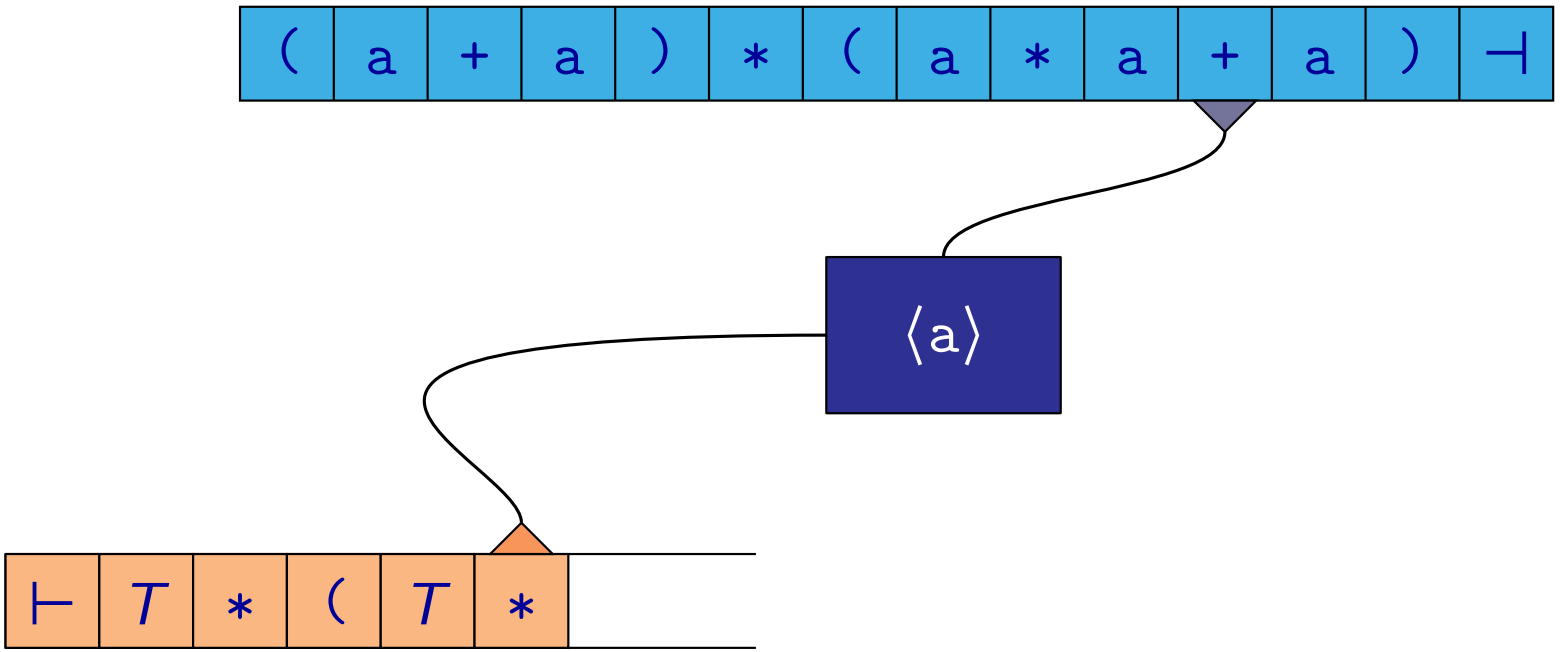
$T*(\underline{T}*a+a)\dashv \Rightarrow T*(\underline{F}*a+a)\dashv \Rightarrow \underline{T}*(a*a+a)\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



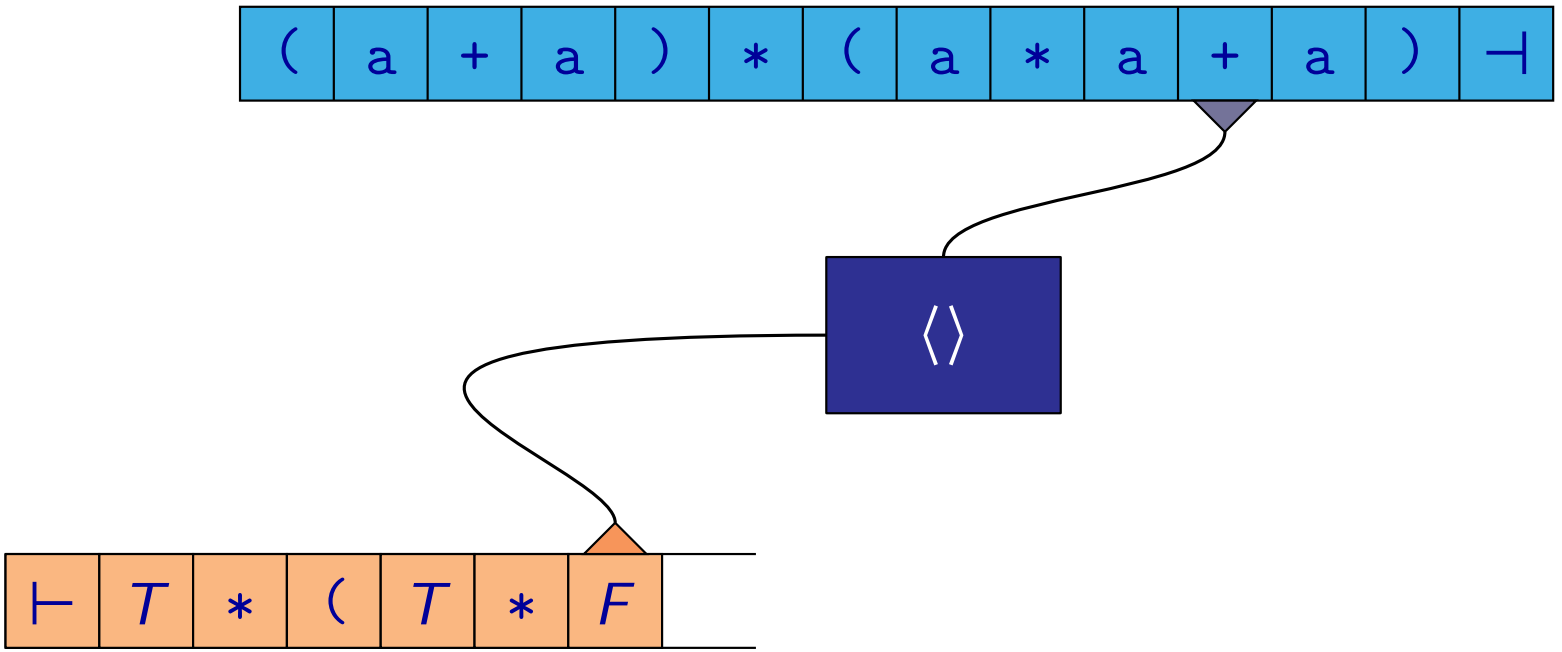
$T*(\underline{T}*a+a)\dashv \Rightarrow T*(\underline{F}*a+a)\dashv \Rightarrow \underline{T}*(a*a+a)\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



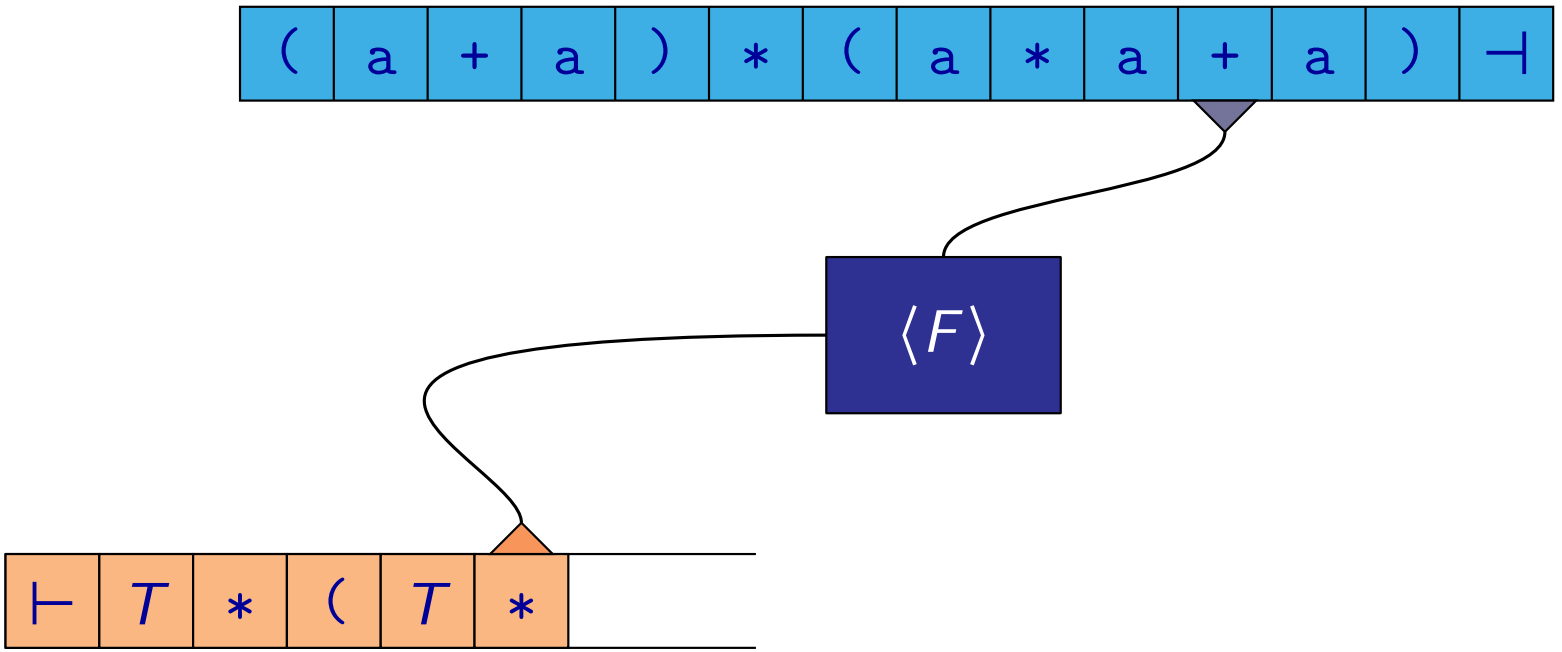
$T*(\underline{T}*a+a)\dashv \Rightarrow T*(\underline{F}*a+a)\dashv \Rightarrow \underline{T}*(a*a+a)\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



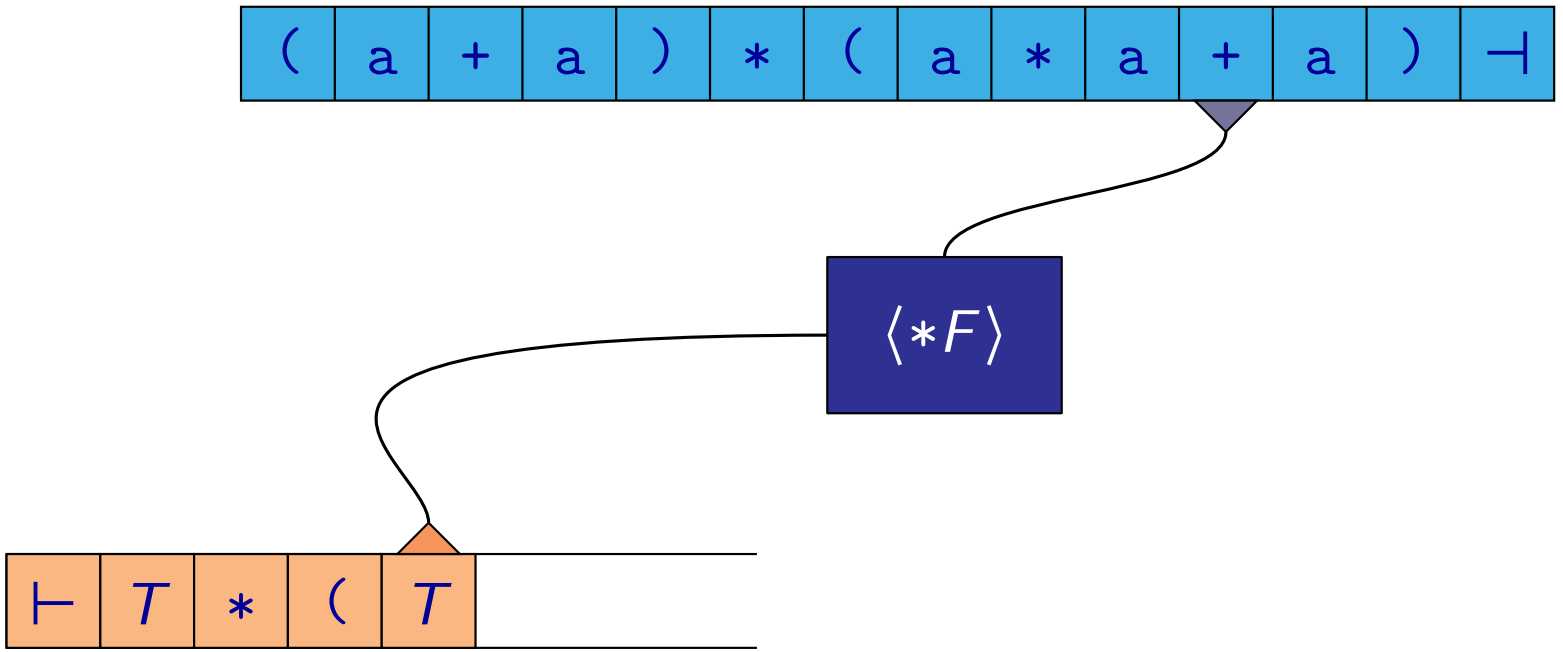
$T*(T*\underline{F}+a)\dashv \Rightarrow T*(\underline{T}*a+a)\dashv \Rightarrow T*(\underline{F}*a+a)\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



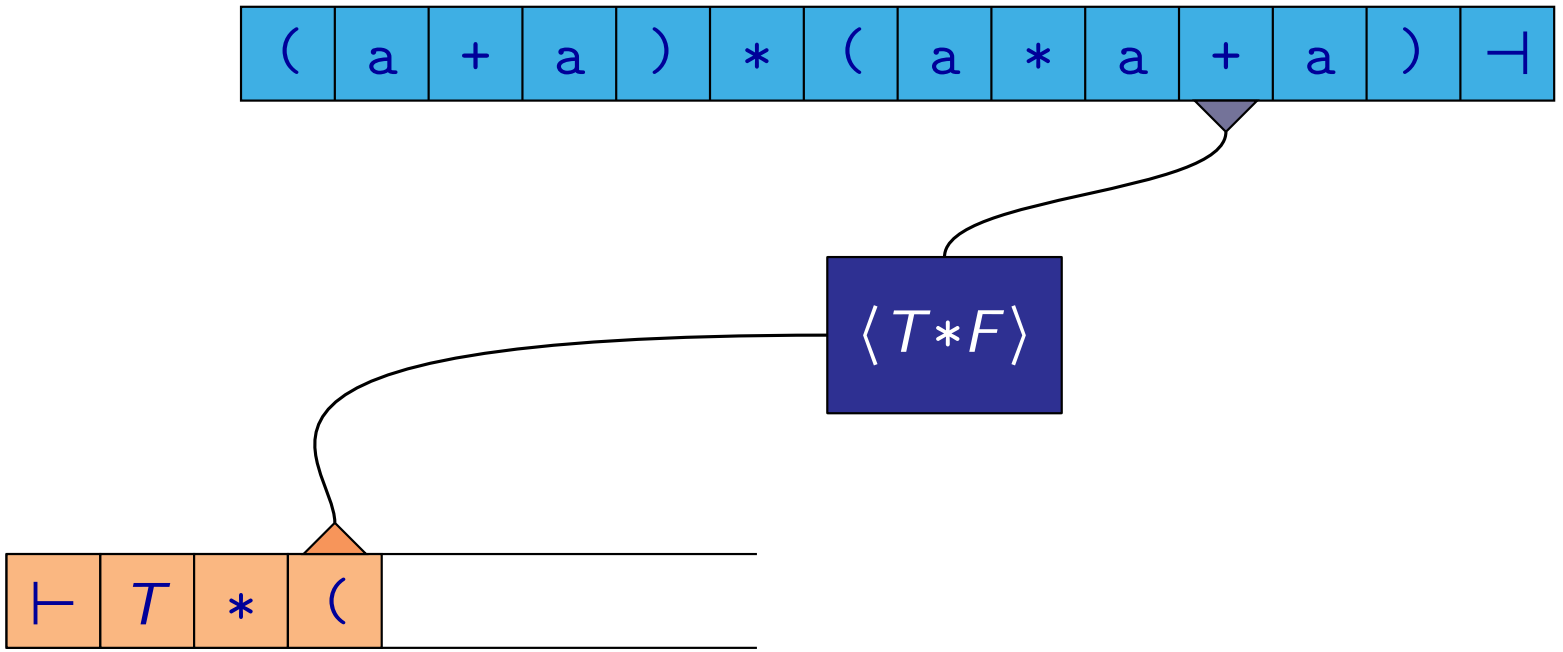
$T*(T*\underline{F}+\mathtt{a})\dashv \;\Rightarrow\; T*(\underline{T}*\mathtt{a}+\mathtt{a})\dashv \;\Rightarrow\; T*(\underline{F}*\mathtt{a}+\mathtt{a})\dashv \;\Rightarrow\; \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



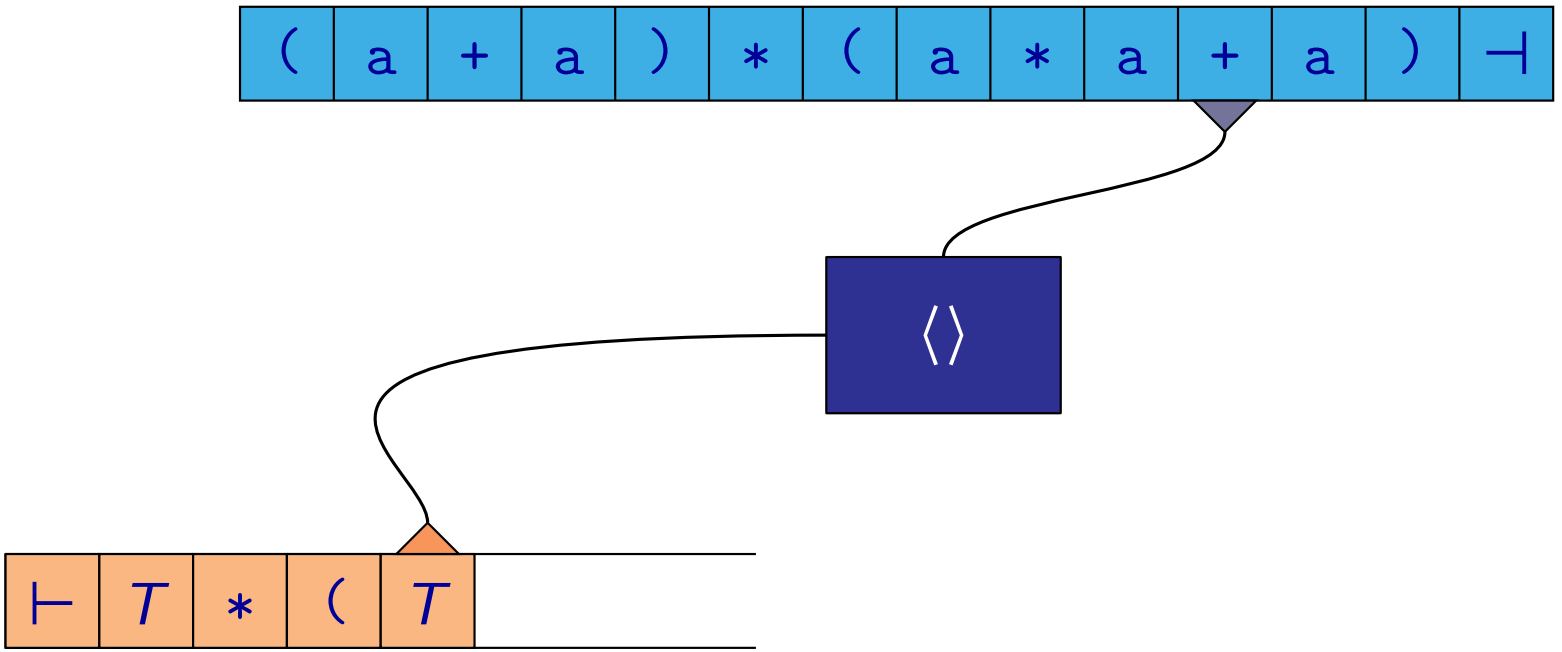
$T*(T*\underline{F}+\mathtt{a})\dashv \;\Rightarrow\; T*(\underline{T}*\mathtt{a}+\mathtt{a})\dashv \;\Rightarrow\; T*(\underline{F}*\mathtt{a}+\mathtt{a})\dashv \;\Rightarrow\; \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



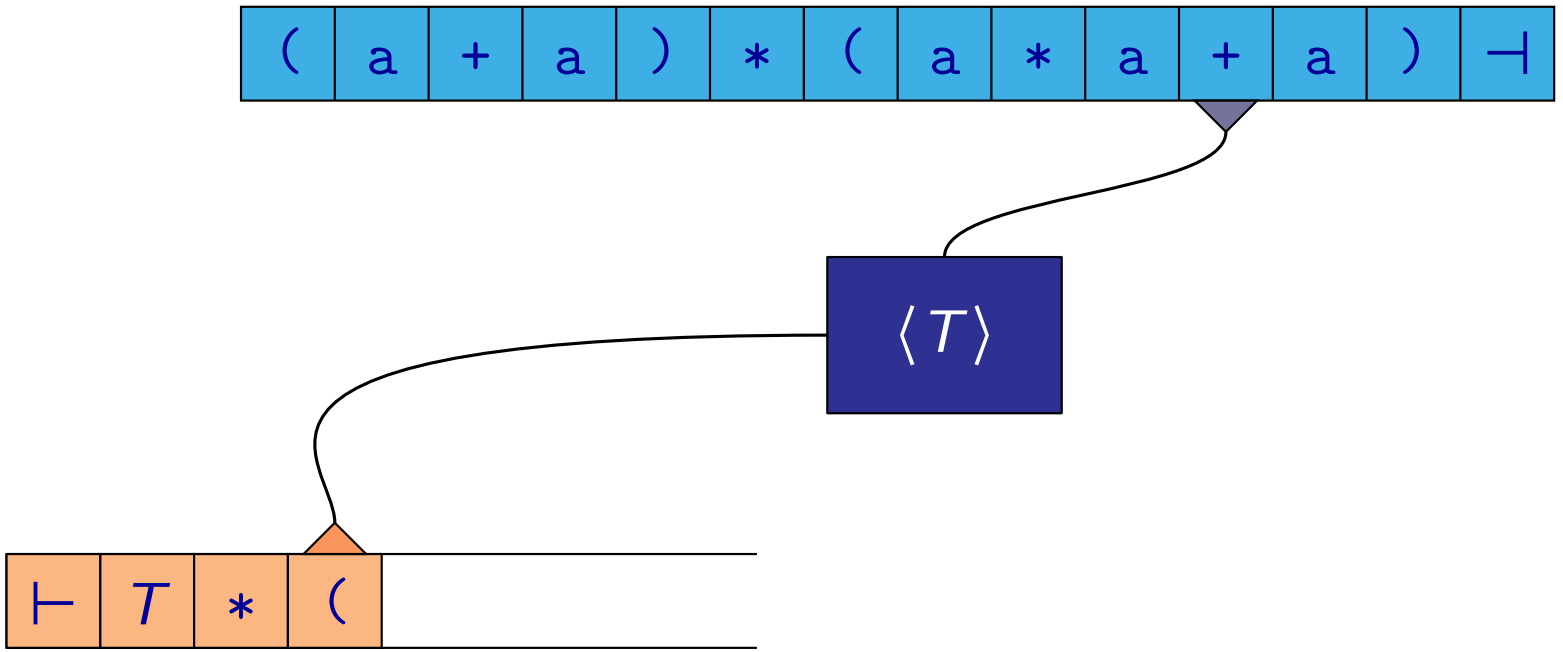
$T*(T*\underline{F}+\mathtt{a})\dashv \;\Rightarrow\; T*(\underline{T}*\mathtt{a}+\mathtt{a})\dashv \;\Rightarrow\; T*(\underline{F}*\mathtt{a}+\mathtt{a})\dashv \;\Rightarrow\; \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



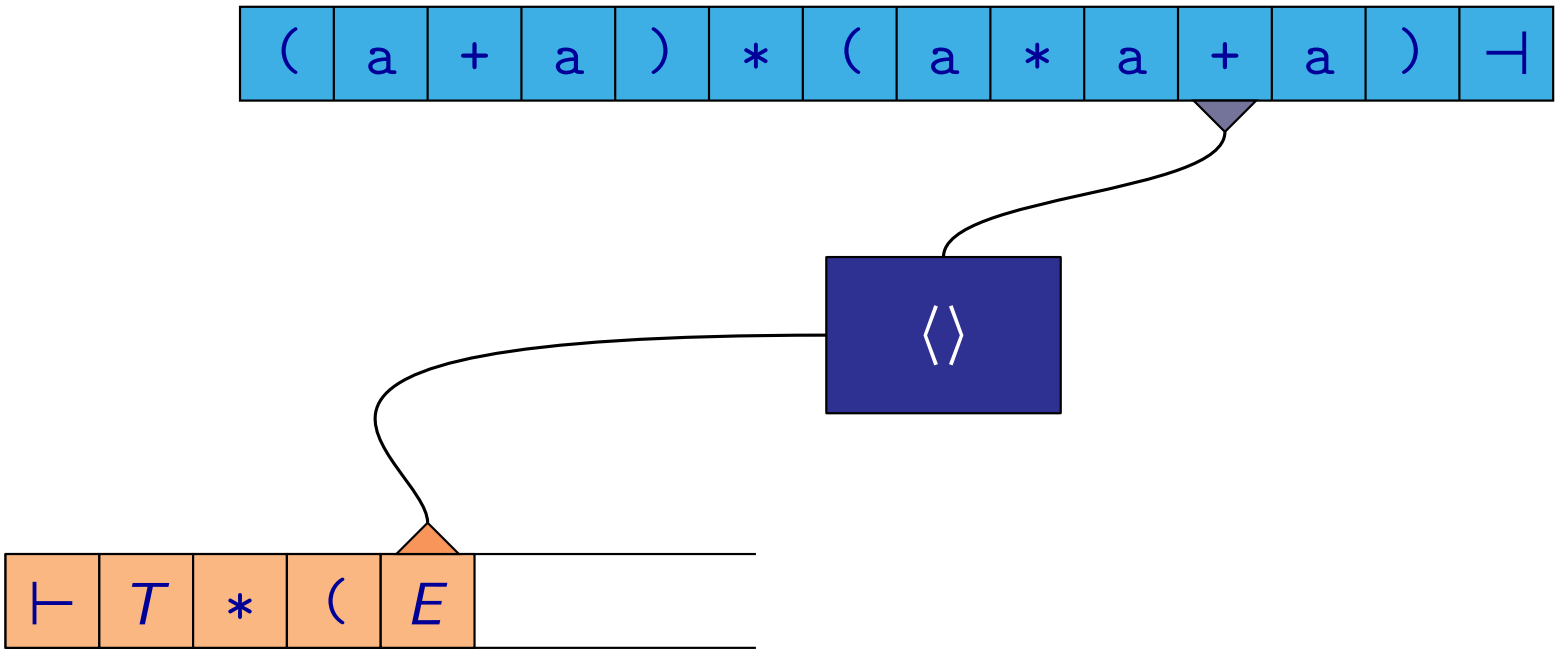
$T*(\underline{T}+\mathtt{a})\dashv \;\Rightarrow\; T*(T*\underline{F}+\mathtt{a})\dashv \;\Rightarrow\; T*(\underline{T}*\mathtt{a}+\mathtt{a})\dashv \;\Rightarrow\; \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



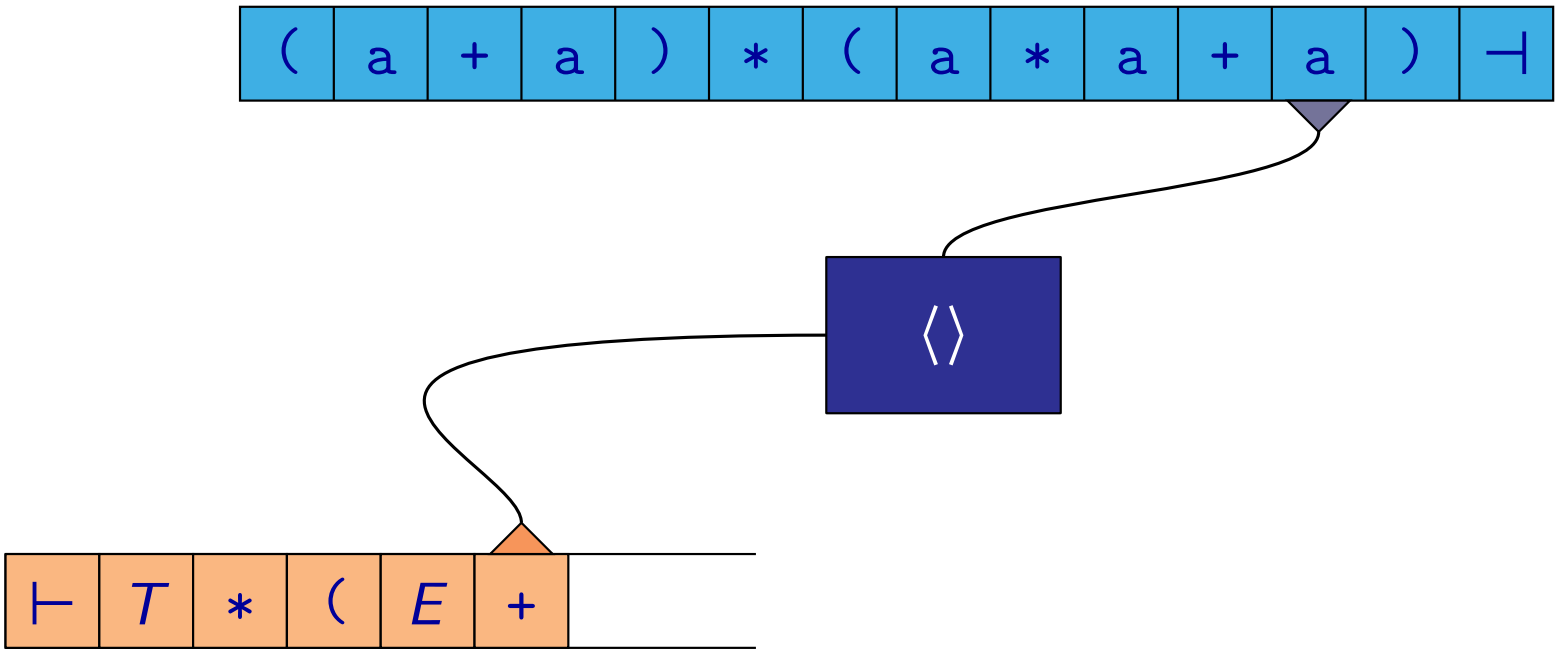
$T*(\underline{T}+\mathtt{a})\dashv \;\Rightarrow\; T*(T*\underline{F}+\mathtt{a})\dashv \;\Rightarrow\; T*(\underline{T}*\mathtt{a}+\mathtt{a})\dashv \;\Rightarrow\; \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



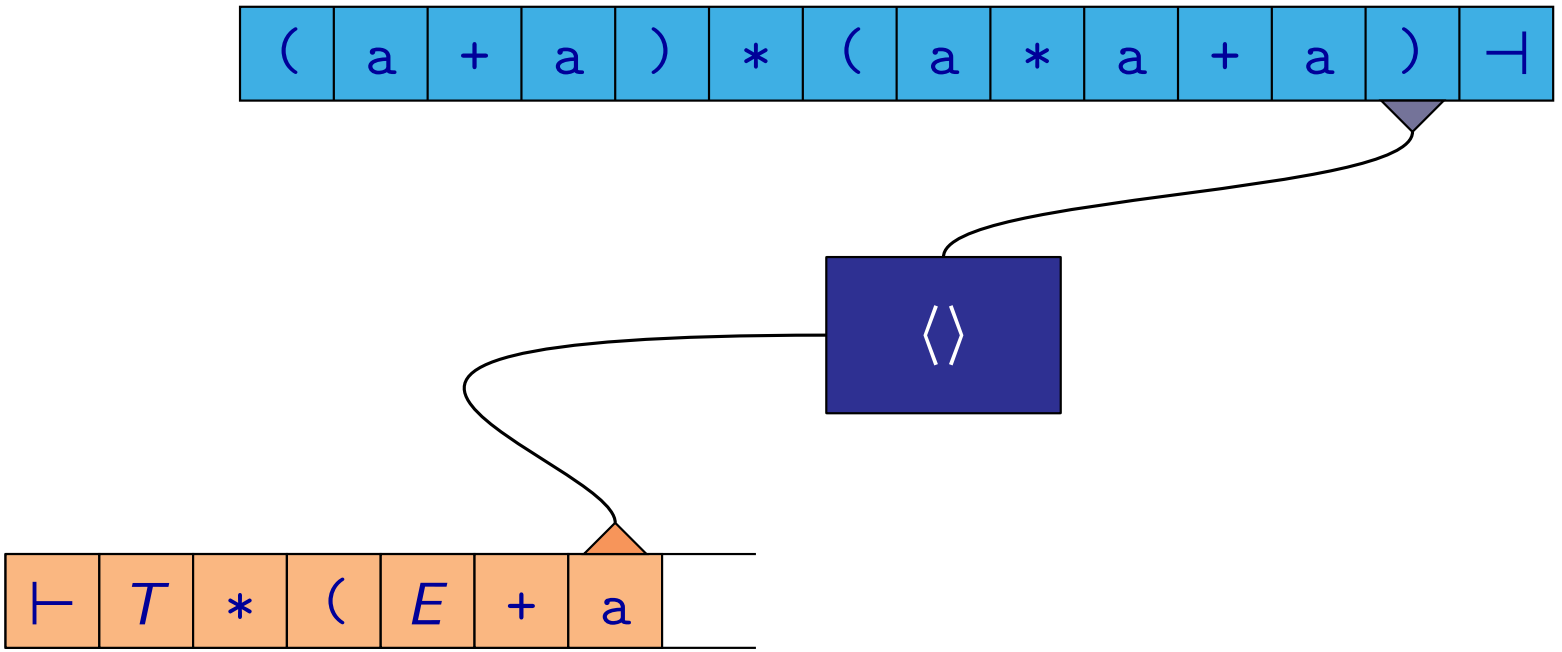
$T*(\underline{E}+\mathtt{a})\dashv \;\Rightarrow\; T*(\underline{T}+\mathtt{a})\dashv \;\Rightarrow\; T*(T*\underline{F}+\mathtt{a})\dashv \;\Rightarrow\; \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



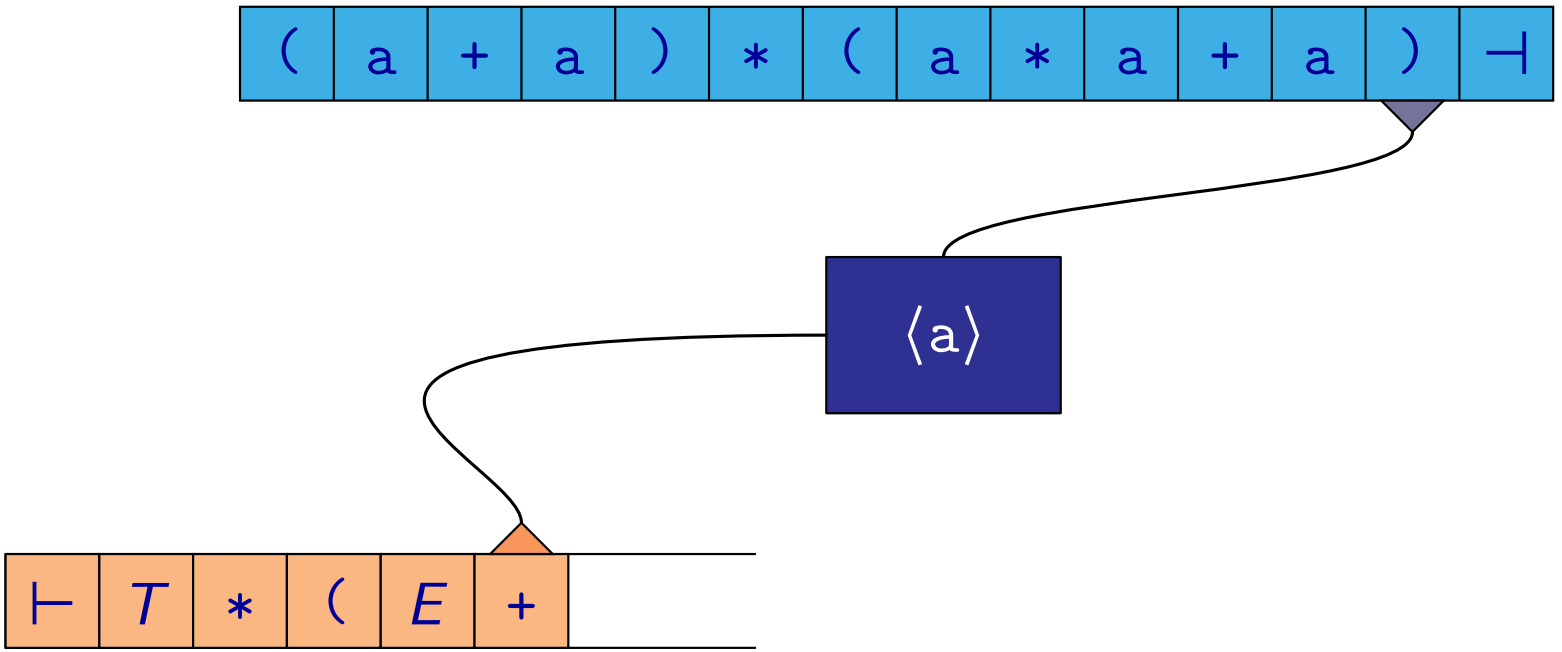
$T*(\underline{E}+\mathtt{a})\dashv \;\Rightarrow\; T*(\underline{T}+\mathtt{a})\dashv \;\Rightarrow\; T*(T*\underline{F}+\mathtt{a})\dashv \;\Rightarrow\; \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



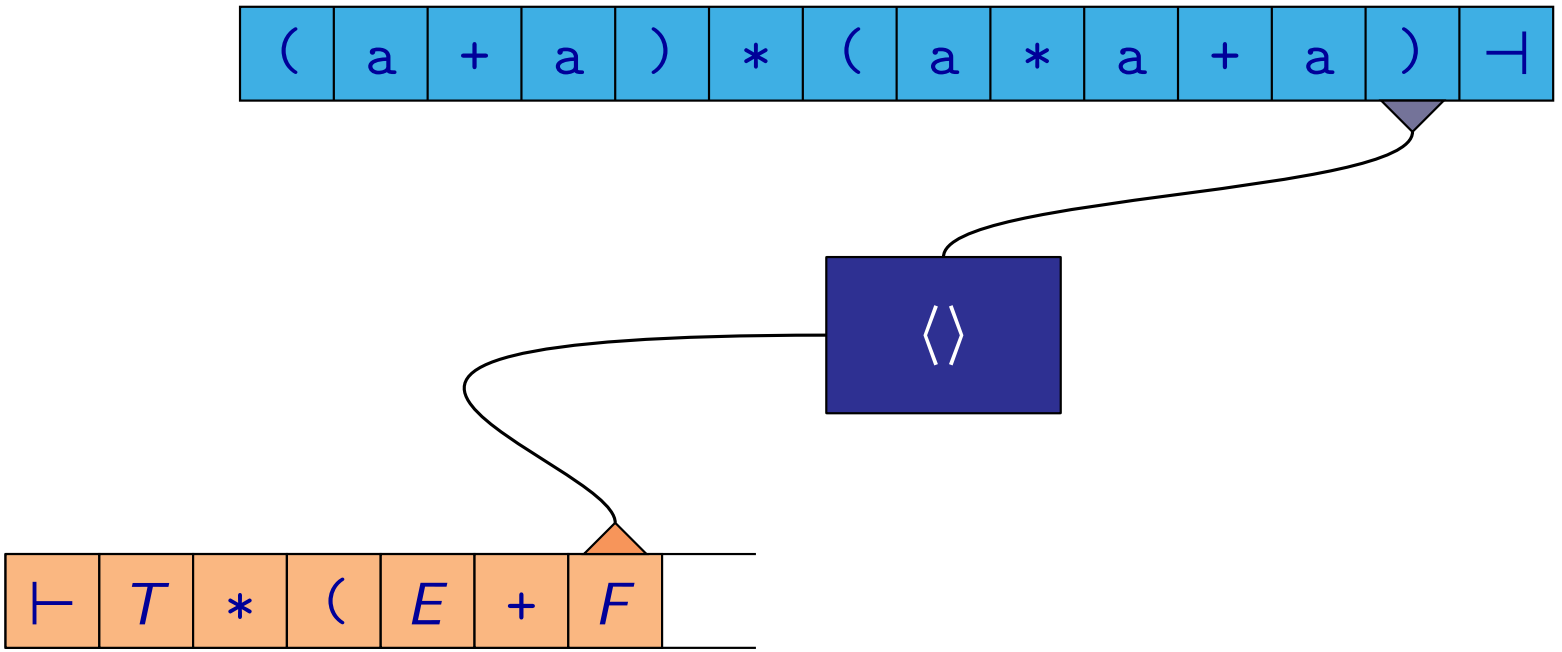
$T*(\underline{E}+\mathtt{a})\dashv \;\Rightarrow\; T*(\underline{T}+\mathtt{a})\dashv \;\Rightarrow\; T*(T*\underline{F}+\mathtt{a})\dashv \;\Rightarrow\; \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



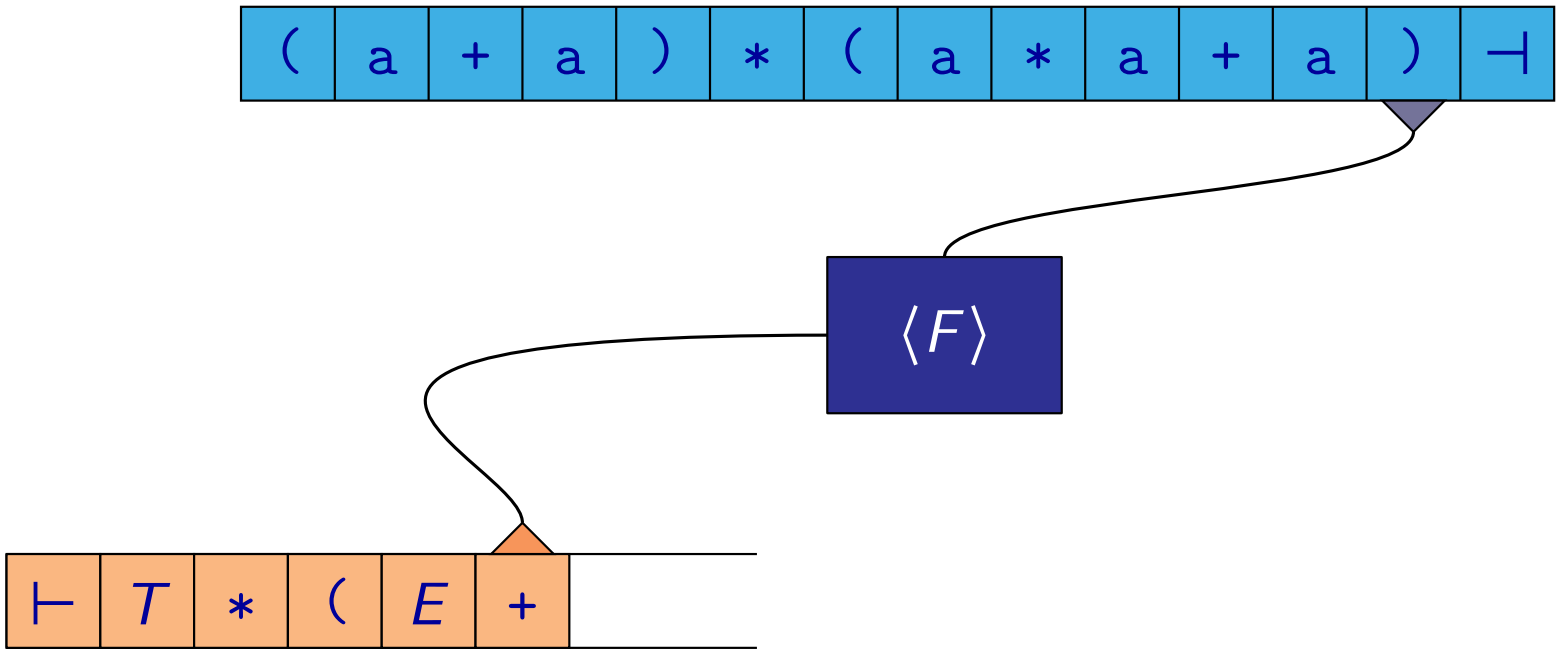
$T*(\underline{E}+\mathtt{a})\dashv \;\Rightarrow\; T*(\underline{T}+\mathtt{a})\dashv \;\Rightarrow\; T*(T*\underline{F}+\mathtt{a})\dashv \;\Rightarrow\; \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



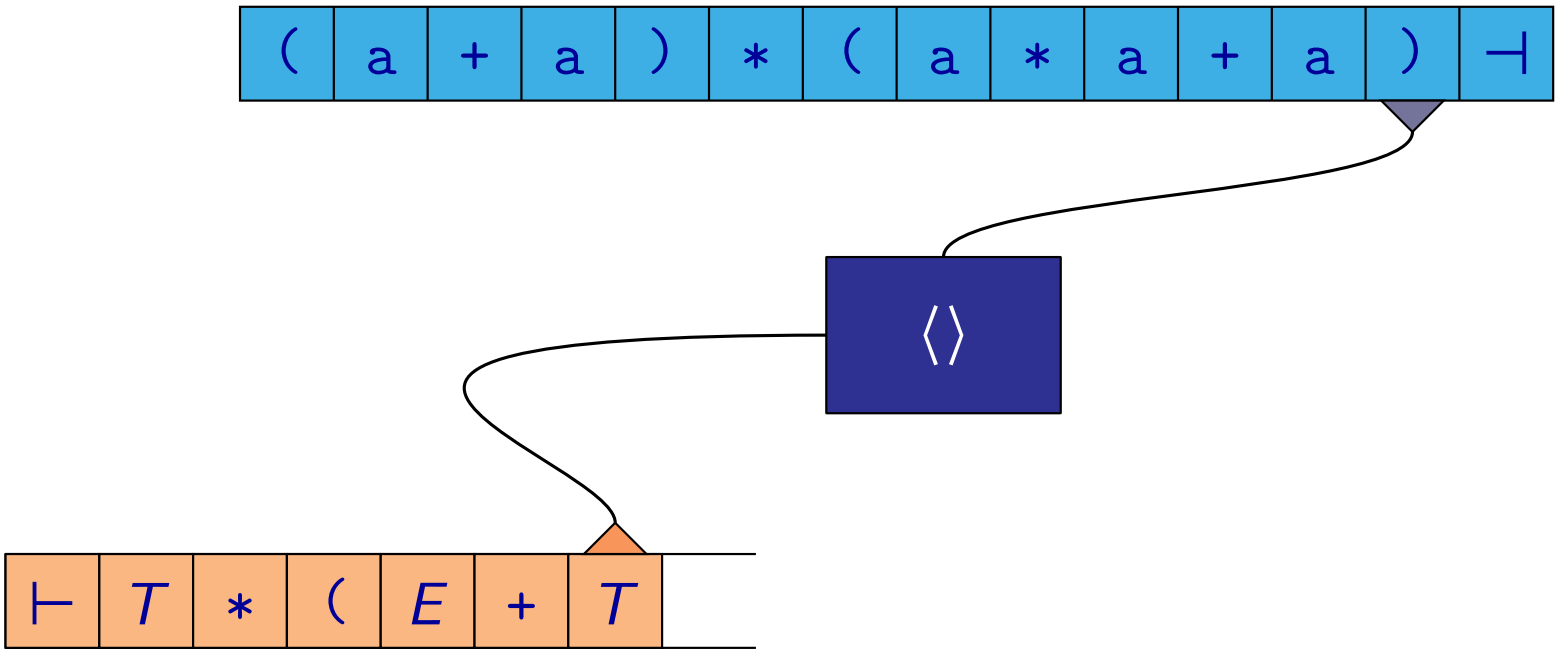
$T*(E+\underline{F})\dashv \Rightarrow T*(\underline{E}+\mathtt{a})\dashv \Rightarrow T*(\underline{T}+\mathtt{a})\dashv \Rightarrow T*(T*\underline{F}+\mathtt{a})\dashv \Rightarrow$
$\cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



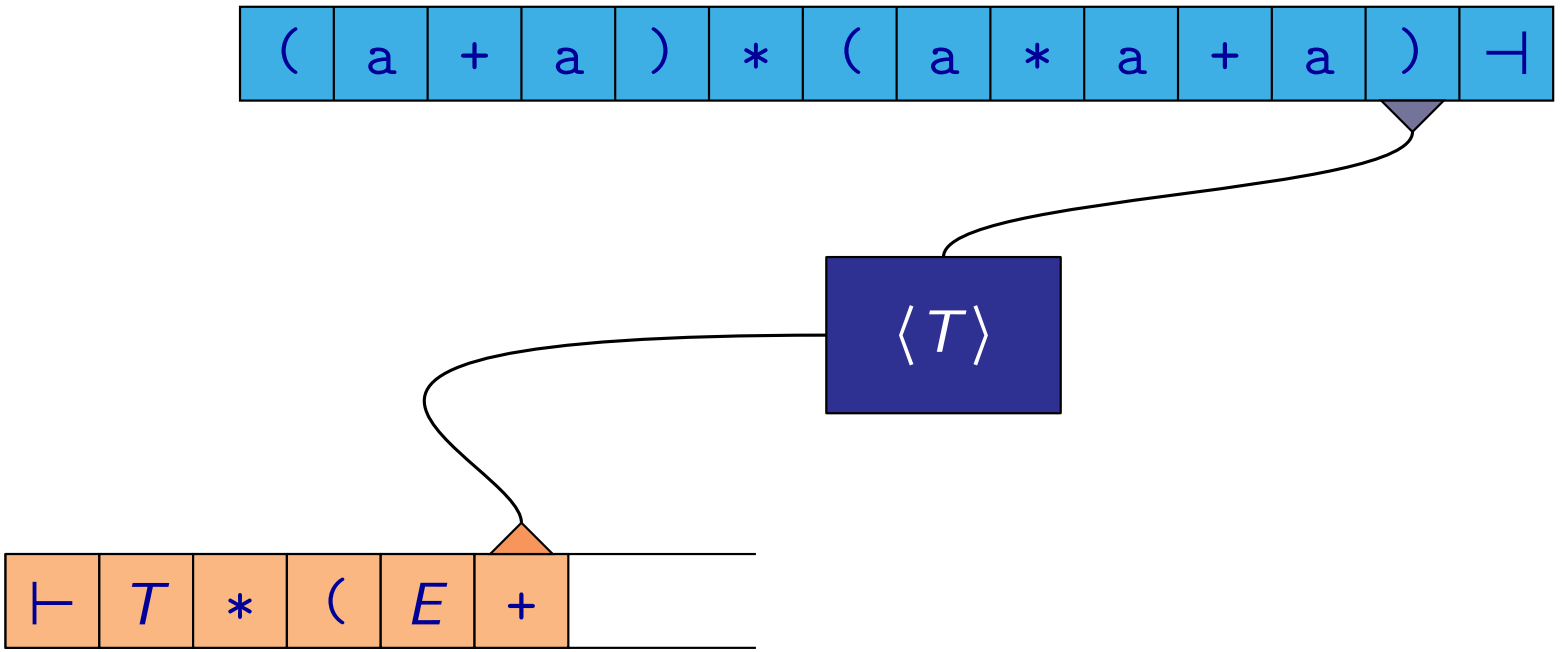
$T*(E+\underline{F})\dashv \Rightarrow T*(\underline{E}+\mathtt{a})\dashv \Rightarrow T*(\underline{T}+\mathtt{a})\dashv \Rightarrow T*(T*\underline{F}+\mathtt{a})\dashv \Rightarrow$
$\cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



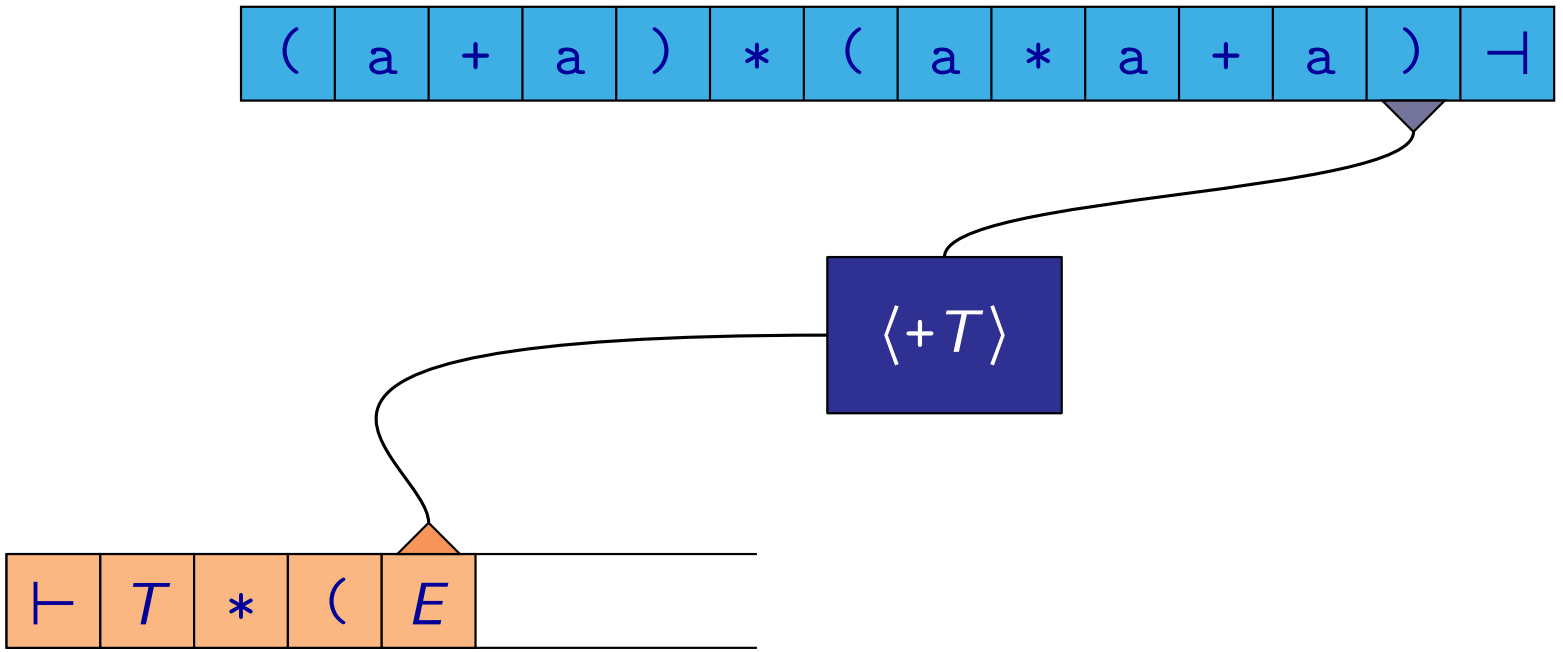
$T*(E+\underline{T})\dashv \Rightarrow T*(E+\underline{F})\dashv \Rightarrow T*(\underline{E}+\mathtt{a})\dashv \Rightarrow T*(\underline{T}+\mathtt{a})\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



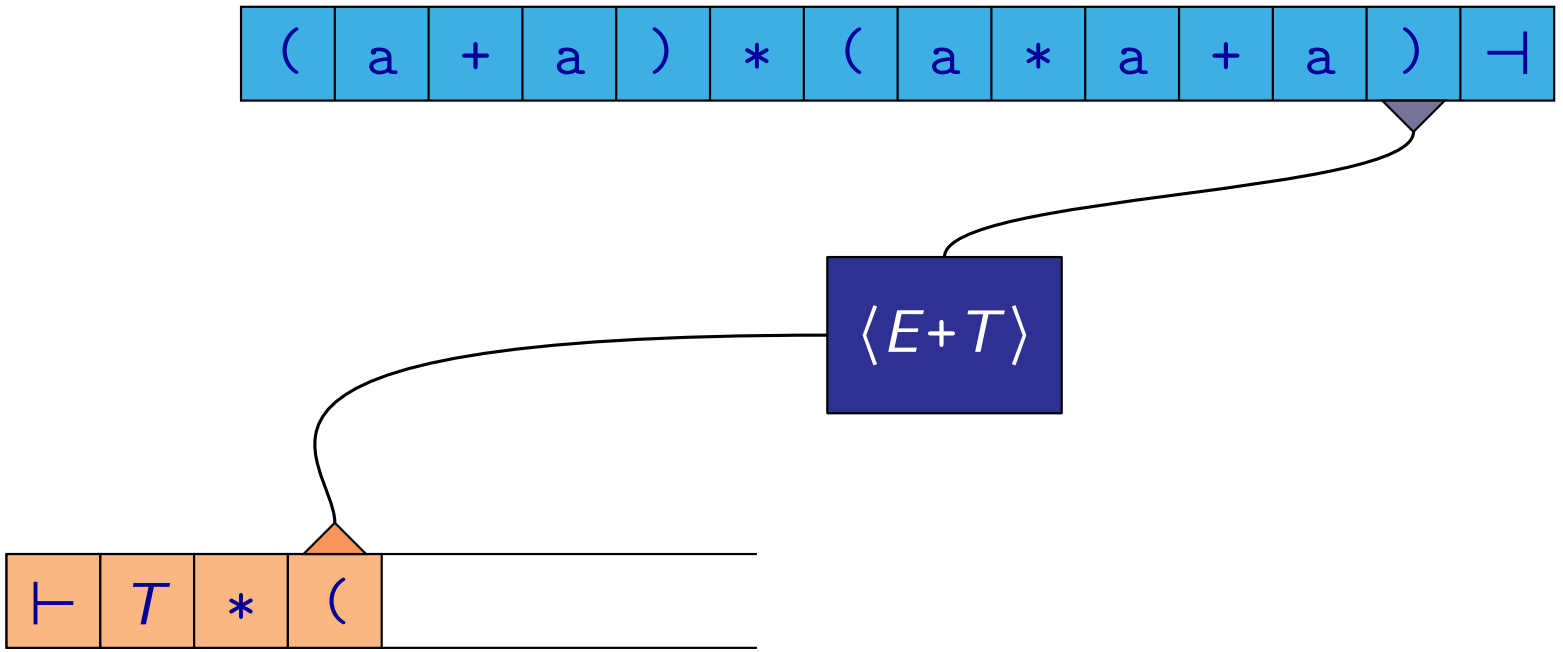
$$T*(E+\underline{T})\dashv \Rightarrow T*(E+\underline{F})\dashv \Rightarrow T*(\underline{E}+\mathtt{a})\dashv \Rightarrow T*(\underline{T}+\mathtt{a})\dashv \Rightarrow \cdots$$

Ekvivalence bezkontextových gramatik a zás. automatů



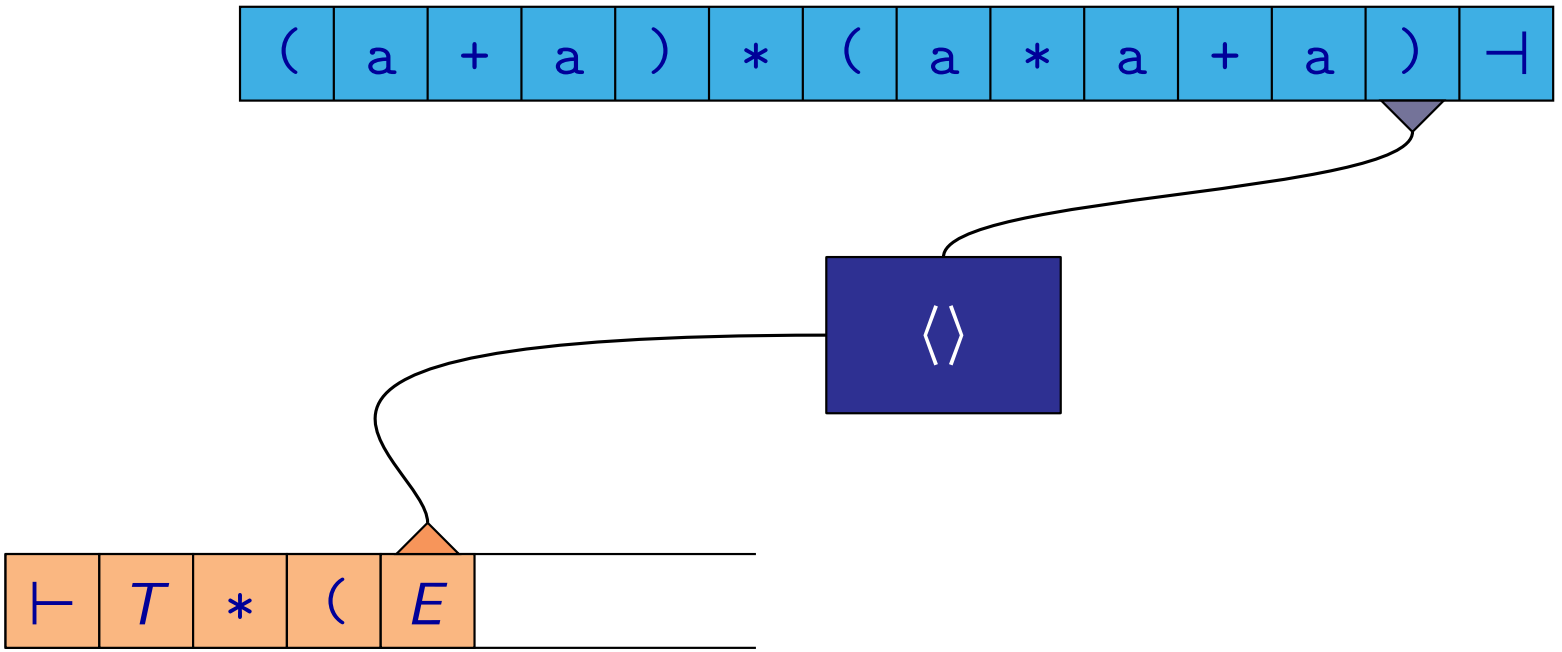
$T*(E+\underline{T})\dashv \Rightarrow T*(E+\underline{F})\dashv \Rightarrow T*(\underline{E}+\mathtt{a})\dashv \Rightarrow T*(\underline{T}+\mathtt{a})\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



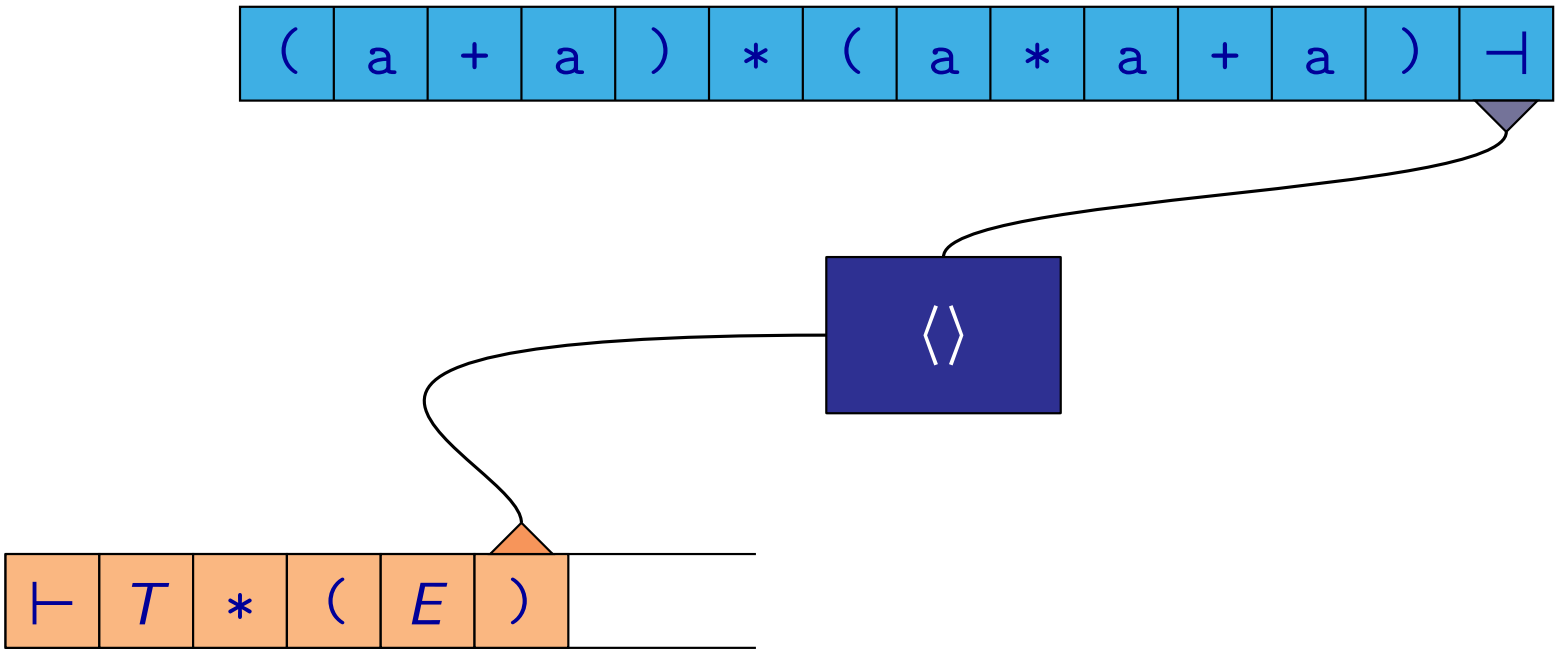
$T*(E+\underline{T})\dashv \Rightarrow T*(E+\underline{F})\dashv \Rightarrow T*(\underline{E}+\mathtt{a})\dashv \Rightarrow T*(\underline{T}+\mathtt{a})\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



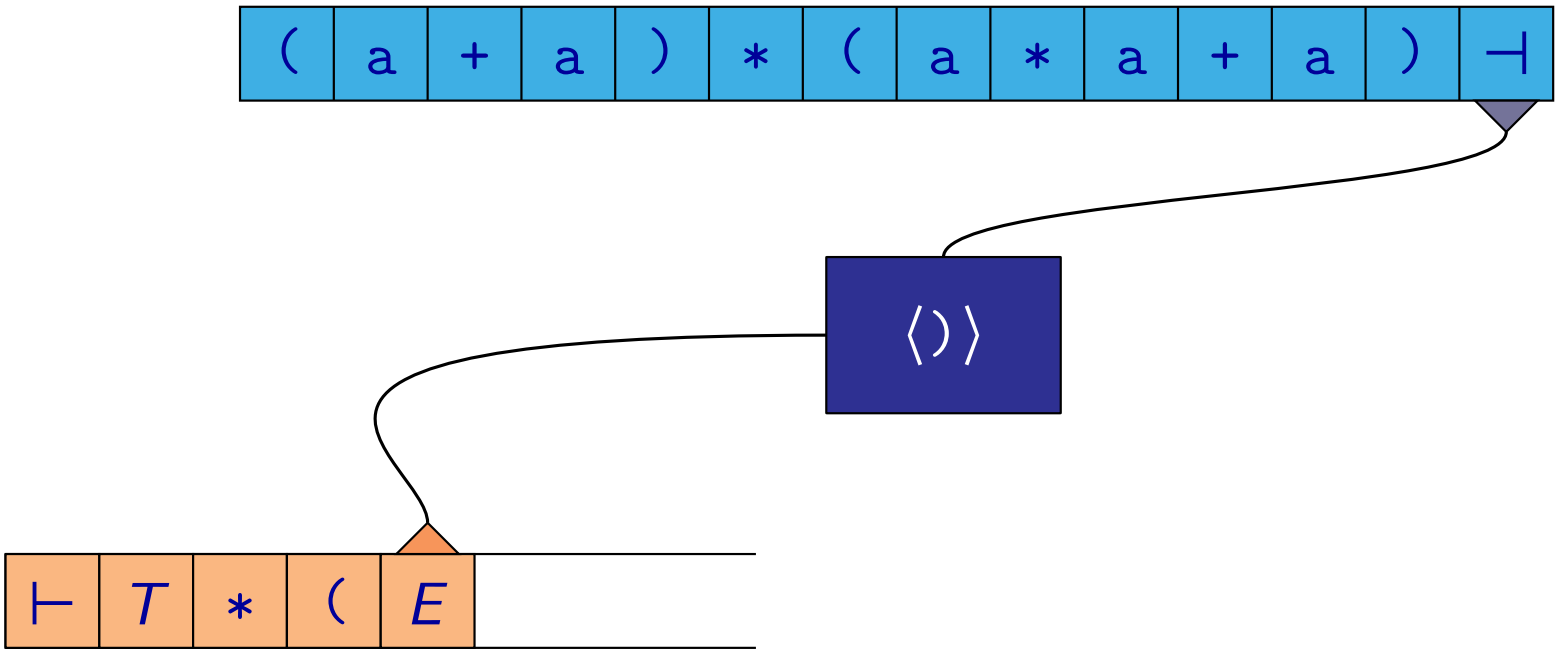
$$T*(\underline{E})\dashv \;\Rightarrow\; T*(E+\underline{T})\dashv \;\Rightarrow\; T*(E+\underline{F})\dashv \;\Rightarrow\; T*(\underline{E}+\mathtt{a})\dashv \;\Rightarrow\; \cdots$$

Ekvivalence bezkontextových gramatik a zás. automatů



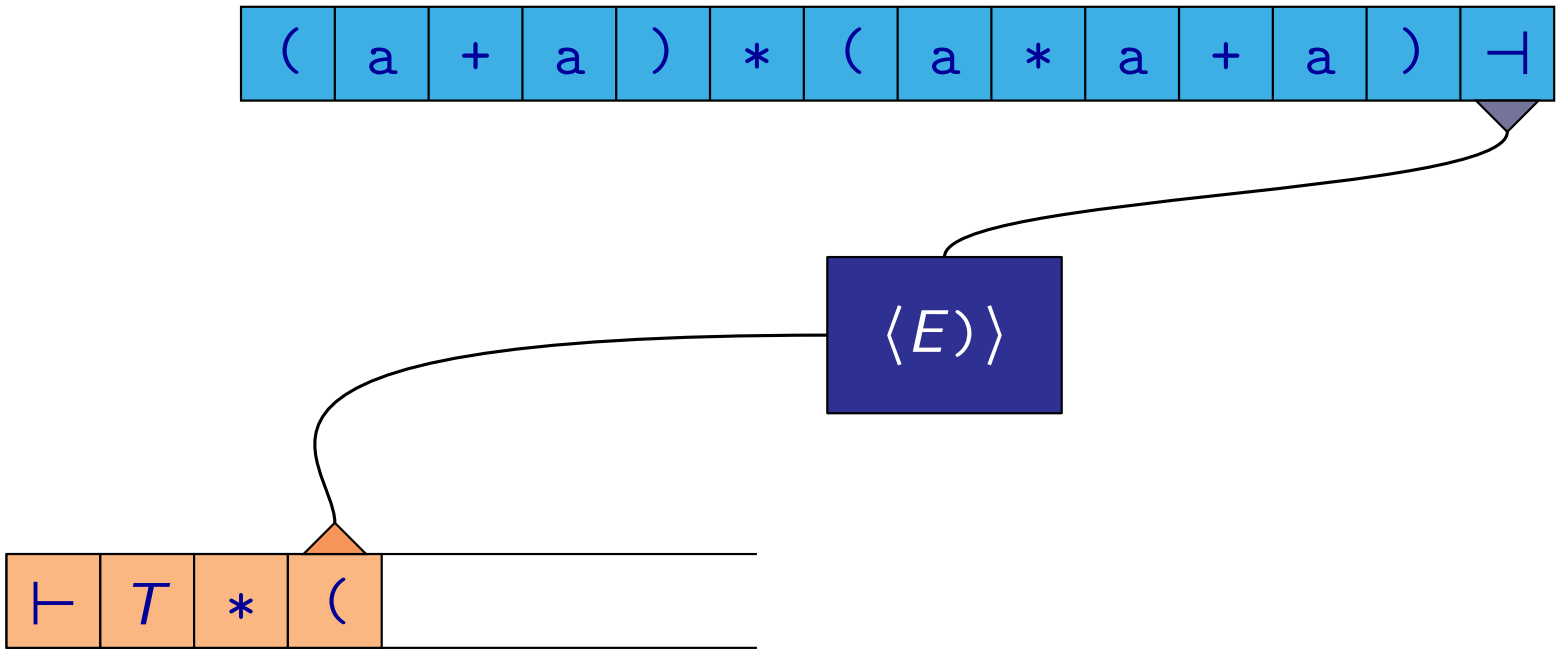
$T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow T*(E+\underline{F})\dashv \Rightarrow T*(\underline{E}+\mathrm{a})\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



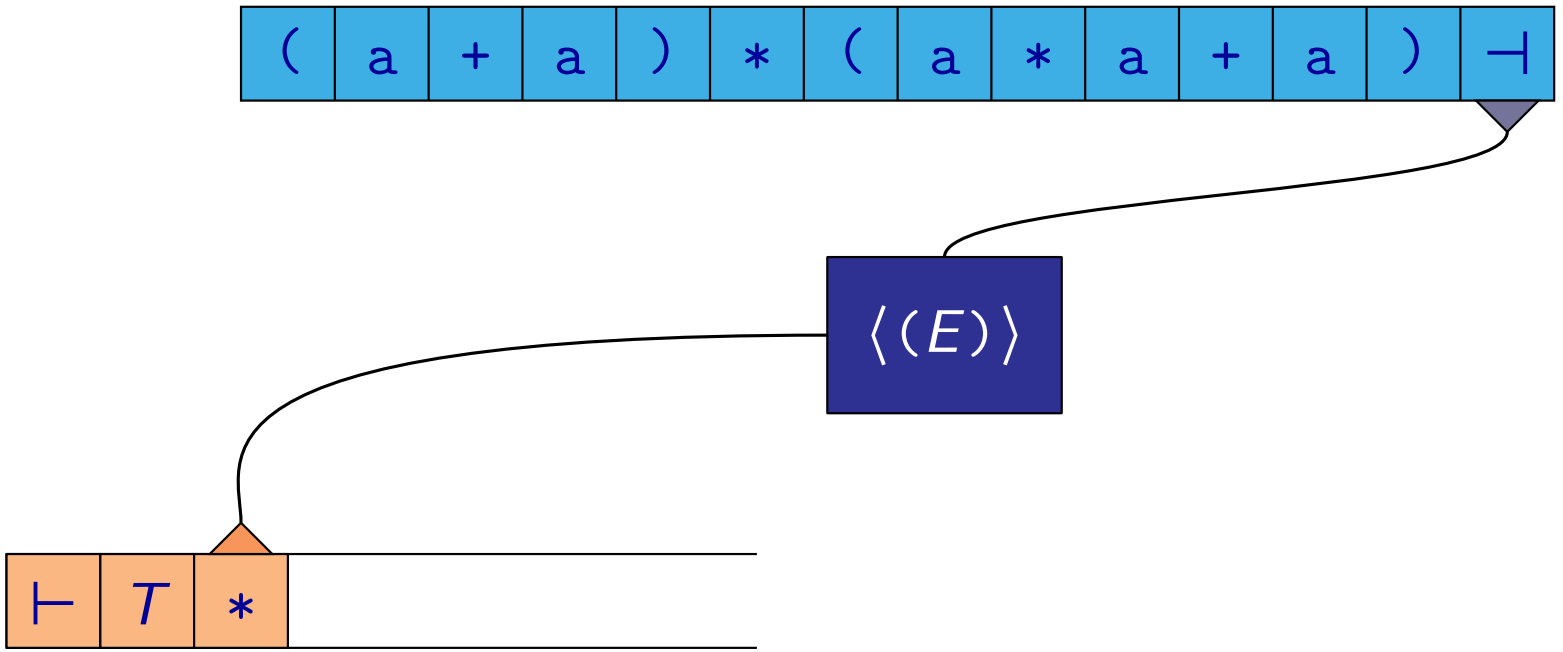
$T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow T*(E+\underline{F})\dashv \Rightarrow T*(\underline{E}+\mathtt{a})\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



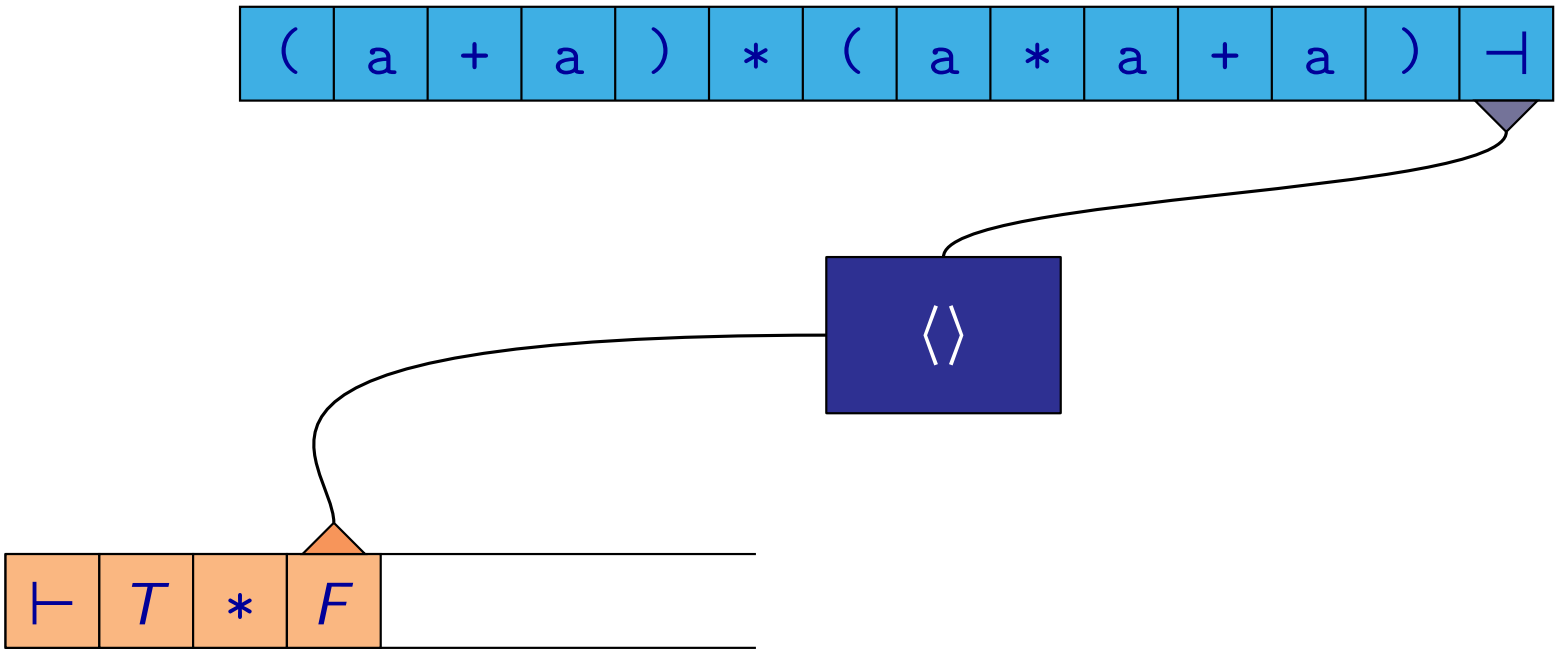
$$T*(\underline{E})\dashv \;\Rightarrow\; T*(E+\underline{T})\dashv \;\Rightarrow\; T*(E+\underline{F})\dashv \;\Rightarrow\; T*(\underline{E}+\mathtt{a})\dashv \;\Rightarrow\; \cdots$$

Ekvivalence bezkontextových gramatik a zás. automatů



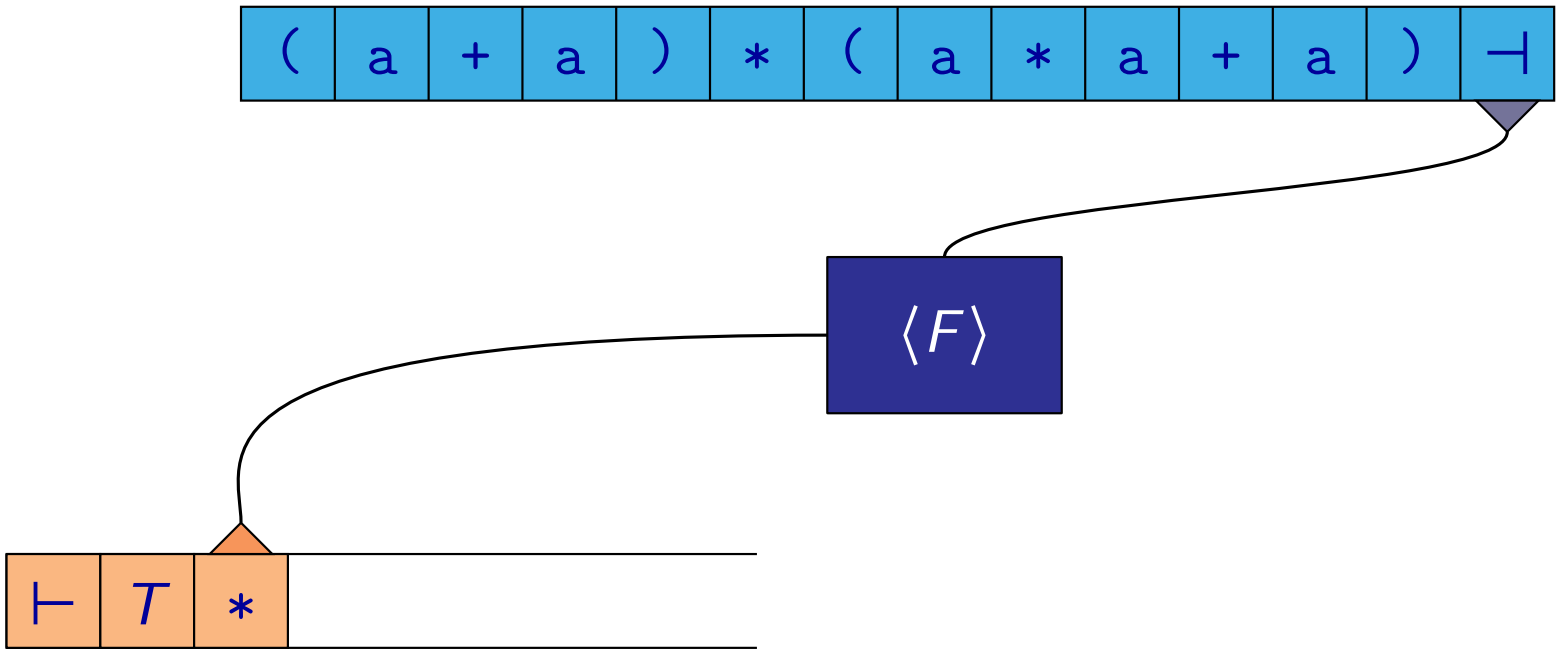
$T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow T*(E+\underline{F})\dashv \Rightarrow T*(\underline{E}+\mathtt{a})\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



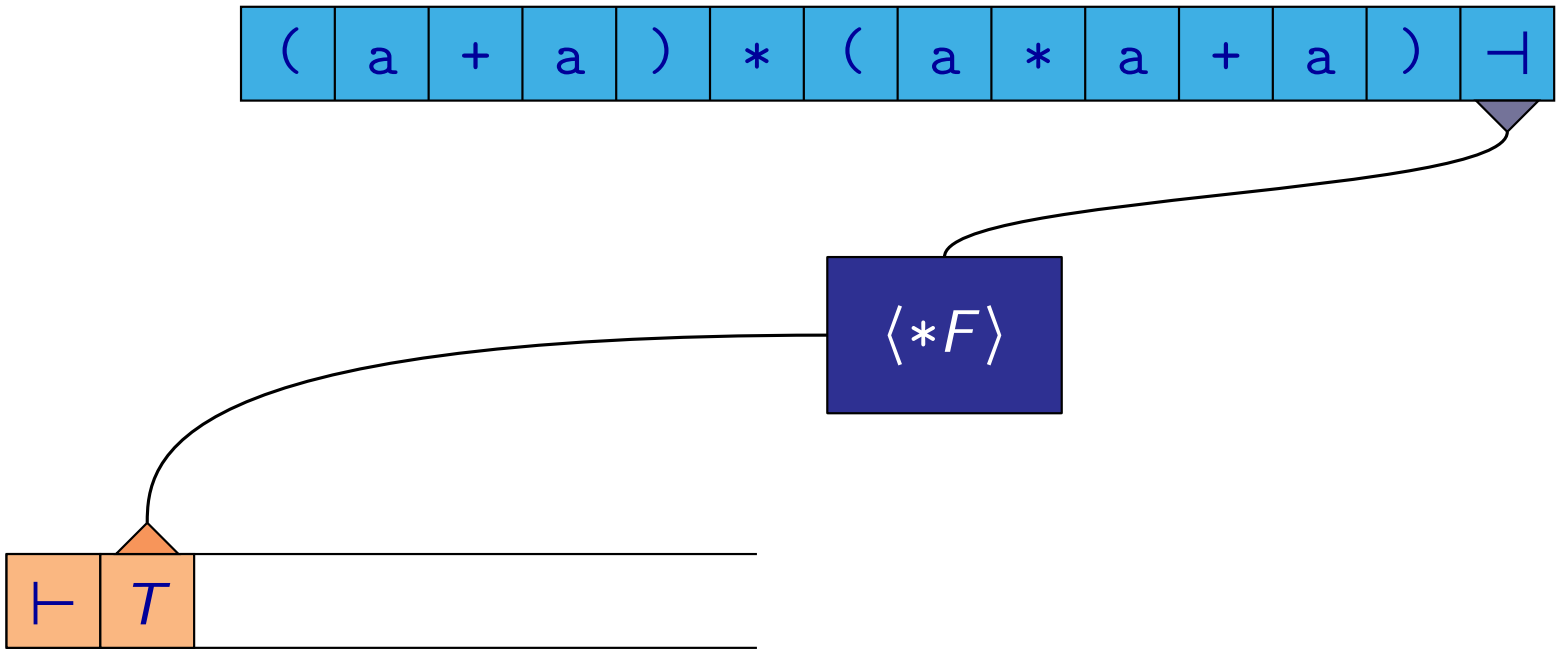
$T*\underline{F}\dashv \Rightarrow T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow T*(E+\underline{F})\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



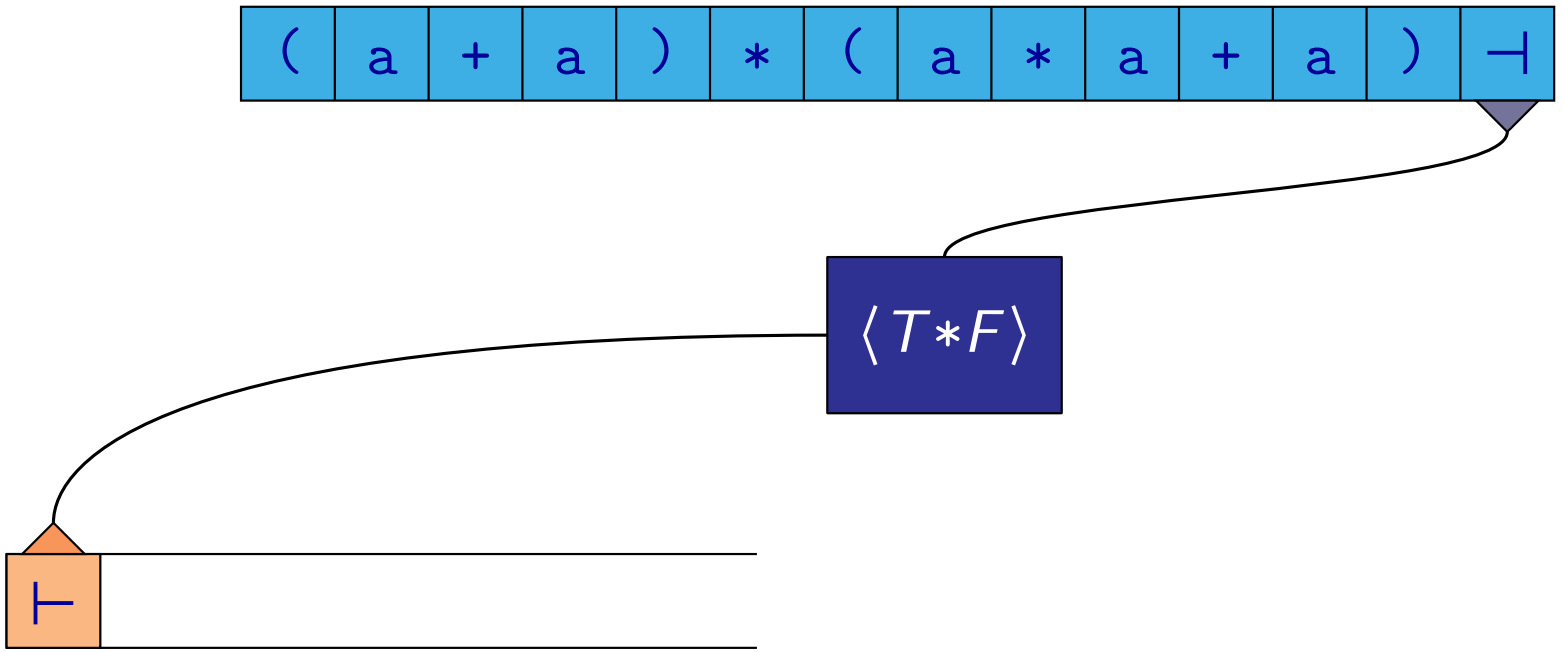
$T*\underline{F}\dashv \Rightarrow T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow T*(E+\underline{F})\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



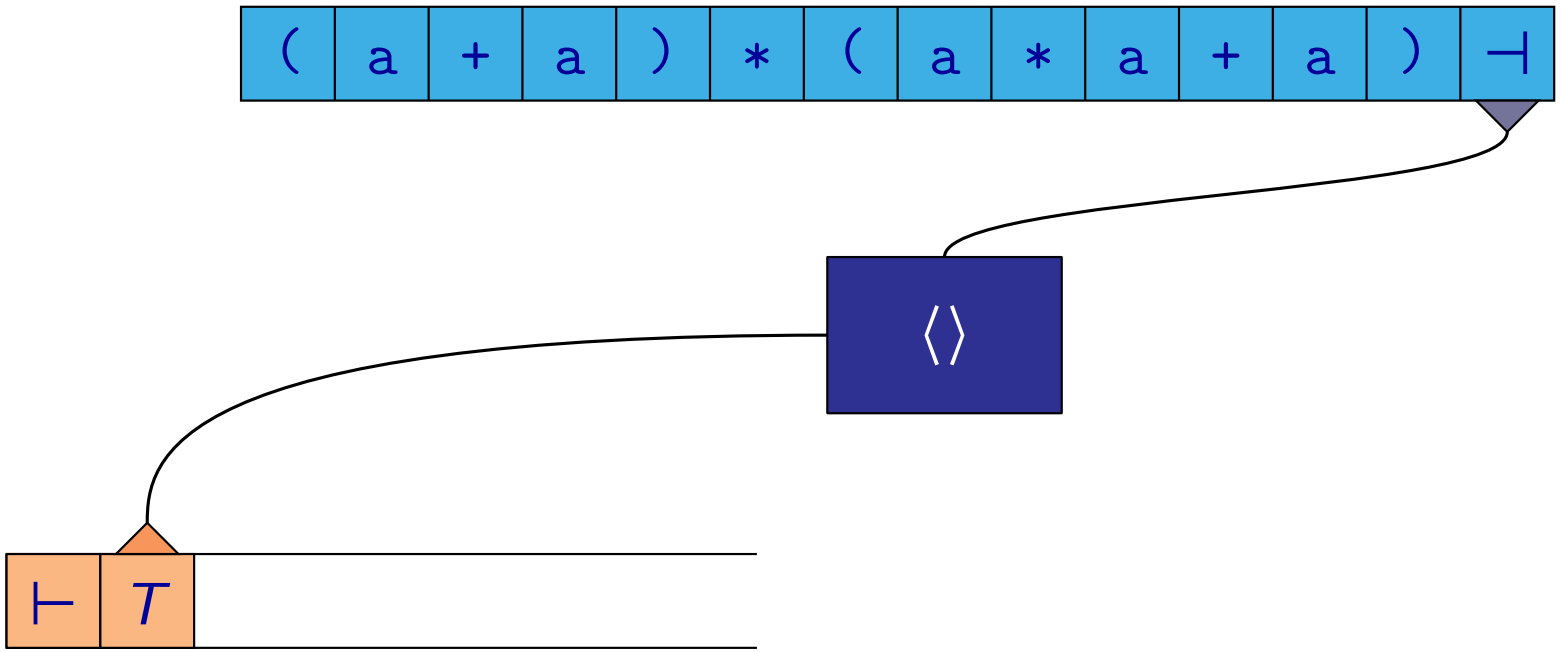
$T*\underline{F}\dashv \Rightarrow T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow T*(E+\underline{F})\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



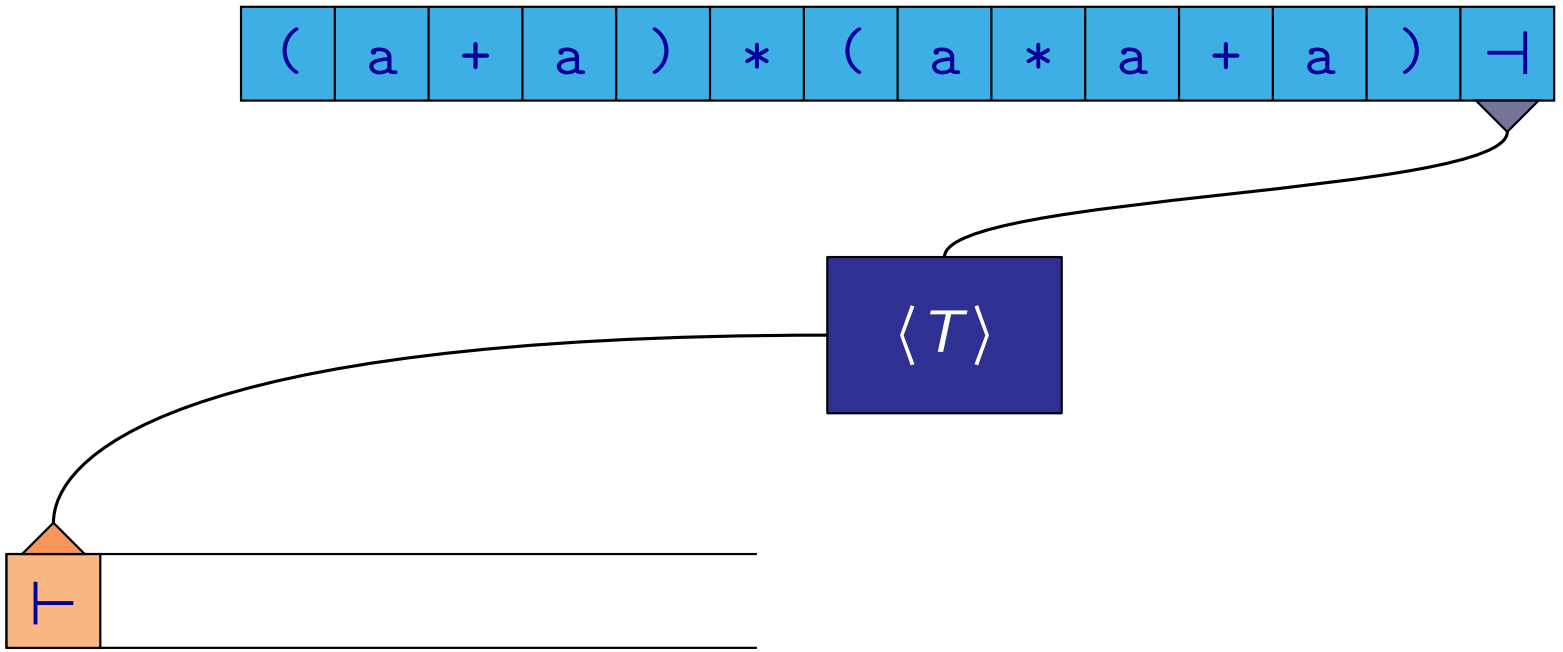
$T*\underline{F}\dashv \Rightarrow T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow T*(E+\underline{F})\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



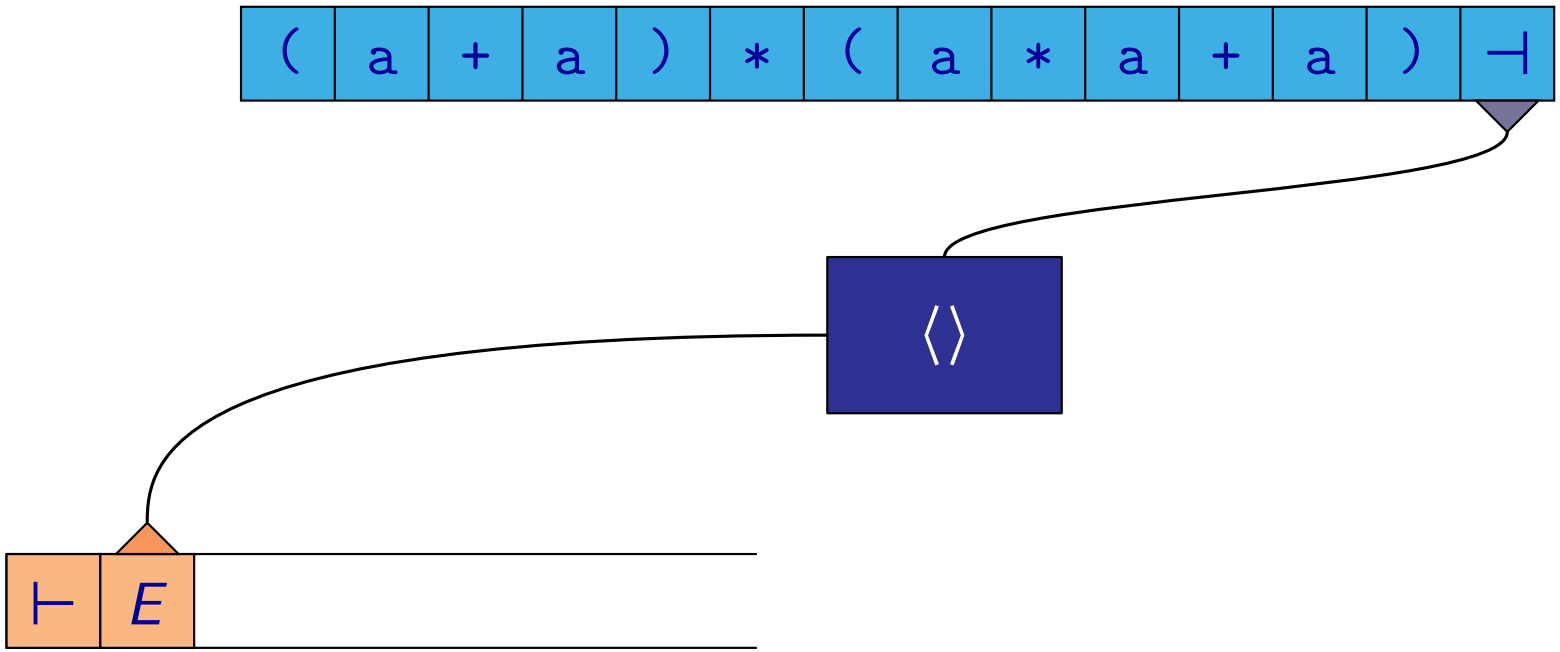
$\underline{T}\dashv \Rightarrow T*\underline{F}\dashv \Rightarrow T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow T*(E+\underline{F})\dashv \Rightarrow$
$\cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



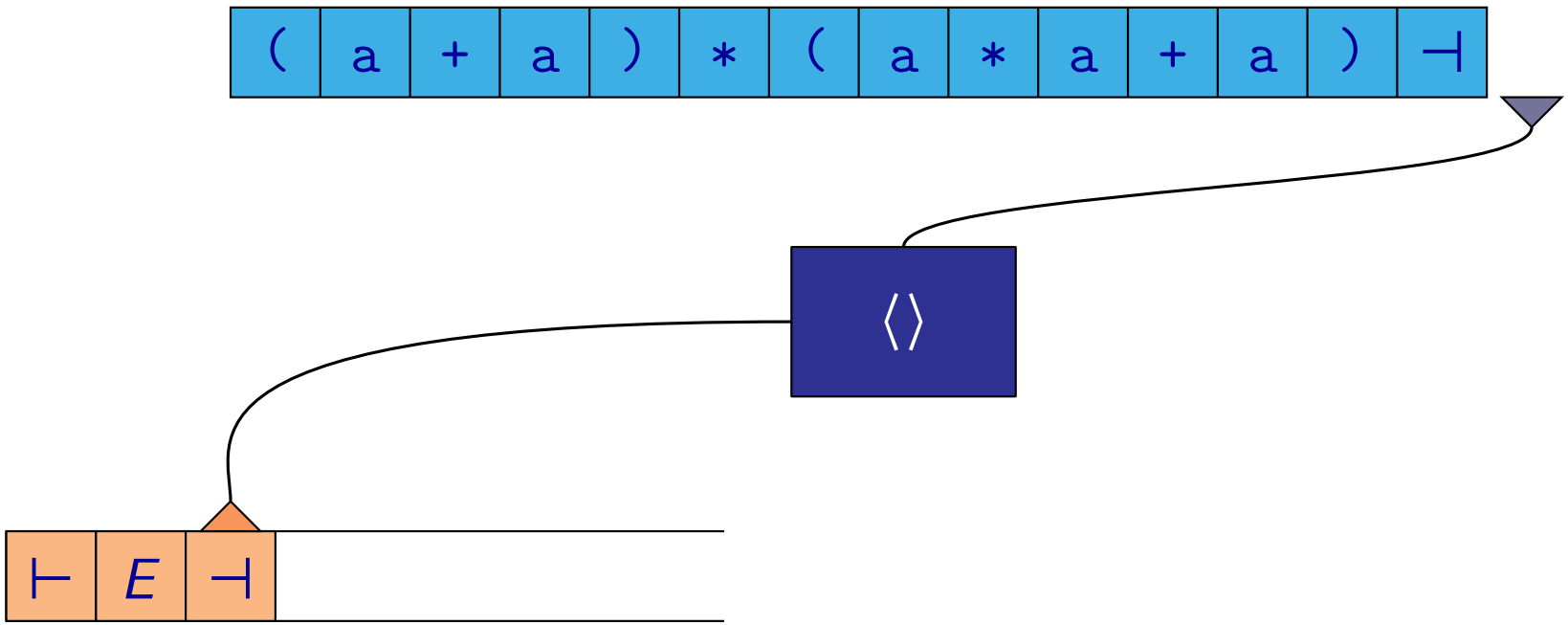
$\underline{T}\dashv \Rightarrow T*\underline{F}\dashv \Rightarrow T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow T*(E+\underline{F})\dashv \Rightarrow$
$\cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



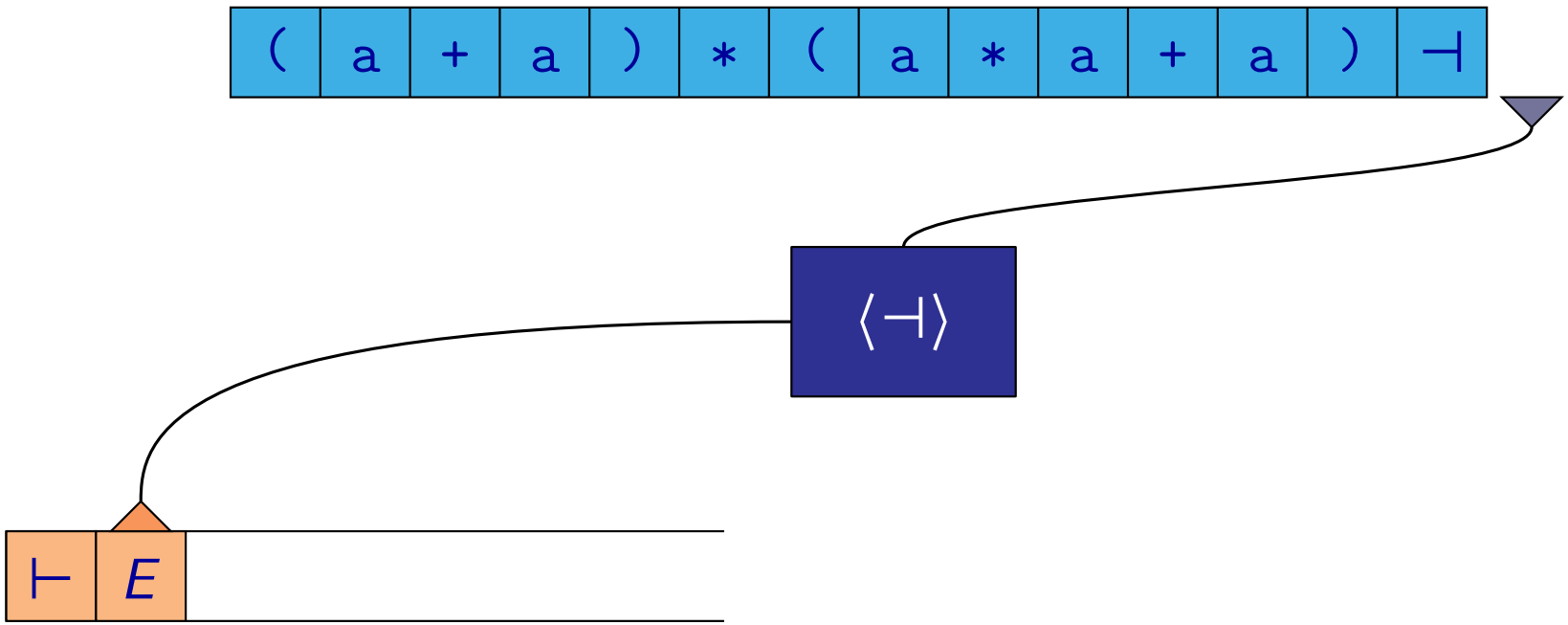
$\underline{E}\dashv \Rightarrow \underline{T}\dashv \Rightarrow T*\underline{F}\dashv \Rightarrow T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



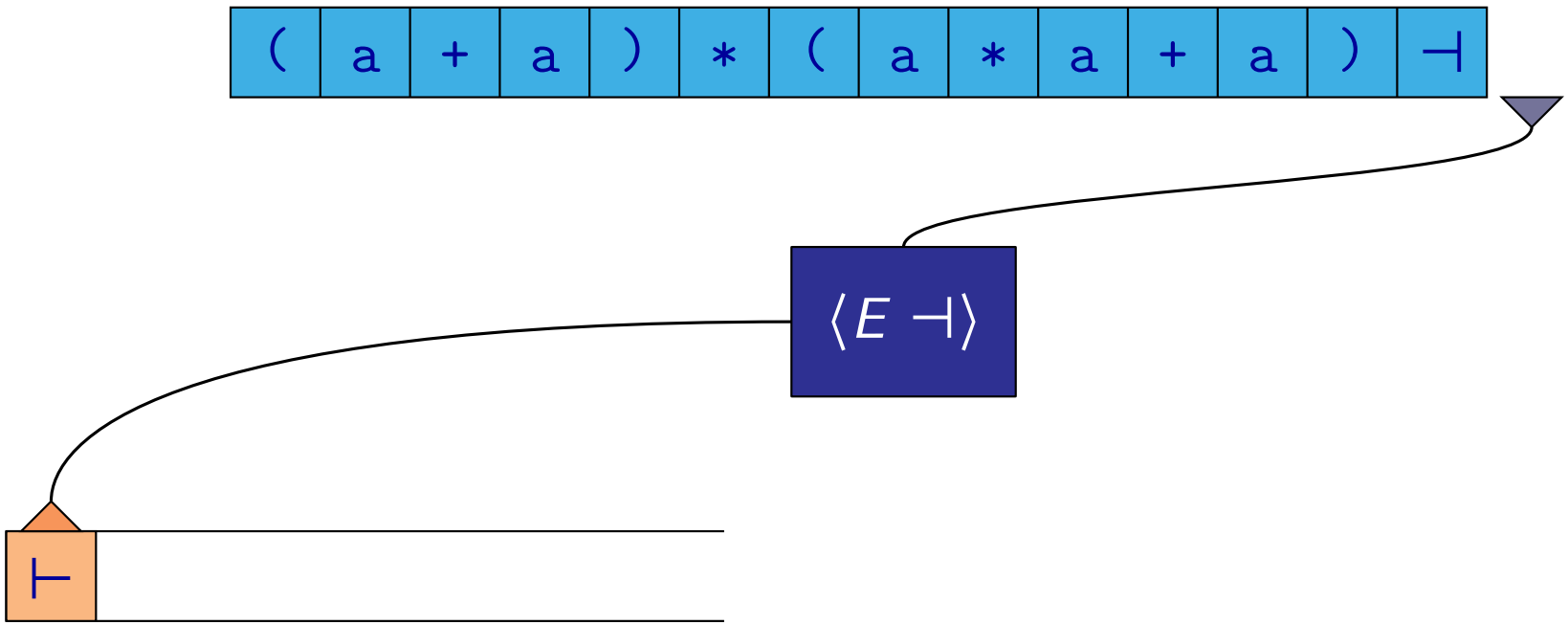
$\underline{E}\dashv \Rightarrow \underline{T}\dashv \Rightarrow T*\underline{F}\dashv \Rightarrow T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



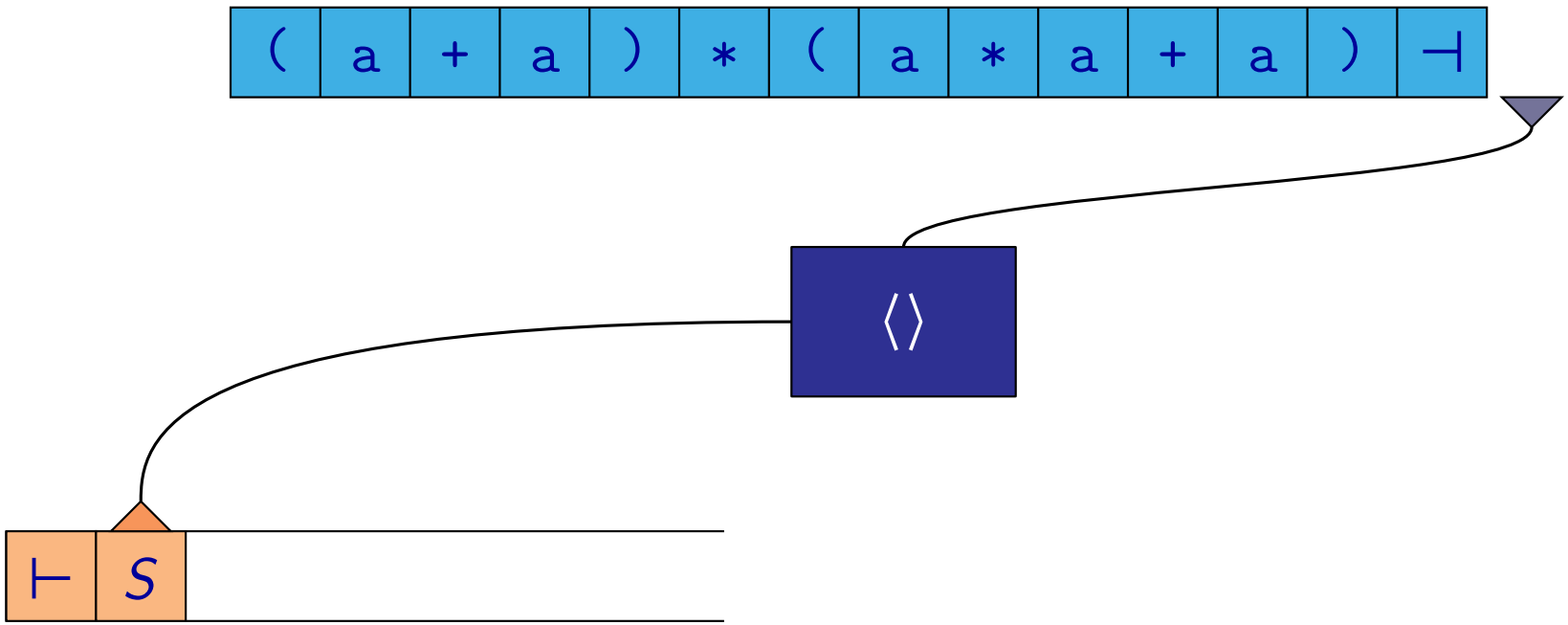
$\underline{E}\dashv \Rightarrow \underline{T}\dashv \Rightarrow T*\underline{F}\dashv \Rightarrow T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



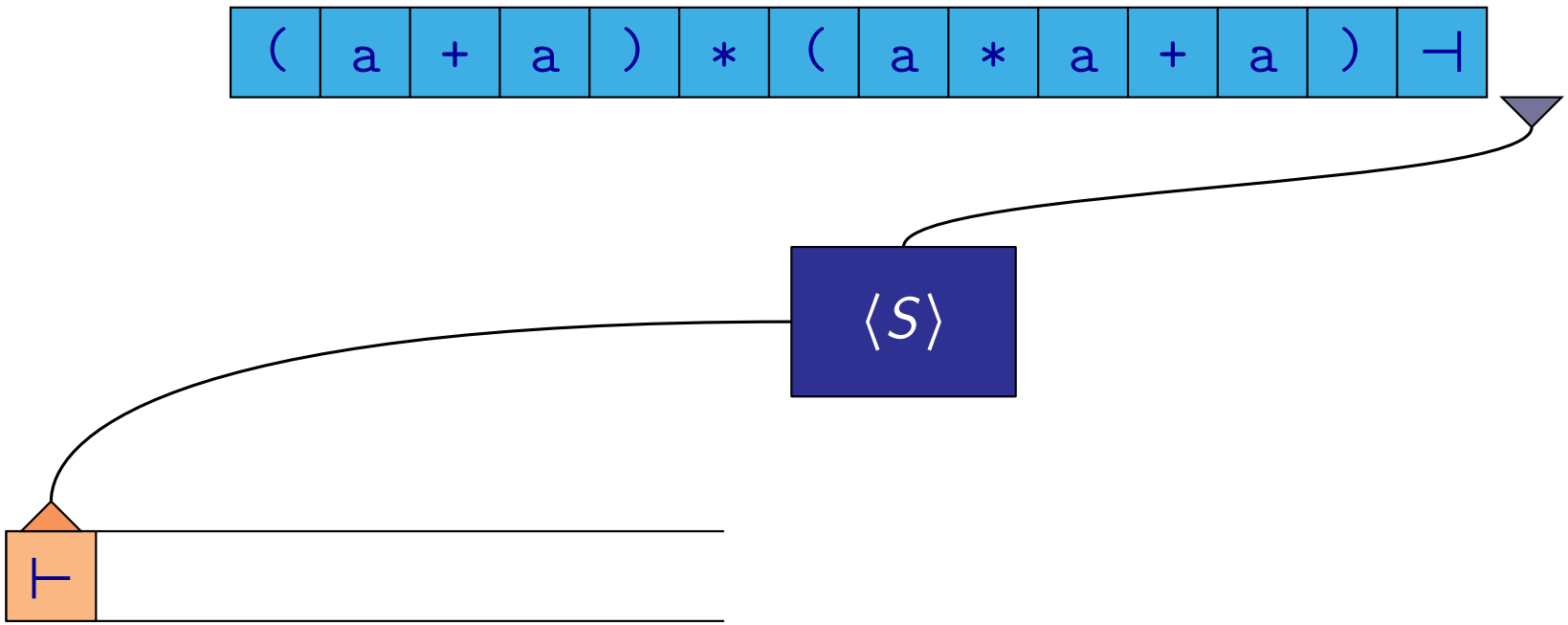
$\underline{E}\dashv \Rightarrow \underline{T}\dashv \Rightarrow T*\underline{F}\dashv \Rightarrow T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow \cdots$

Ekvivalence bezkontextových gramatik a zás. automatů



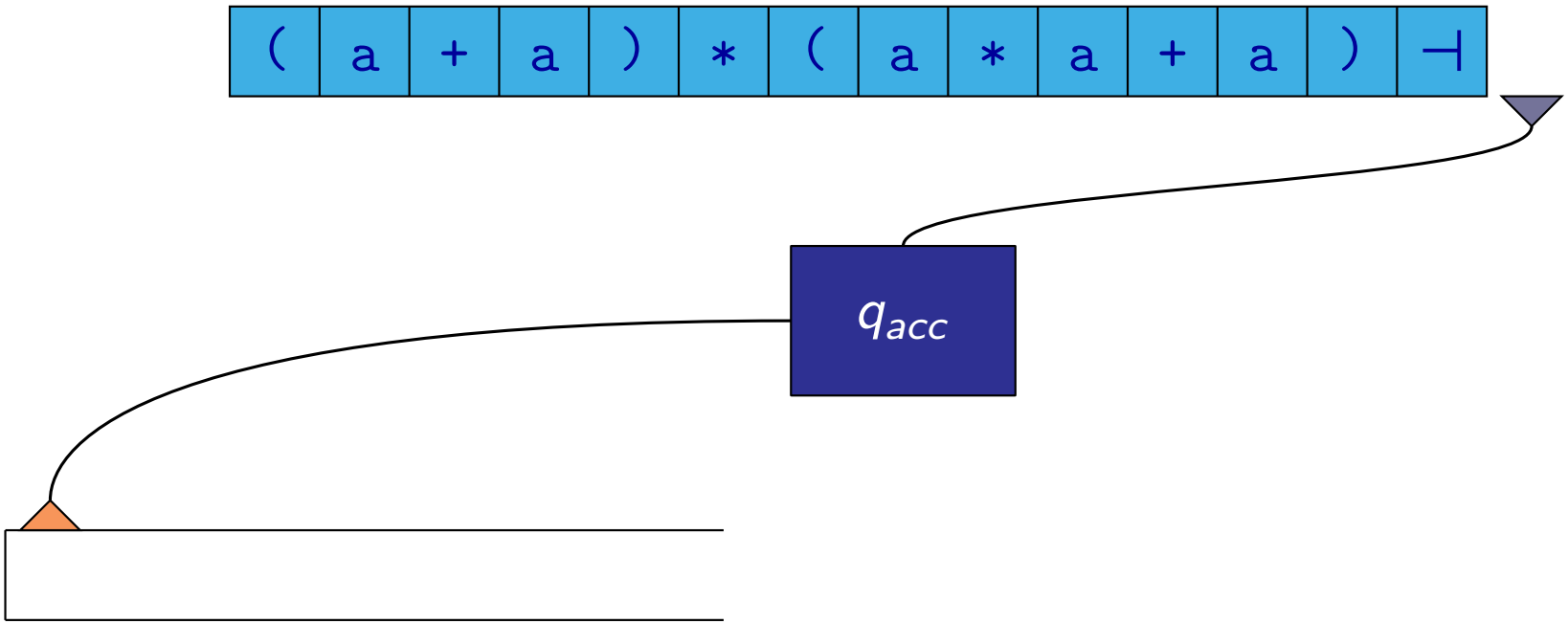
$$\underline{S} \Rightarrow \underline{E}\dashv \Rightarrow \underline{T}\dashv \Rightarrow T*\underline{F}\dashv \Rightarrow T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow \cdots$$

Ekvivalence bezkontextových gramatik a zás. automatů



$$\underline{S} \Rightarrow \underline{E}\dashv \Rightarrow \underline{T}\dashv \Rightarrow T*\underline{F}\dashv \Rightarrow T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow \cdots$$

Ekvivalence bezkontextových gramatik a zás. automatů



$$\underline{S} \Rightarrow \underline{E}\dashv \Rightarrow \underline{T}\dashv \Rightarrow T*\underline{F}\dashv \Rightarrow T*(\underline{E})\dashv \Rightarrow T*(E+\underline{T})\dashv \Rightarrow \cdots$$

Ekvivalence bezkontextových gramatik a zás. automatů

Jak je vidět z předchozího příkladu, zásobníkový automat $\mathcal{M}$ v zásadě provádí **pravou derivaci** v gramatice $\mathcal{G}$ pozpátku.

Další třídy bezkontextových gramatik

Existuje řada různých tříd bezkontextových gramatik, pro které je možné sestrojit daný zásobníkový automat tak, aby byl deterministický:

- Přístup **shora dolů** — vytváří levou derivaci:
  - LL(0), LL(1), LL(2), ...
- Přístup **zdola nahoru** — vytváří pravou derivaci pozpátku:
  - LR(0), LR(1), LR(2), ...
  - LALR (resp. LALR(1), ...)
  - SLR (resp. SLR(1), ...)

Generátory parserů

Generátory parserů — nástroje, které umožňují z popisu dané gramatiky automaticky vygenerovat kód v nějakém programovacím jazyce, který de facto implementuje činnost odpovídajícího zásobníkového automatu.

Příklady generátorů parserů:

- Yacc
- Bison
- ANTLR
- JavaCC
- Menhir
- ...

Ekvivalence bezkontextových gramatik a zás. automatů

Věta

Ke každému zásobníkovému automatu $\mathcal{M}$ s jedním stavem a přijímajícím prázdným zásobníkem lze sestrojit bezkontextovou gramatiku $\mathcal{G}$ takovou, že $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{M})$.

Důkaz: Pro ZA $\mathcal{M} = (\{q_0\}, \Sigma, \Gamma, \delta, q_0, X_0)$, kde $\Sigma \cap \Gamma = \varnothing$, vytvoříme BG $\mathcal{G} = (\Gamma, \Sigma, X_0, P)$, kde

$$(A \to a\alpha) \in P \qquad \Longleftrightarrow \qquad (q_0, \alpha) \in \delta(q_0, a, A)$$

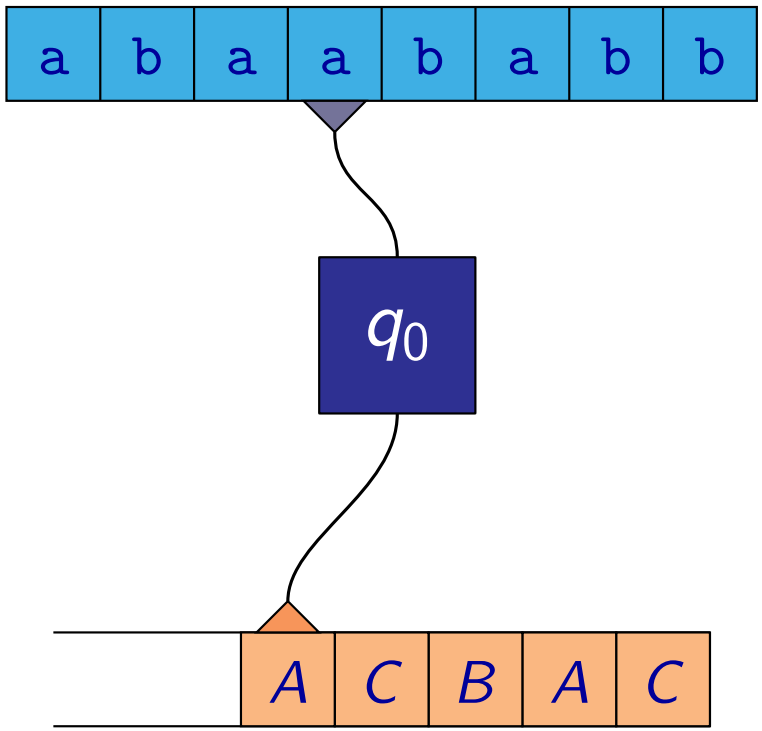
pro každé $A \in \Gamma$, $a \in \Sigma \cup \{\varepsilon\}$, $\alpha \in \Gamma^*$.

Indukcí můžeme dokázat

$$X_0 \Rightarrow^* u\alpha \ \text{ (v } \mathcal{G}) \qquad \Longleftrightarrow \qquad q_0 X_0 \xrightarrow{u} q_0\alpha \ \text{ (v } \mathcal{M})$$

kde $u \in \Sigma^*$ a $\alpha \in \Gamma^*$ (přičemž v $\mathcal{G}$ uvažujeme pouze levé derivace).

Ekvivalence bezkontextových gramatik a zás. automatů



$\mathcal{M}$:

$\vdots$

$q_0 A \xrightarrow{a} q_0 BC$

$q_0 B \xrightarrow{b} q_0$

$\vdots$

$\mathcal{G}$:

$\vdots$

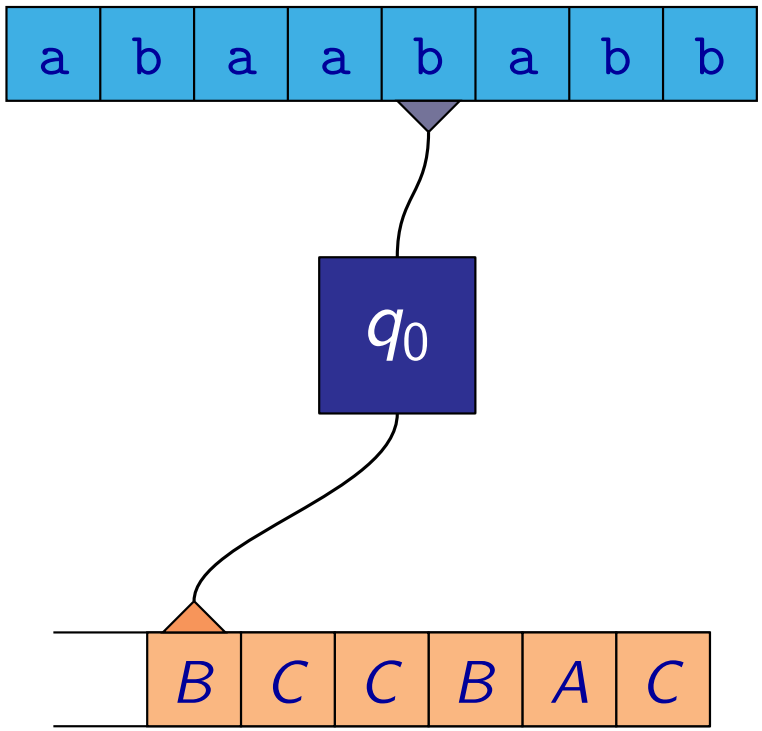
$A \to aBC$

$B \to b$

$\vdots$

a b a $\underline{A}$ C B A C

Ekvivalence bezkontextových gramatik a zás. automatů



$\mathcal{M}$:

$$\vdots$$

$$q_0 A \xrightarrow{a} q_0 BC$$

$$q_0 B \xrightarrow{b} q_0$$

$$\vdots$$

$\mathcal{G}$:

$$\vdots$$

$$A \to aBC$$

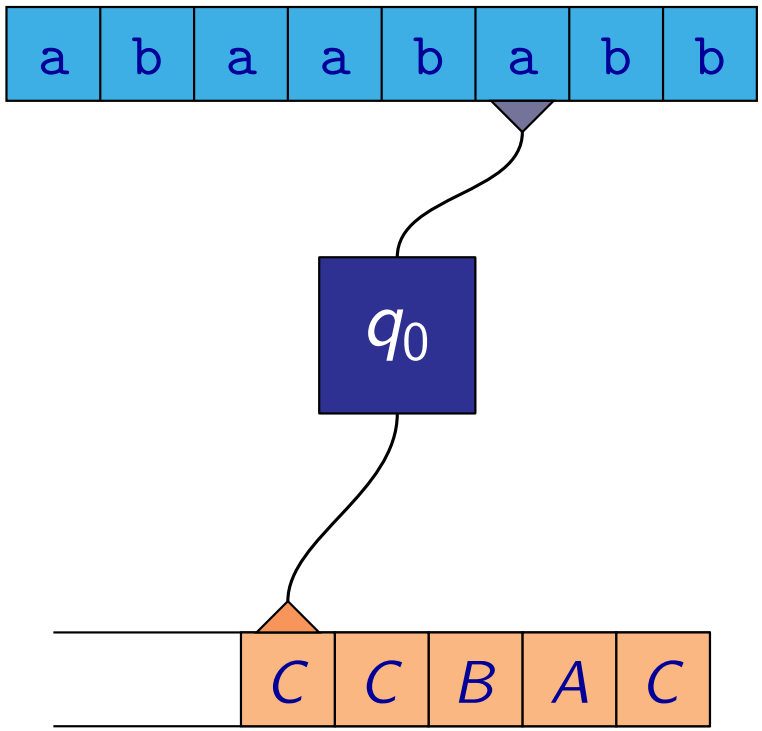
$$B \to b$$

$$\vdots$$

$$\mathtt{a\,b\,a}\ \underline{A}\ C\ B\ A\ C$$

$$\Rightarrow\ \mathtt{a\,b\,a\,a}\ \underline{B}\ C\ C\ B\ A\ C$$

Ekvivalence bezkontextových gramatik a zás. automatů



$\mathcal{M}$:

$$\vdots$$

$$q_0 A \xrightarrow{a} q_0 BC$$

$$q_0 B \xrightarrow{b} q_0$$

$$\vdots$$

$\mathcal{G}$:

$$\vdots$$

$$A \to aBC$$

$$B \to b$$

$$\vdots$$

$$\begin{aligned} & \mathtt{a\,b\,a}\,\underline{A}\,C\,B\,A\,C \\ \Rightarrow\ & \mathtt{a\,b\,a\,a}\,\underline{B}\,C\,C\,B\,A\,C \\ \Rightarrow\ & \mathtt{a\,b\,a\,a\,b}\,\underline{C}\,C\,B\,A\,C \end{aligned}$$

Ekvivalence bezkontextových gramatik a zás. automatů

Věta

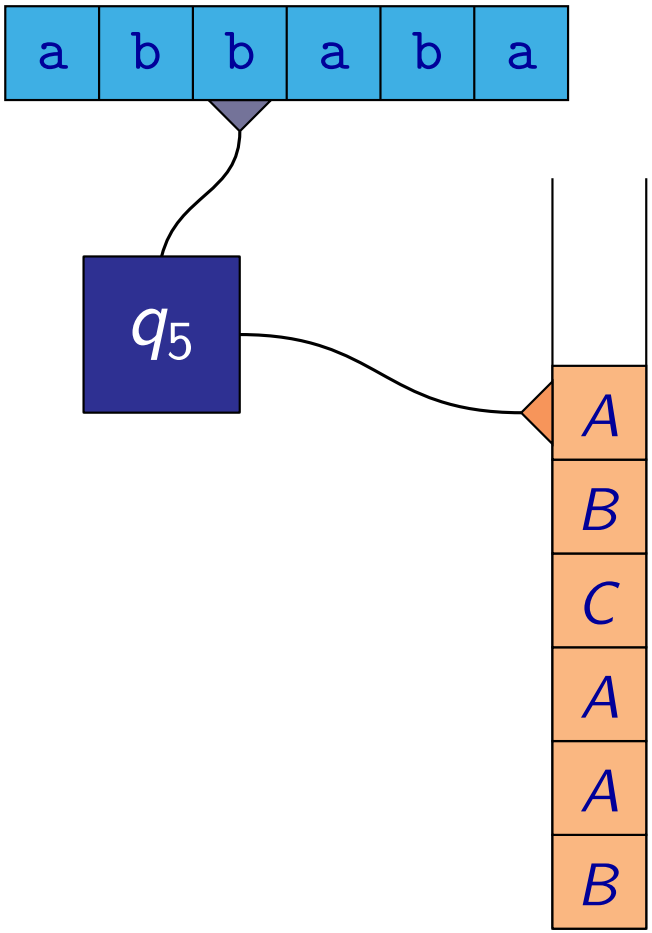
Ke každému zásobníkovému automatu $\mathcal{M}$ lze sestrojit zásobníkový automat $\mathcal{M}'$ s jedním stavem tž. $\mathcal{L}(\mathcal{M}') = \mathcal{L}(\mathcal{M})$.

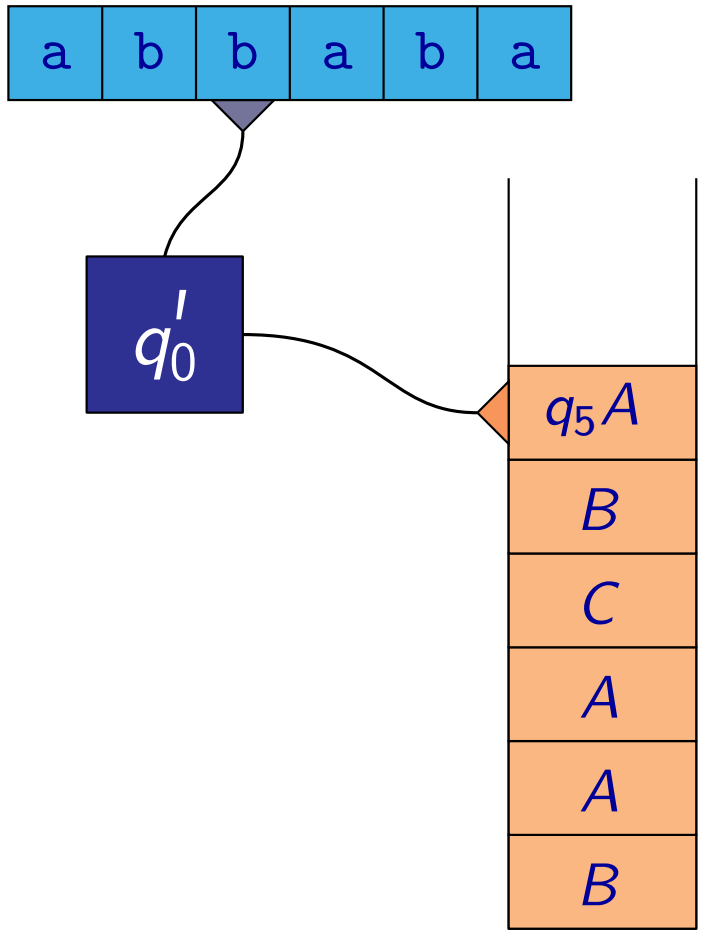
Myšlenka důkazu:

- Stav automatu $\mathcal{M}$ si budeme pamatovat na zásobníku.
- Pro $\delta(q, a, X) = \{(q', \varepsilon)\}$ musíme kontrolovat nejen, že jsme ve stavu q, ale také, že se dostaneme do stavu q'. (Další případy jsou přímočaré.)
- Každý zásobníkový symbol automatu $\mathcal{M}'$ je tedy trojice, kde si pamatujeme zásobníkový symbol, aktuální stav a aktuální stav ze symbolu o jedna níže na zásobníku.
- ZA $\mathcal{M}'$ nedeterministicky „hádá“ řídící stavy, do kterých se dostane $\mathcal{M}$ v okamžiku, kdy se daný zásobníkový symbol ocitne na vrcholu zásobníku.

Ekvivalence bezkontextových gramatik a zás. automatů

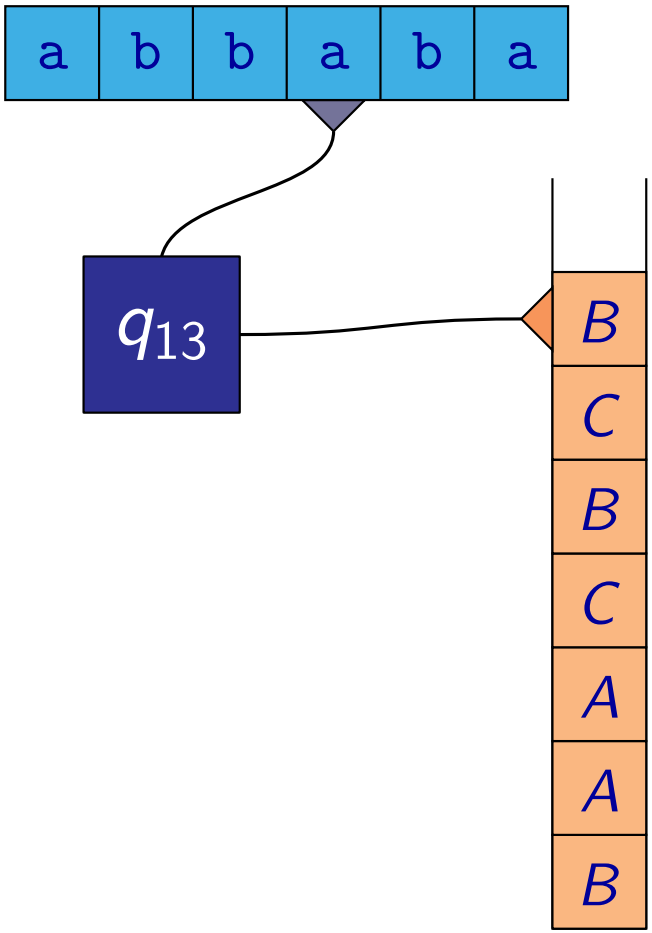
Chybná myšlenka:

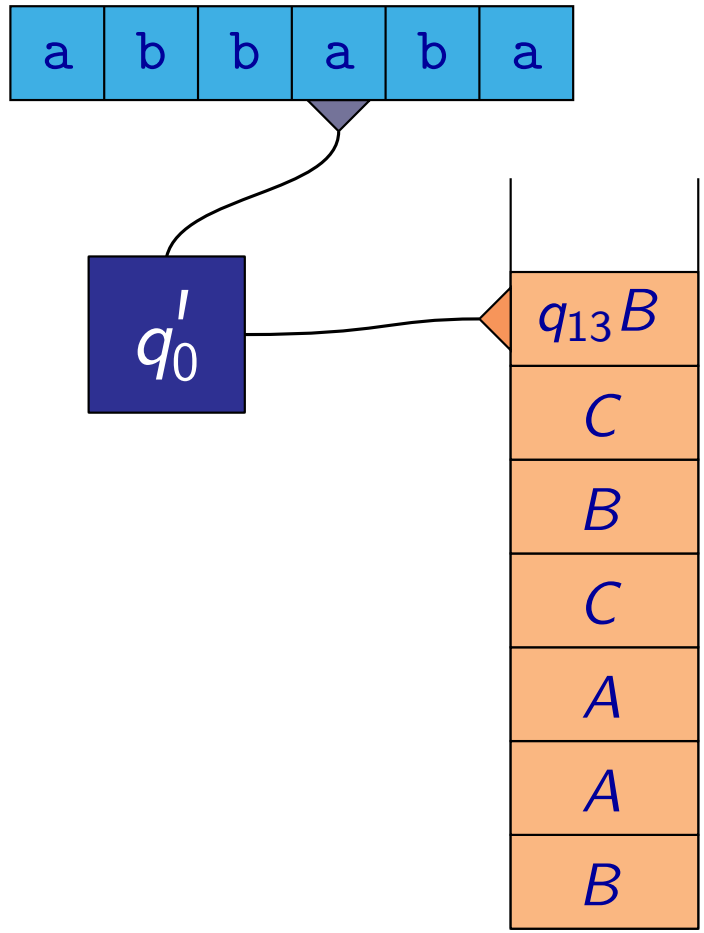




Ekvivalence bezkontextových gramatik a zás. automatů

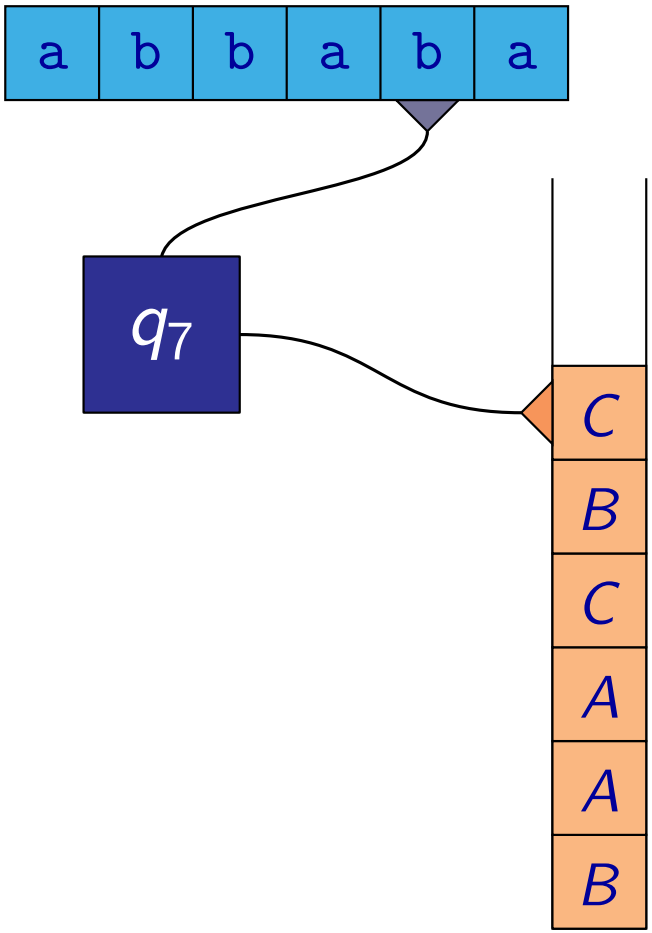
Chybná myšlenka:

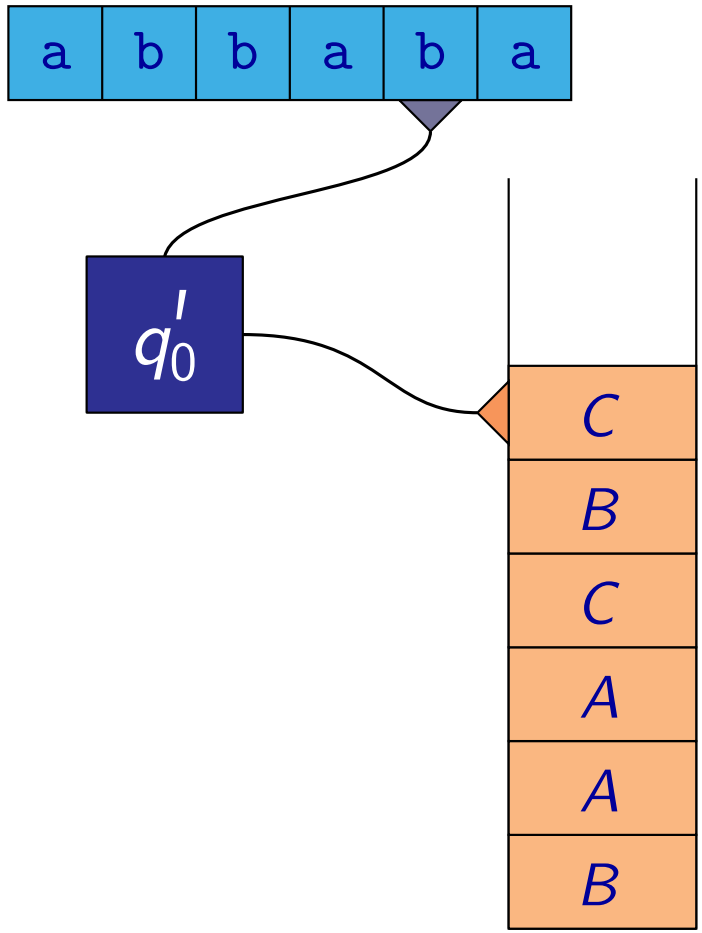




Ekvivalence bezkontextových gramatik a zás. automatů

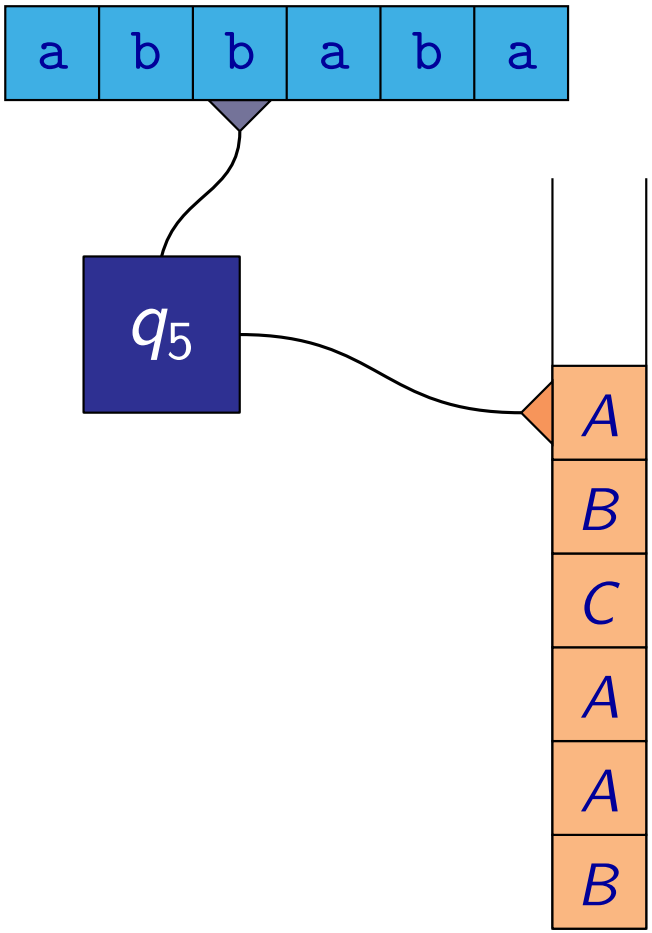
Chybná myšlenka:

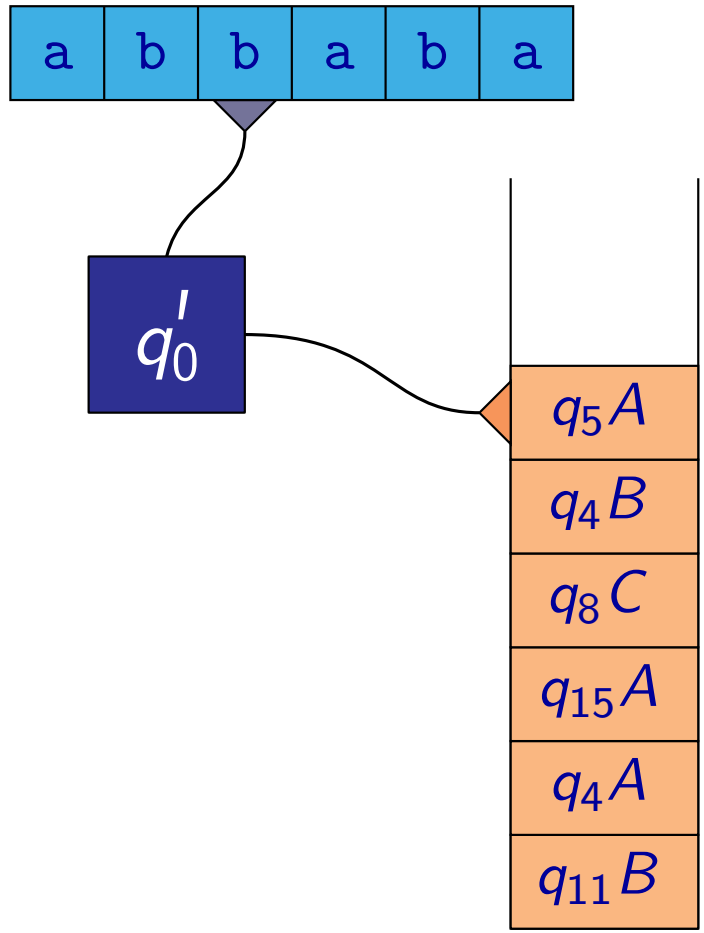




Ekvivalence bezkontextových gramatik a zás. automatů

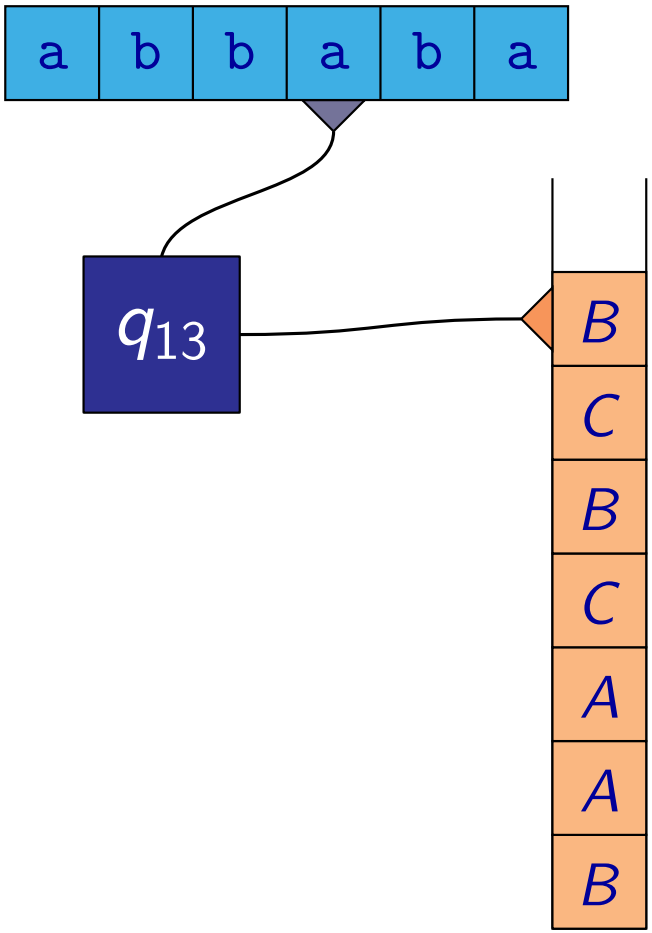
Další chybná myšlenka:

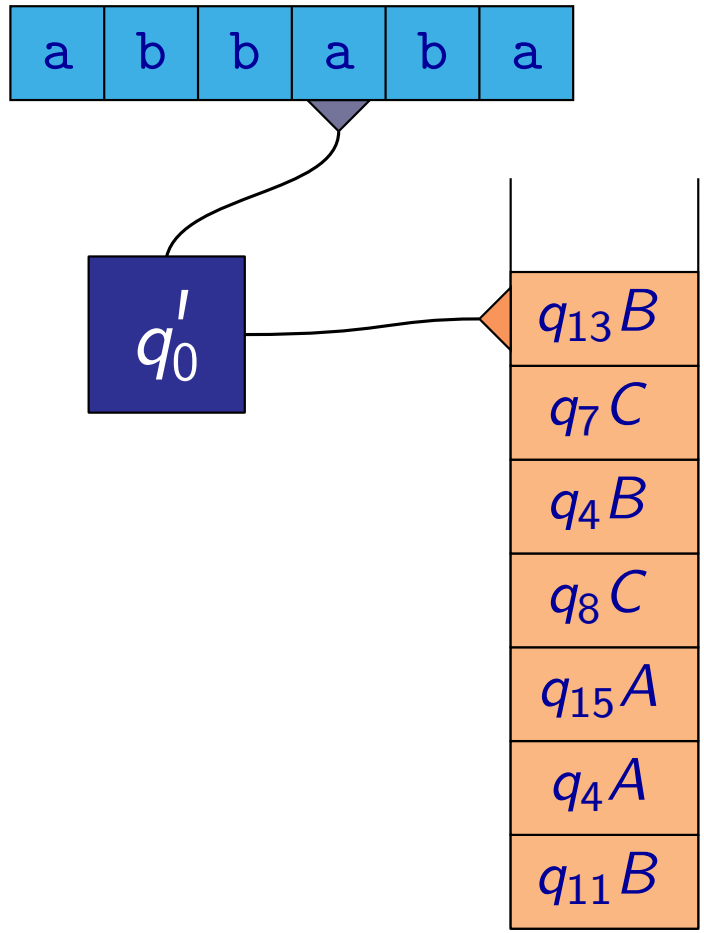




Ekvivalence bezkontextových gramatik a zás. automatů

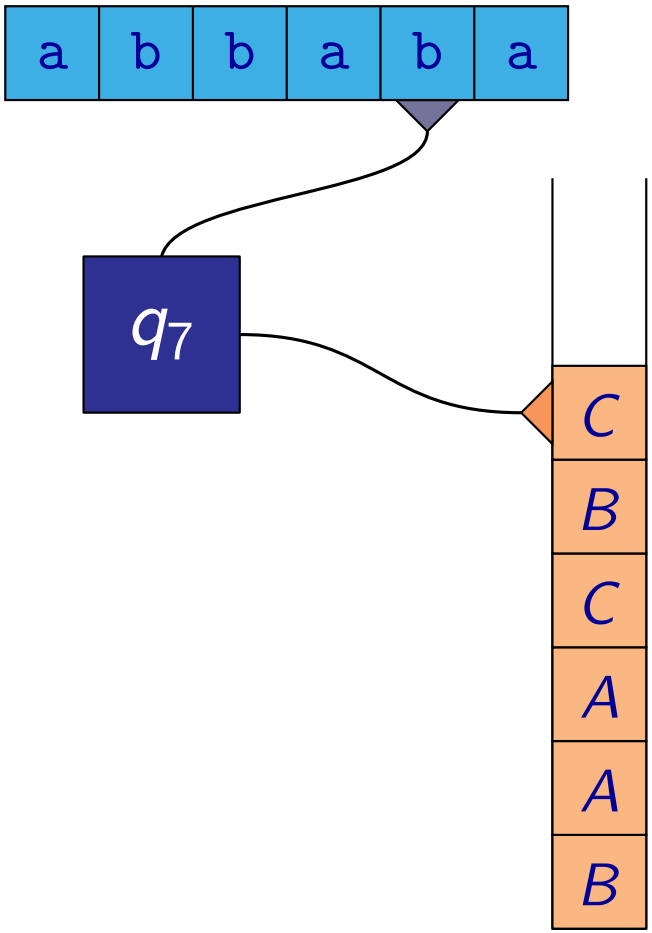
Další chybná myšlenka:

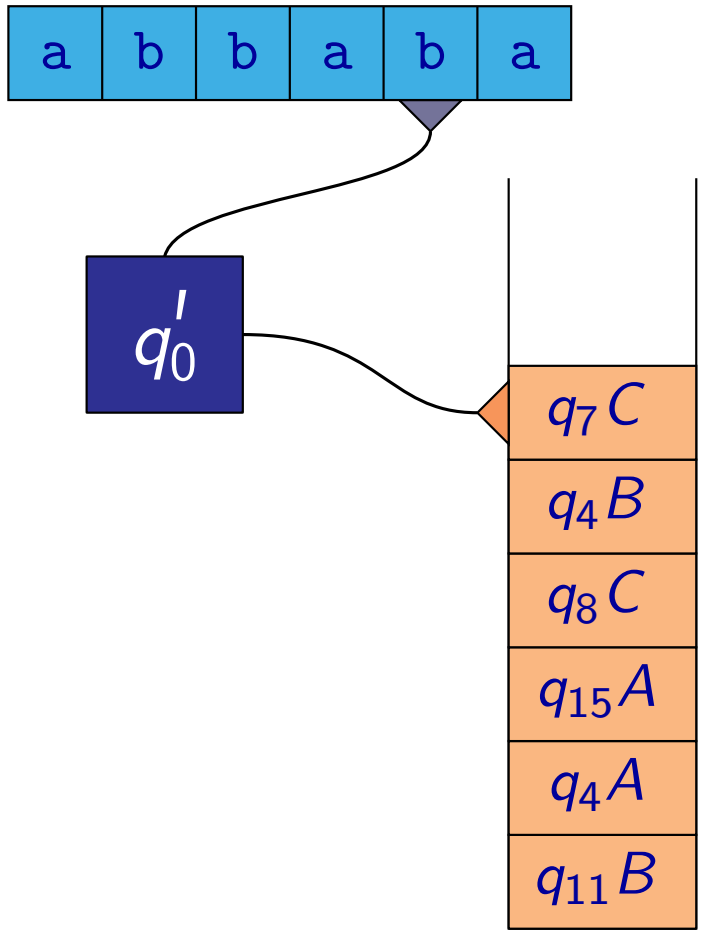




Ekvivalence bezkontextových gramatik a zás. automatů

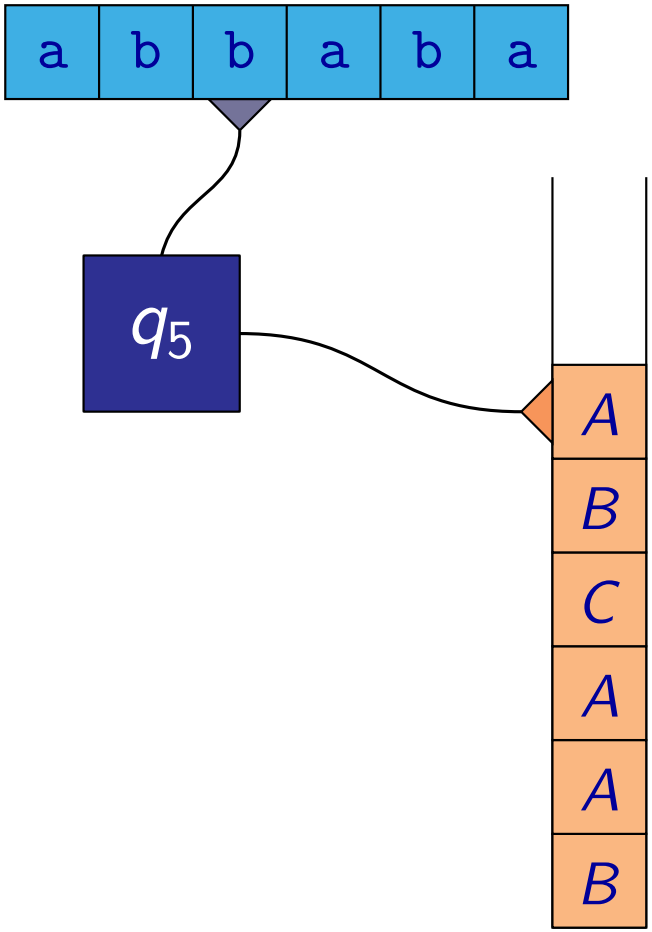
Další chybná myšlenka:

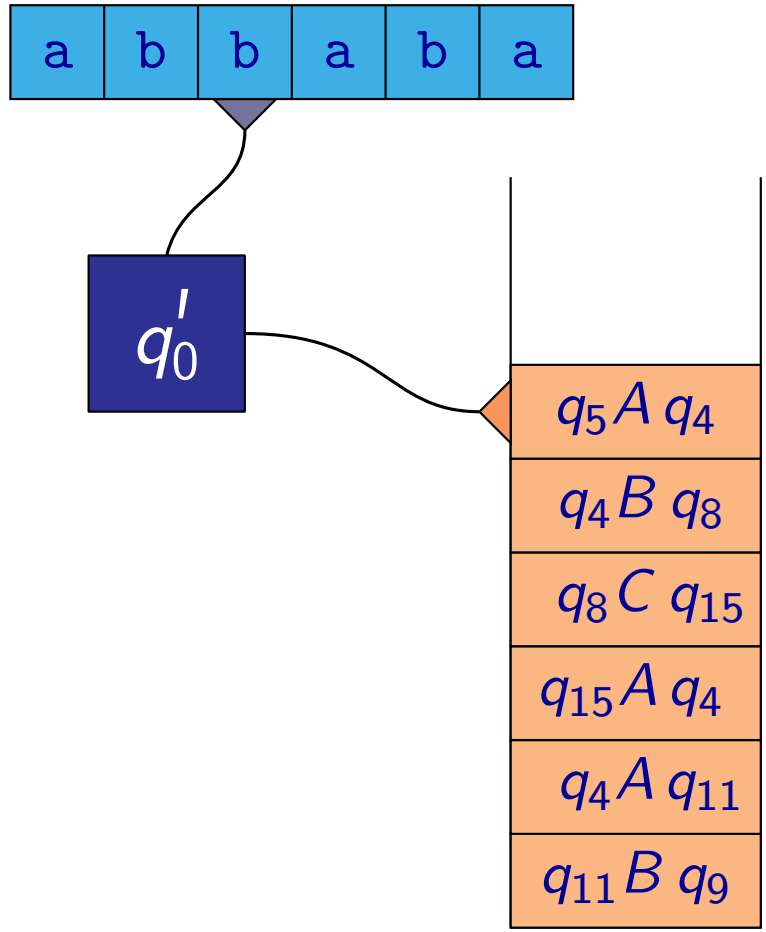




Ekvivalence bezkontextových gramatik a zás. automatů

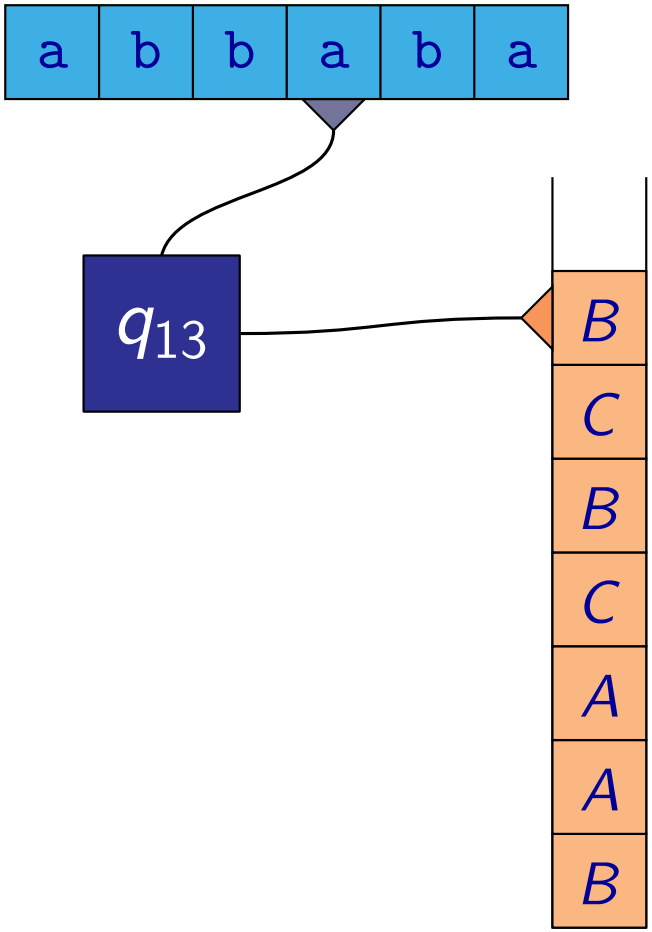
Korektní konstrukce:

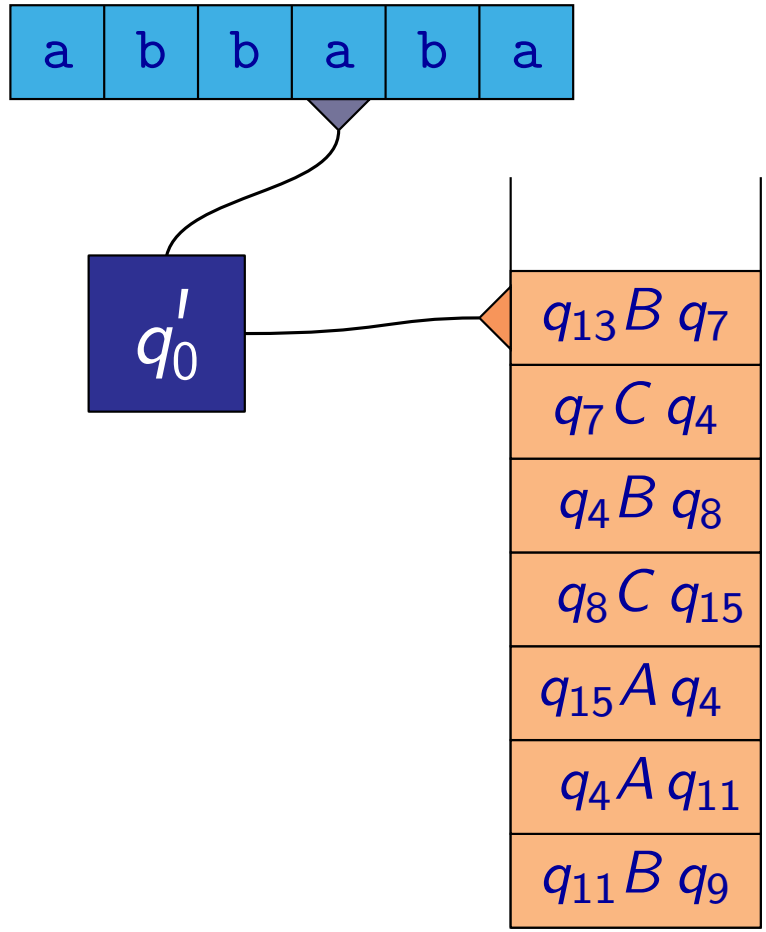




Ekvivalence bezkontextových gramatik a zás. automatů

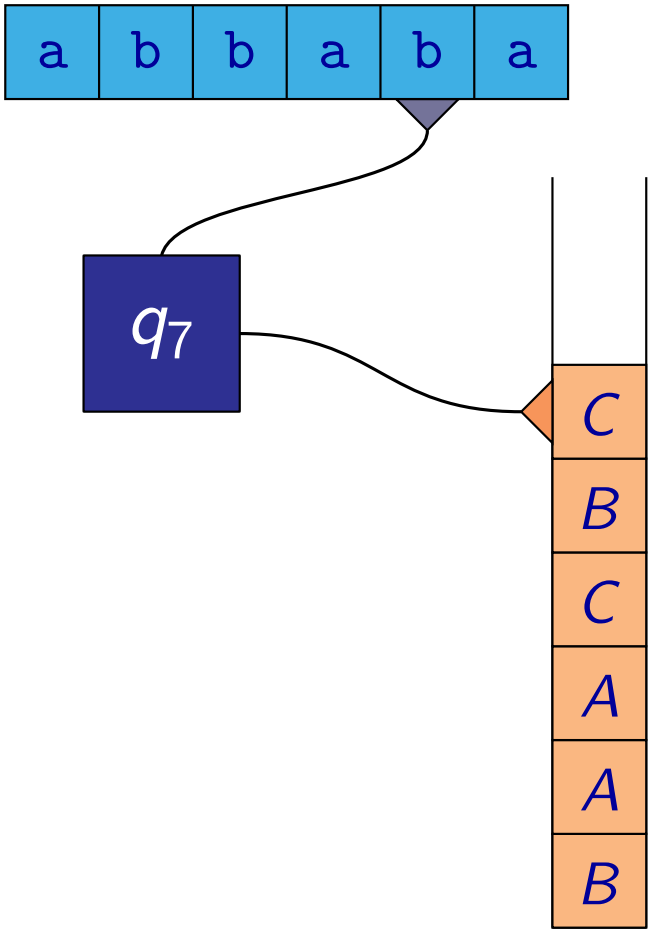
Korektní konstrukce:

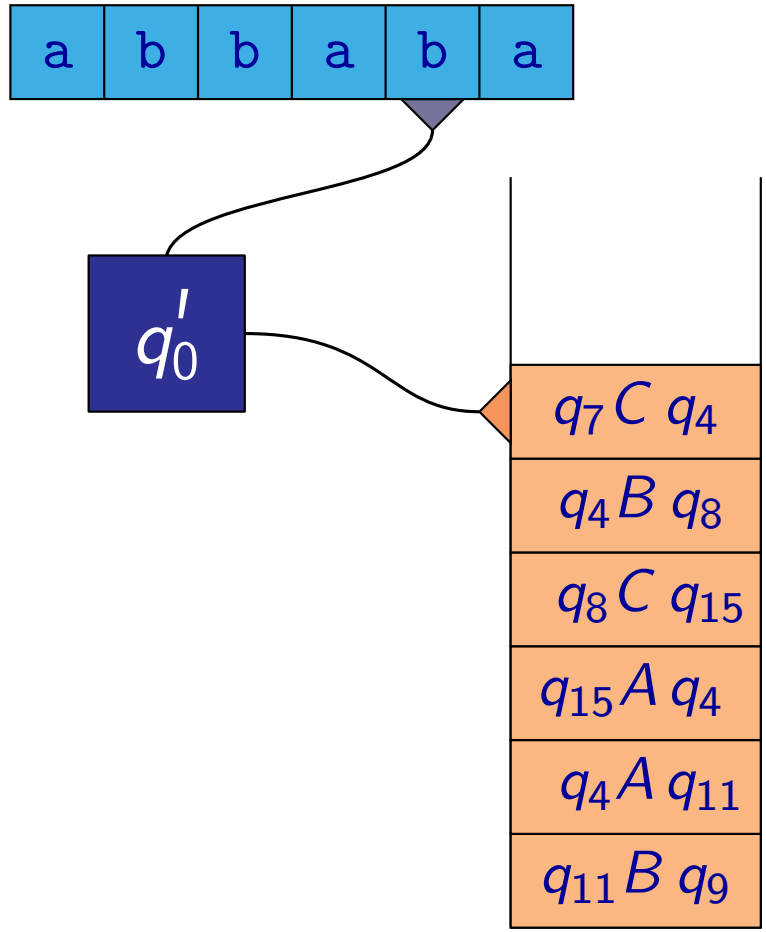




Ekvivalence bezkontextových gramatik a zás. automatů

Korektní konstrukce:





Ekvivalence bezkontextových gramatik a zás. automatů

Tvrzení

K libovolné bezkontextové gramatice $\mathcal{G}$ je možné sestrojit (nedeterministický) zásobníkový automat $\mathcal{M}$ takový, že $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{M})$.

Tvrzení

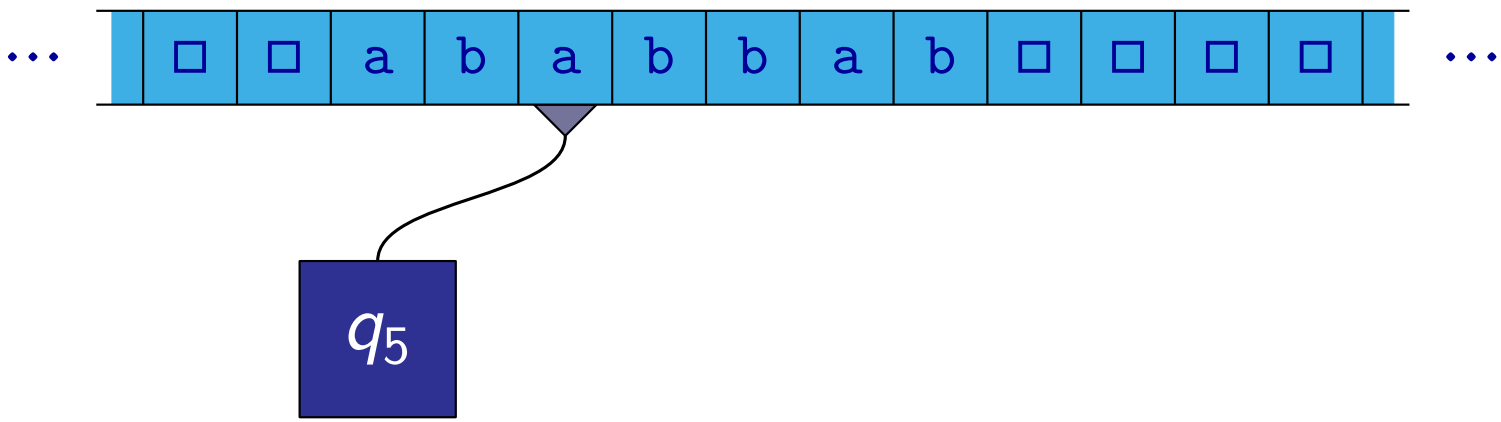
K libovolnému zásobníkovému automatu $\mathcal{M}$ je možné sestrojit bezkontextovou gramatiku $\mathcal{G}$ takovou, že $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{G})$.

Turingovy stroje

Turingův stroj

Turingův stroj — zařízení podobné konečnému automatu s následujícími rozdíly:

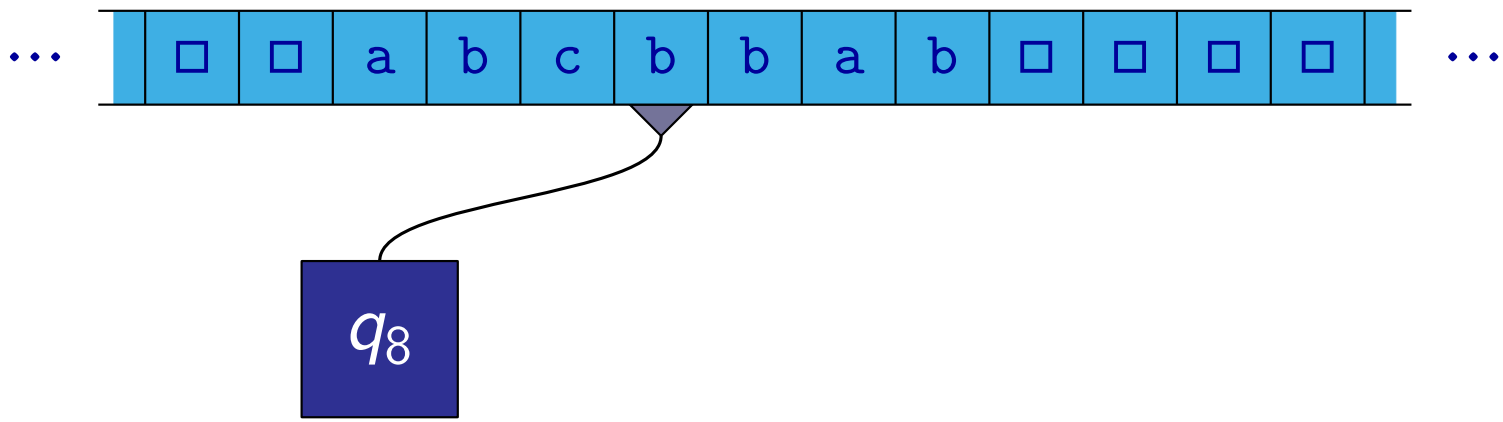
- pohyb hlavy oběma směry
- možnost zápisu na pásku na aktuální pozici hlavy
- páska je nekonečná



Turingův stroj

Turingův stroj — zařízení podobné konečnému automatu s následujícími rozdíly:

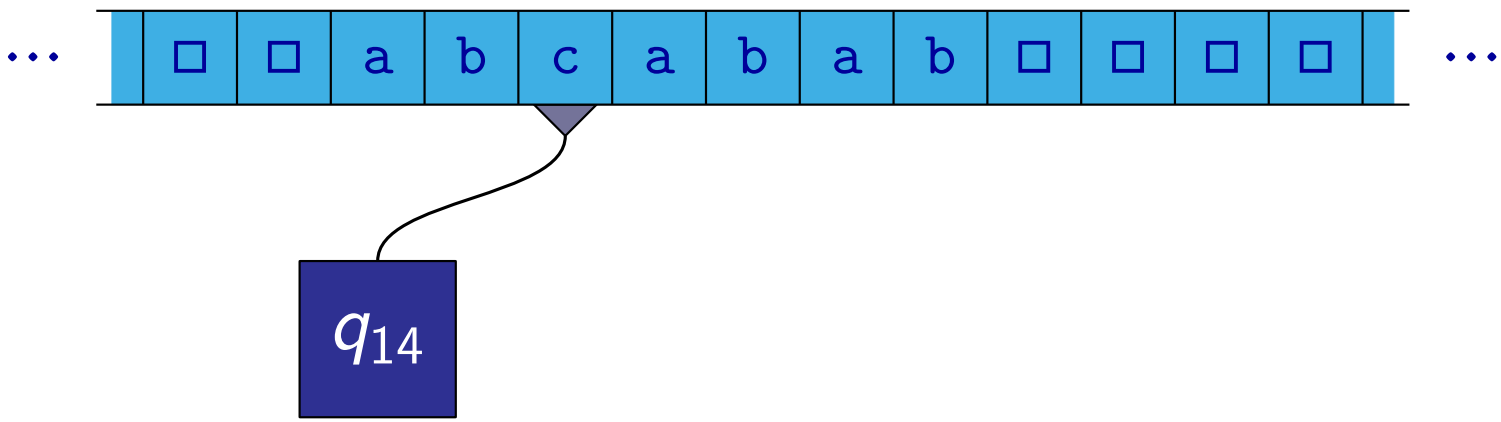
- pohyb hlavy oběma směry
- možnost zápisu na pásku na aktuální pozici hlavy
- páska je nekonečná



Turingův stroj

Turingův stroj — zařízení podobné konečnému automatu s následujícími rozdíly:

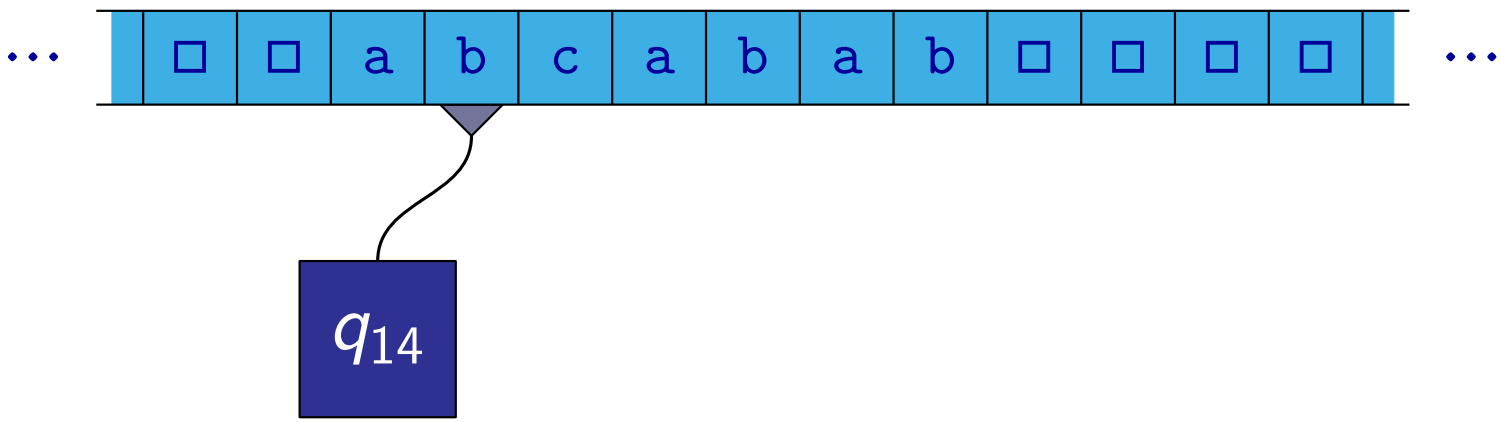
- pohyb hlavy oběma směry
- možnost zápisu na pásku na aktuální pozici hlavy
- páska je nekonečná



Turingův stroj

Turingův stroj — zařízení podobné konečnému automatu s následujícími rozdíly:

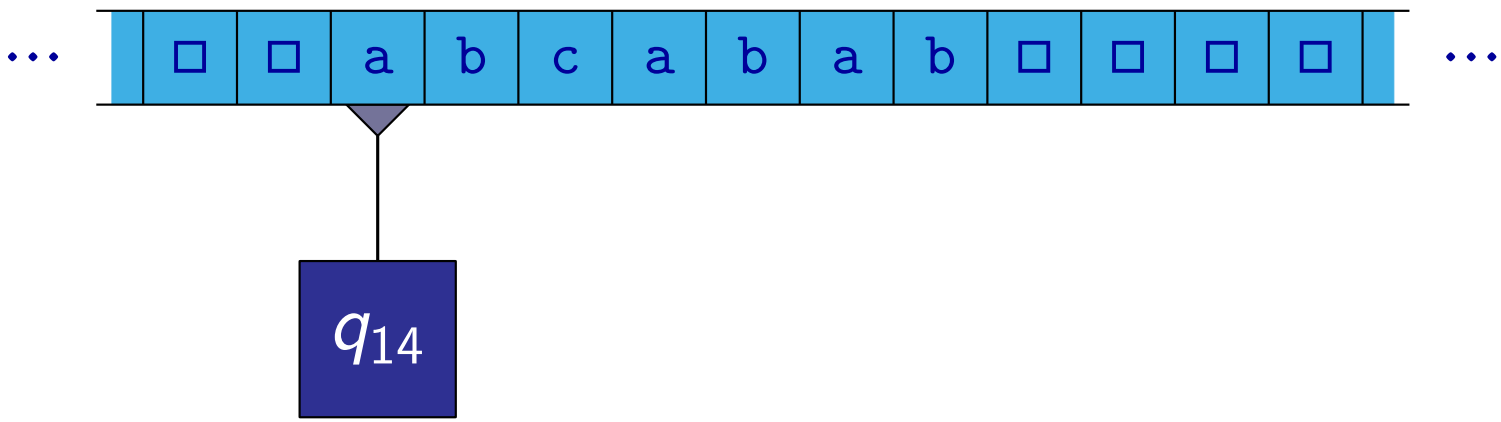
- pohyb hlavy oběma směry
- možnost zápisu na pásku na aktuální pozici hlavy
- páska je nekonečná



Turingův stroj

Turingův stroj — zařízení podobné konečnému automatu s následujícími rozdíly:

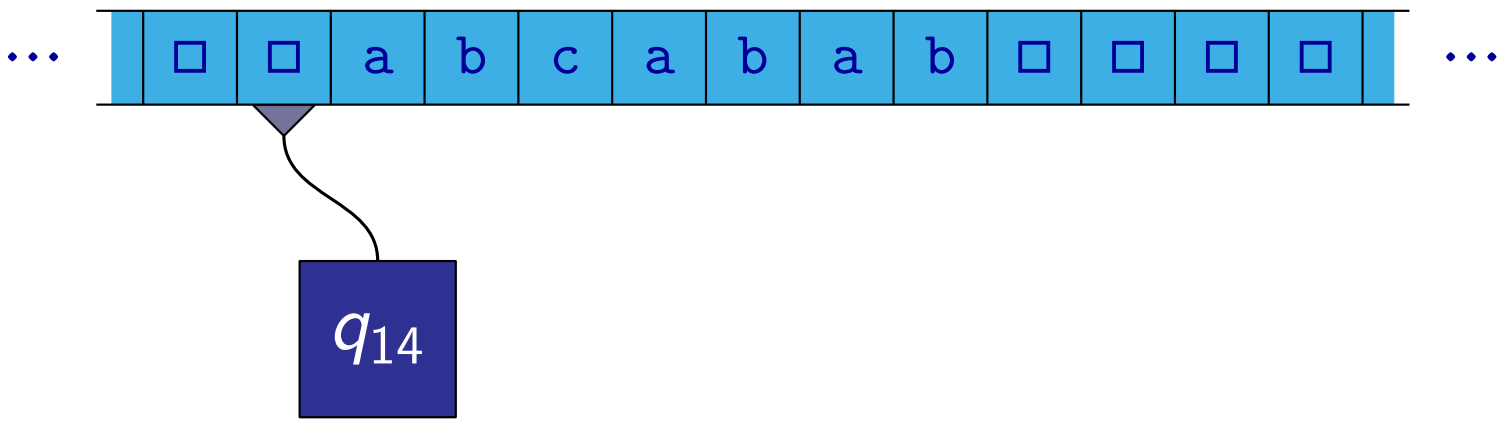
- pohyb hlavy oběma směry
- možnost zápisu na pásku na aktuální pozici hlavy
- páska je nekonečná



Turingův stroj

Turingův stroj — zařízení podobné konečnému automatu s následujícími rozdíly:

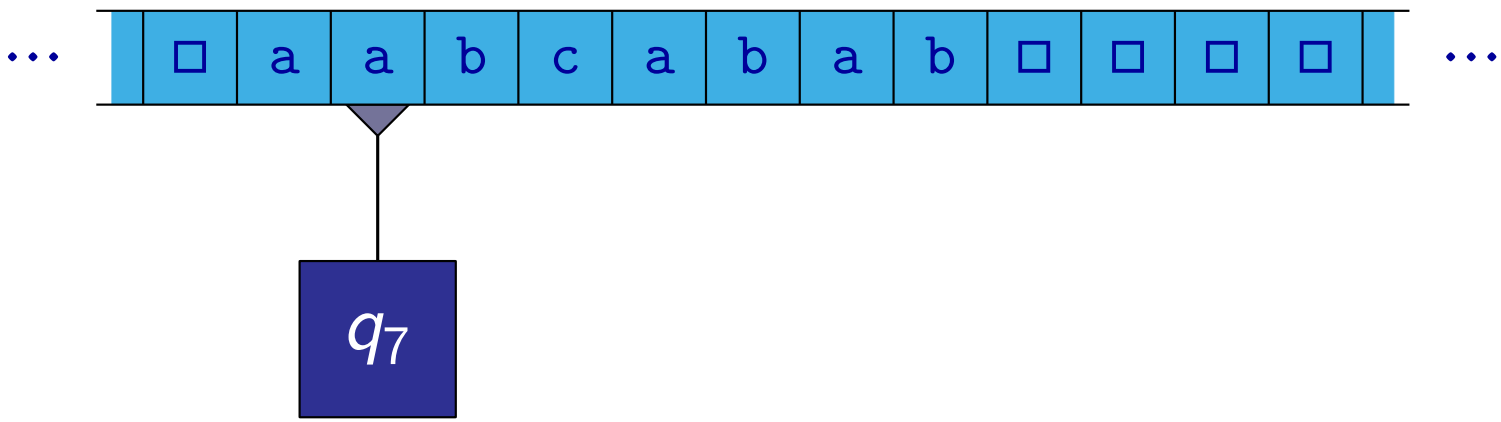
- pohyb hlavy oběma směry
- možnost zápisu na pásku na aktuální pozici hlavy
- páska je nekonečná



Turingův stroj

Turingův stroj — zařízení podobné konečnému automatu s následujícími rozdíly:

- pohyb hlavy oběma směry
- možnost zápisu na pásku na aktuální pozici hlavy
- páska je nekonečná



Turingův stroj

Alan M. Turing, „On Computable Numbers, with an application to the Entscheidungsproblem“, *Proceedings of the London Mathematical Society*, 42 (1936), pp. 230–265, Erratum: Ibid., 43 (1937), pp. 544–546.

Turingův stroj

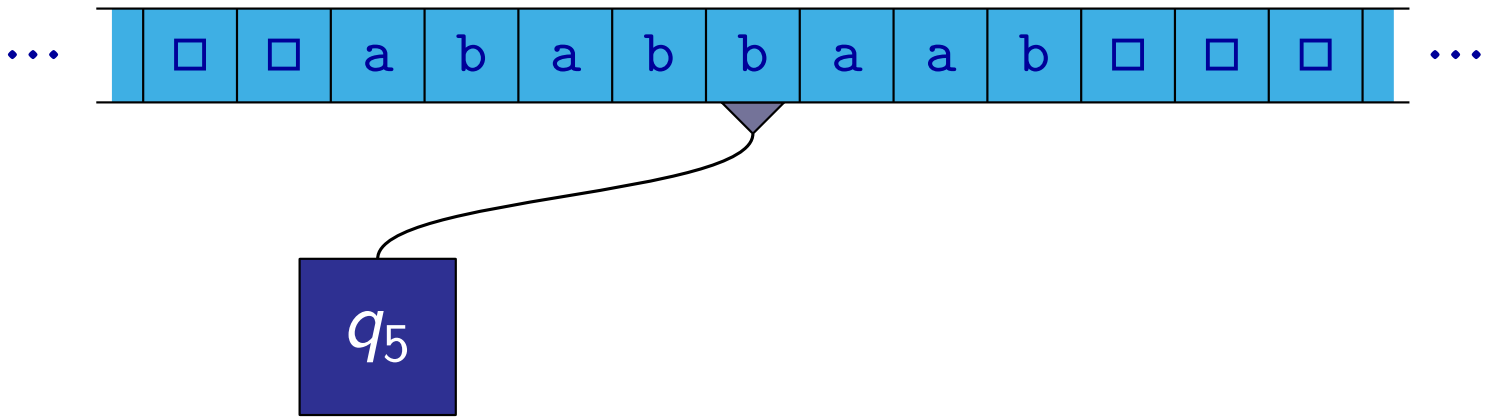
Definice

Formálně je **Turingův stroj** definován jako šestice $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F)$ kde:

- Q je konečná neprázdná množina **stavů**
- Γ je konečná neprázdná množina **páskových symbolů** (**pásková abeceda**)
- $\Sigma \subseteq \Gamma$ je konečná neprázdná množina **vstupních symbolů** (**vstupní abeceda**)
- $\delta : (Q - F) \times \Gamma \to Q \times \Gamma \times \{-1, 0, +1\}$ je **přechodová funkce**
- $q_0 \in Q$ je **počáteční stav**
- $F \subseteq Q$ je množina **koncových stavů**

Předpokládáme, že v $\Gamma - \Sigma$ je vždy speciální prvek $\square$ označující prázdný znak (blank).

Konfigurace Turingova stroje

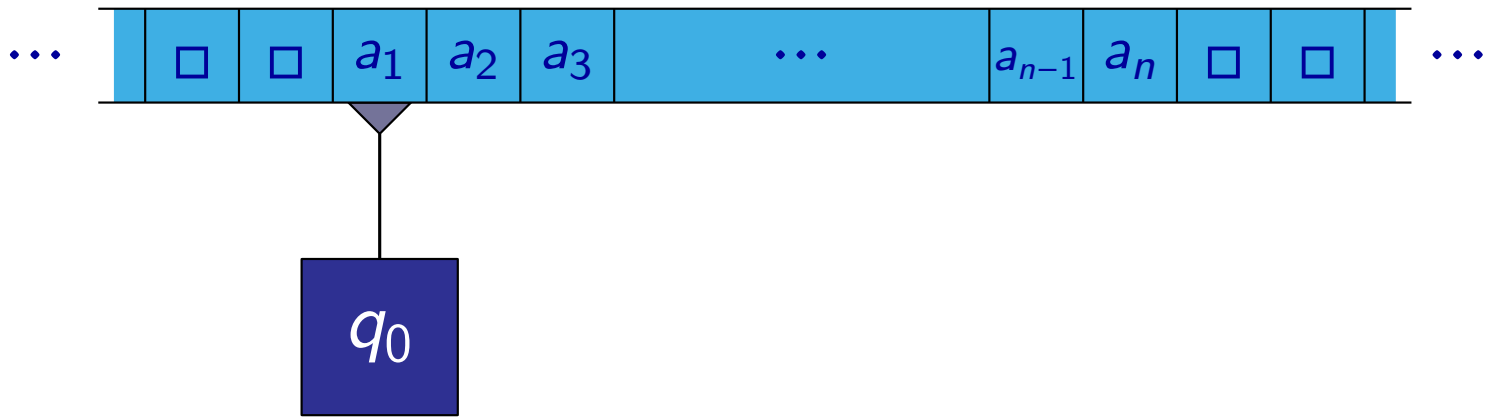


Konfigurace Turingova stroje je dána:

- stavem řídící jednotky
- obsahem pásky
- pozicí hlavy

Konfigurace Turingova stroje

Výpočet Turingova stroje $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F)$ nad slovem $w \in \Sigma^*$, kde $w = a_1 a_2 \cdots a_n$, začíná v **počáteční konfiguraci**:



- stav řídící jednotky je q_0
- na pásce je zapsáno slovo w, zbývající políčka pásky jsou vyplněna prázdnými symboly ($\square$)
- hlava se nachází na prvním symbolu slova w (nebo na symbolu $\square$, pokud je $w = \varepsilon$)

Turingův stroj

Jeden krok Turingova stroje:

Předpokládejme, že:

- stav řídící jednotky je q
- na políčku, kde se právě nachází hlava, je zapsán symbol b

Řekněme, že $\delta(q, b) = (q', b', d)$, kde $d \in \{-1, 0, +1\}$.

Jeden krok Turingova stroje se provede následovně:

- stav řídící jednotky se změní na q'
- na políčko na pozici hlavy se místo symbolu b zapíše symbol b'
- V závislosti na hodnotě d se hlava posune:
  - pro $d = -1$ se posune o jedno políčko doleva
  - pro $d = +1$ se posune o jedno políčko doprava
  - pro $d = 0$ se pozice hlavy nezmění

Turingův stroj

- Turingův stroj provádí kroky tak dlouho, dokud stav řídící jednotky není stav z množiny F.
- Konfigurace, kde stav řídící jednotky patří do množiny F, jsou **koncové konfigurace**.
- V koncových konfiguracích výpočet končí.
- Výpočet stroje $\mathcal{M}$ nad slovem w může být nekonečný.

Turingův stroj

Často volíme množinu koncových stavů $F = \{q_{acc}, q_{rej}\}$.

Můžeme pak pro slovo $w \in \Sigma^*$ definovat, zda ho daný Turingův stroj přijímá:

- Pokud je po skončení výpočtu nad slovem w řídící jednotka ve stavu q_{acc}, stroj slovo w přijímá.
- Pokud je po skončení výpočtu nad slovem w řídící jednotka ve stavu q_{rej}, stroj slovo w nepřijímá.
- Pokud je výpočet nad slovem w nekonečný, stroj slovo w nepřijímá.

Jazyk $\mathcal{L}(\mathcal{M})$ Turingova stroje $\mathcal{M}$ je množina všech slov nad abecedou Σ^*, která stroj $\mathcal{M}$ přijímá.

Turingův stroj

Jazyk $L \subseteq \Sigma^*$ je Turingovým strojem $\mathcal{M}$ **přijímán** (accepted), jestliže:

- pro každé slovo $w \in \Sigma^*$ platí, že $w \in L$ právě tehdy, když výpočet stroje $\mathcal{M}$ nad w skončí v koncovém stavu q_{acc}.

(Výpočty nad slovy, která nepatří do L, tedy mohou skončit ve stavu q_{rej} nebo být nekonečné.)

Jazyk $L \subseteq \Sigma^*$ je Turingovým strojem $\mathcal{M}$ **rozpoznáván** (recognized), jestliže:

- pro každé slovo $w \in L$ výpočet stroje $\mathcal{M}$ nad w skončí v koncovém stavu q_{acc}.
- pro každé slovo $w \in (\Sigma^* - L)$ výpočet stroje $\mathcal{M}$ nad w skončí v koncovém stavu q_{rej}.

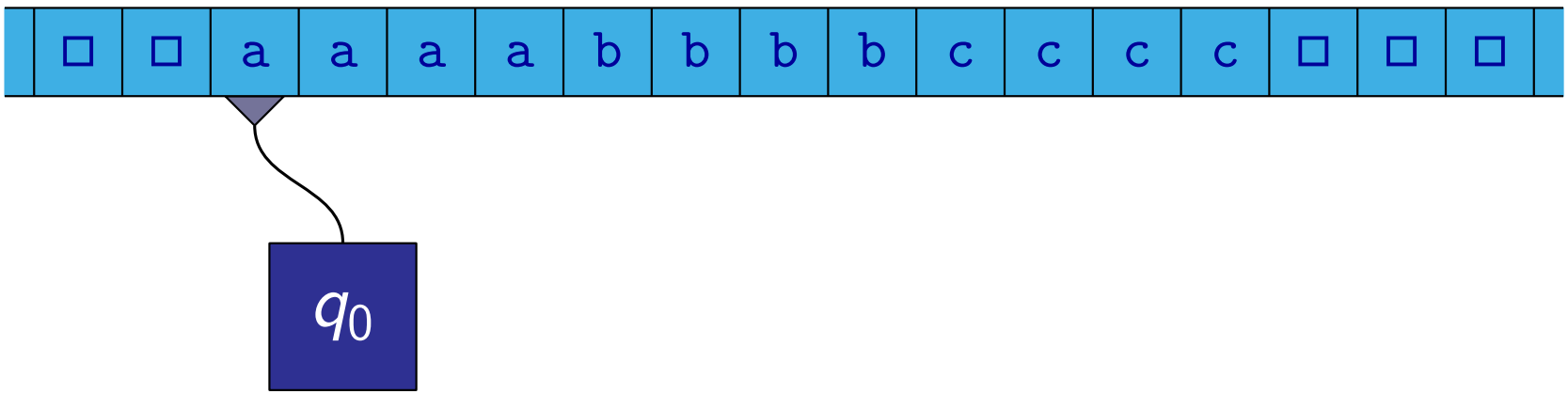
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



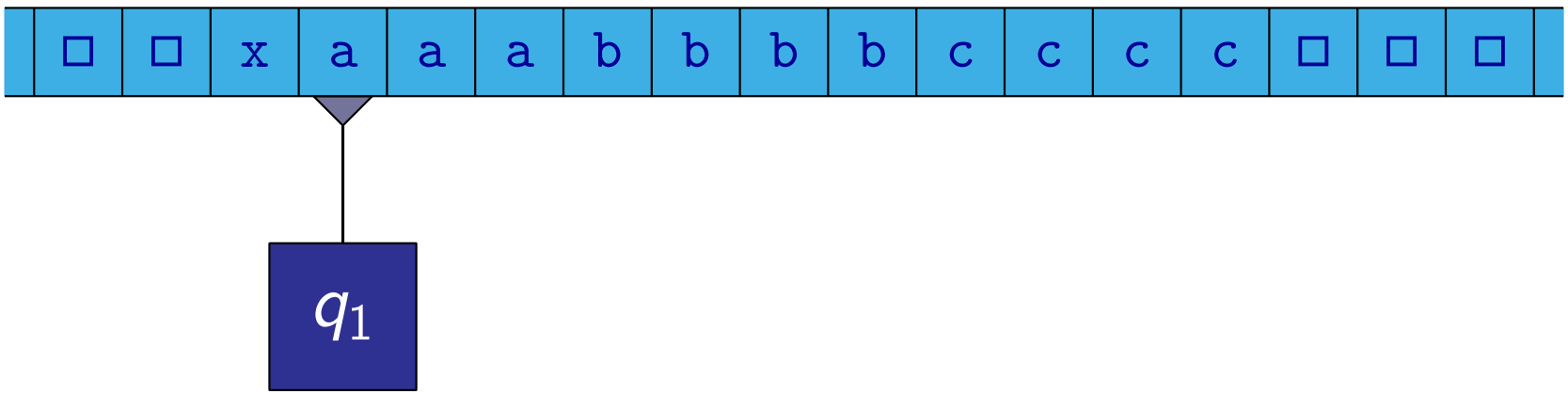
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



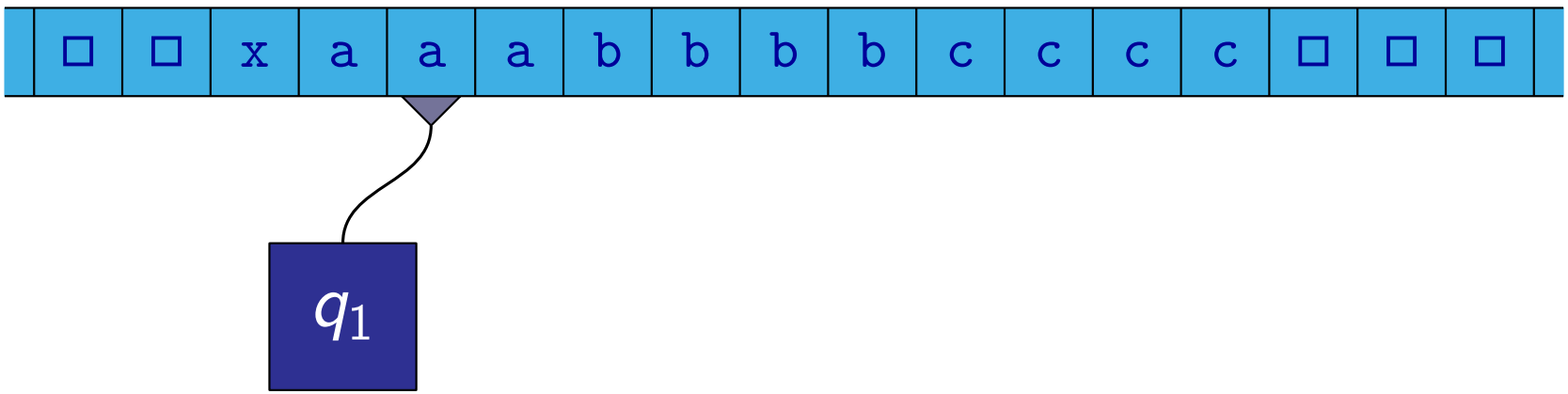
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



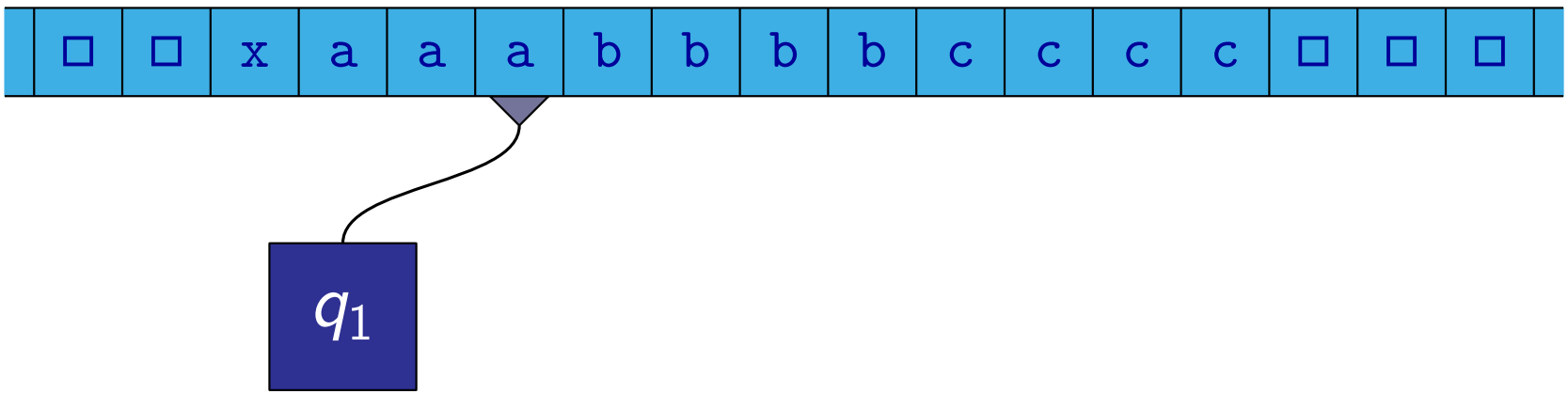
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



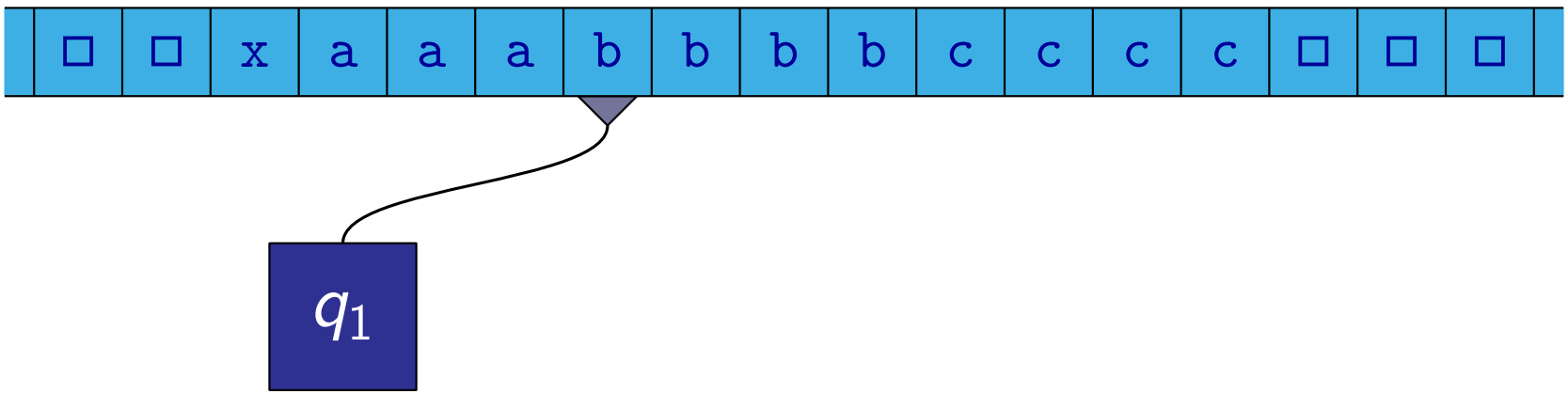
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



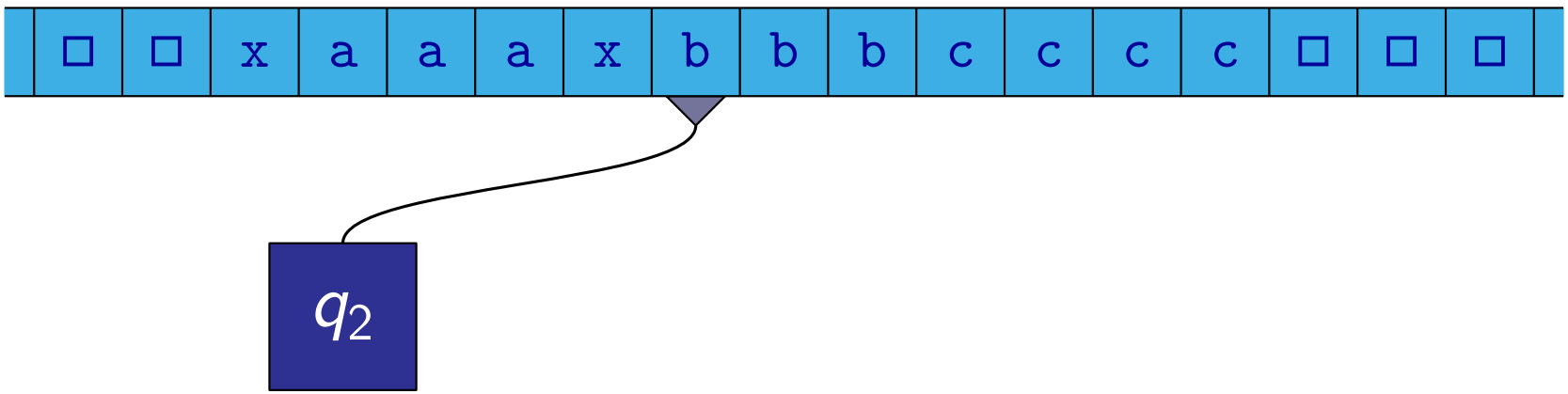
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



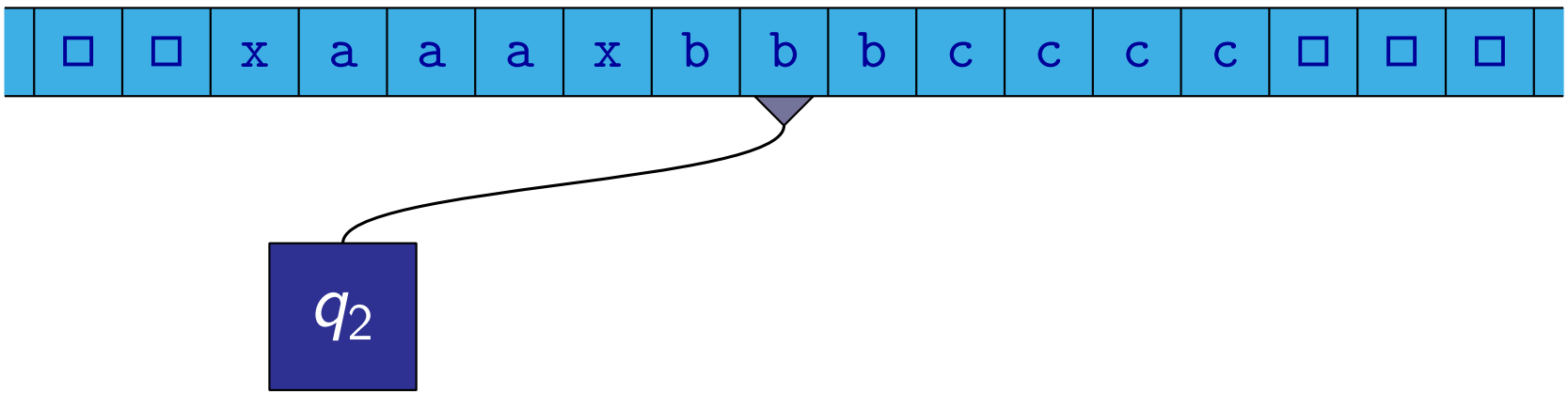
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



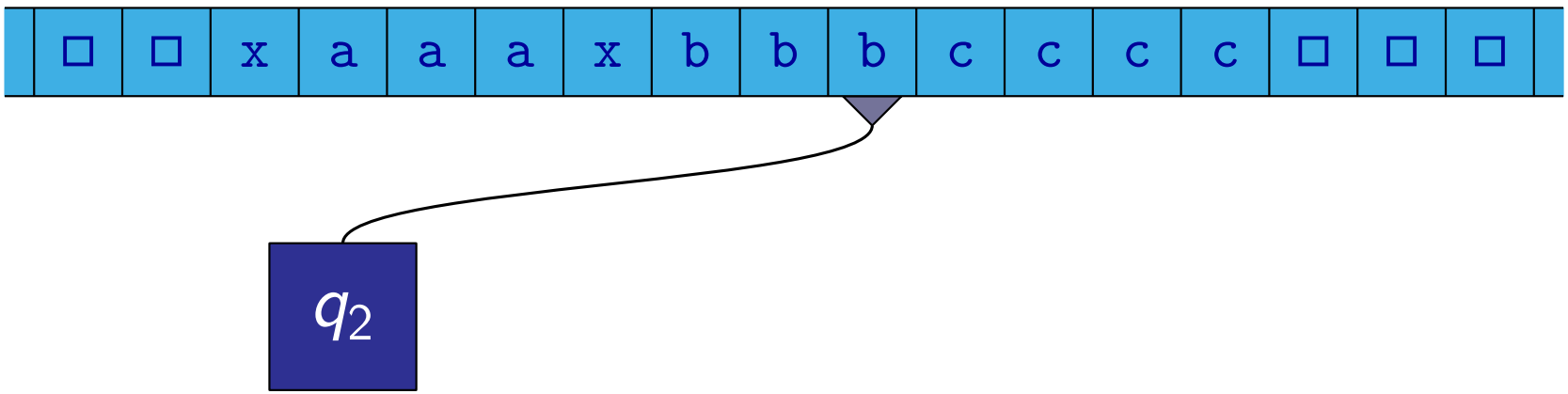
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



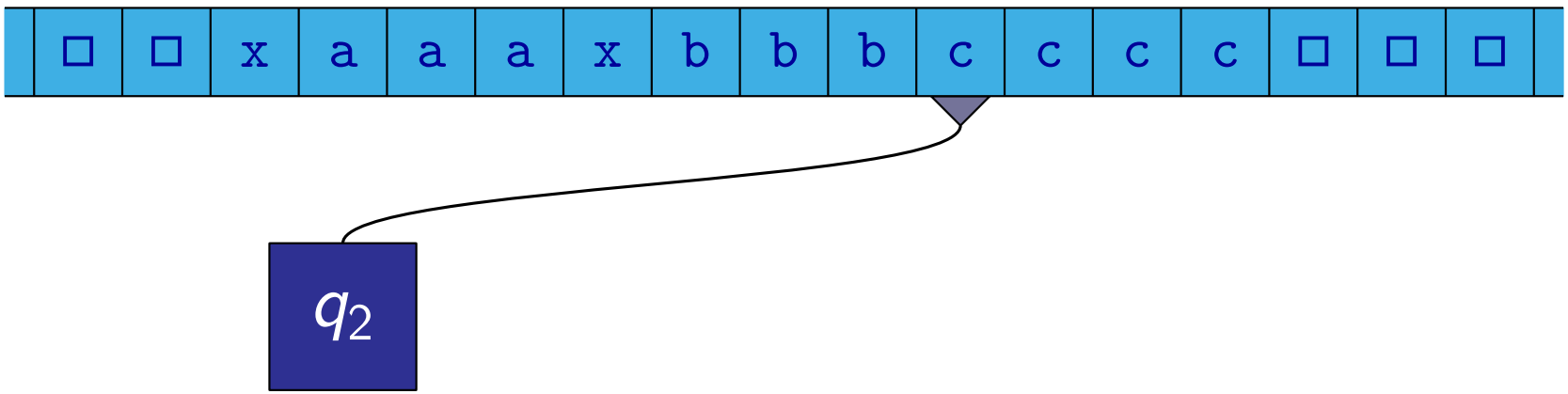
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



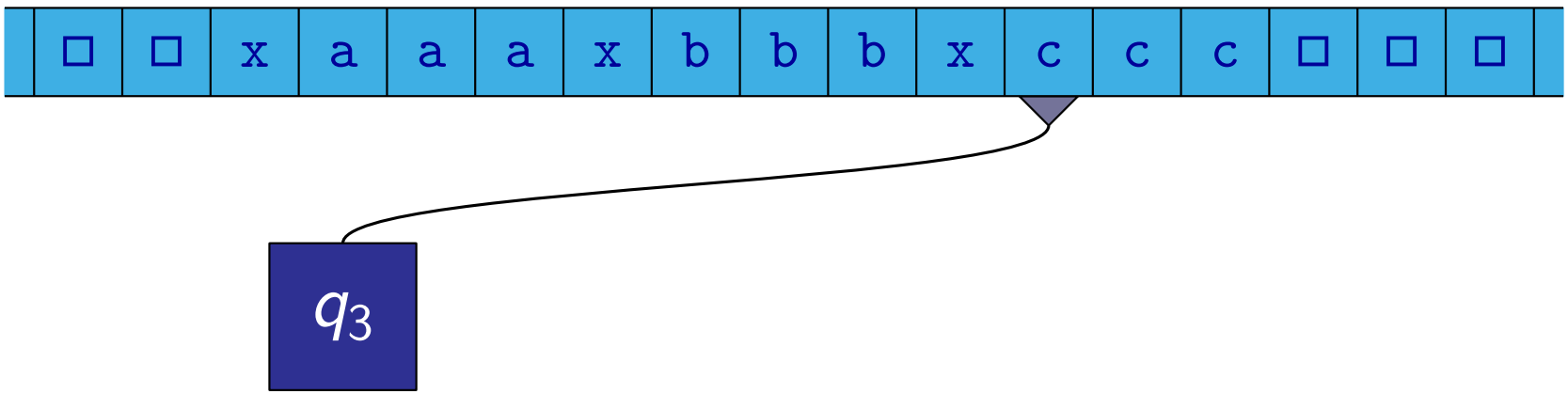
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



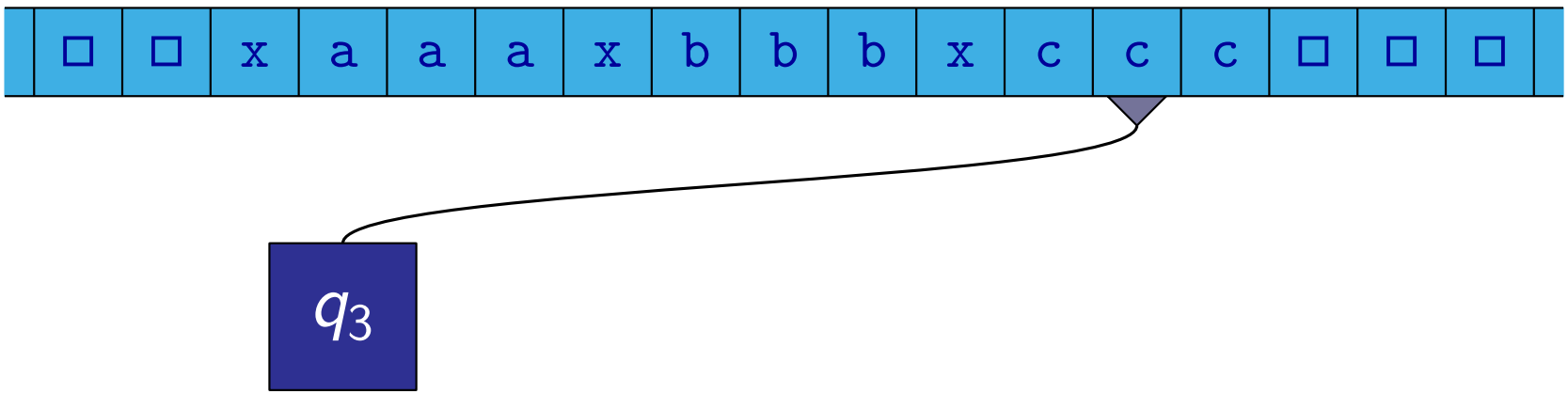
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



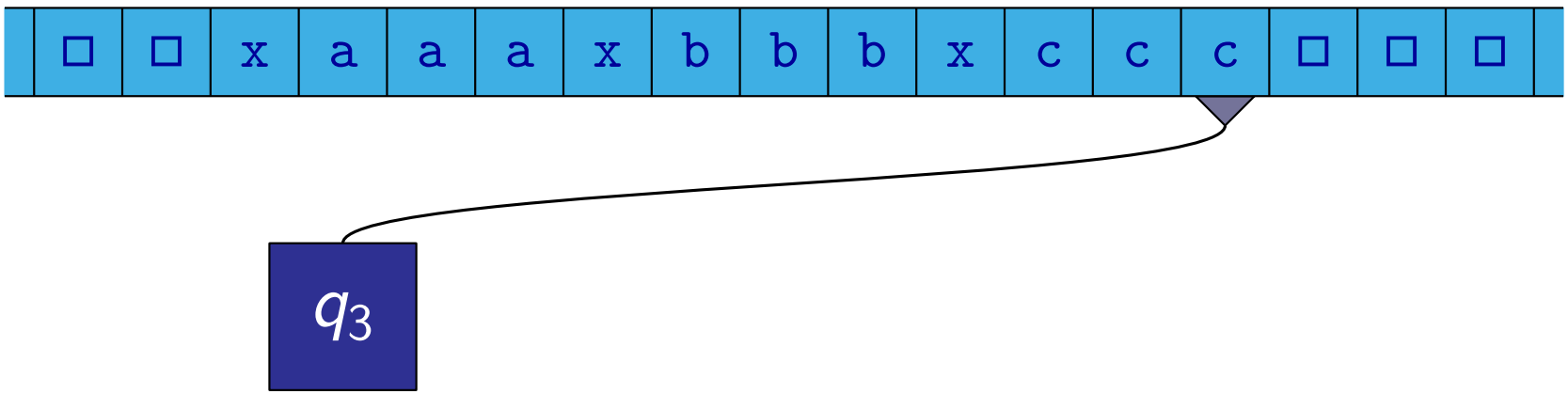
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



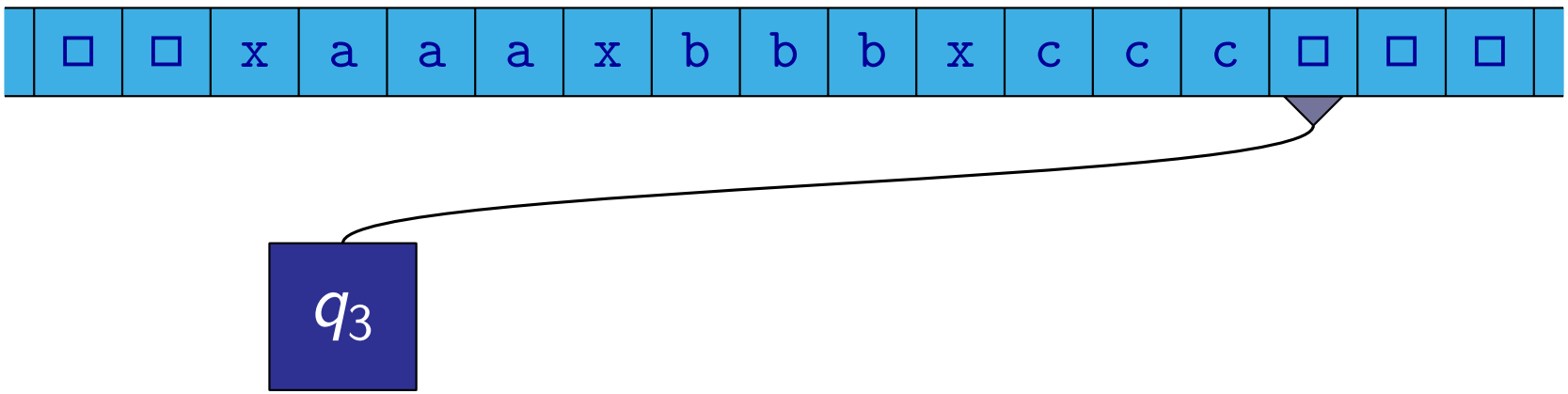
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



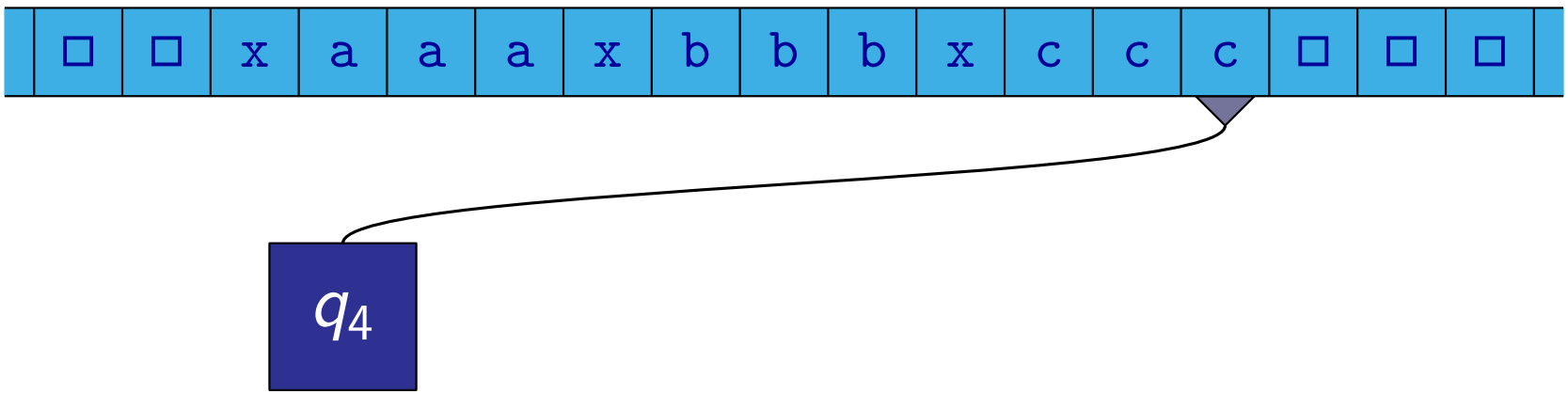
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



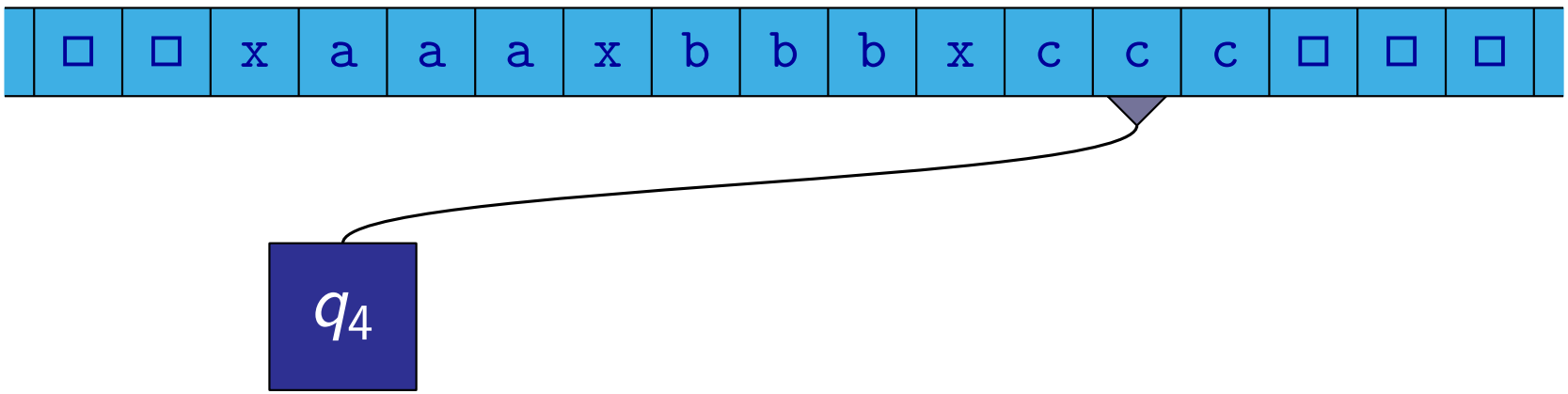
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



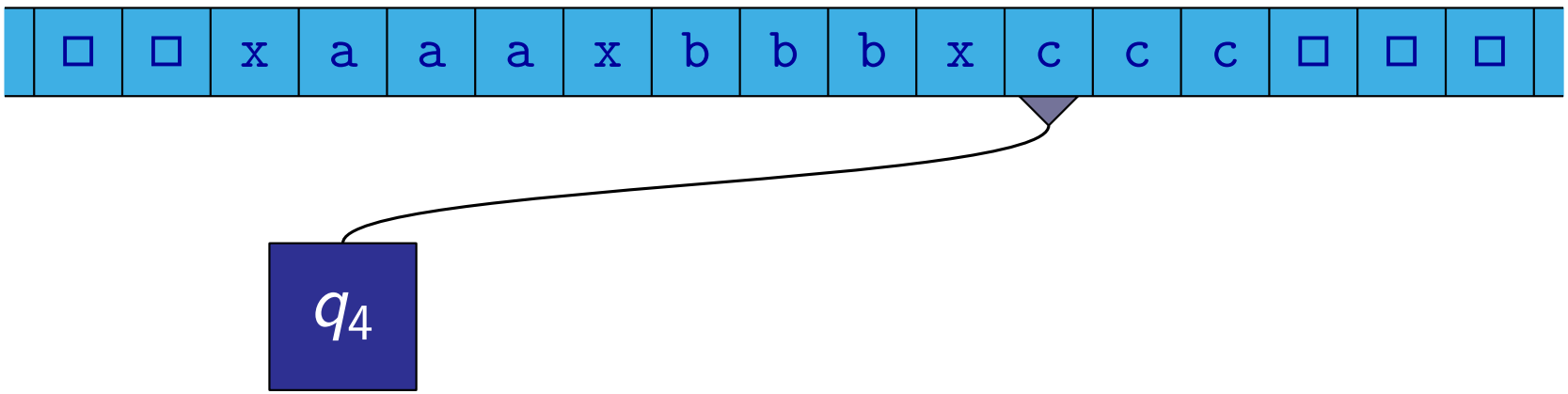
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



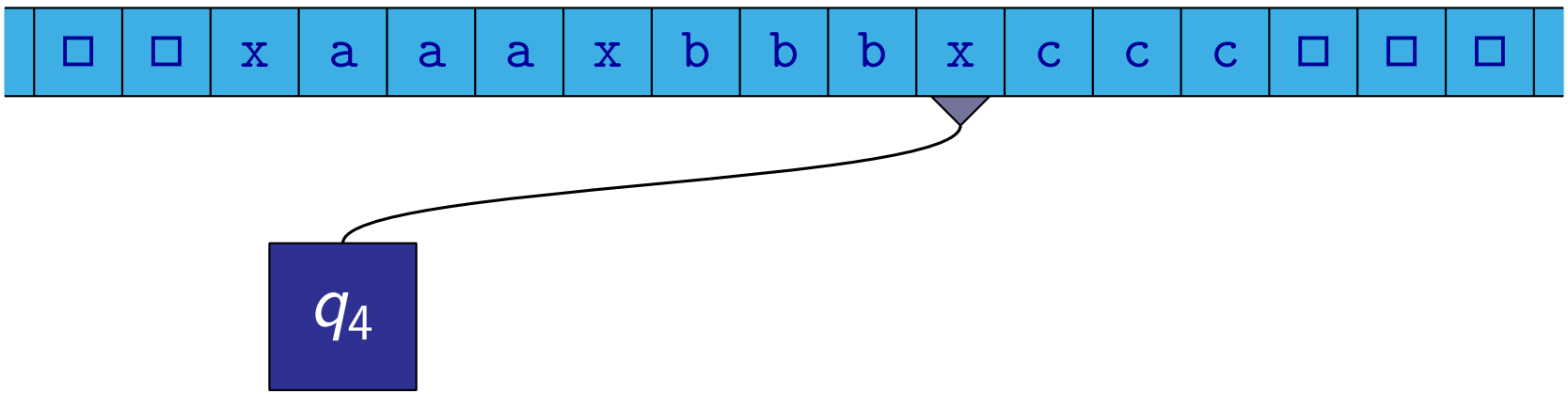
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



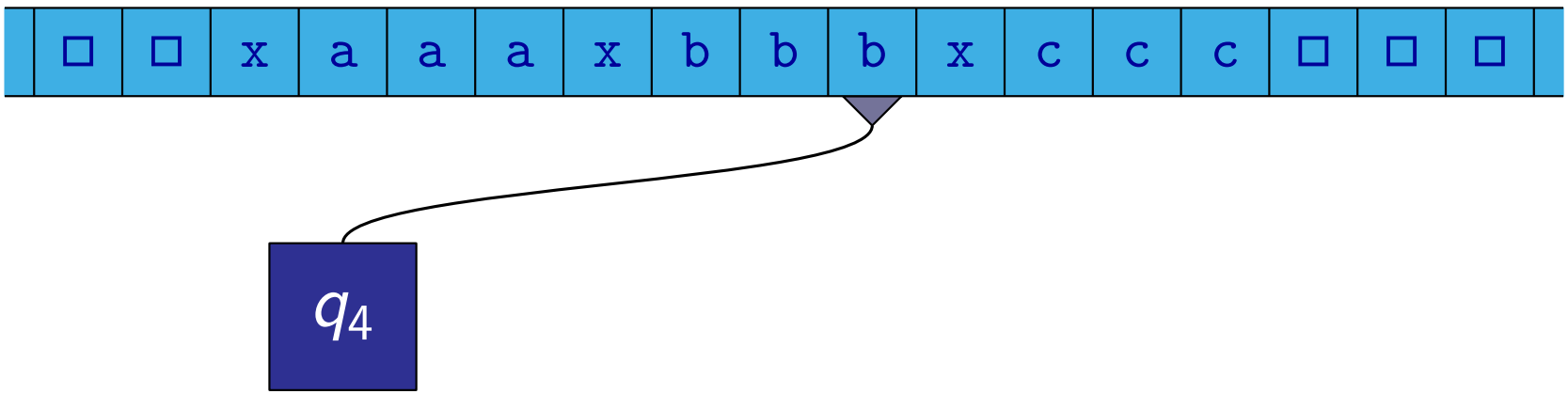
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



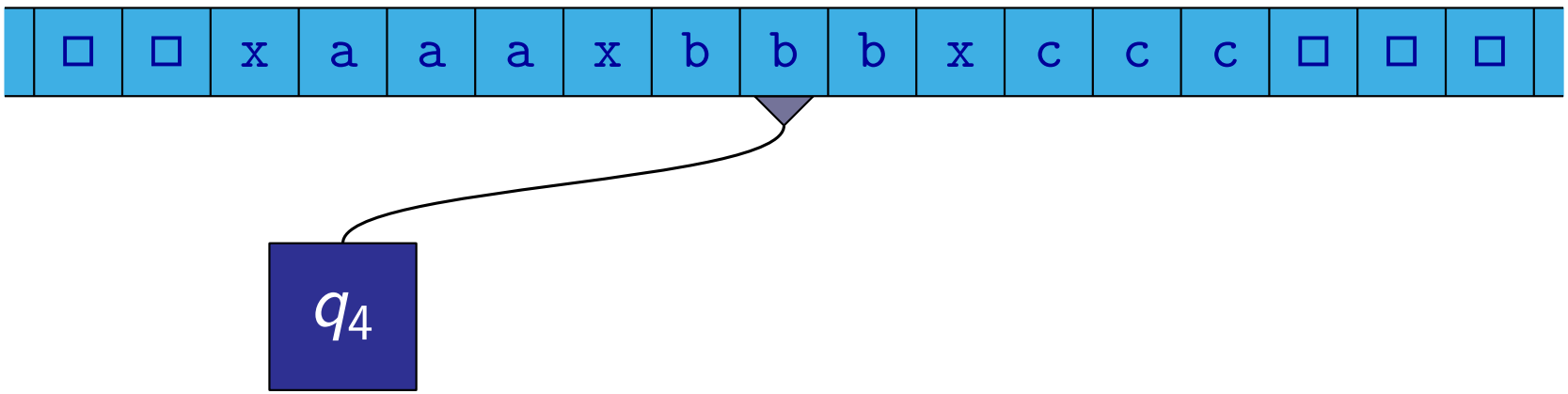
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



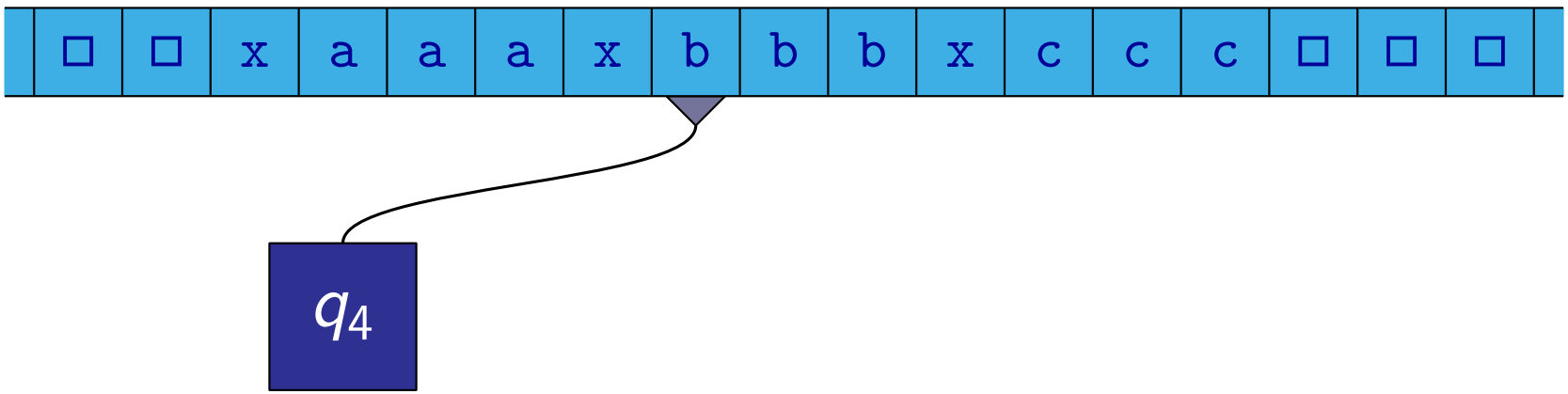
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



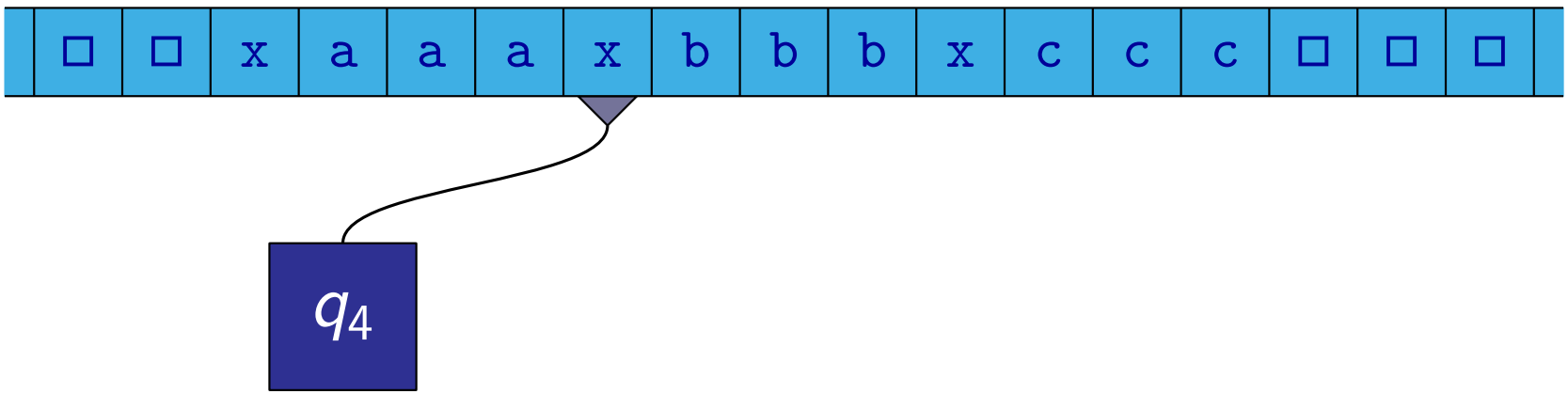
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



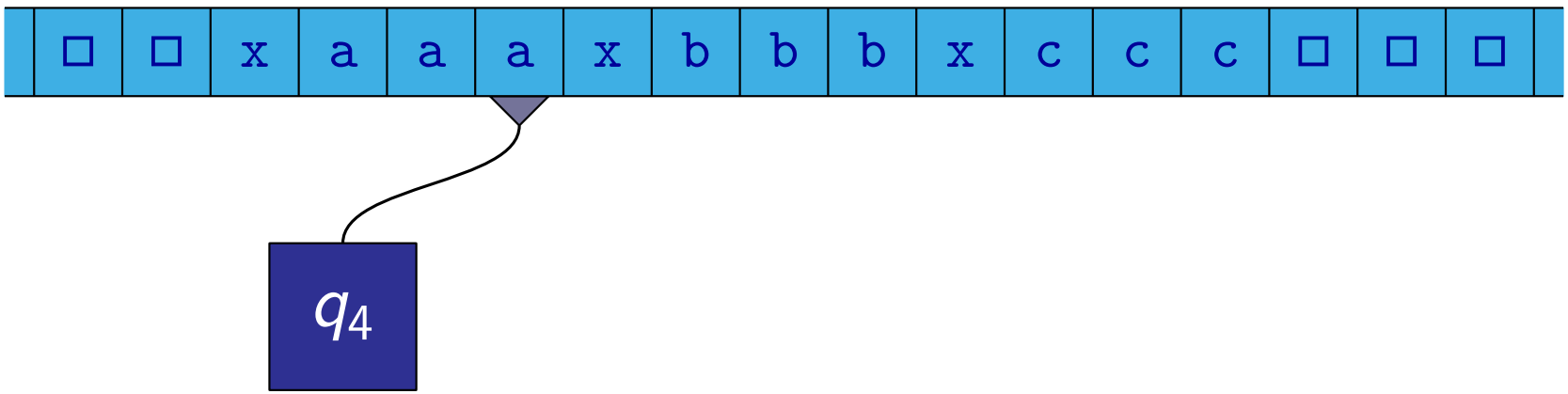
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



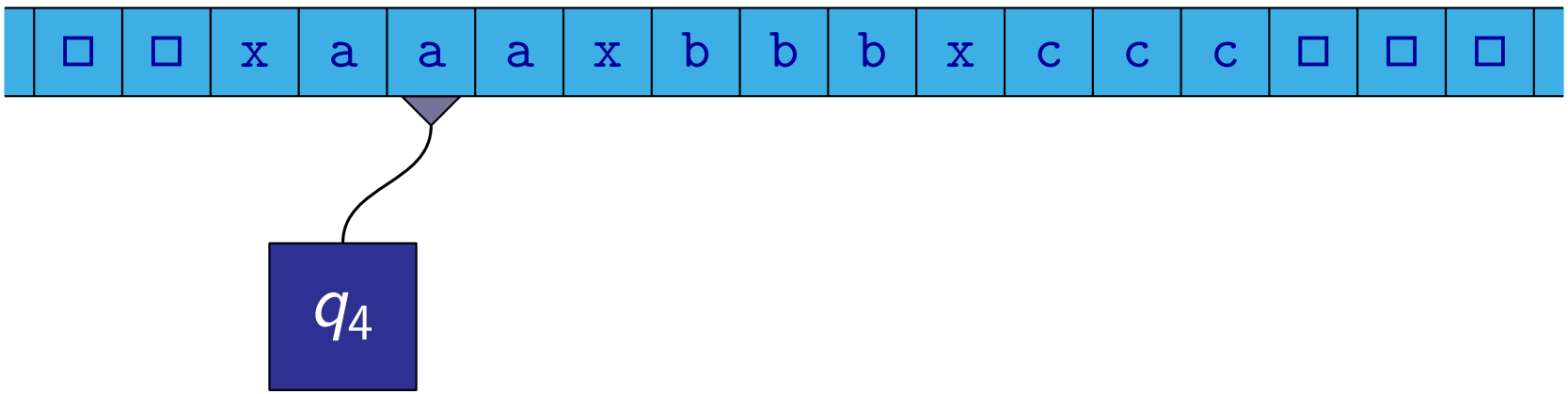
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



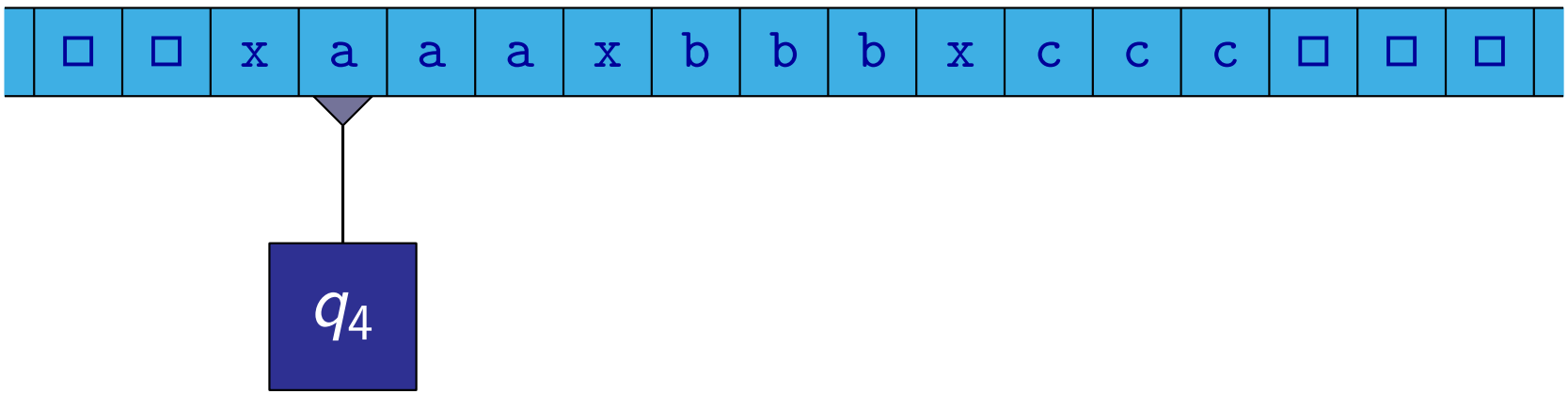
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



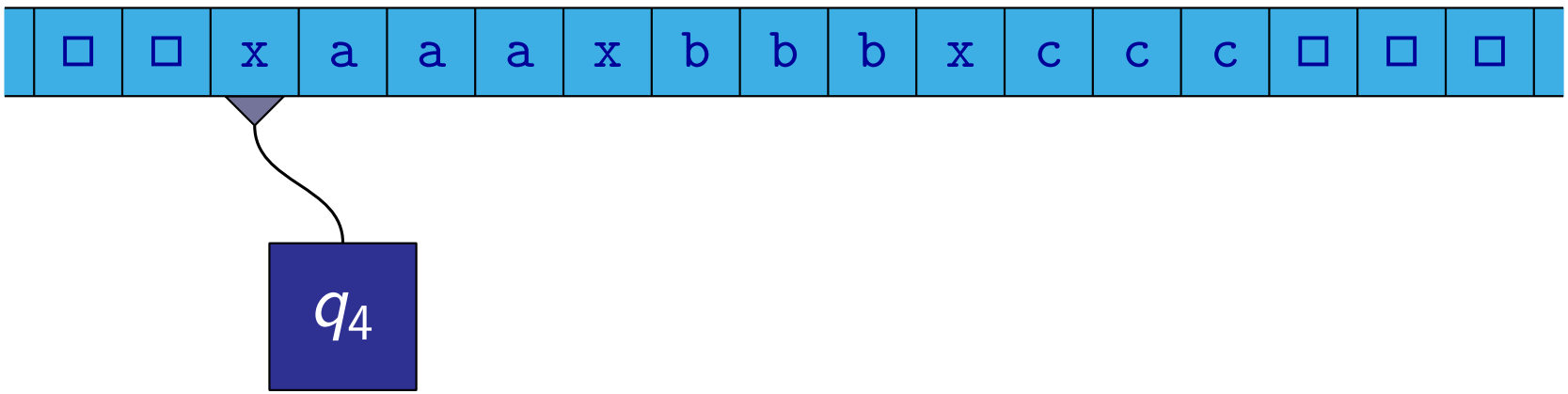
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



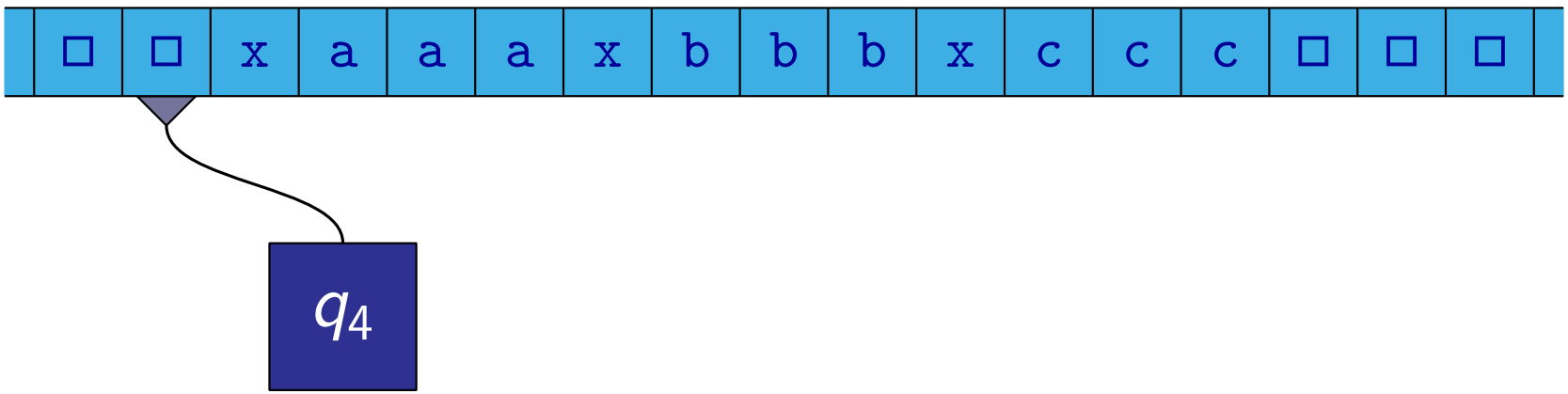
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



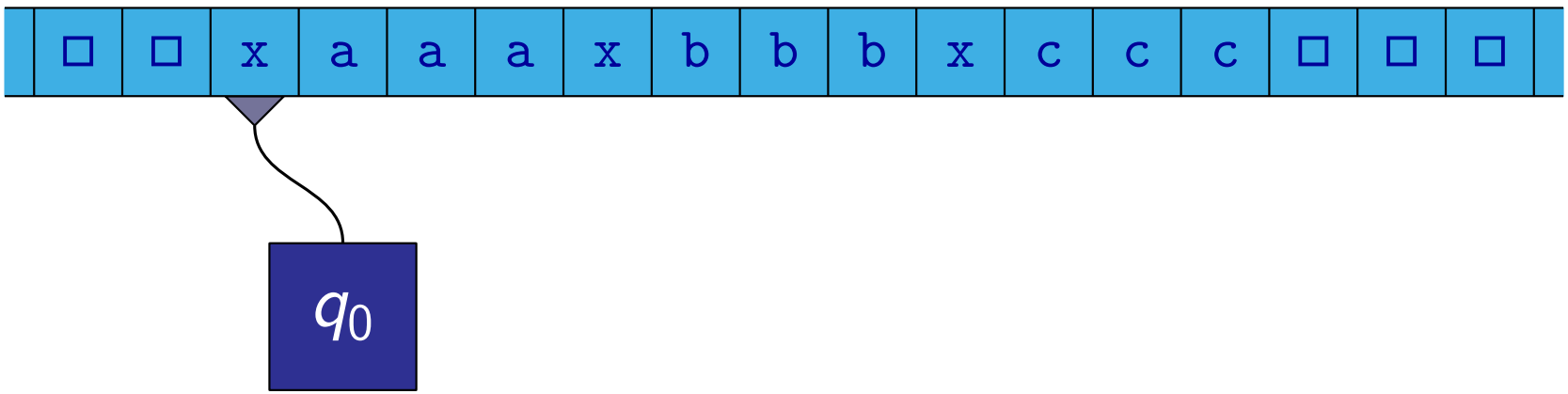
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



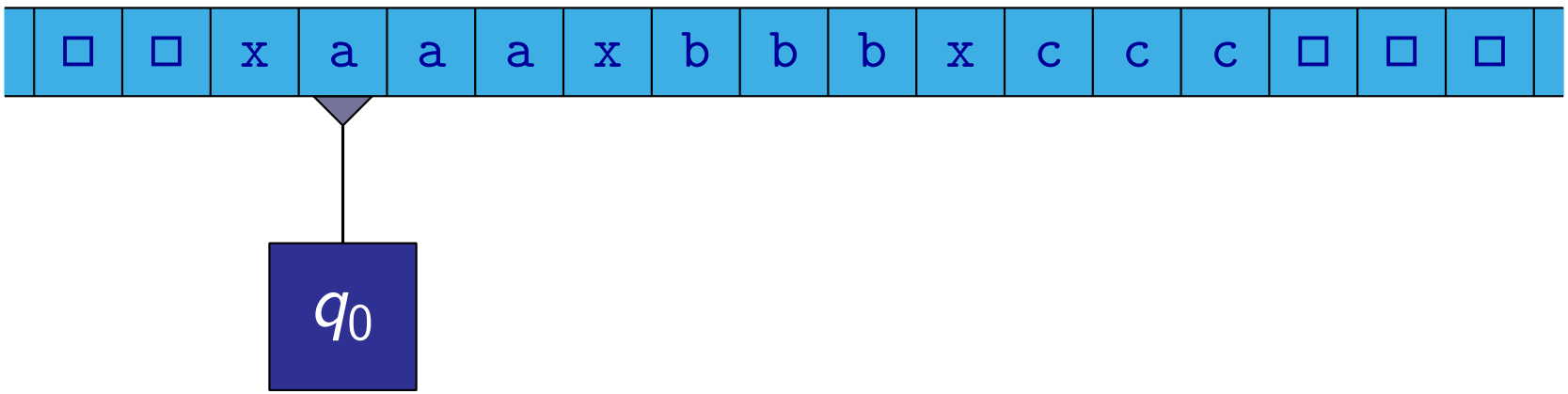
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



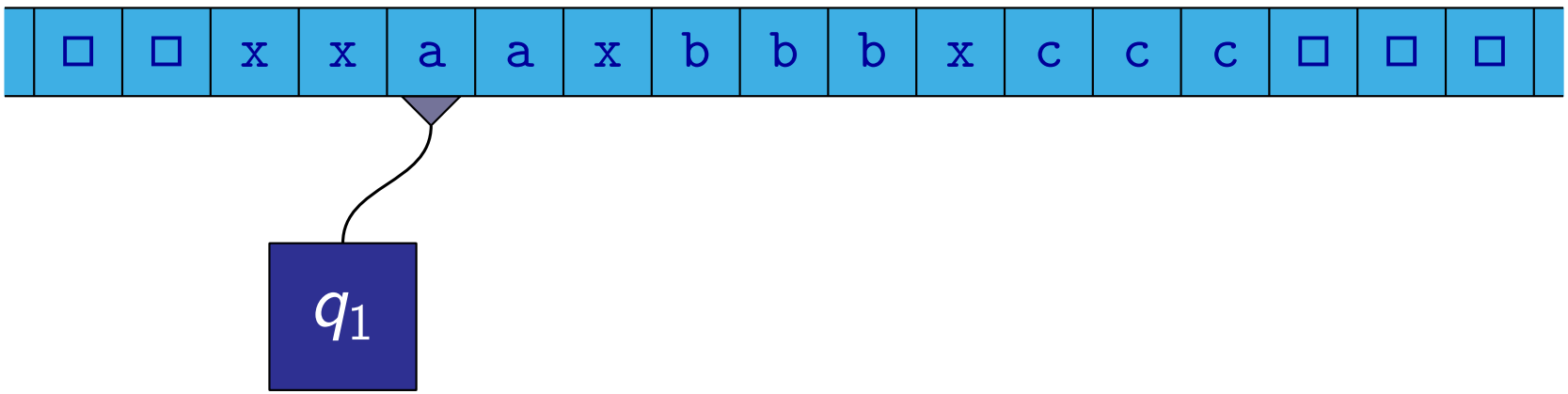
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



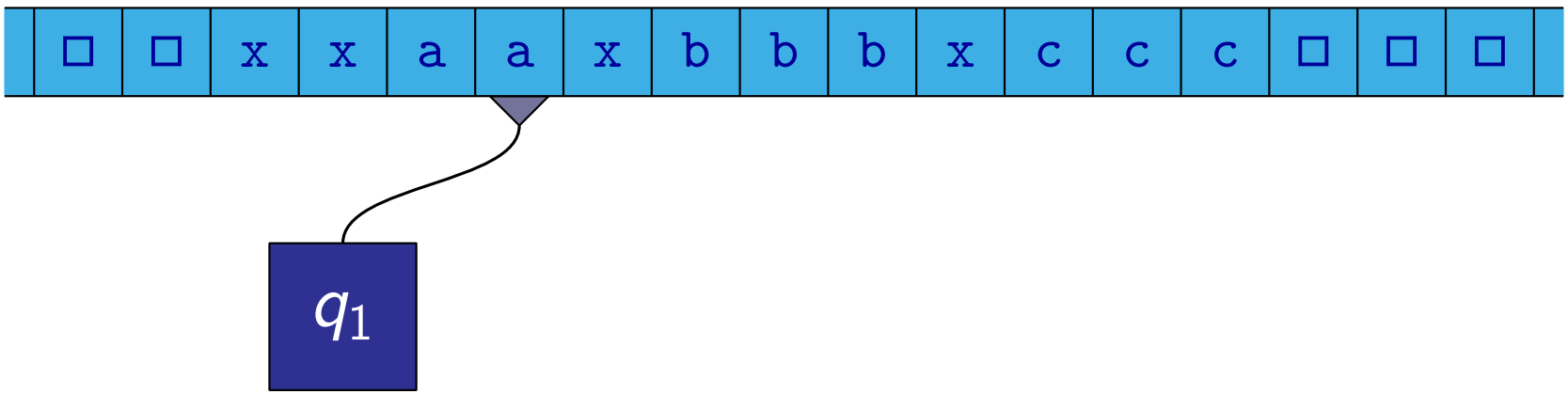
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



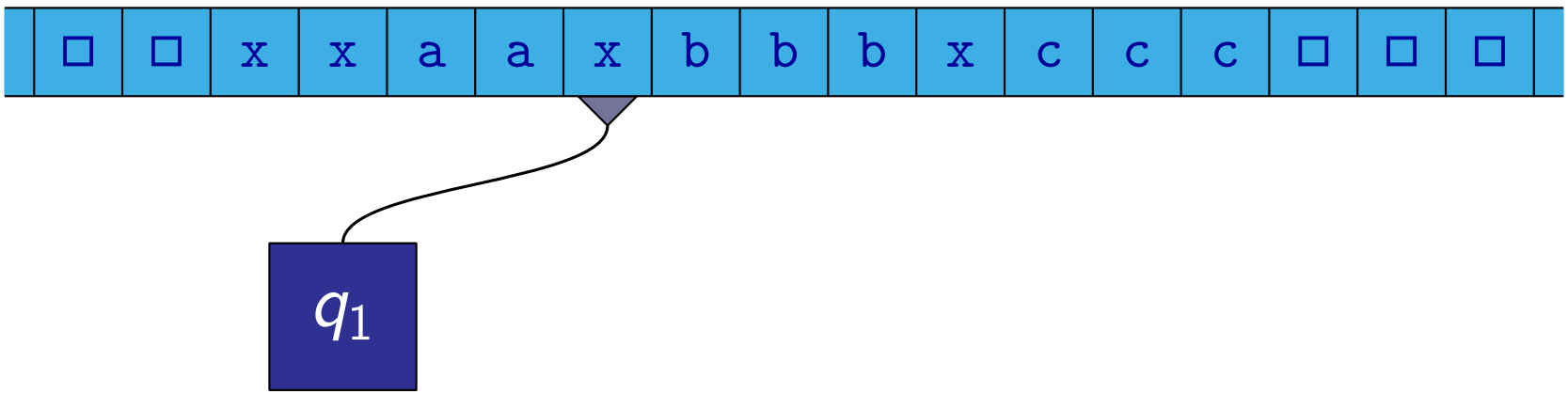
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



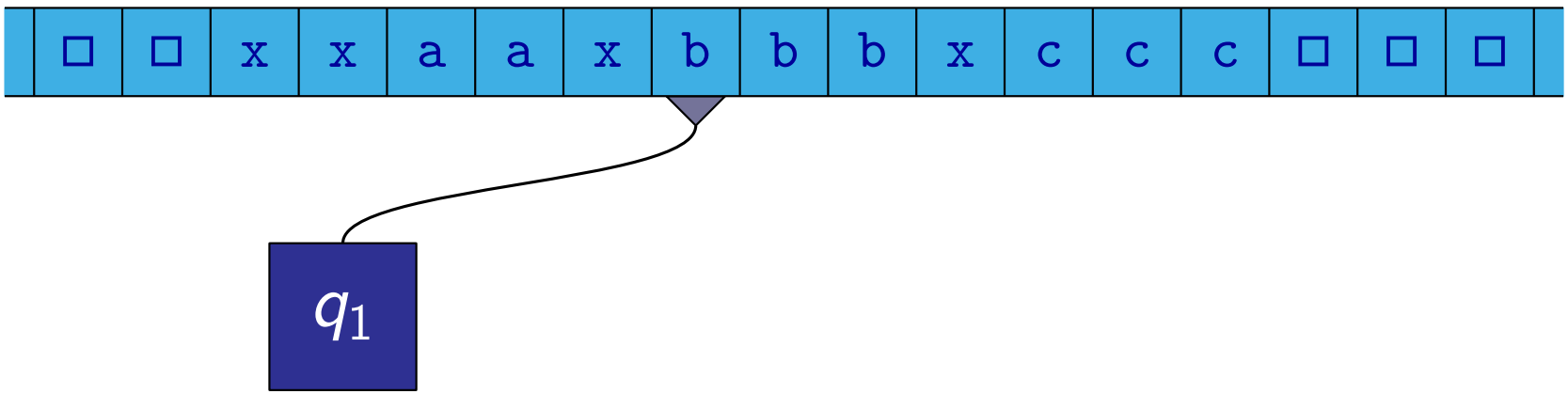
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



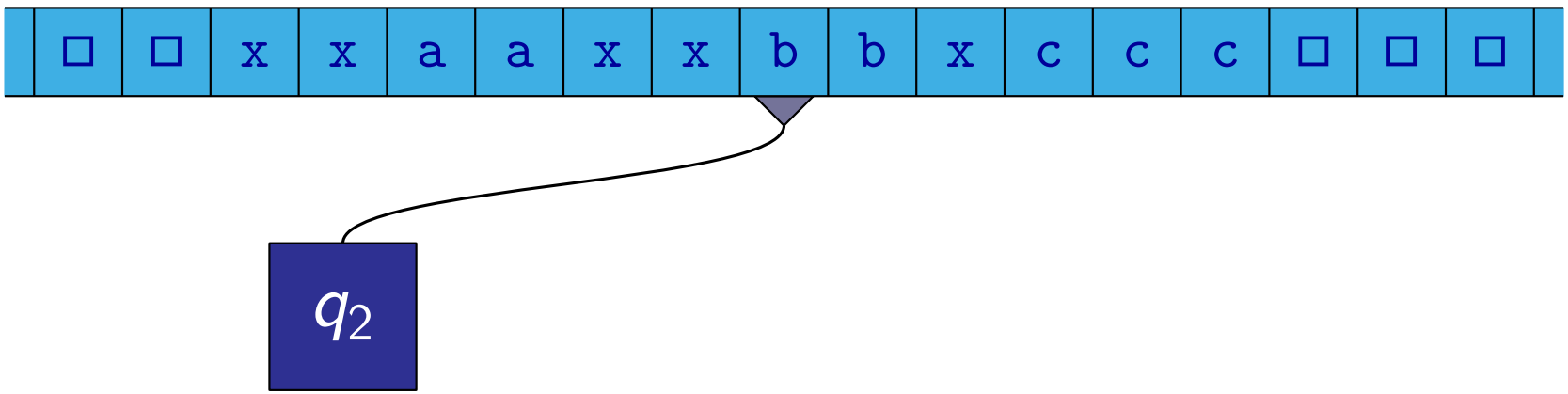
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



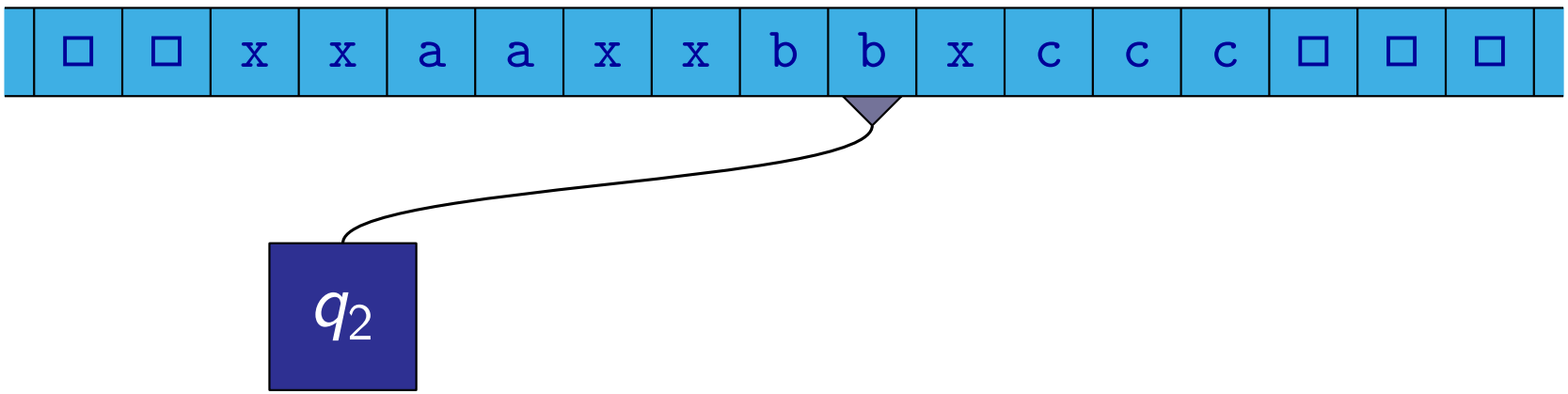
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



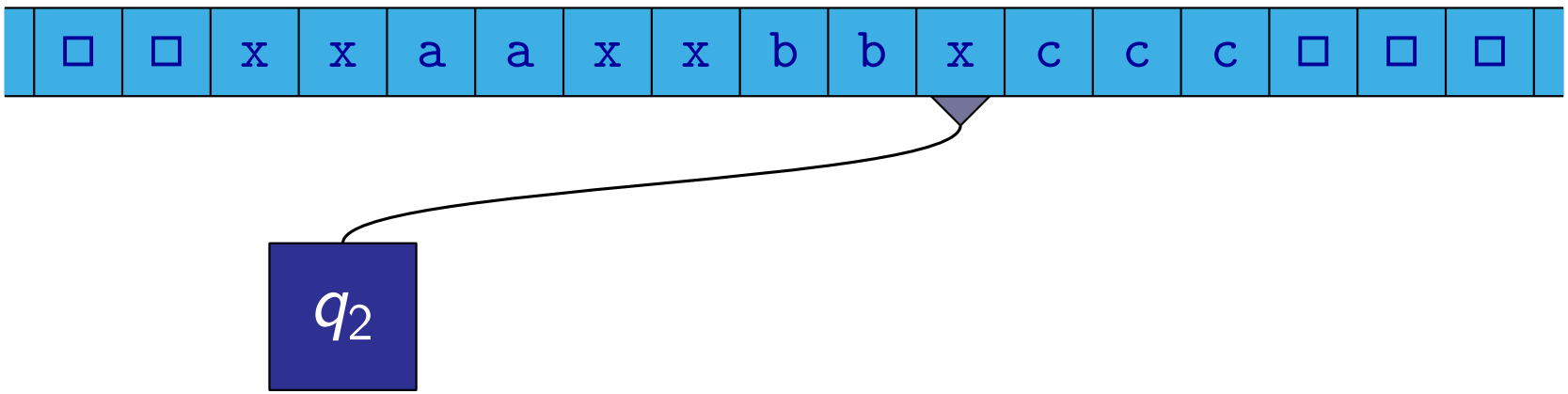
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



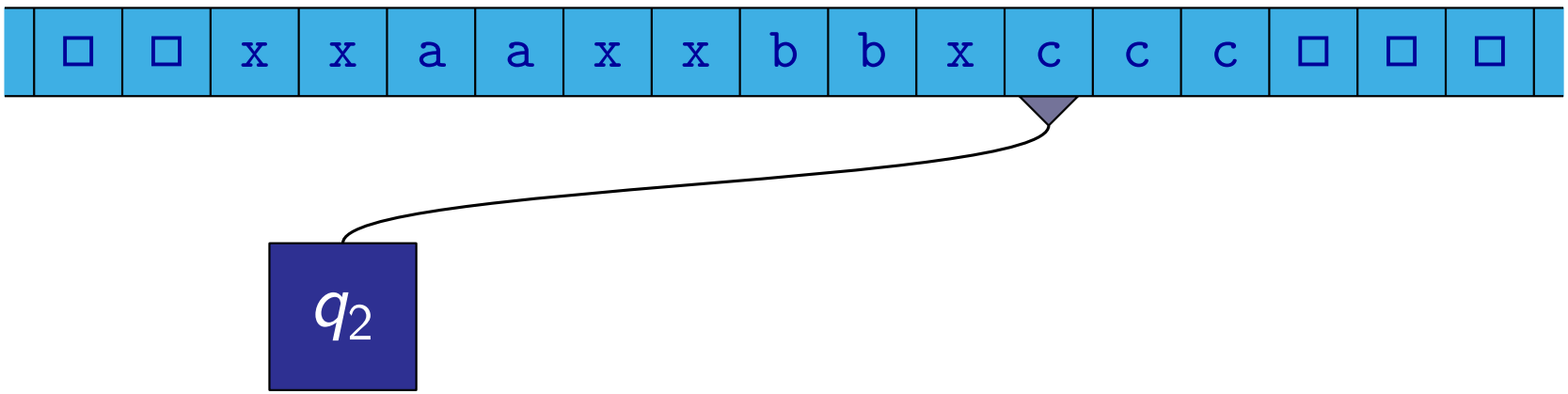
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



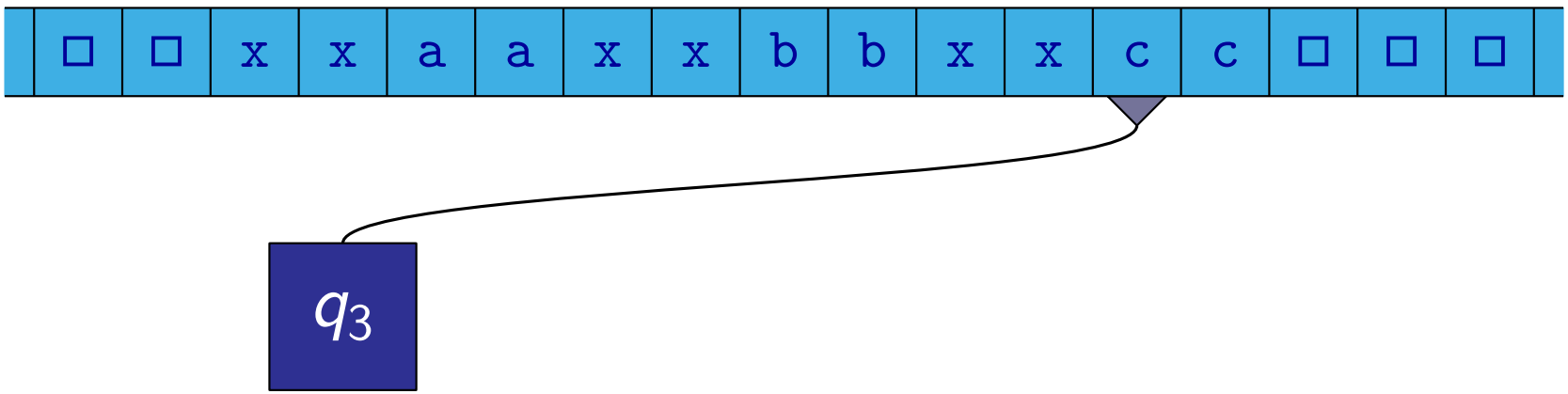
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



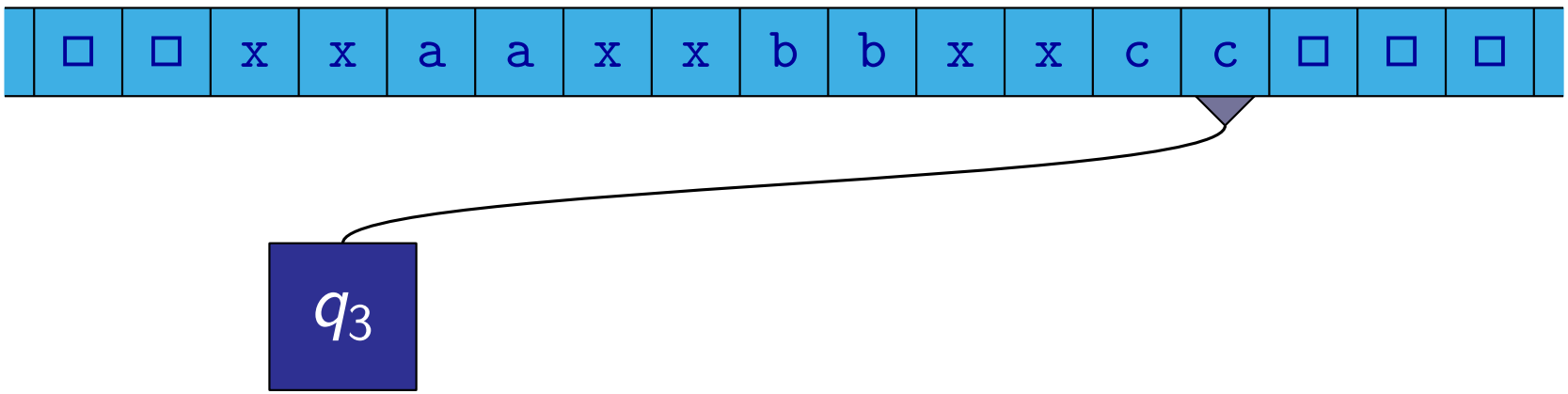
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



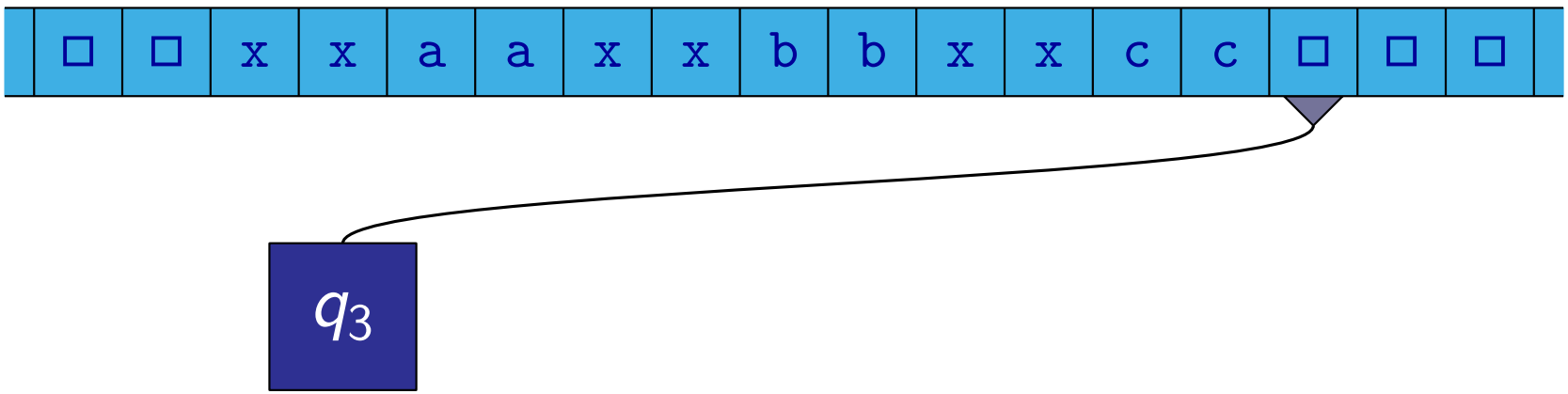
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



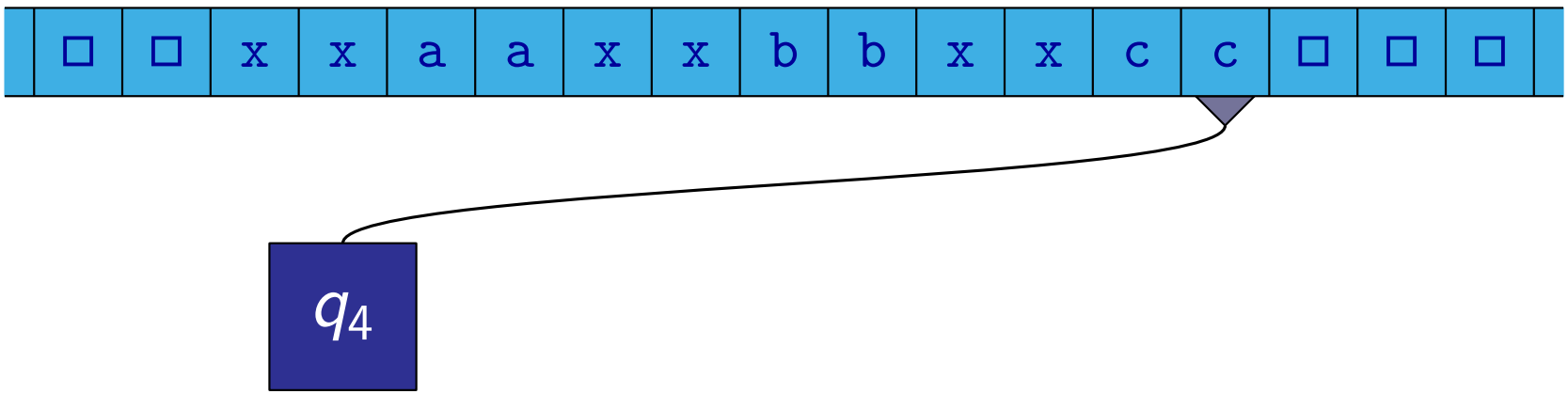
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



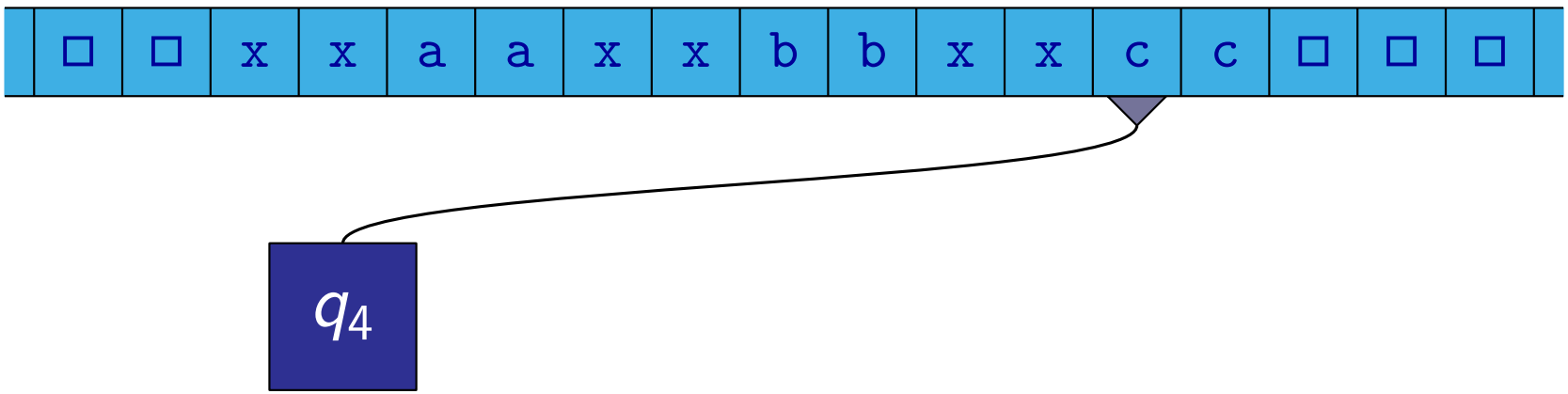
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



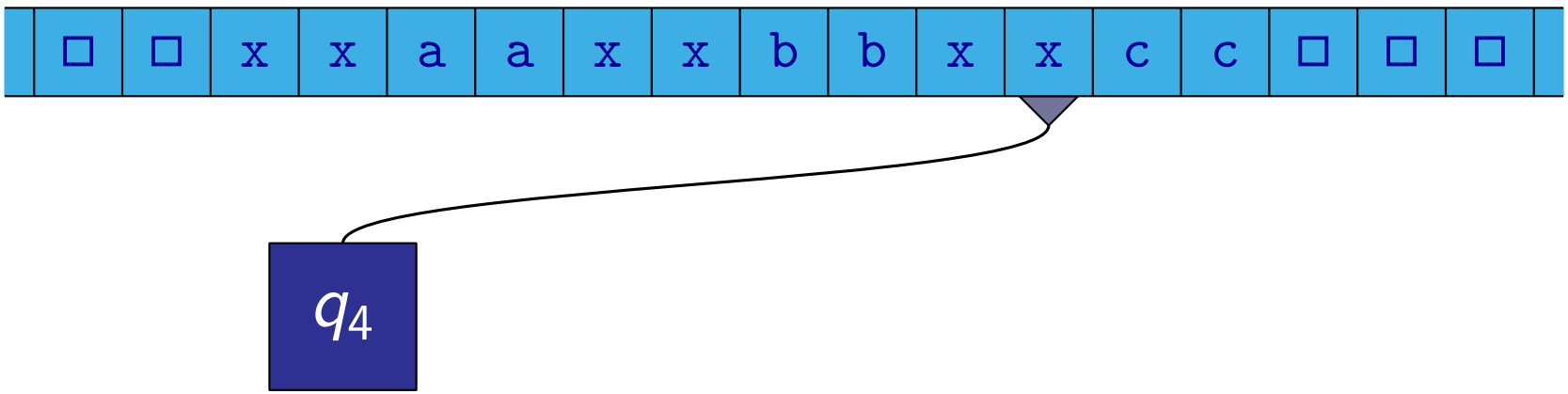
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



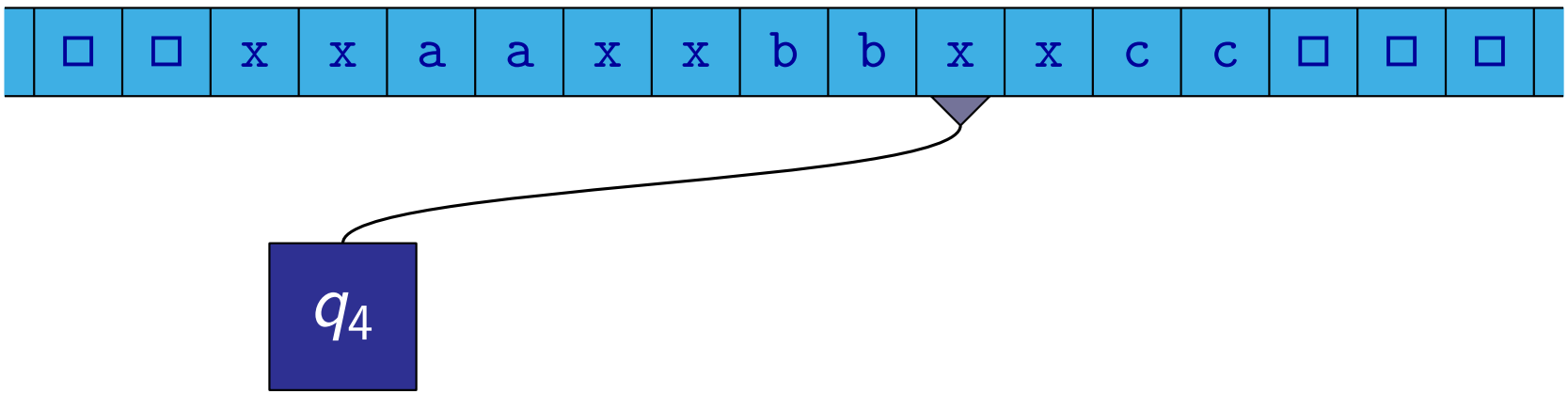
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



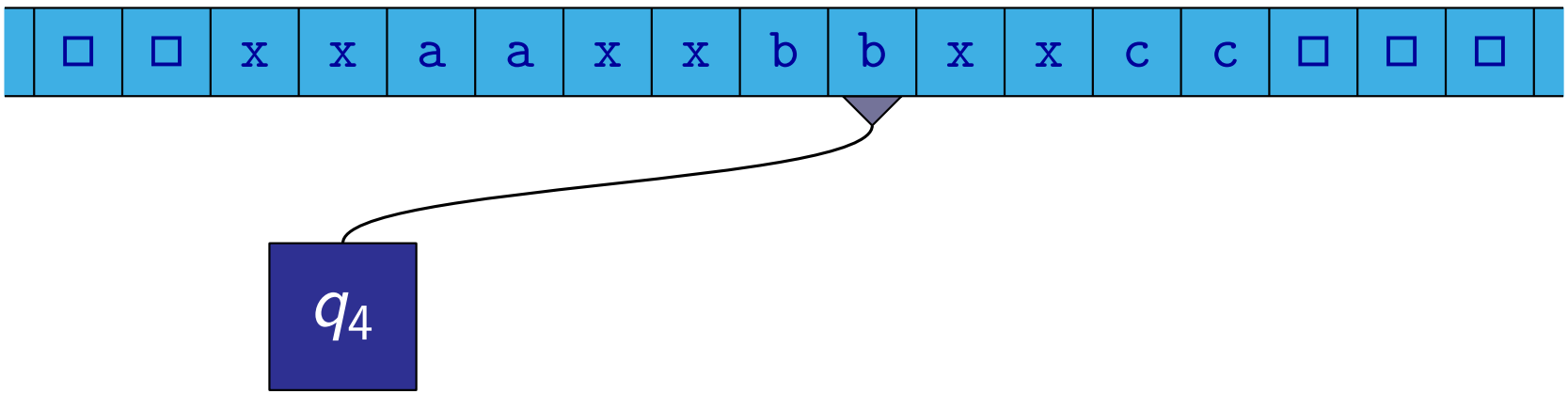
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



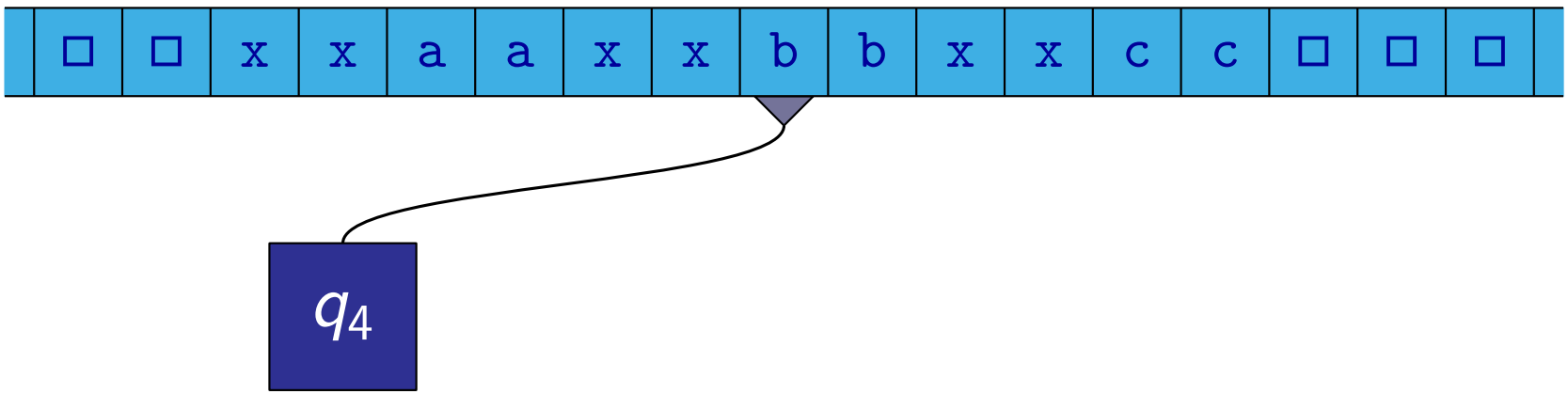
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



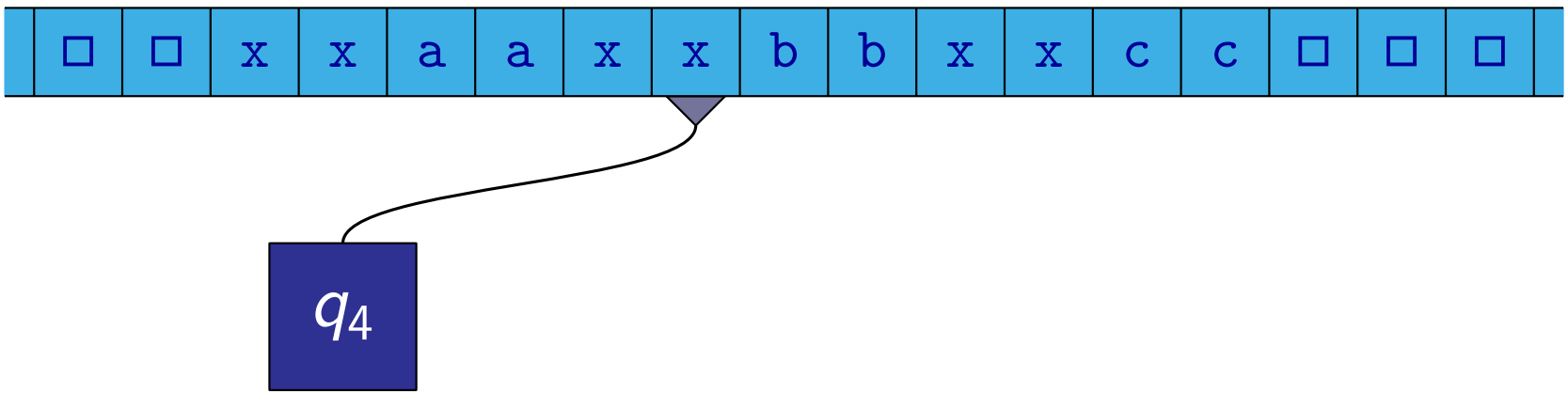
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



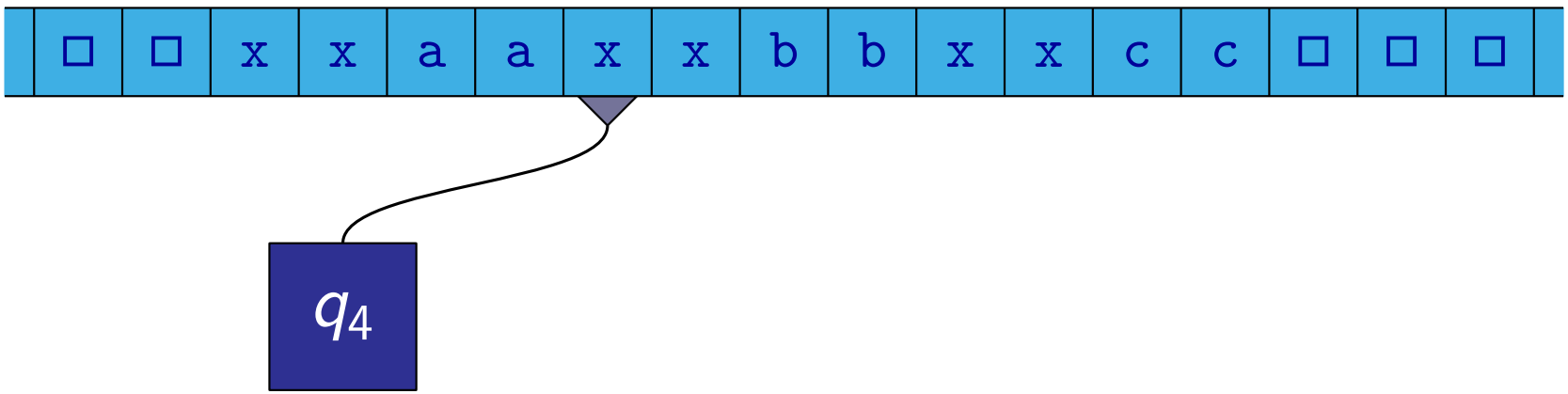
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



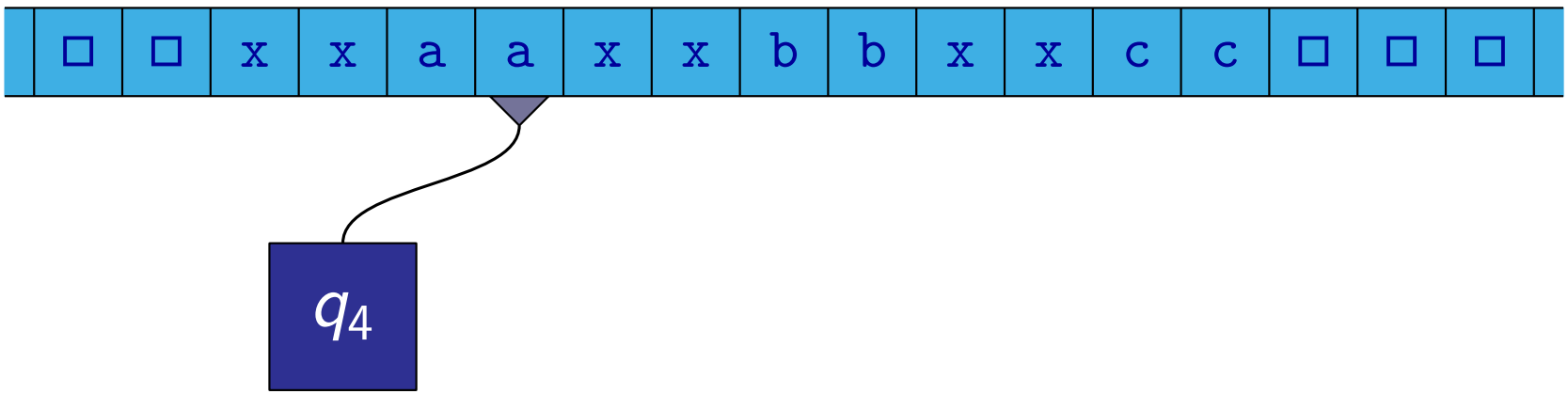
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



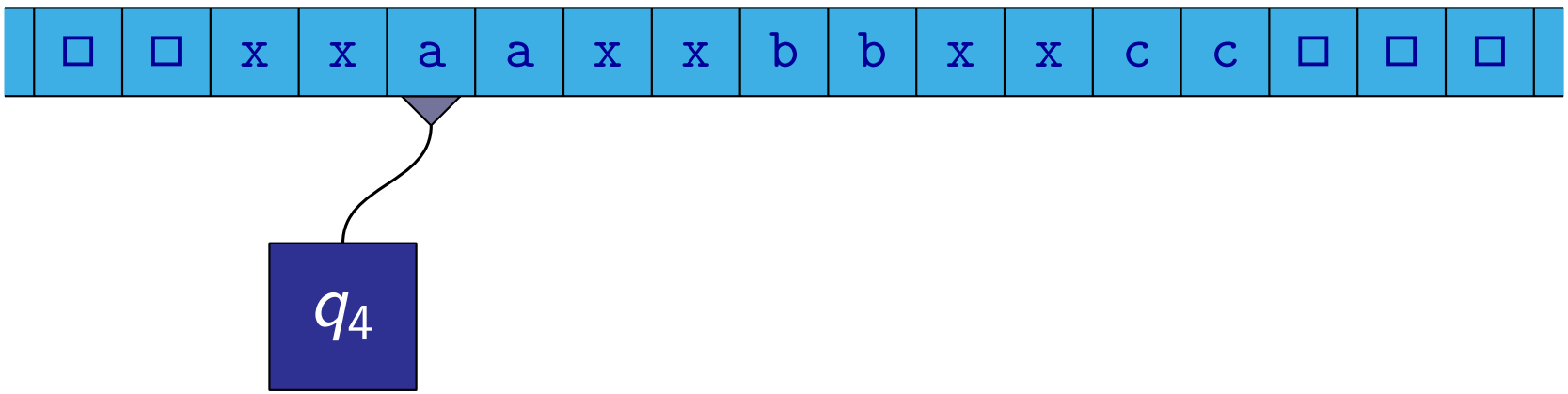
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



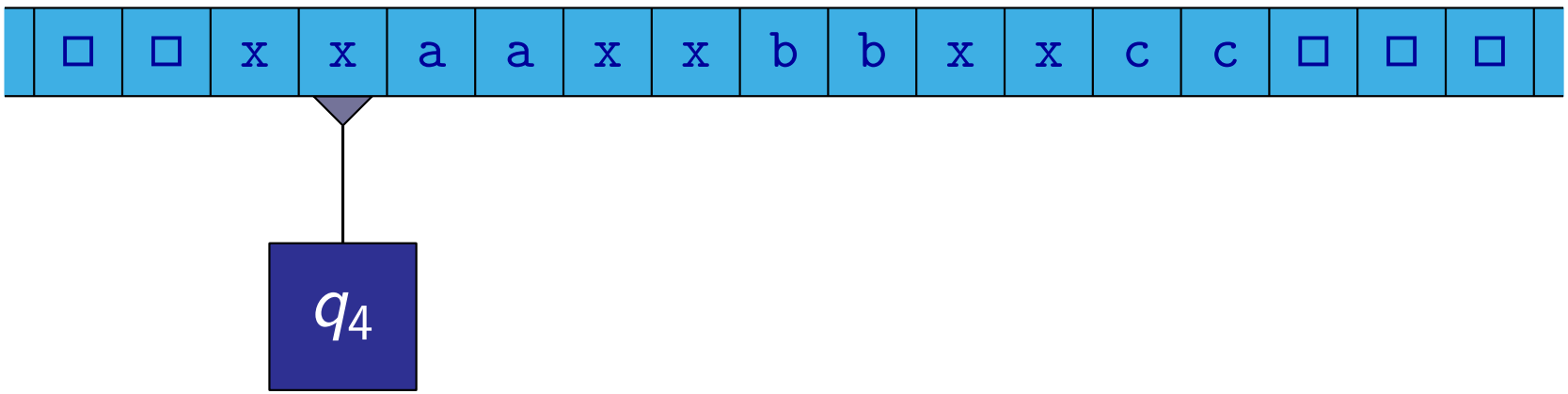
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



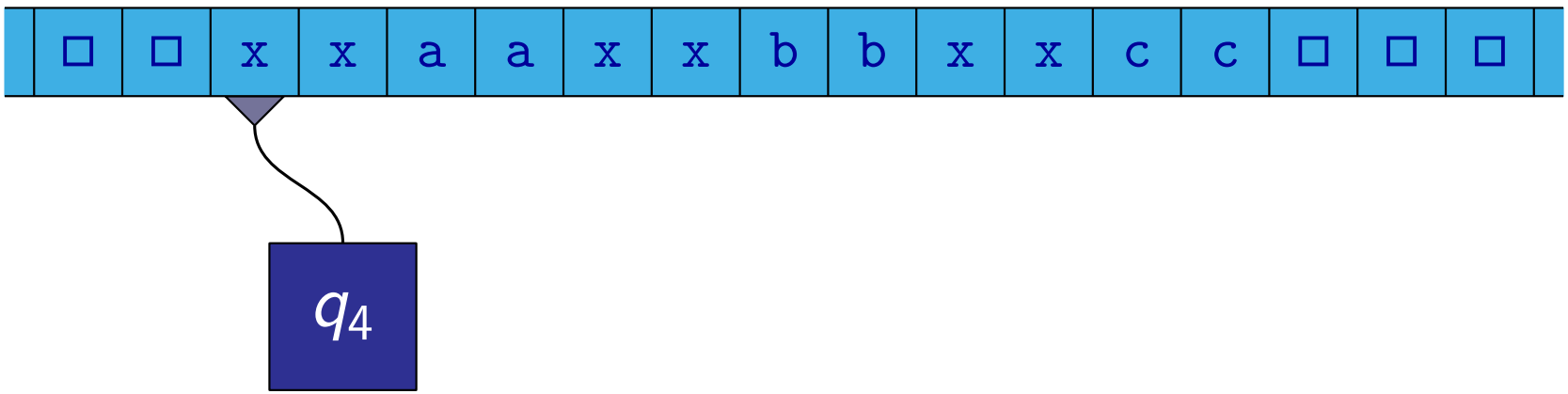
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



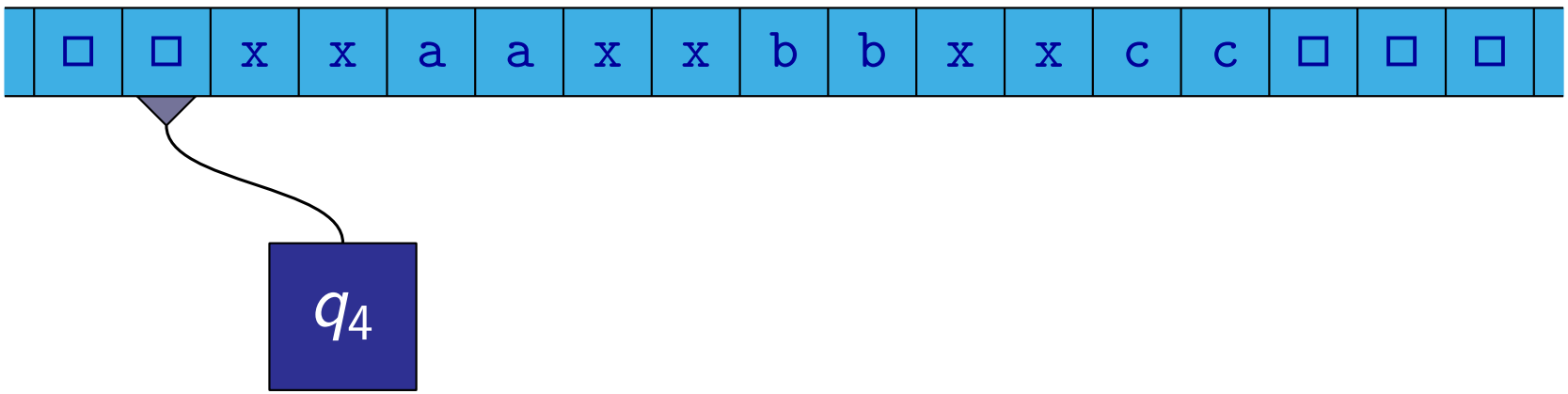
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



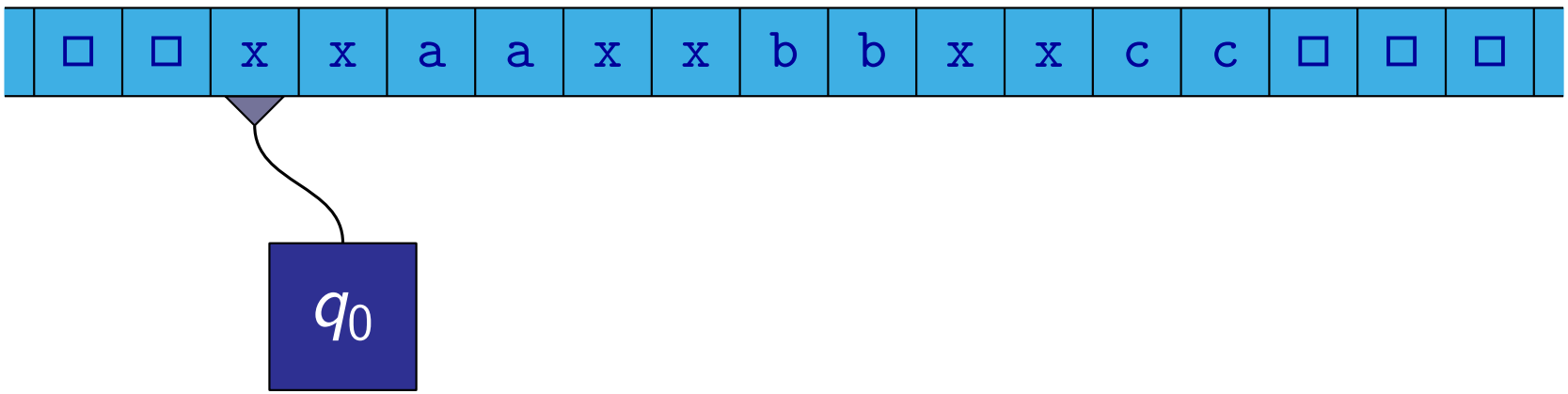
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



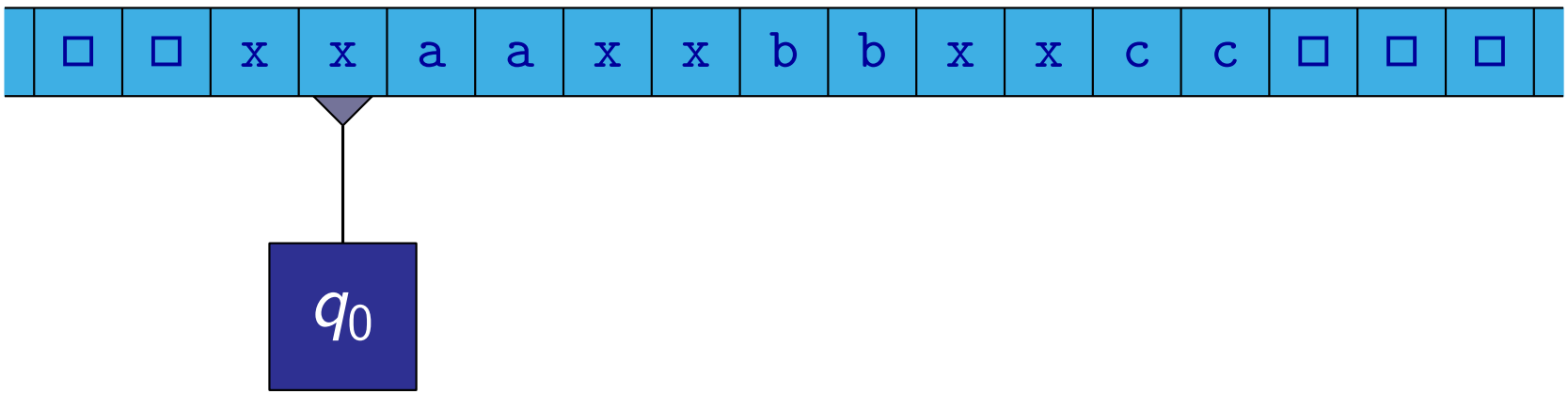
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



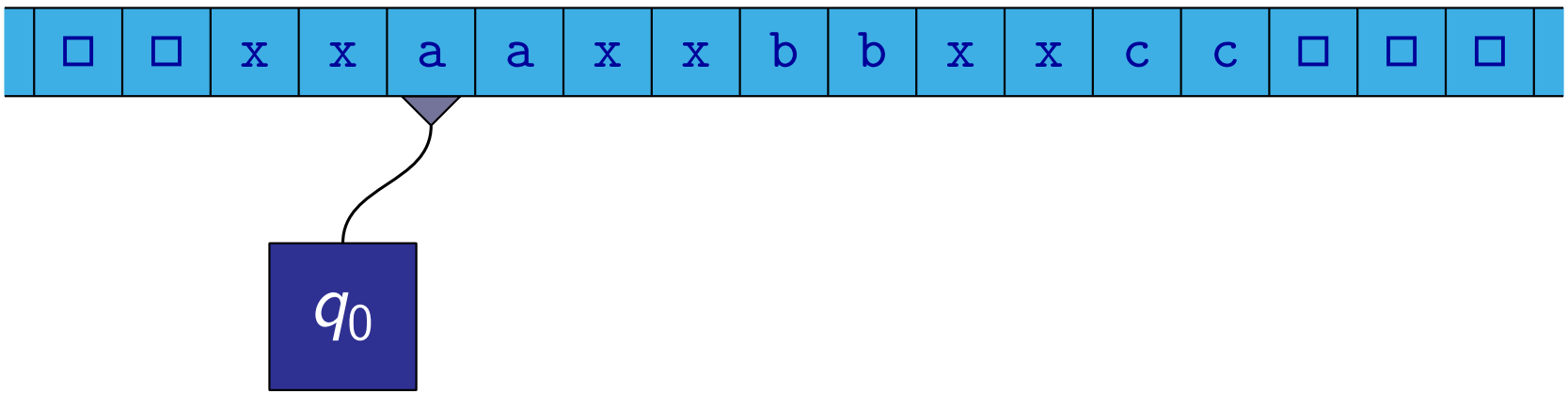
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



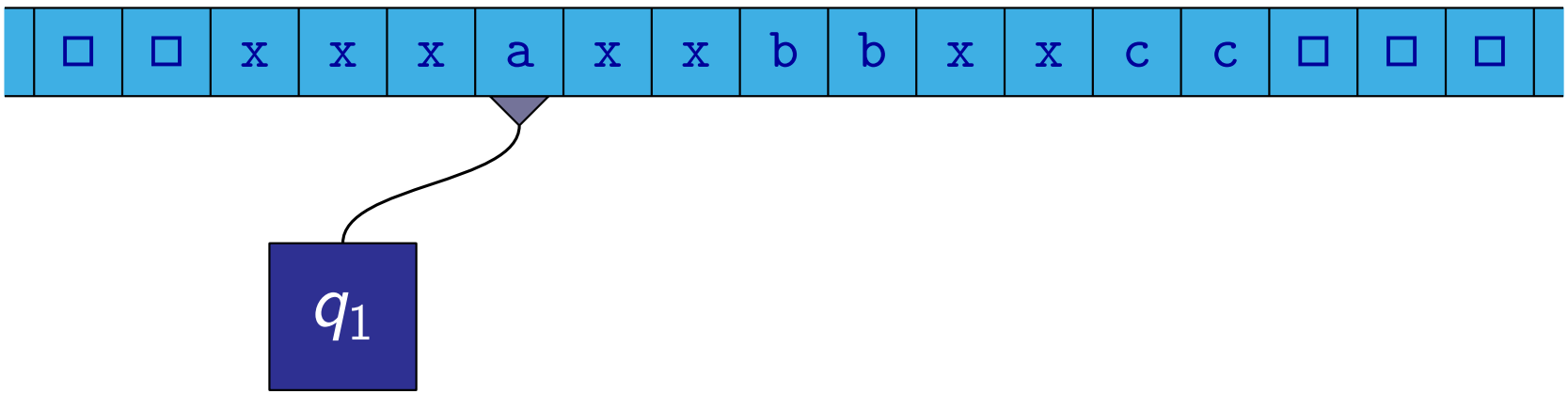
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



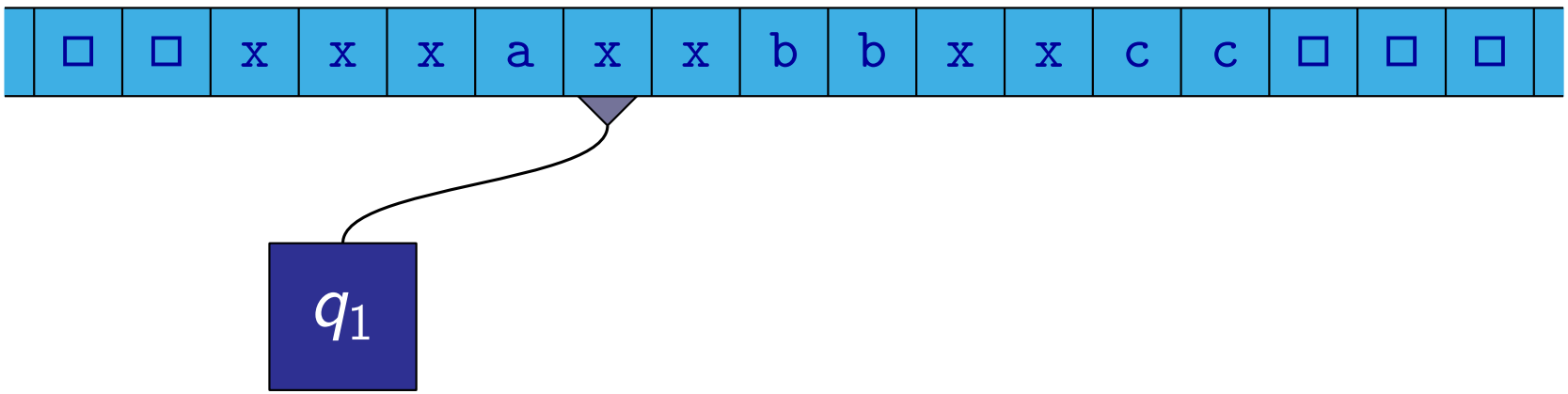
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



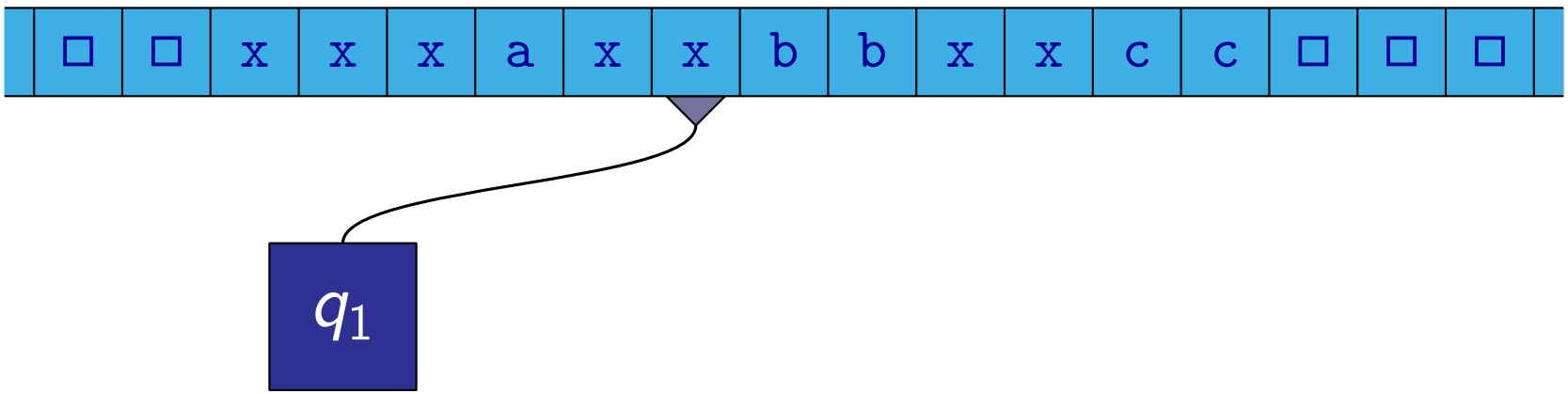
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



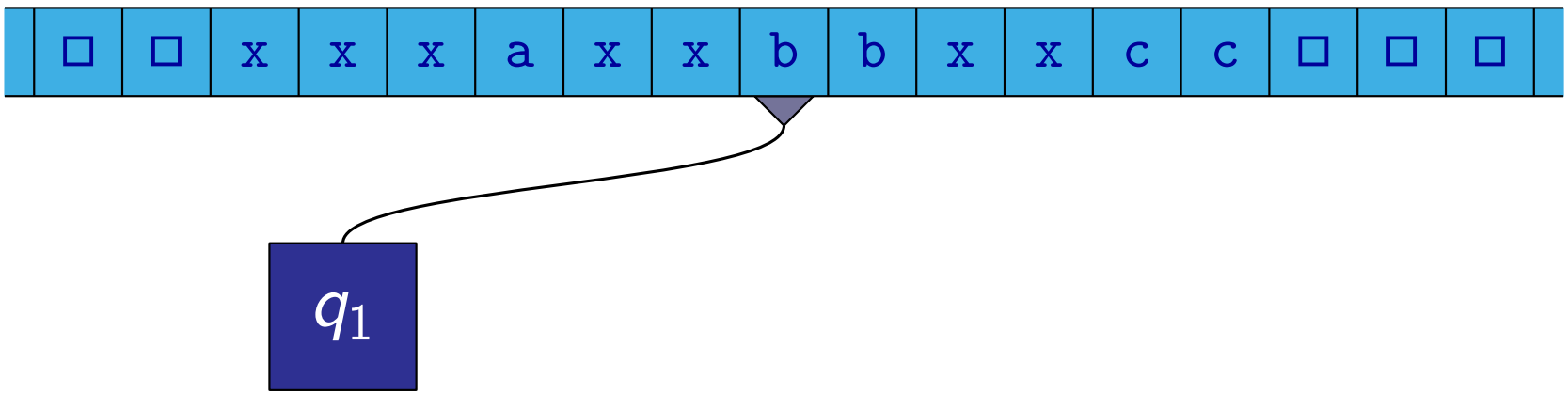
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



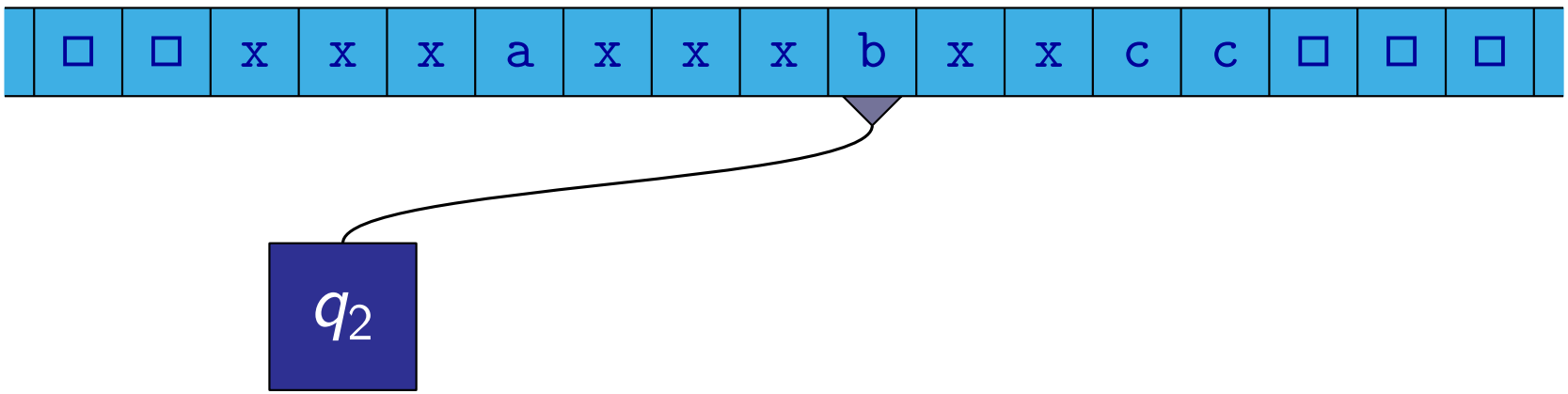
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



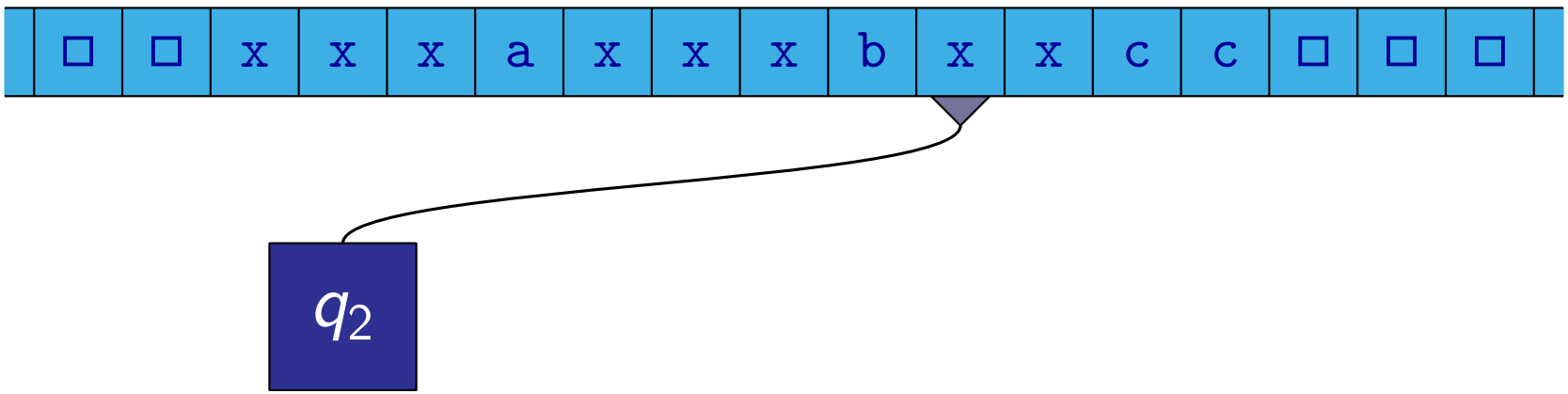
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



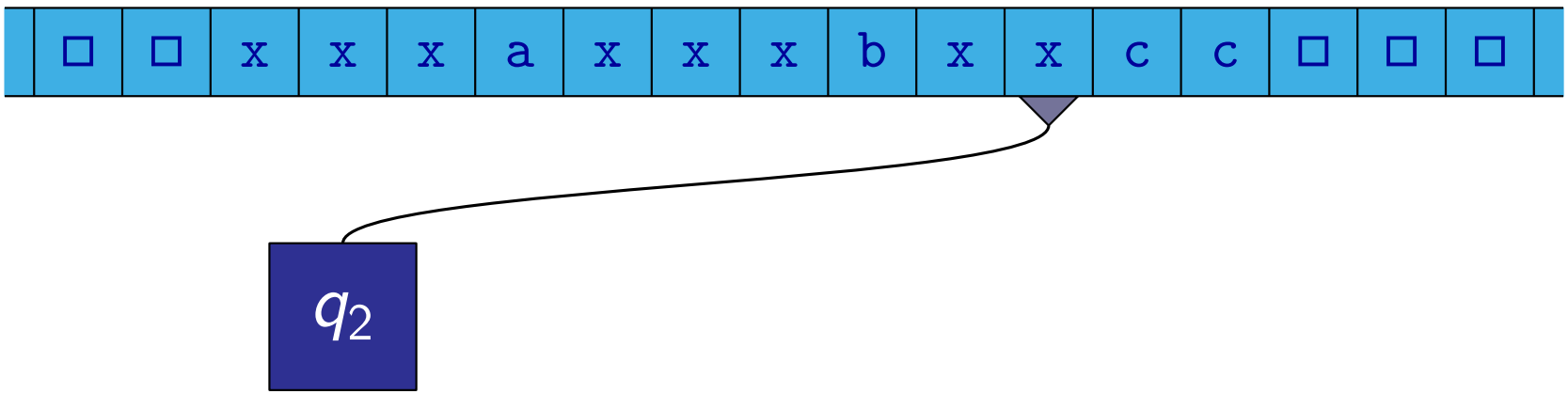
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



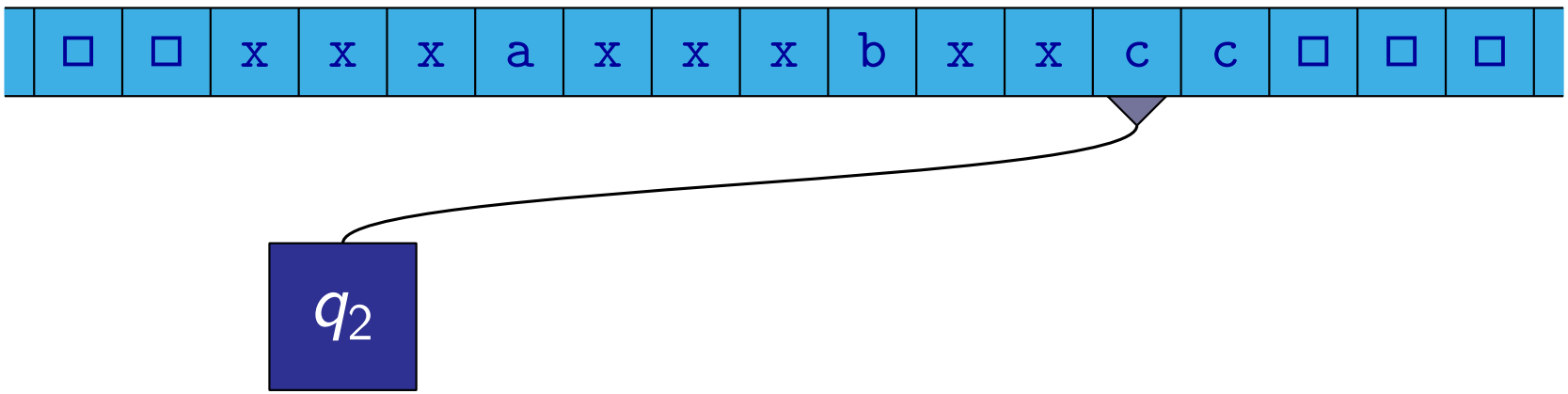
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



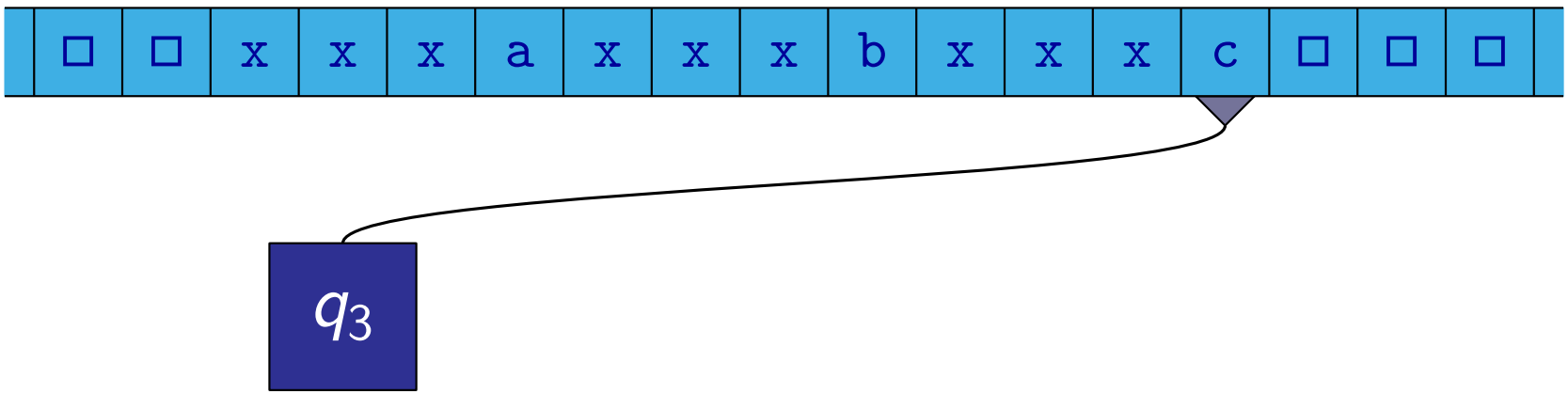
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



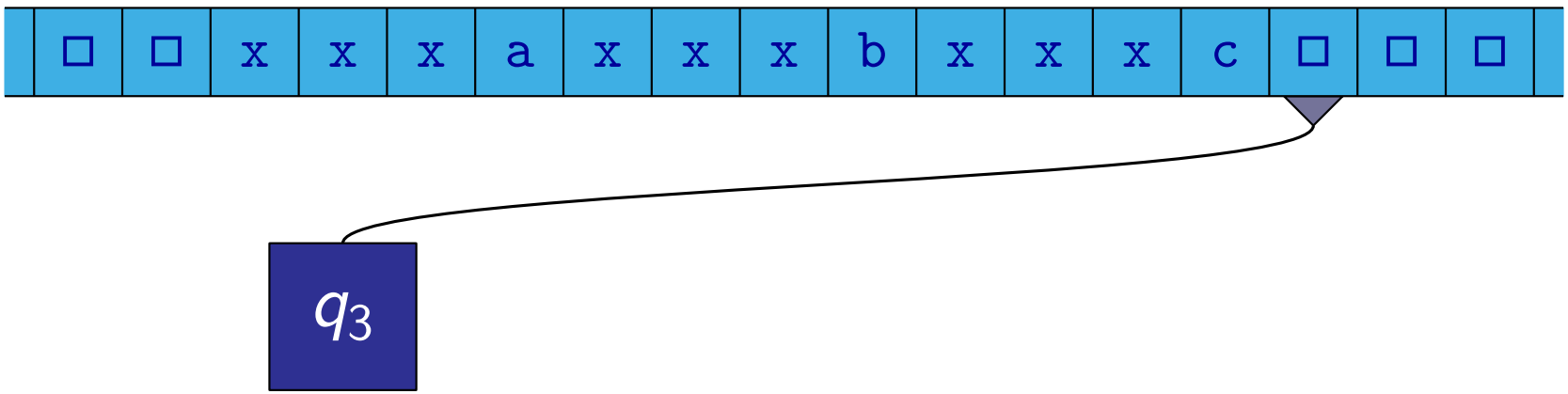
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



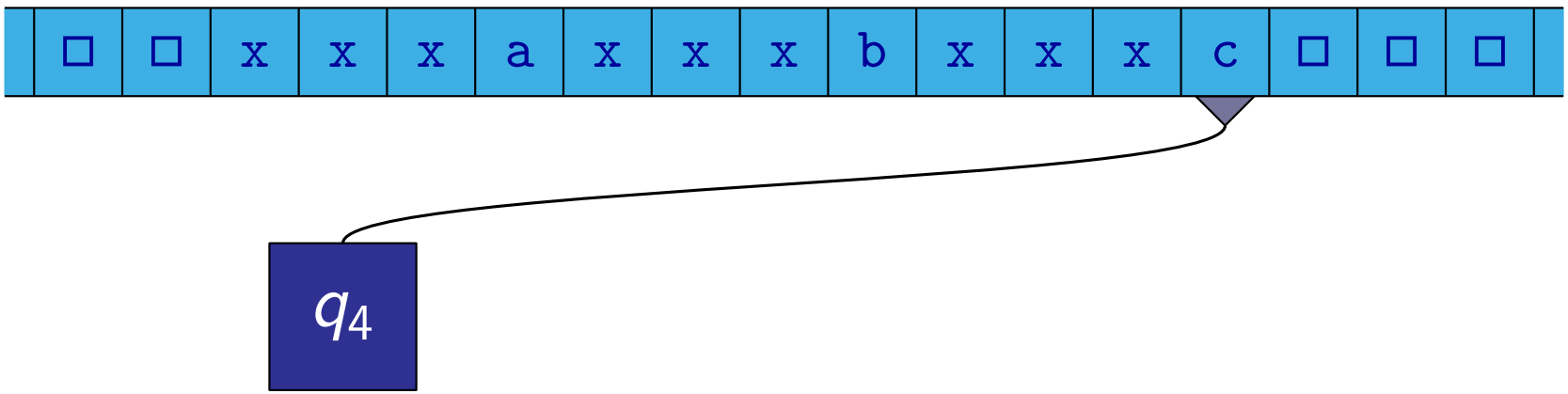
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



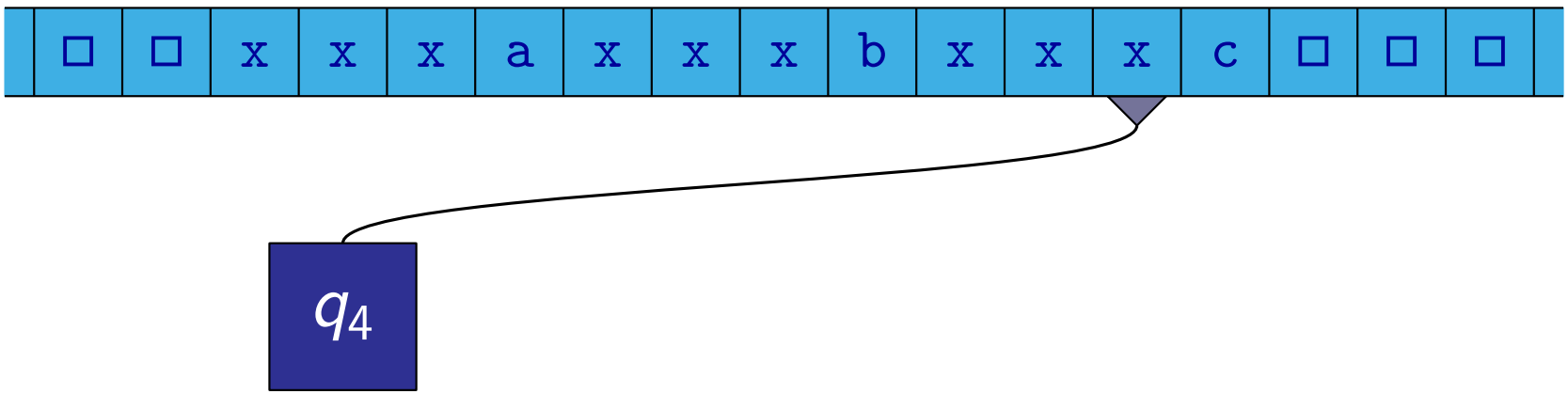
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



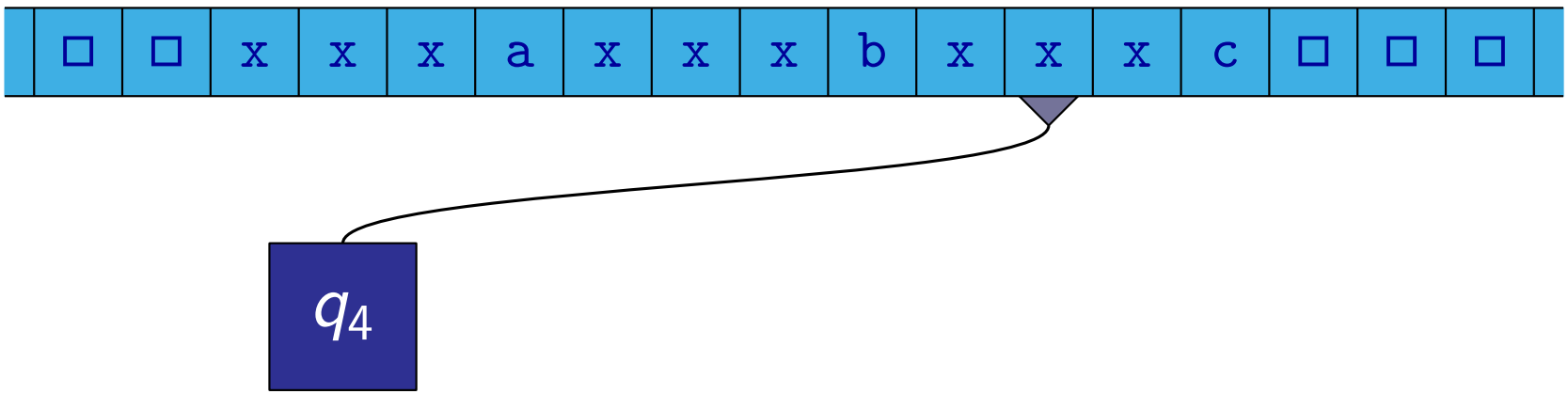
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



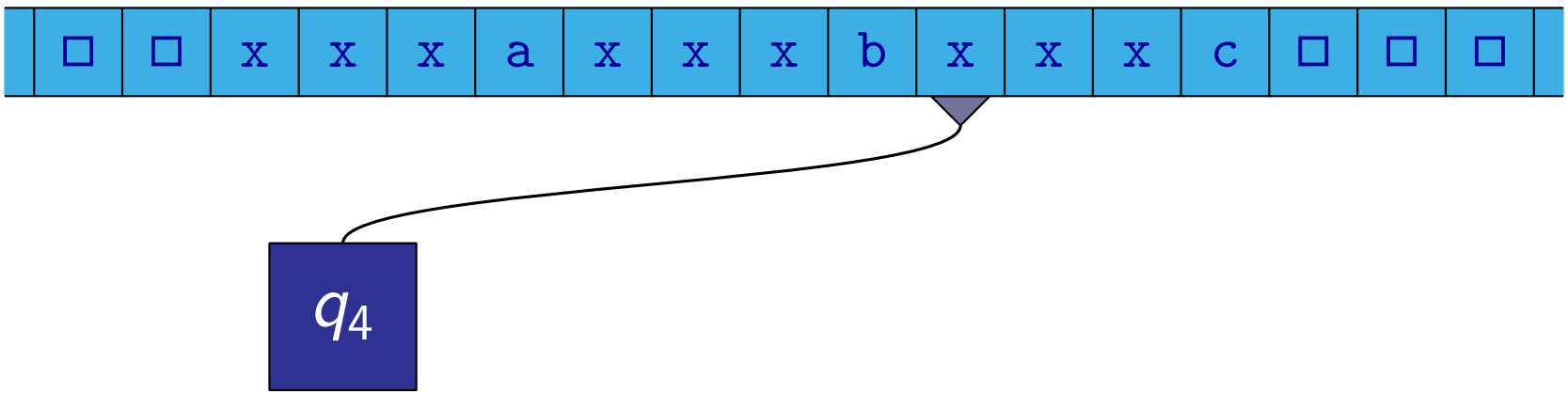
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



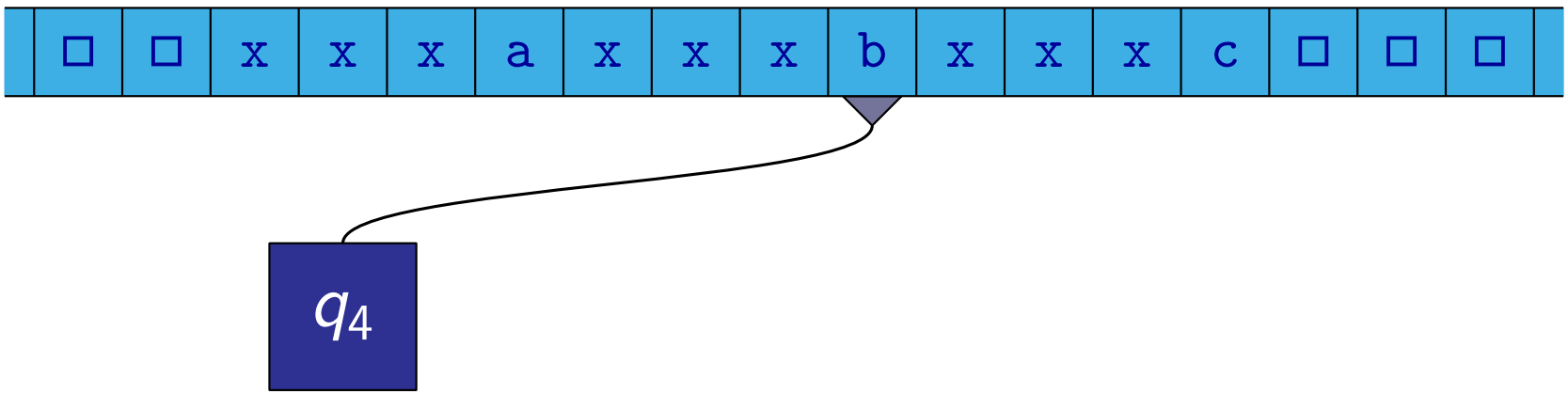
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



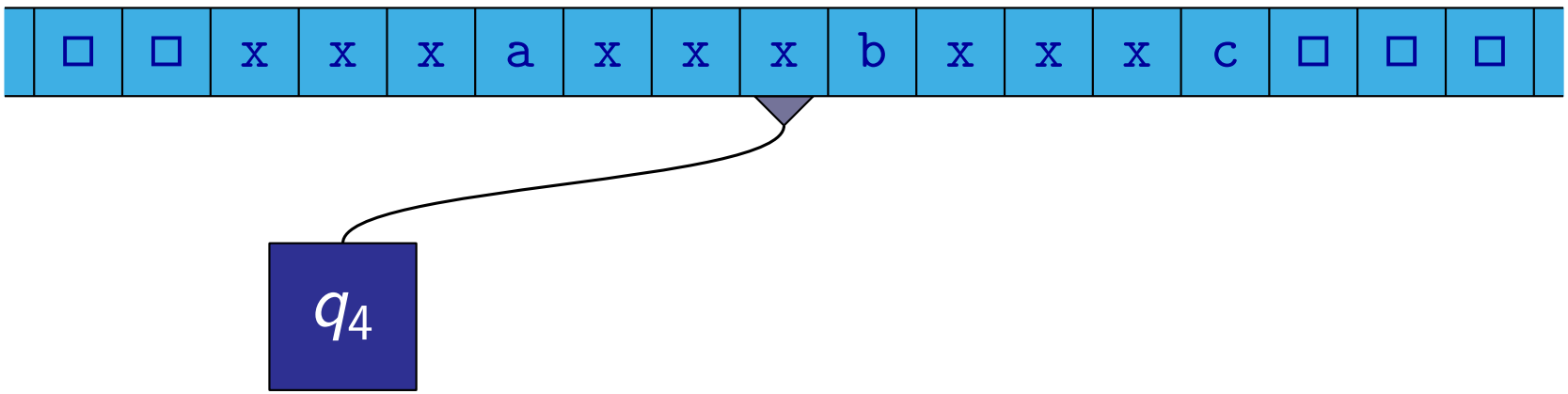
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



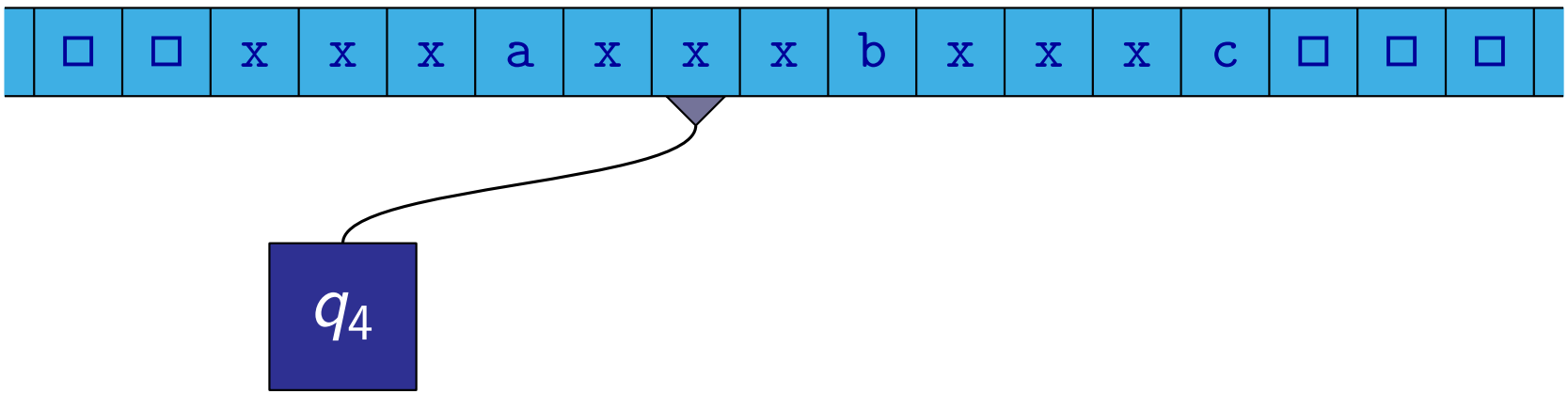
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



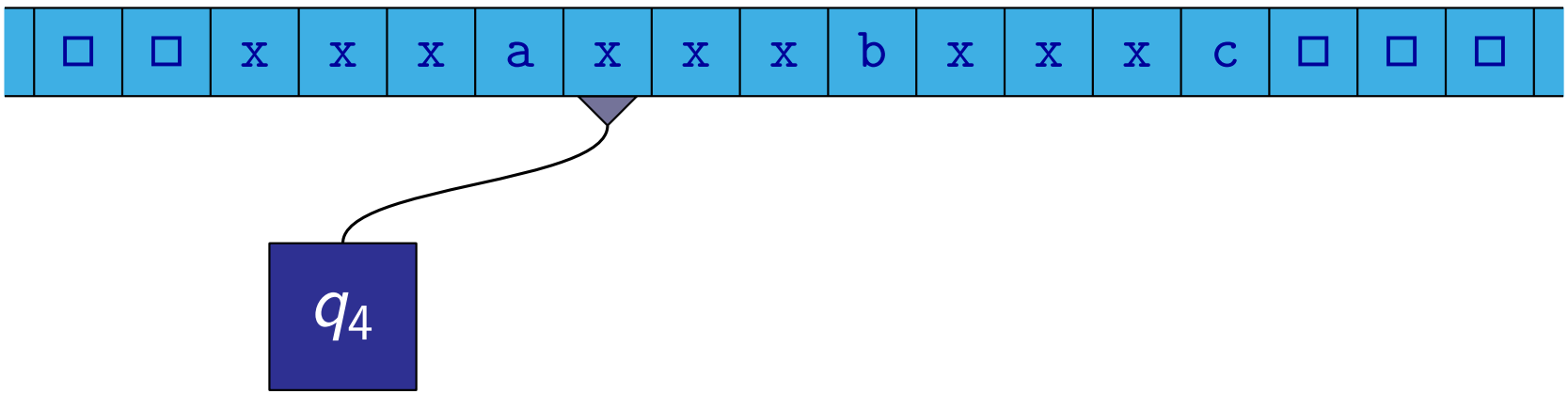
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



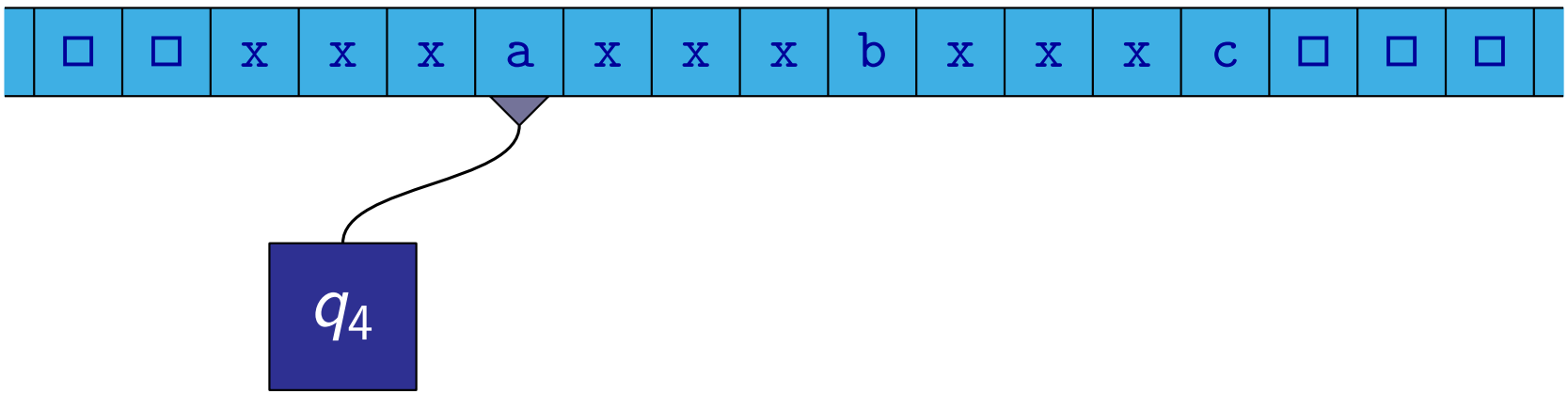
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



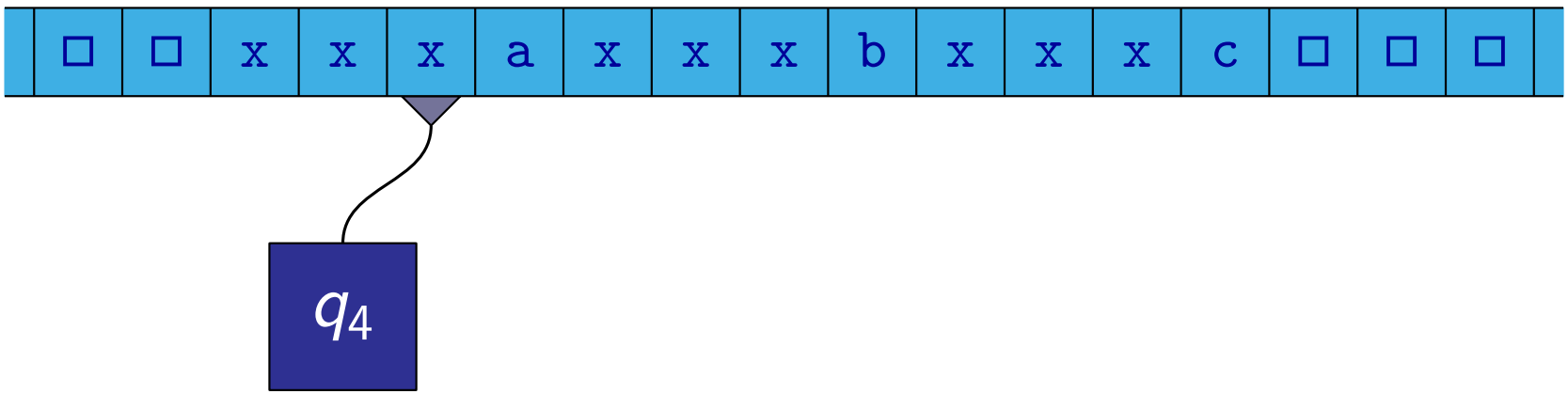
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



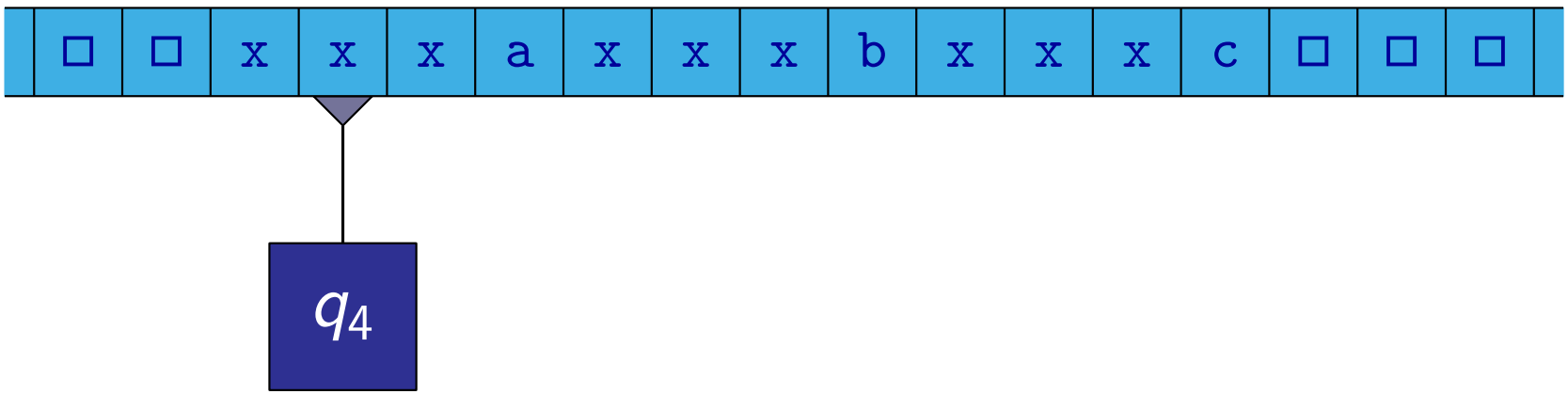
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



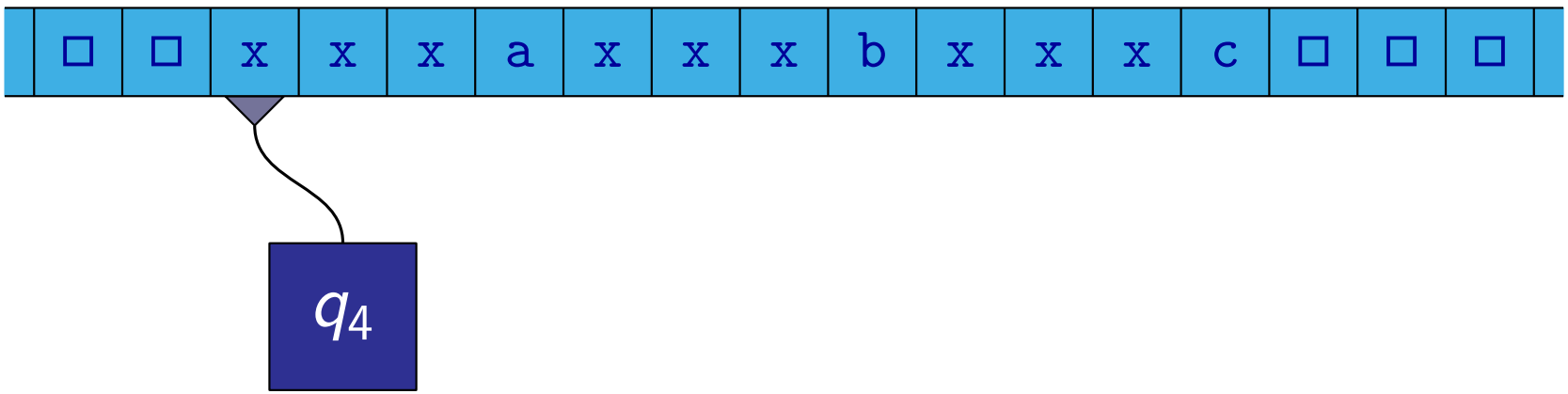
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



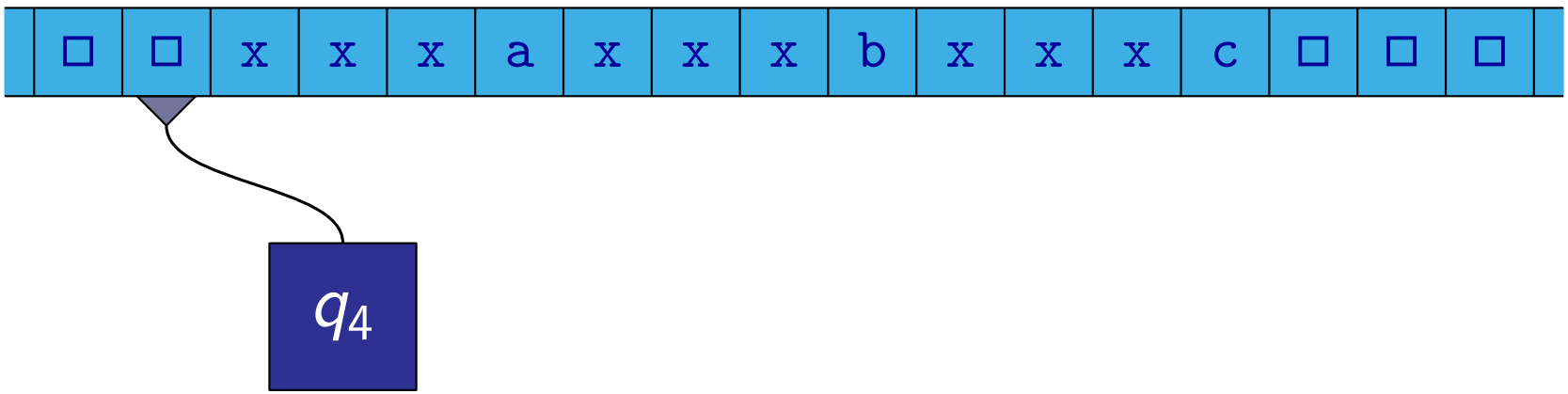
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



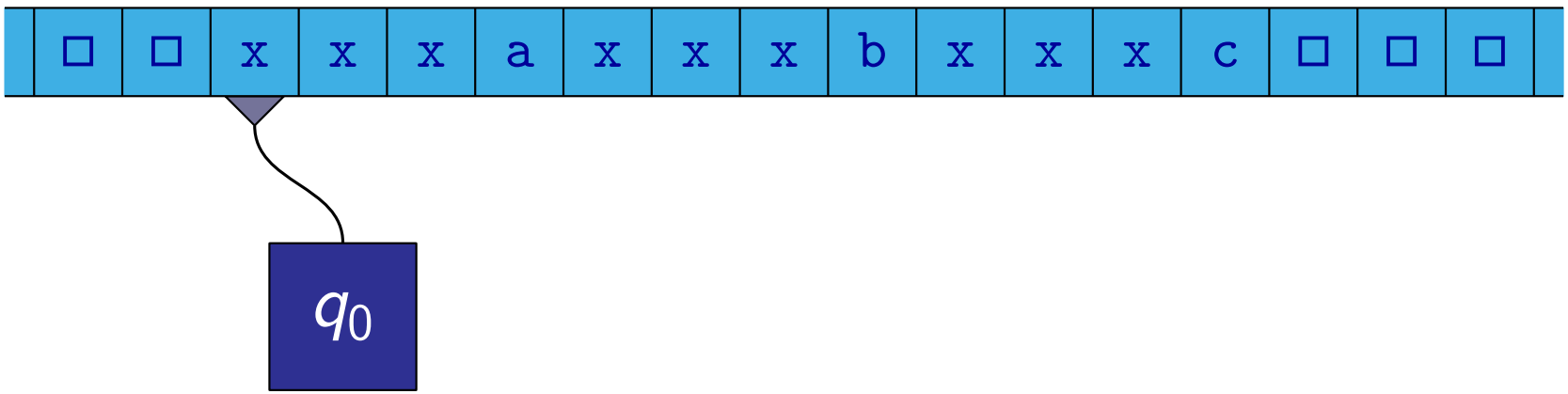
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



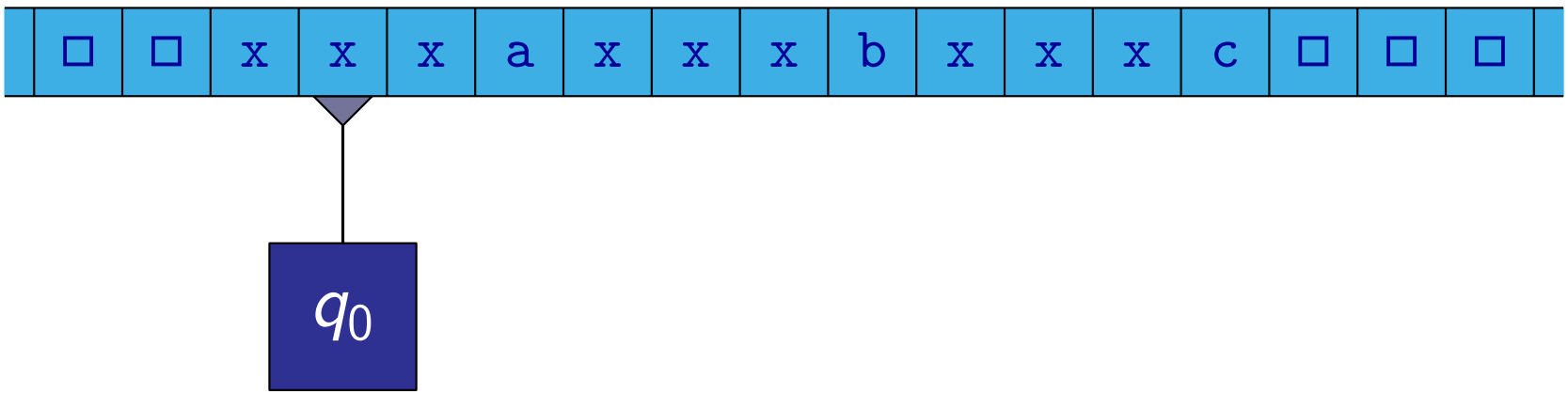
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



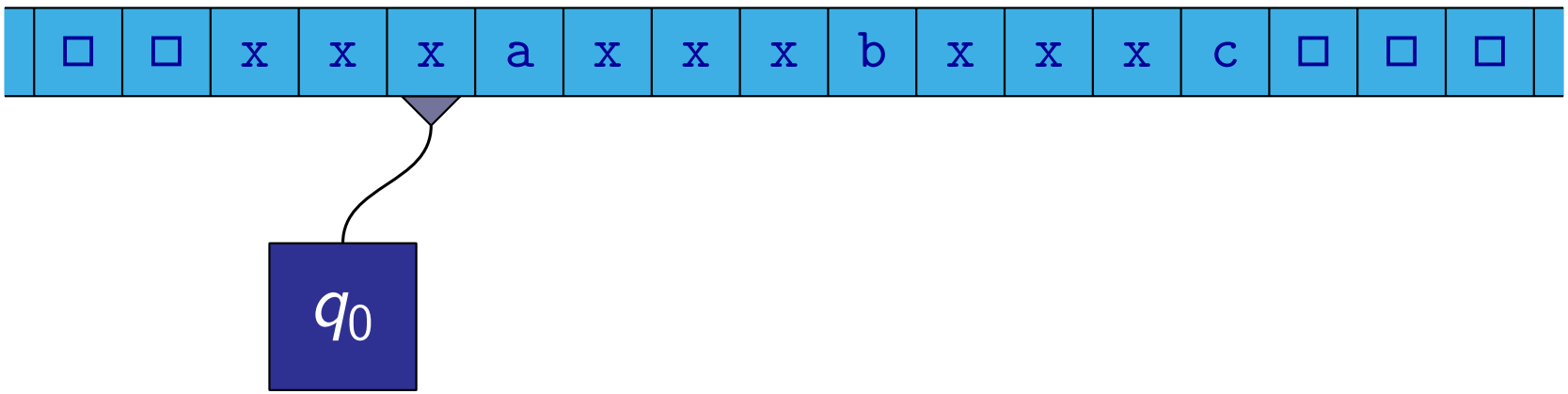
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



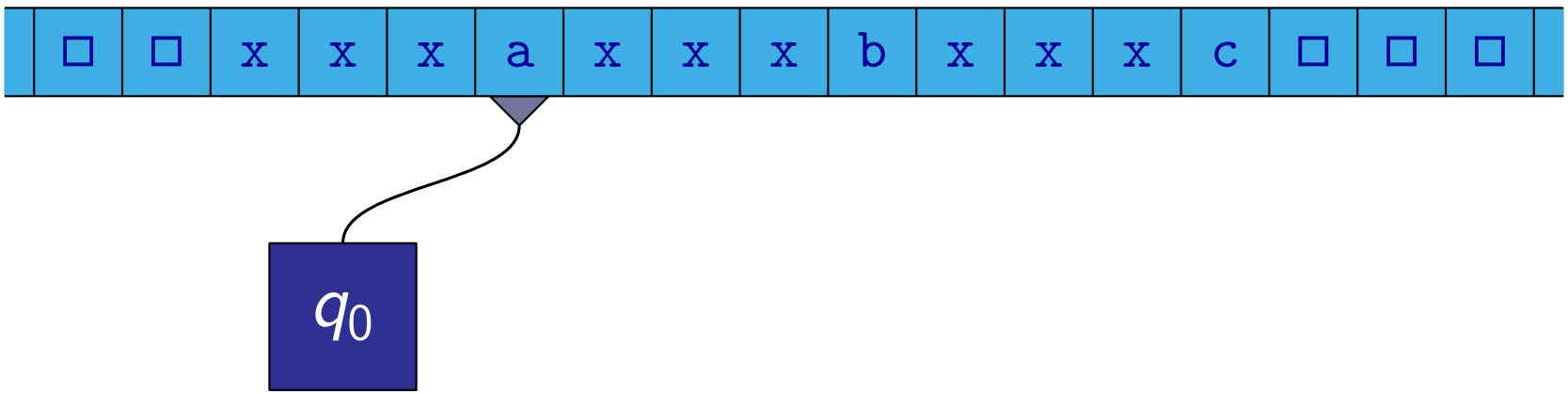
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



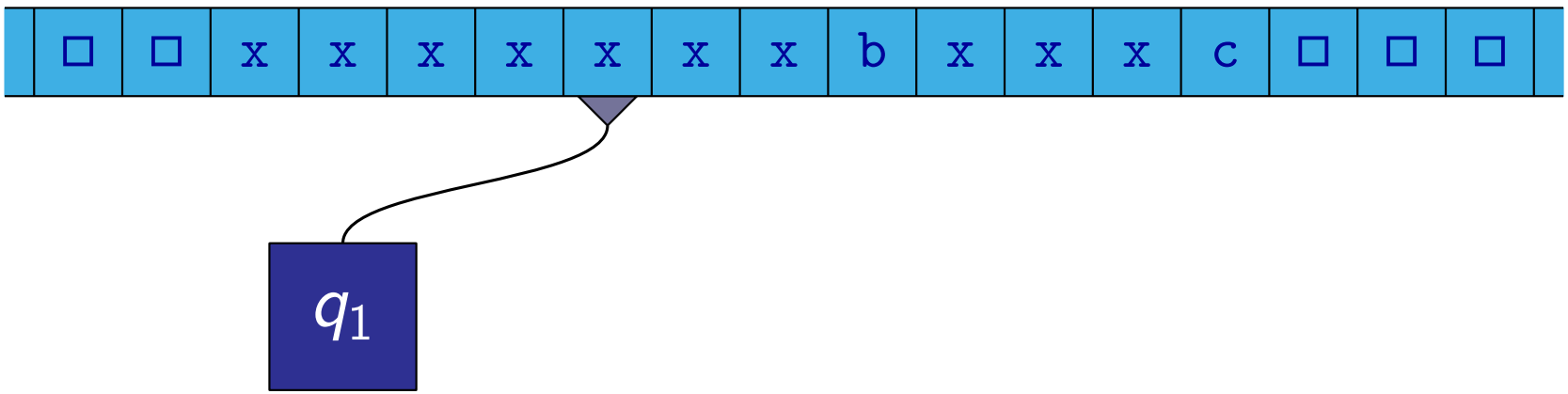
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



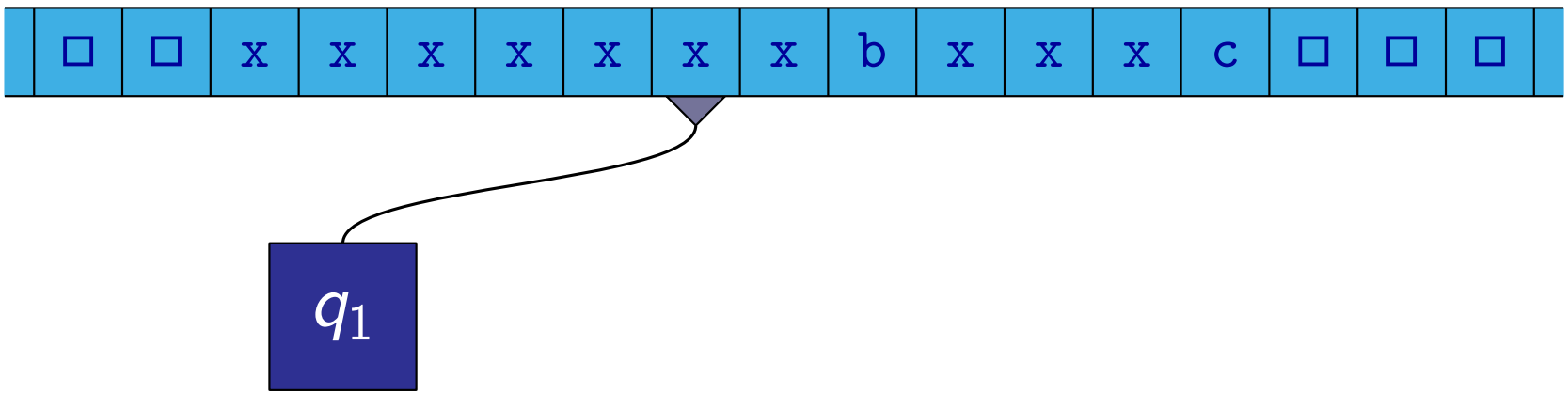
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



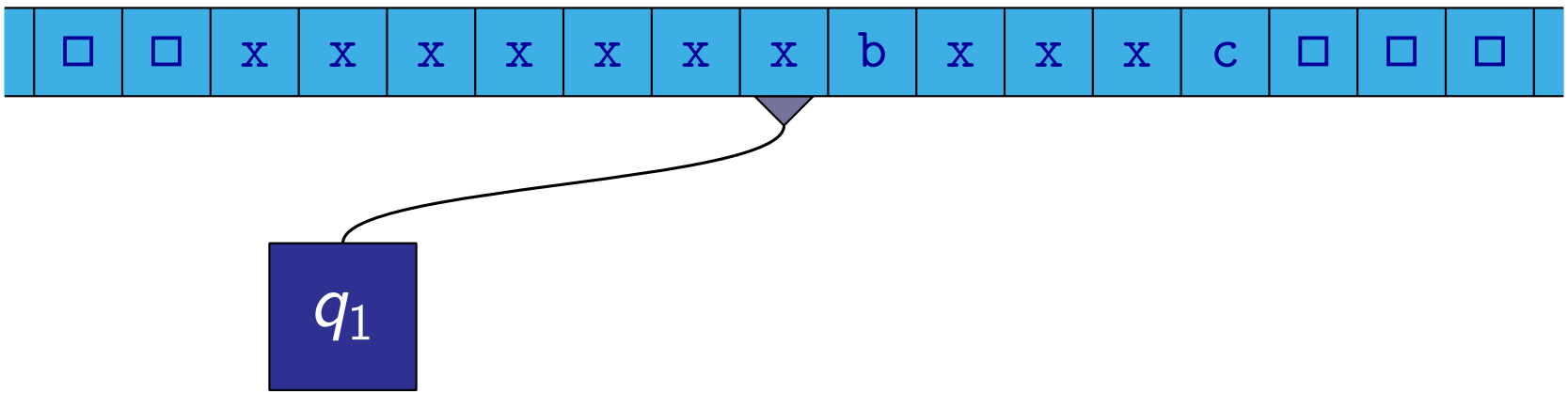
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



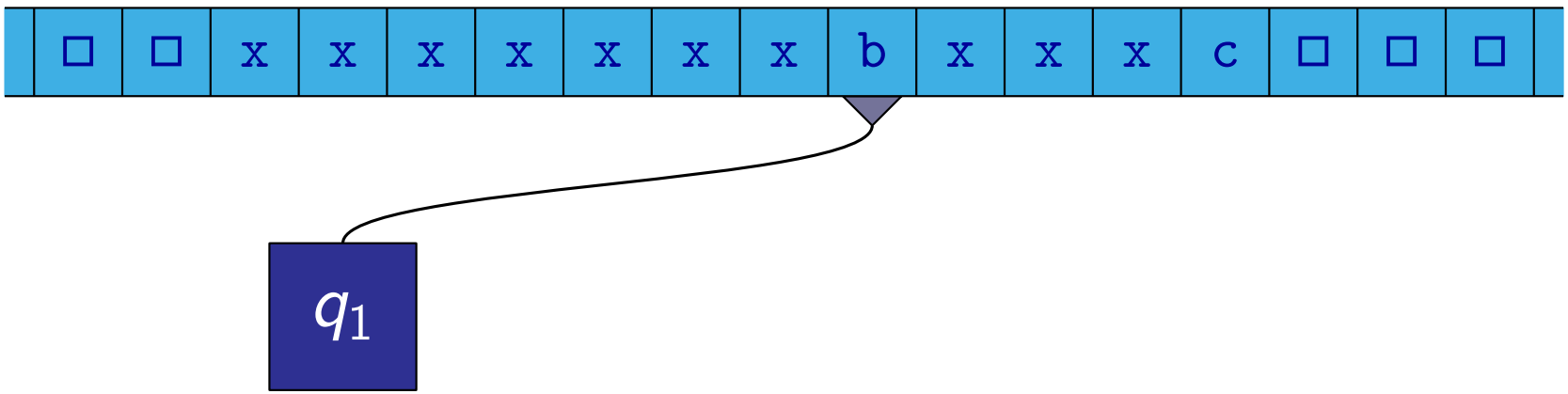
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



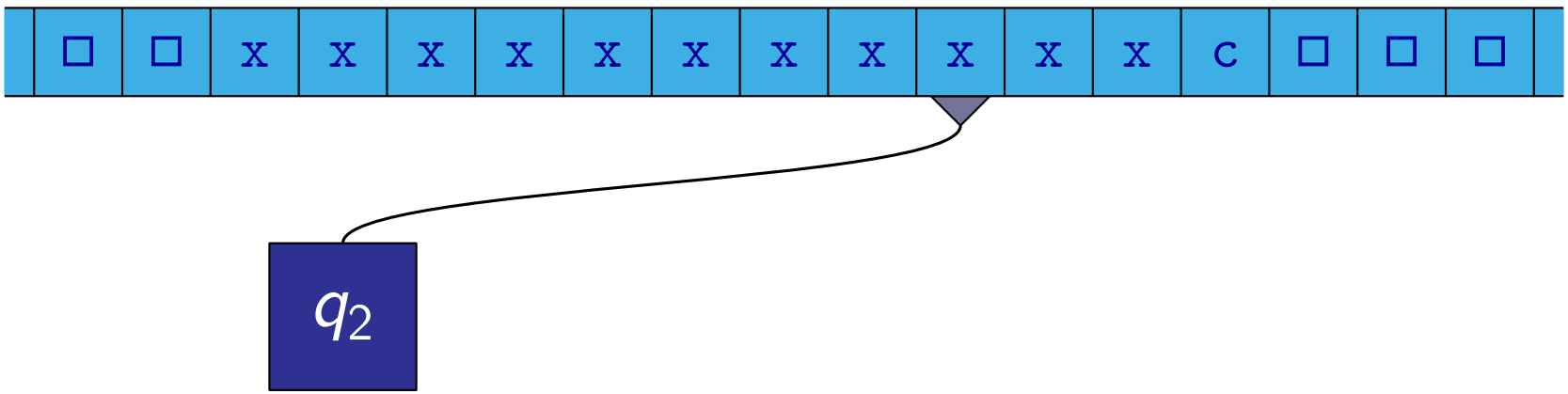
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



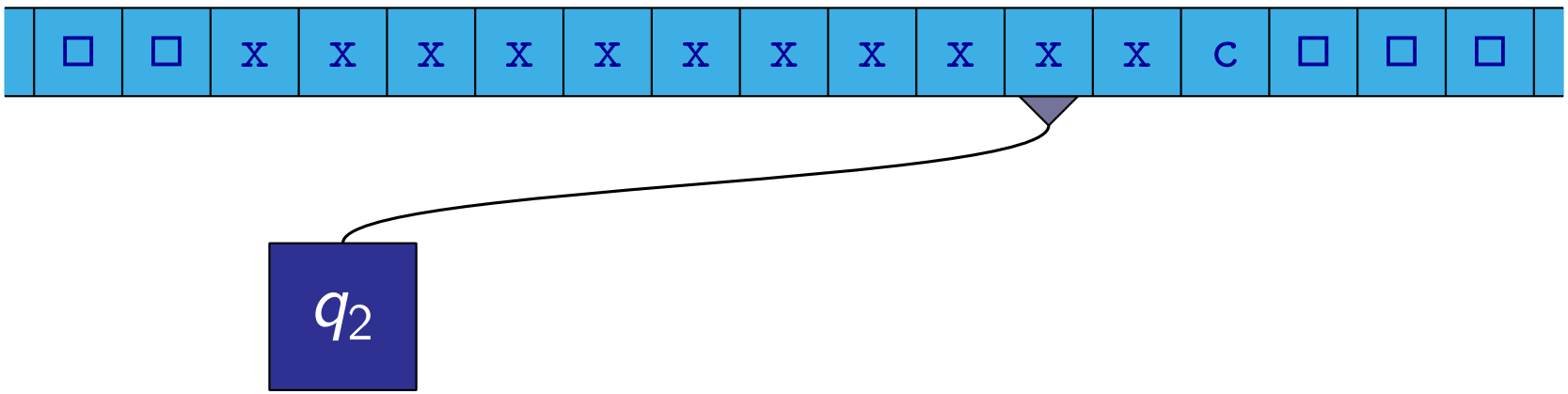
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



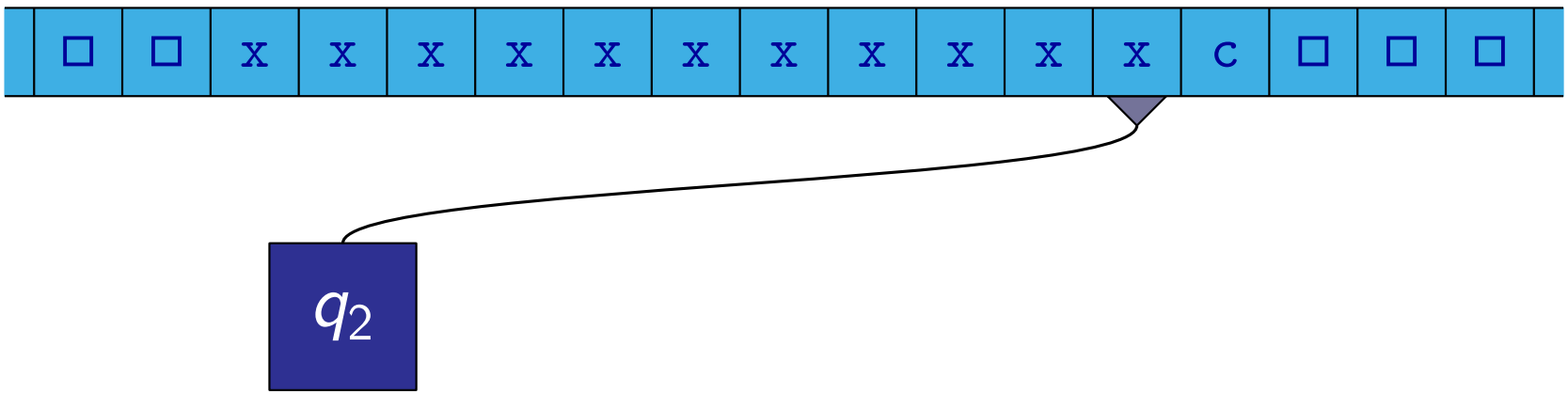
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



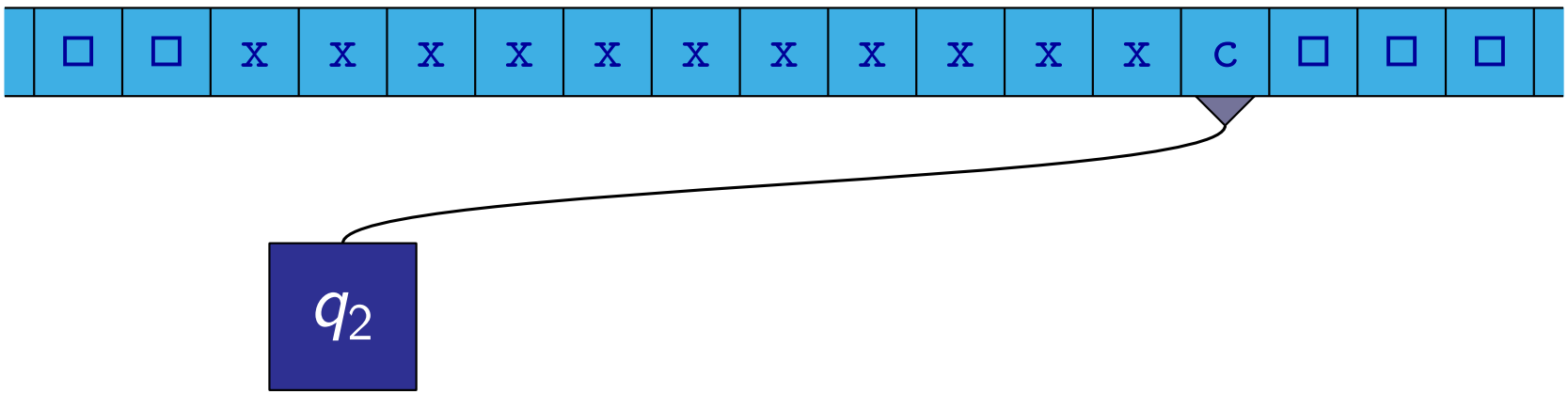
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



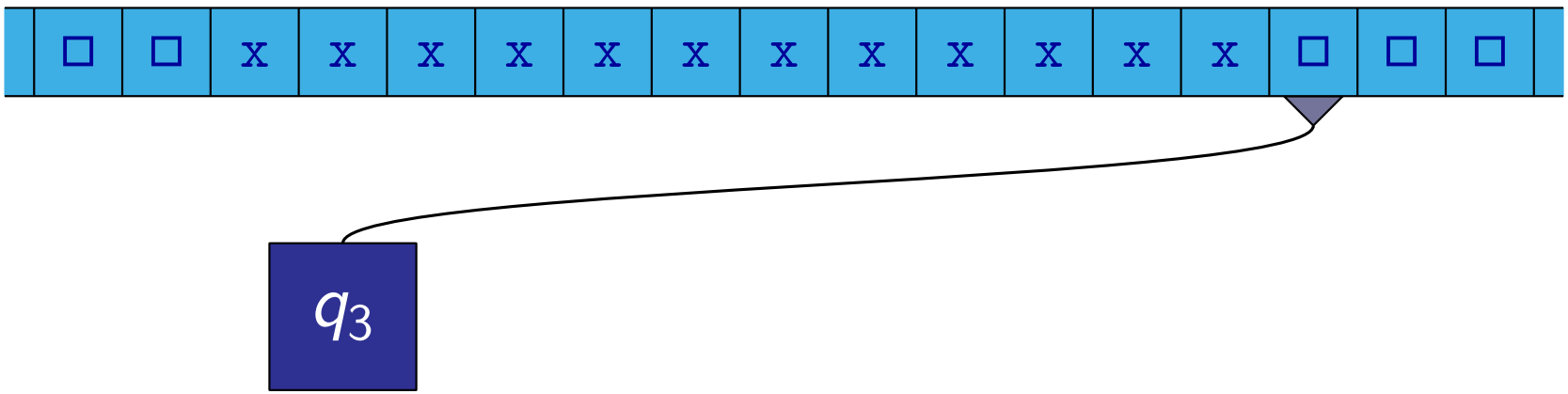
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



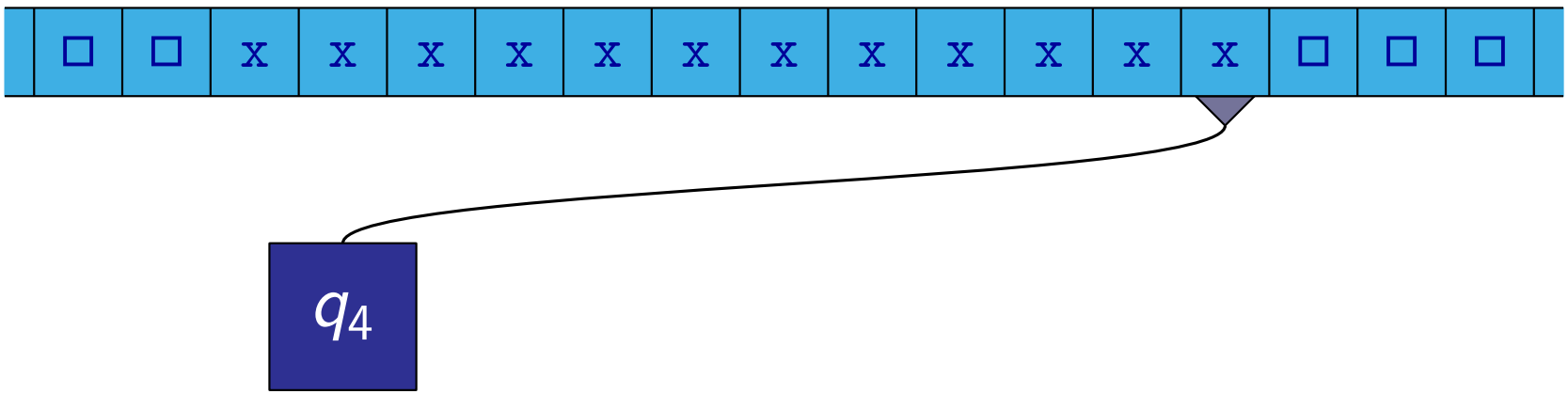
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



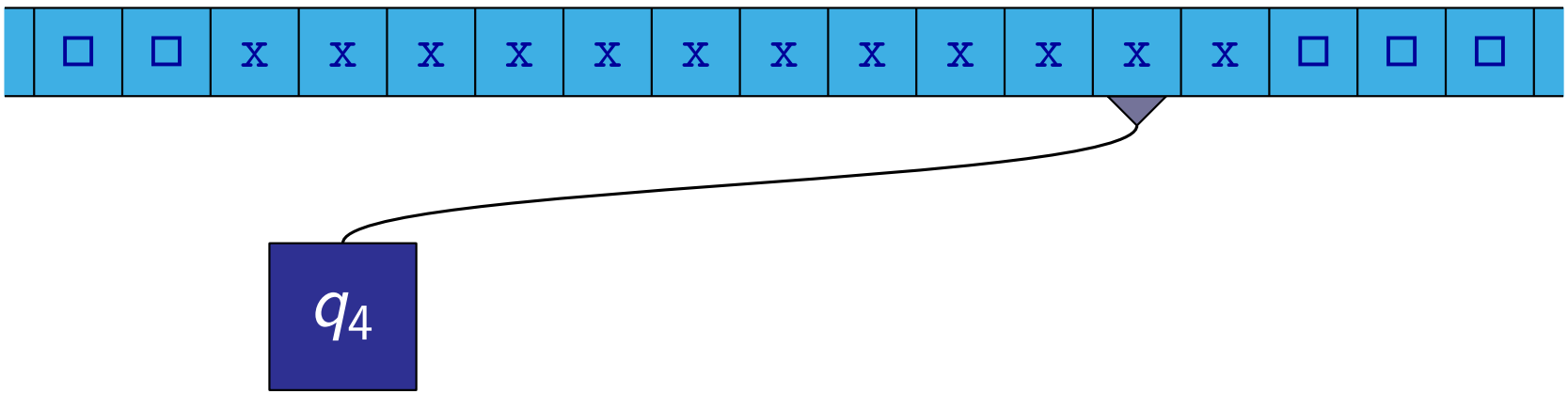
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



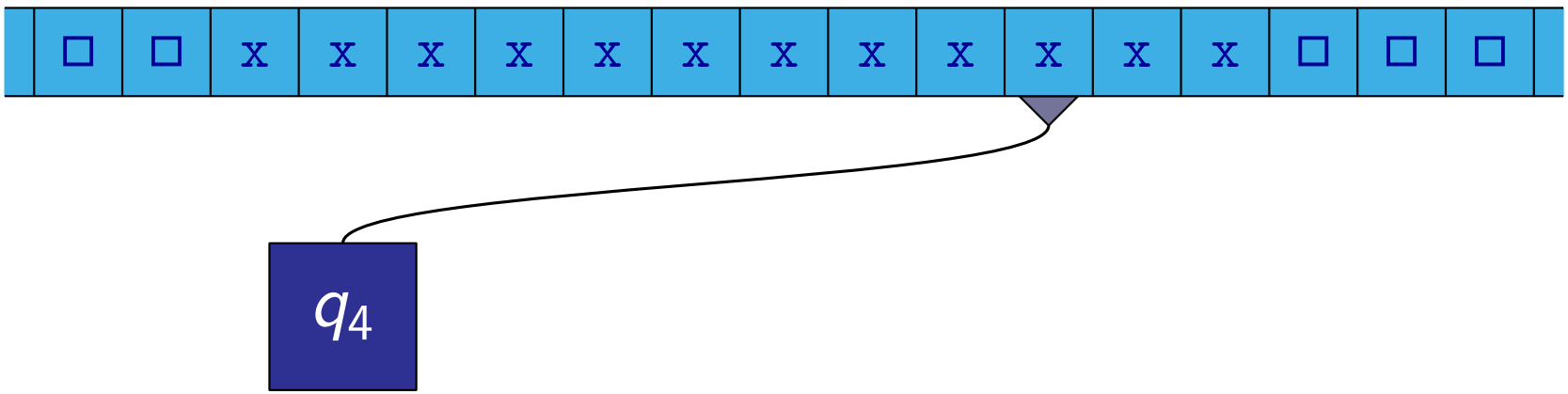
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



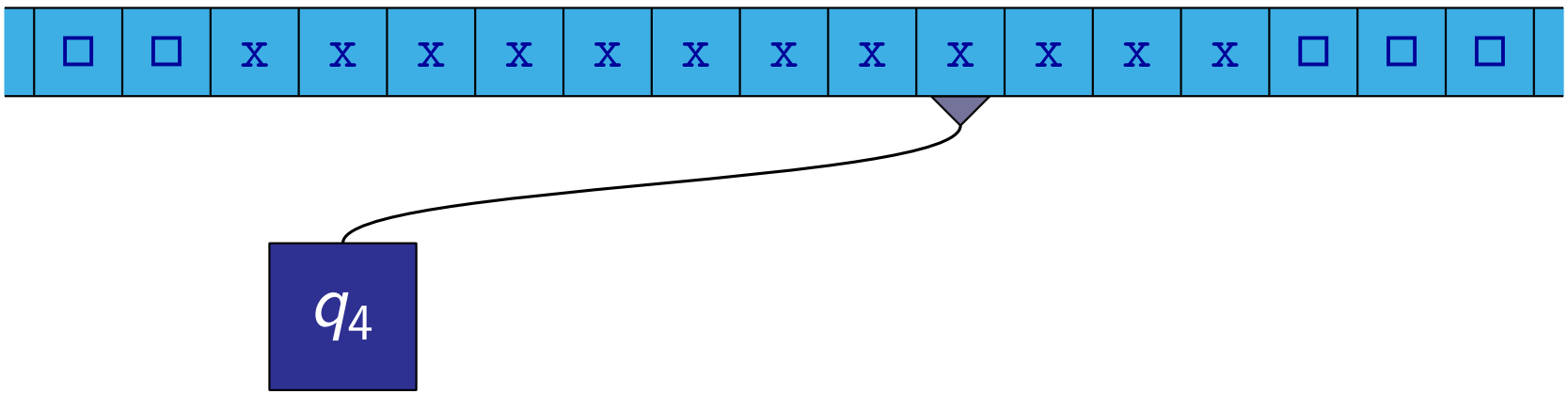
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



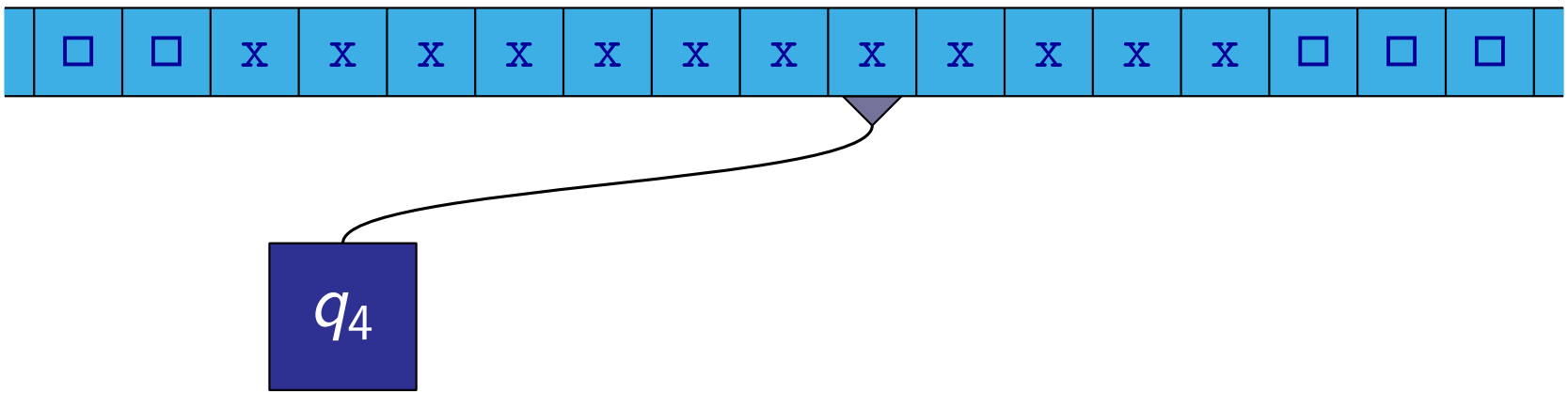
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



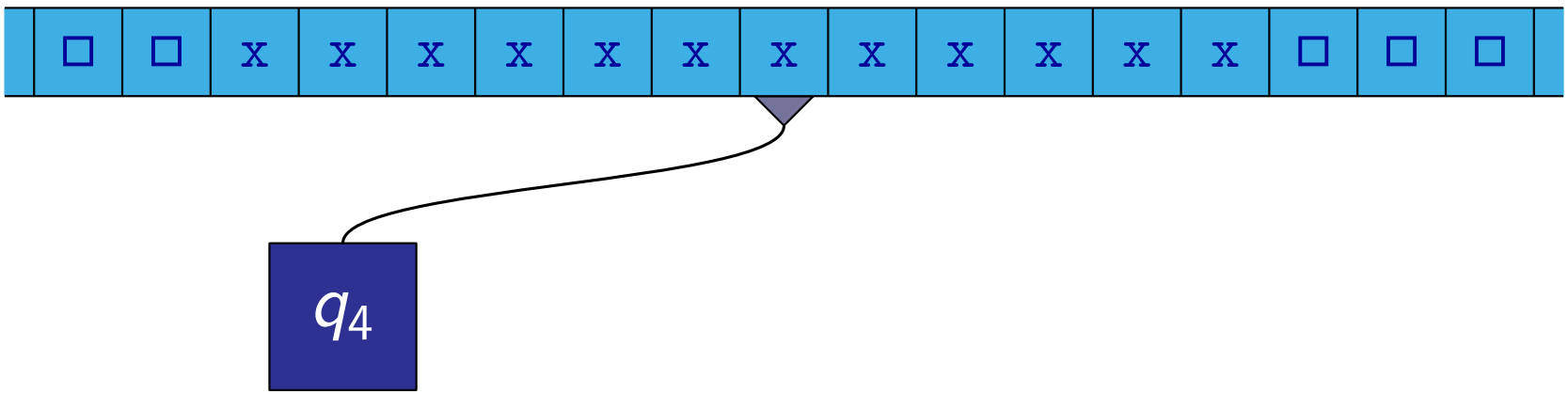
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



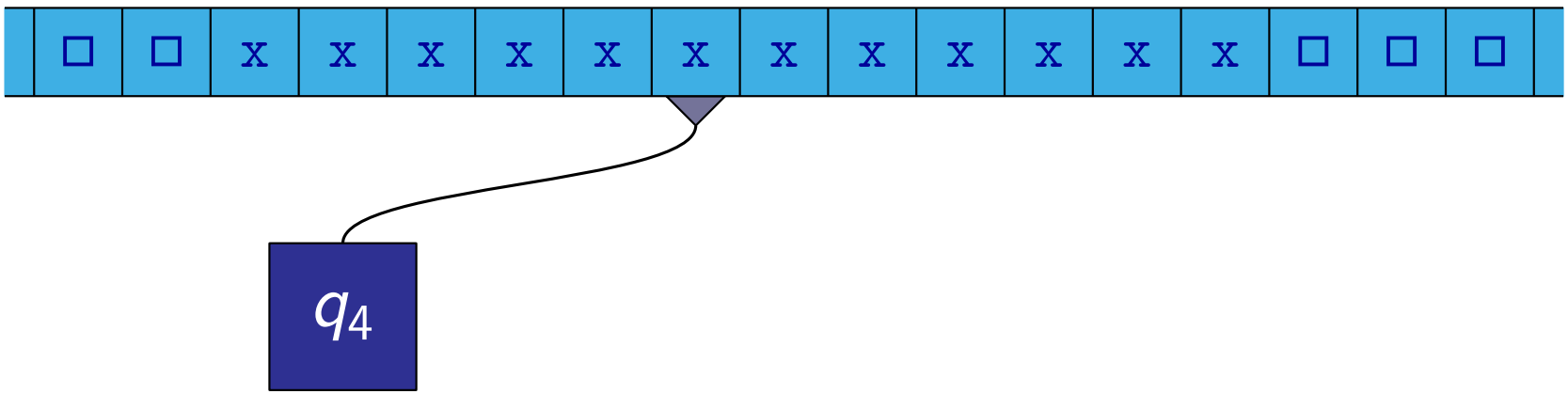
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



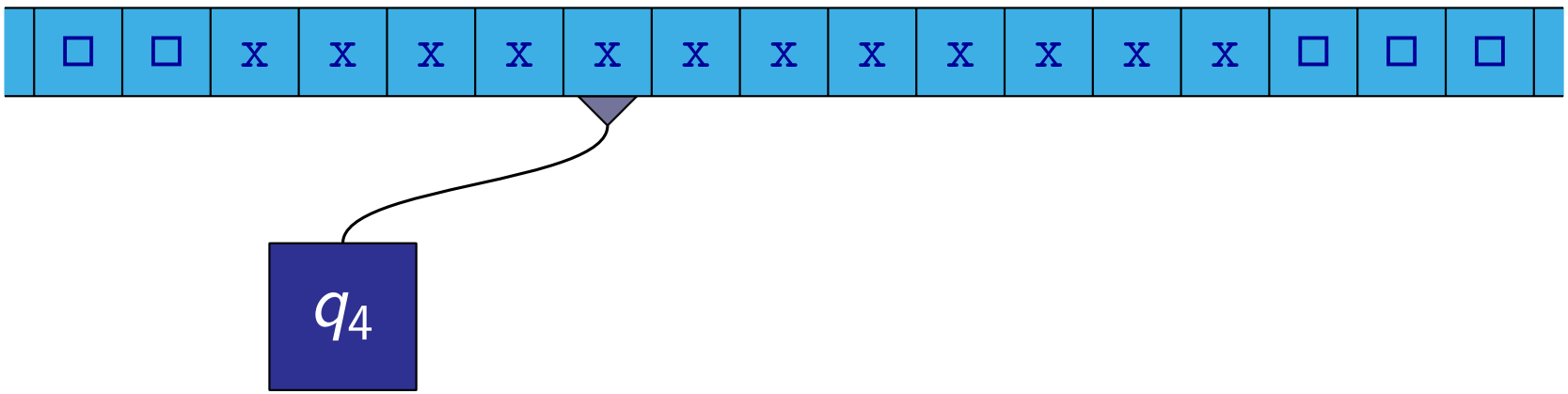
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



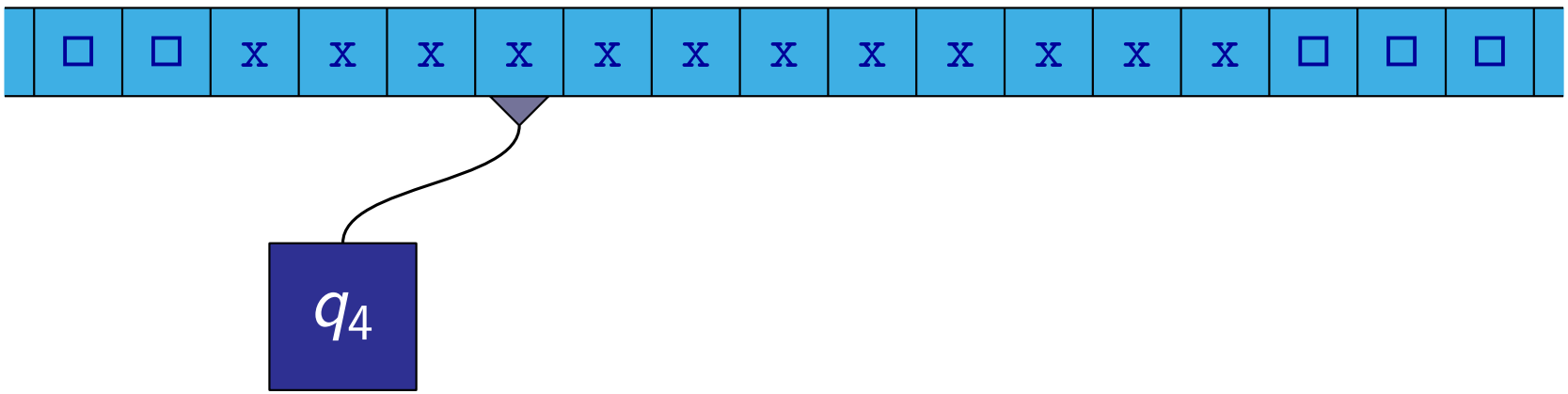
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



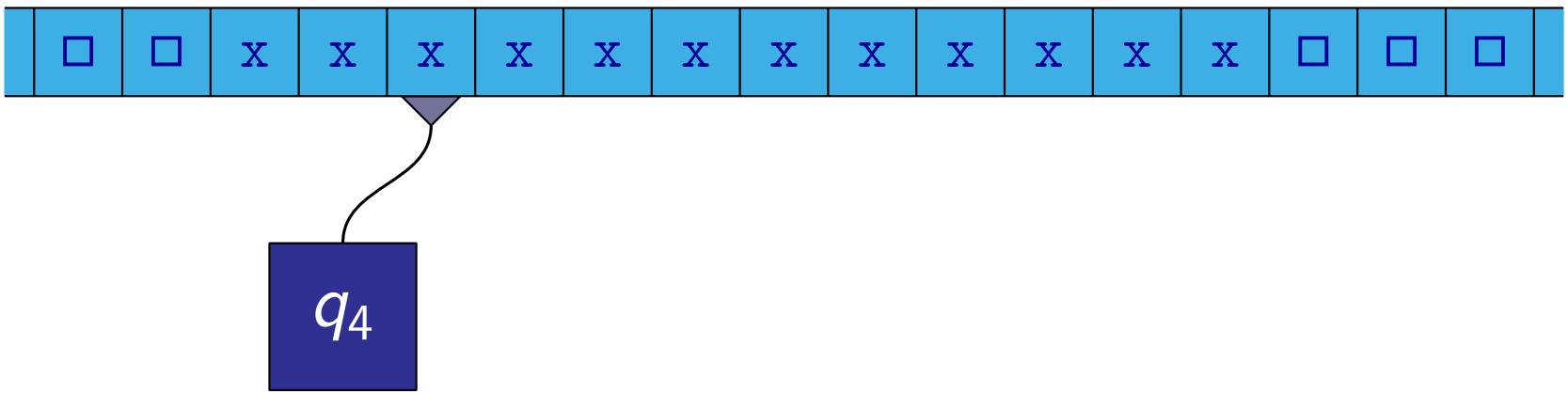
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



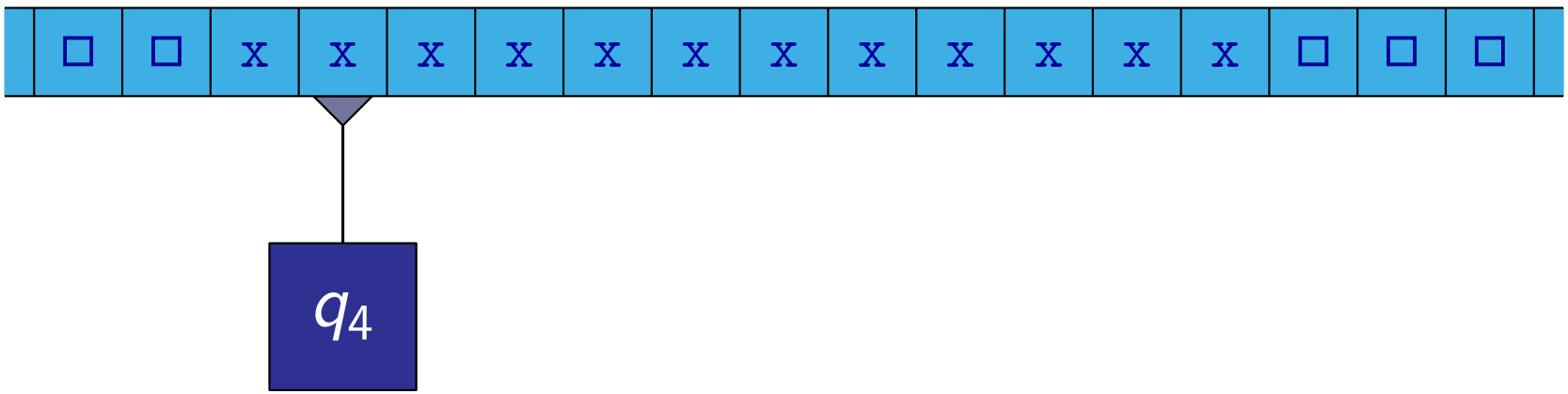
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



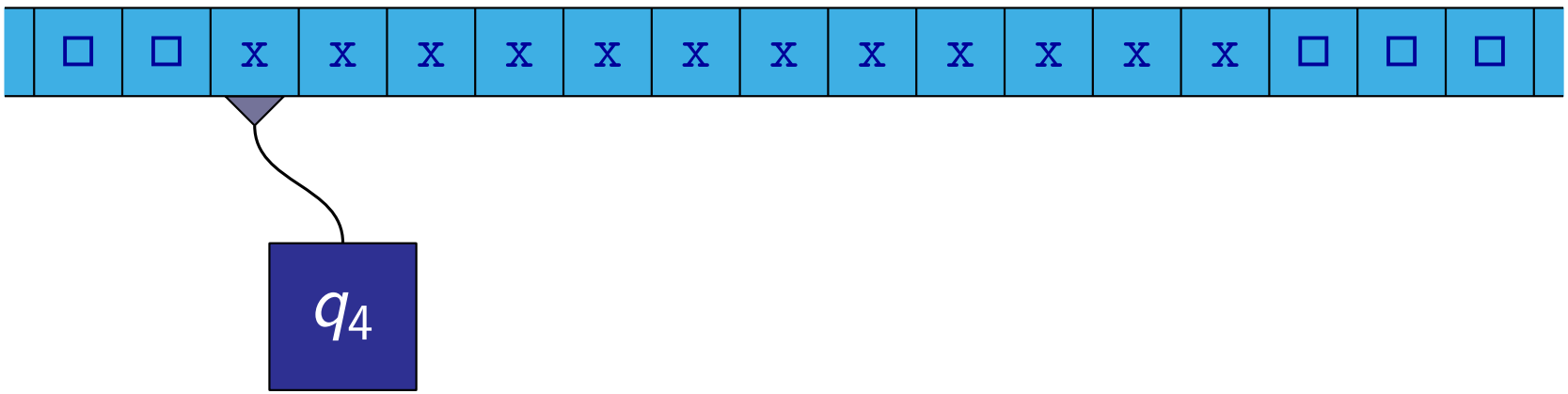
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



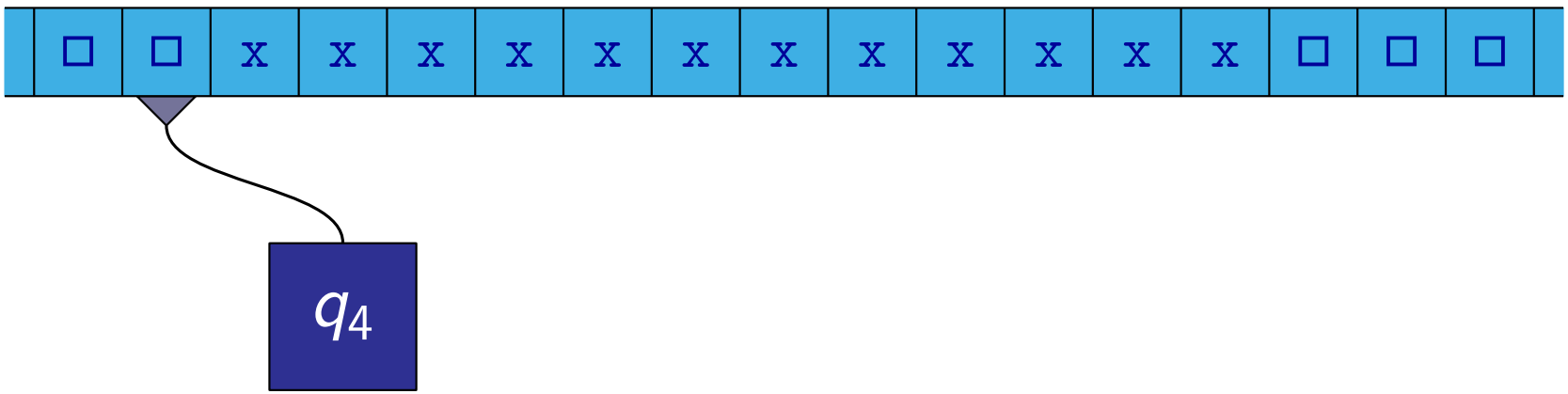
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



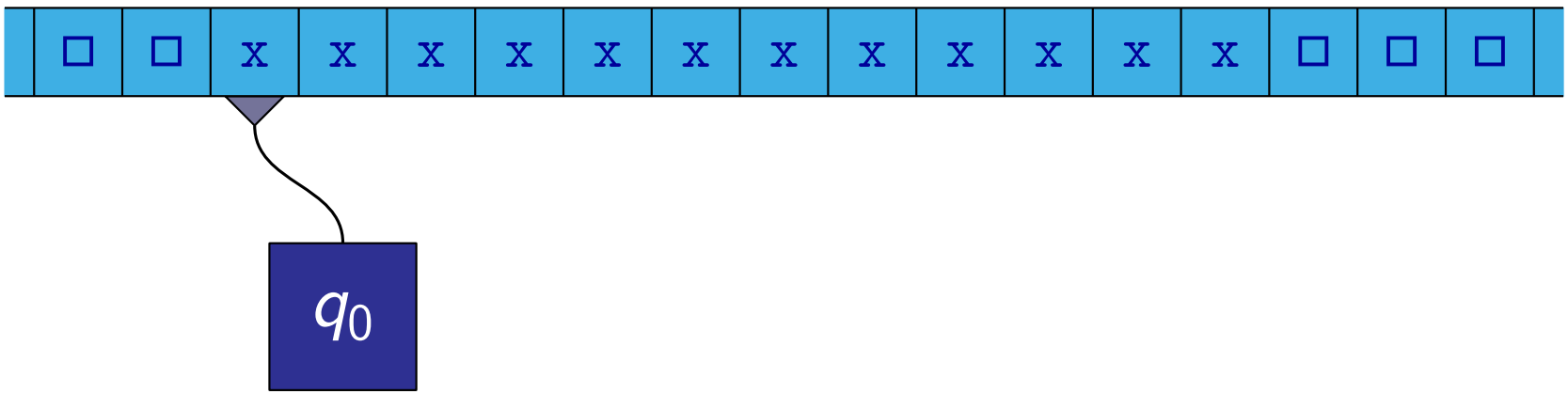
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



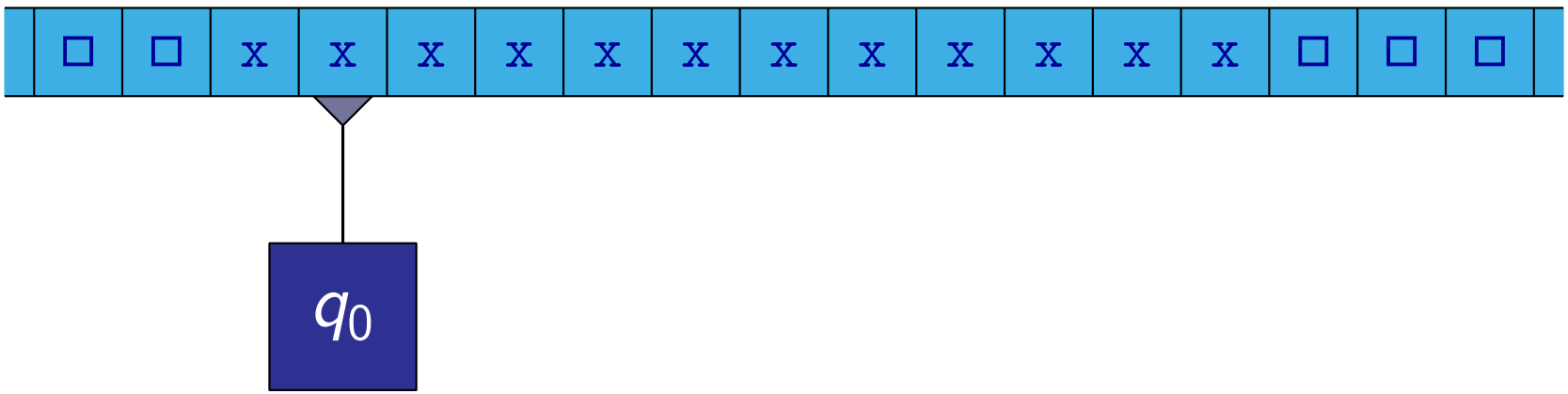
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



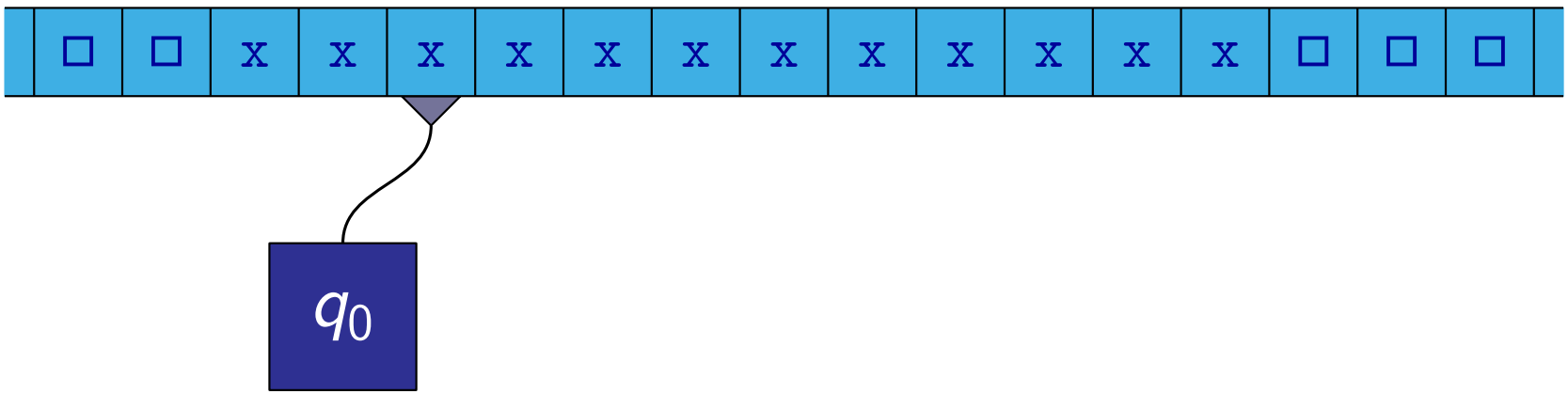
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



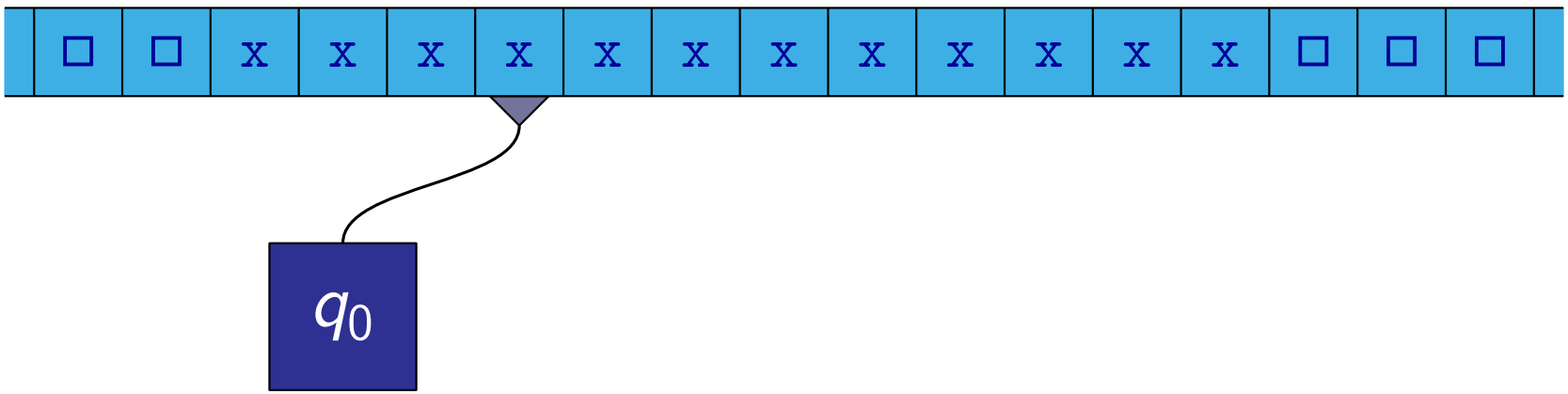
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



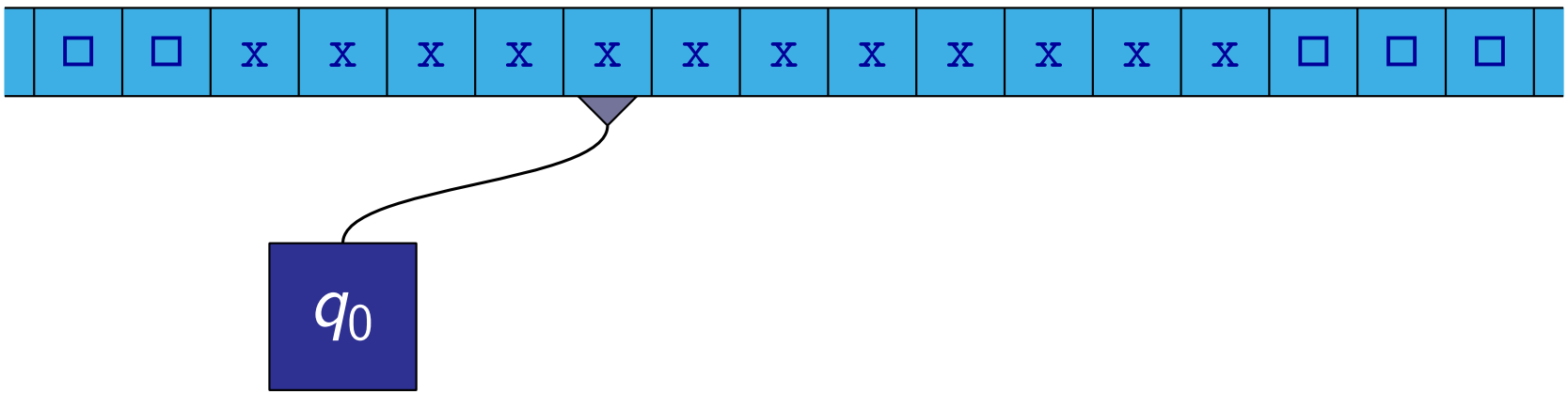
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



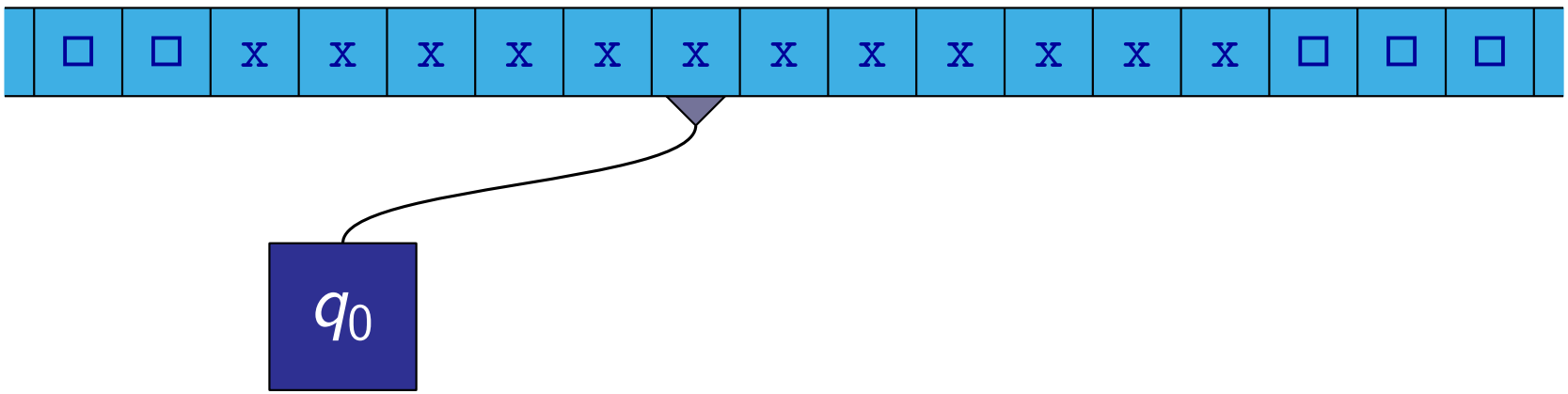
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



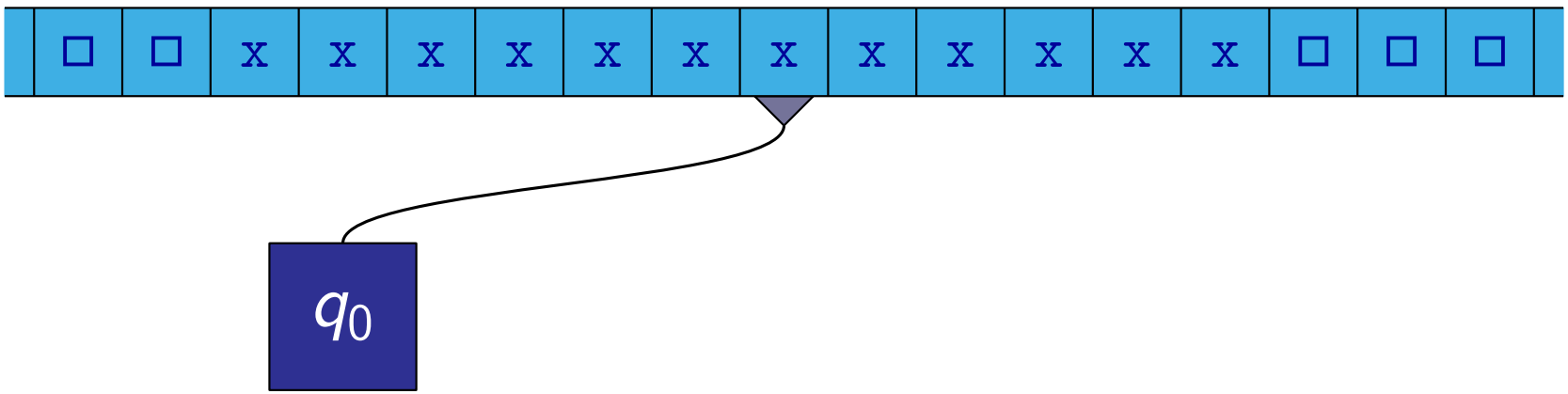
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



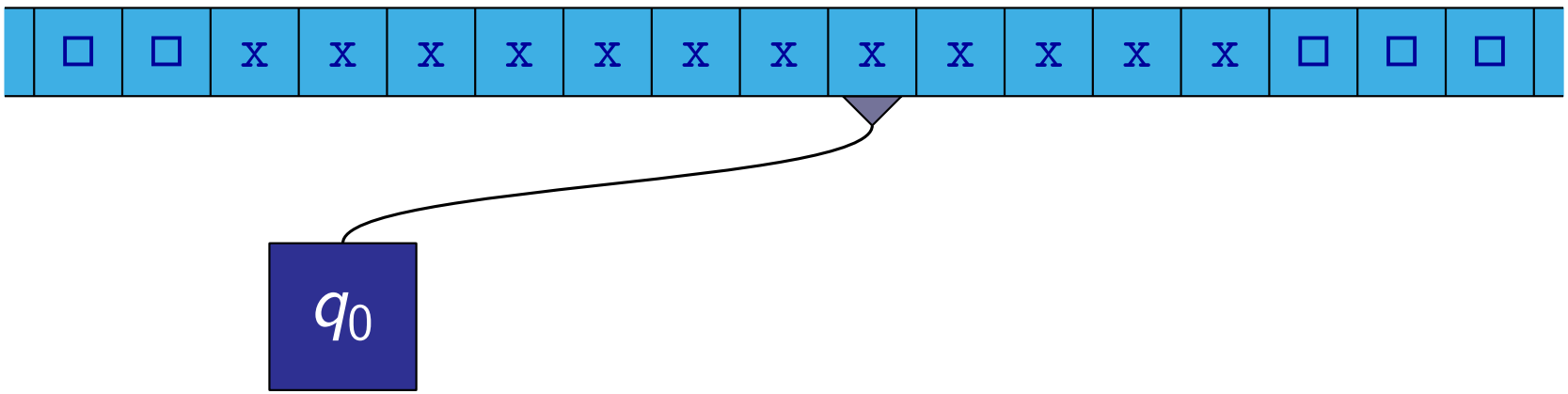
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



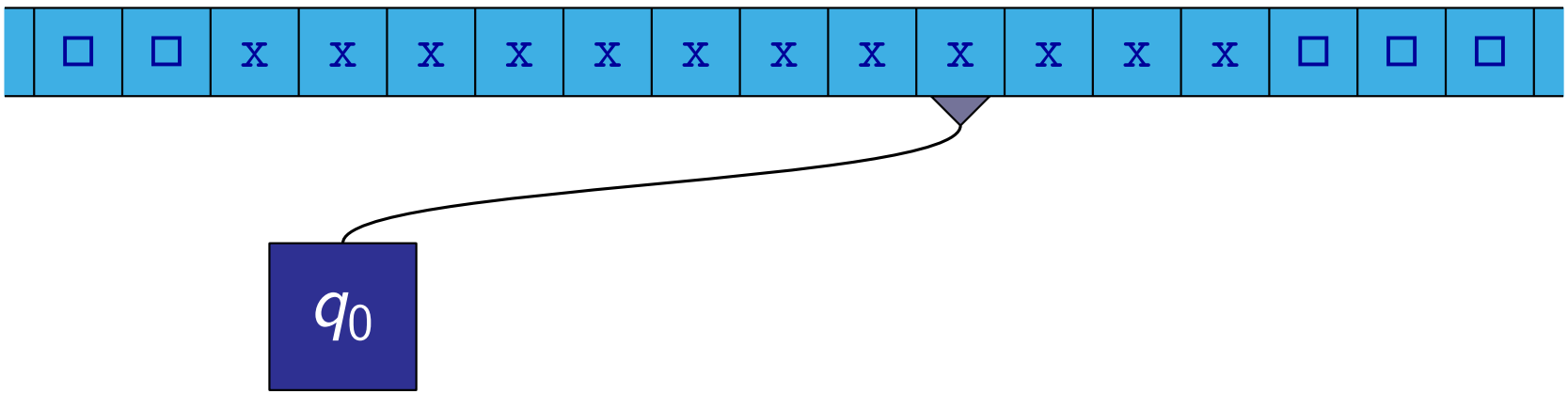
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



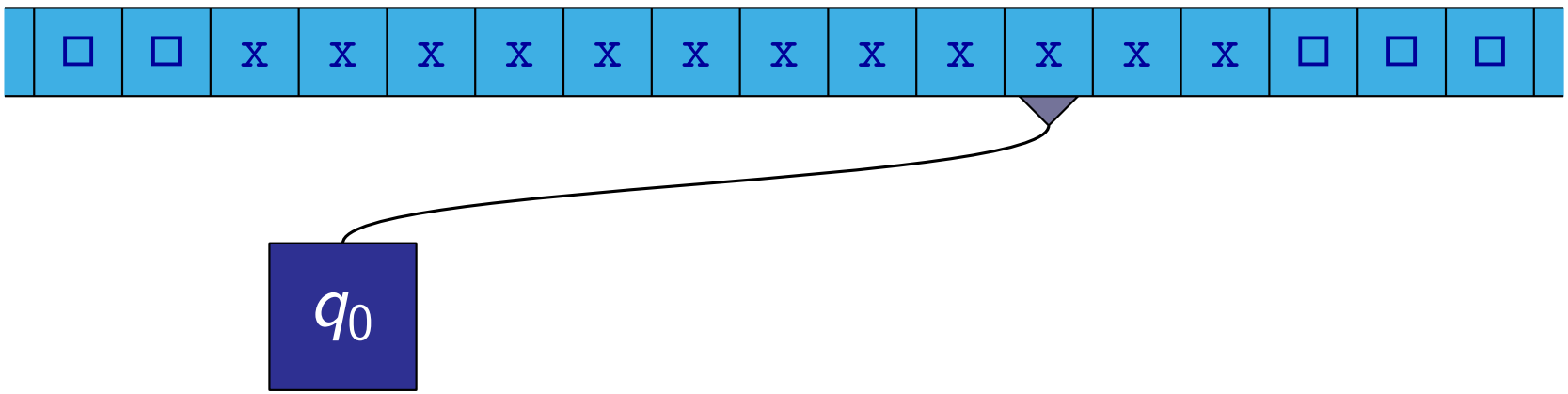
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



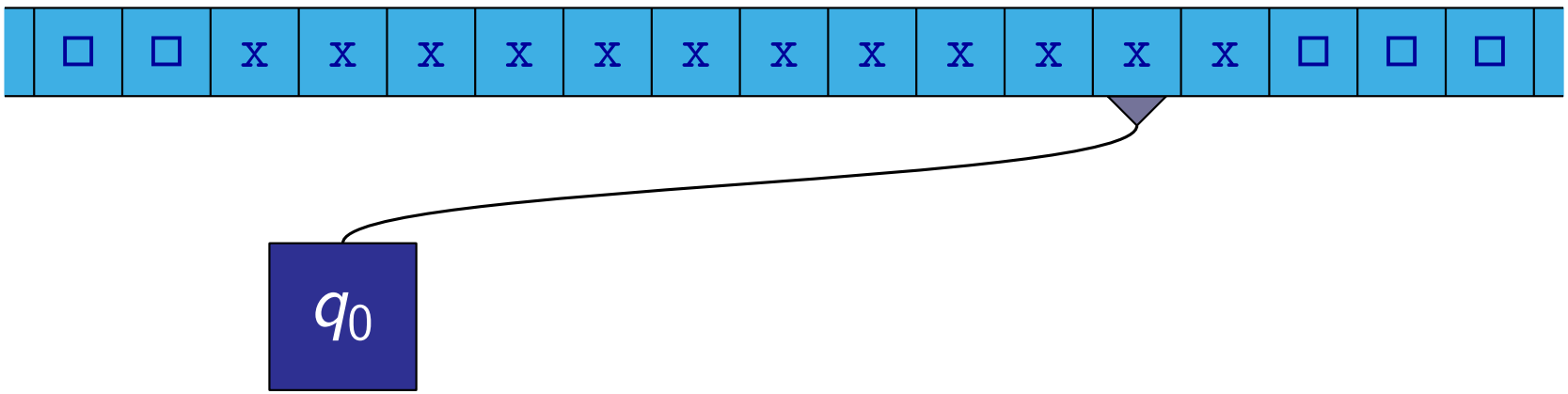
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



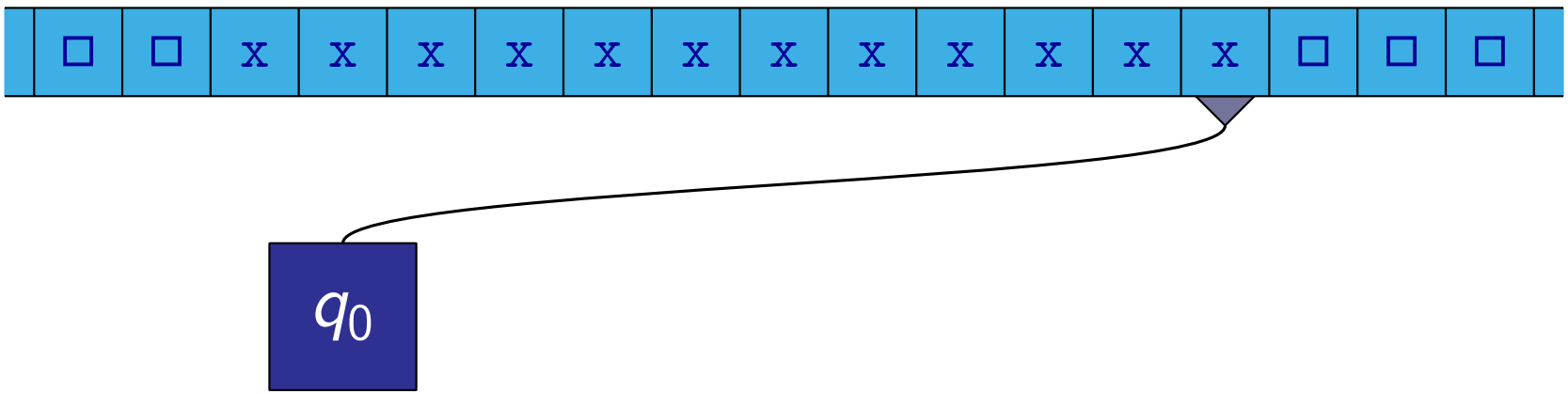
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



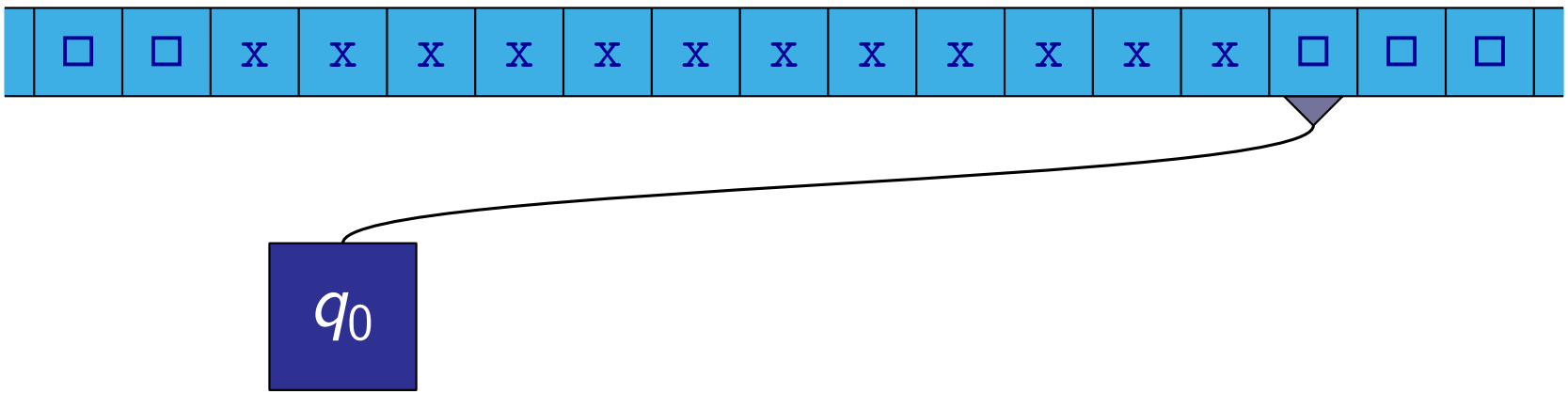
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



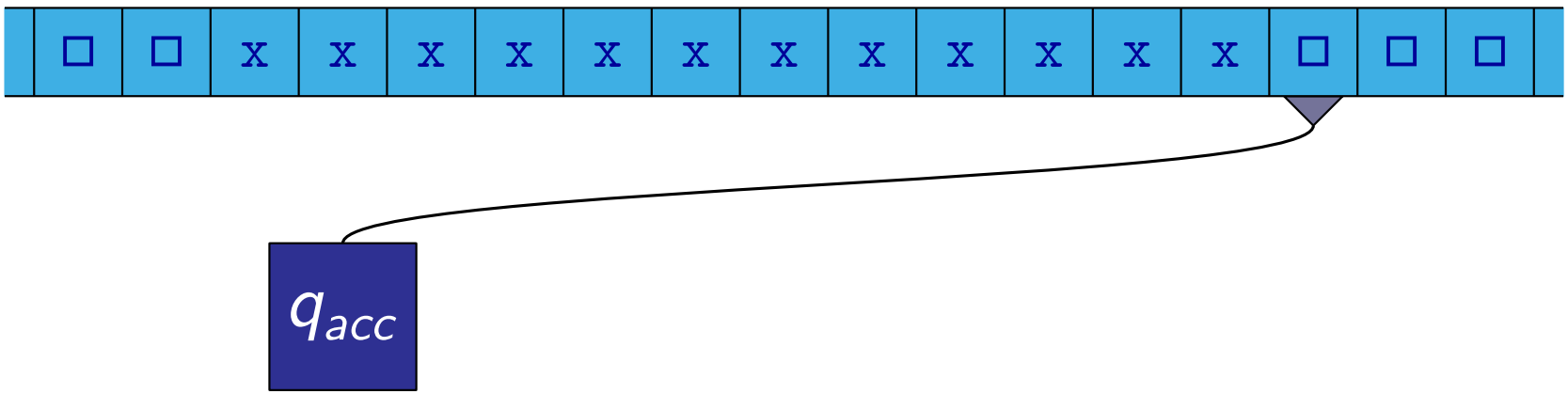
Turingův stroj

Jazyk $L = \{a^n b^n c^n \mid n \geq 0\}$

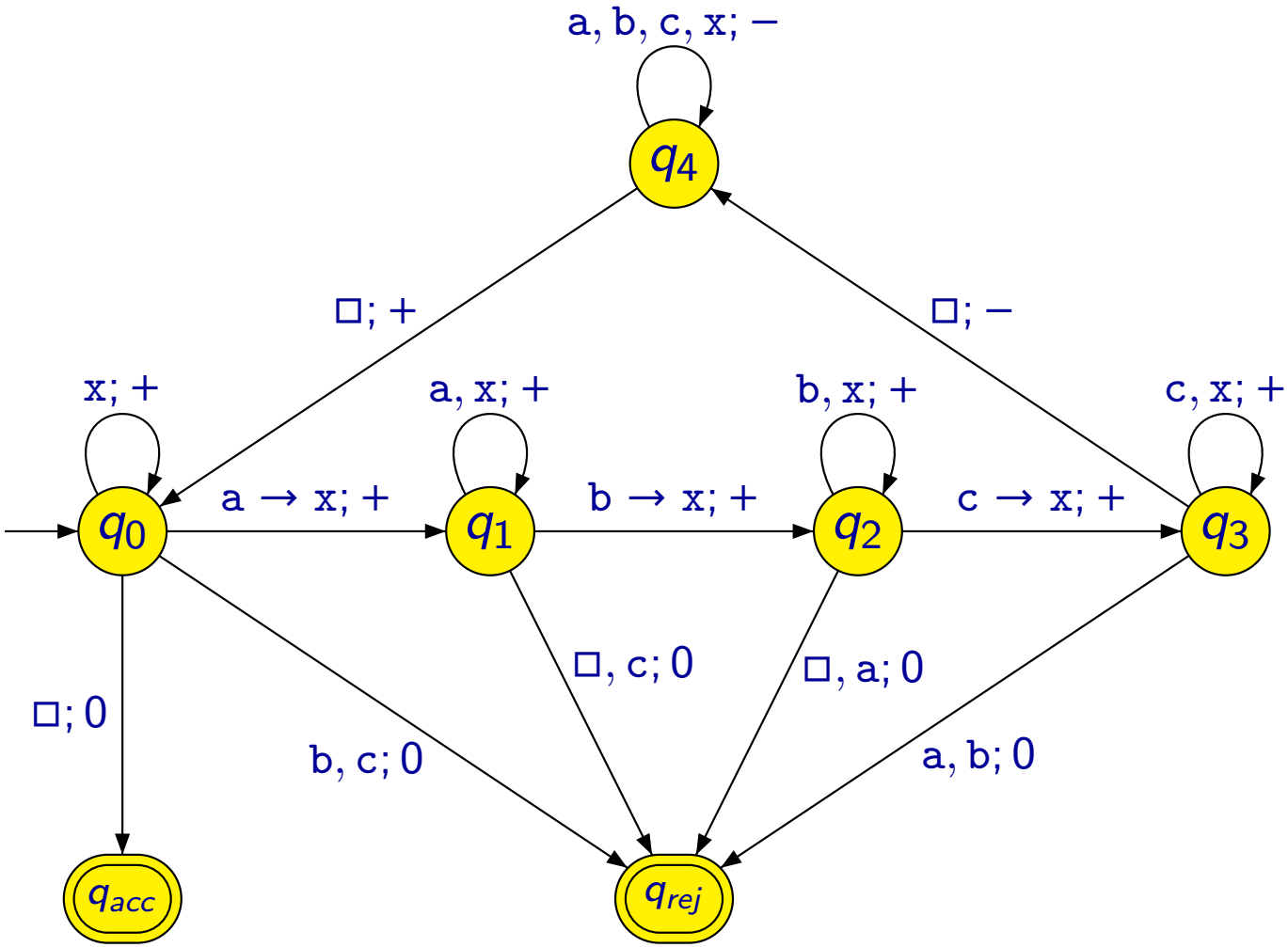
$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$ $\quad F = \{q_{acc}, q_{rej}\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ $\quad \Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{x}\}$

δ	$\square$	a	b	c	x
q_0	$(q_{acc}, \square, 0)$	$(q_1, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_0, \mathtt{x}, +1)$
q_1	$(q_{rej}, \square, 0)$	$(q_1, \mathtt{a}, +1)$	$(q_2, \mathtt{x}, +1)$	$(q_{rej}, \mathtt{c}, 0)$	$(q_1, \mathtt{x}, +1)$
q_2	$(q_{rej}, \square, 0)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_2, \mathtt{b}, +1)$	$(q_3, \mathtt{x}, +1)$	$(q_2, \mathtt{x}, +1)$
q_3	$(q_4, \square, -1)$	$(q_{rej}, \mathtt{a}, 0)$	$(q_{rej}, \mathtt{b}, 0)$	$(q_3, \mathtt{c}, +1)$	$(q_3, \mathtt{x}, +1)$
q_4	$(q_0, \square, +1)$	$(q_4, \mathtt{a}, -1)$	$(q_4, \mathtt{b}, -1)$	$(q_4, \mathtt{c}, -1)$	$(q_4, \mathtt{x}, -1)$



Turingův stroj

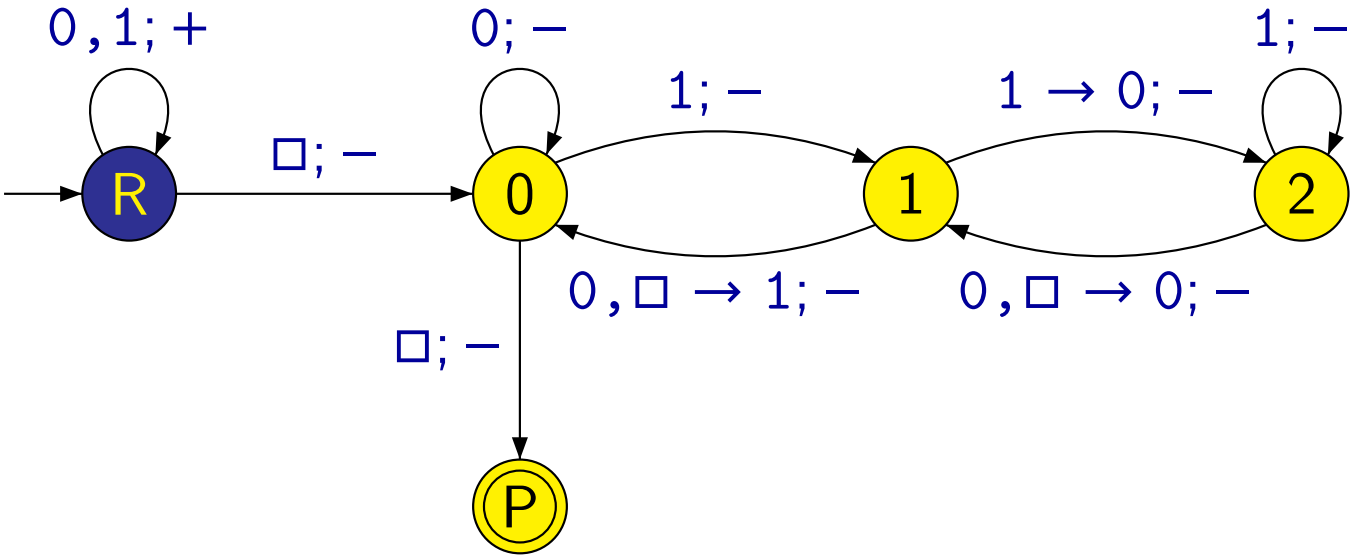


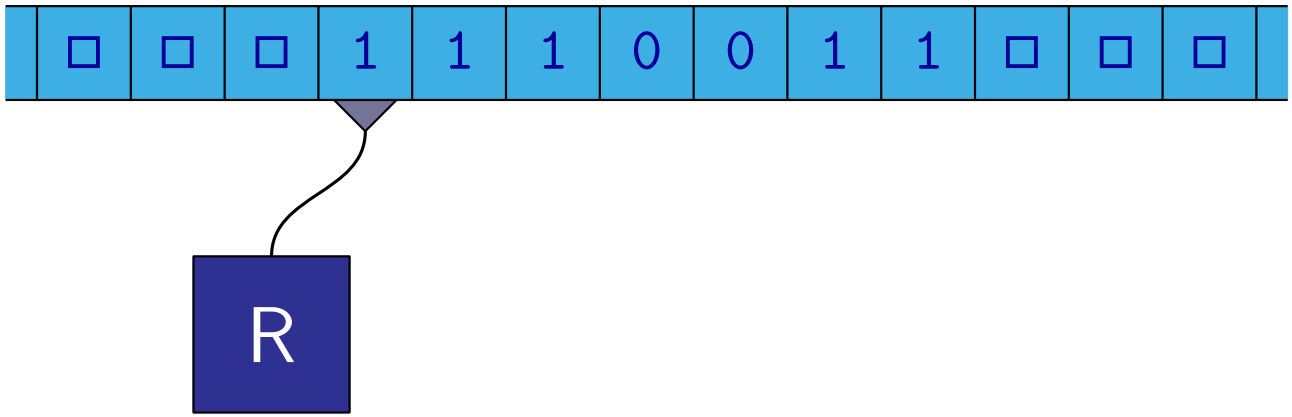
Turingův stroj

- Turingův stroj nemusí dávat jen odpověď Ano nebo Ne, ale může realizovat nějakou funkci, která každému slovu ze Σ^* přiřazuje nějaké jiné slovo (z Γ^*).

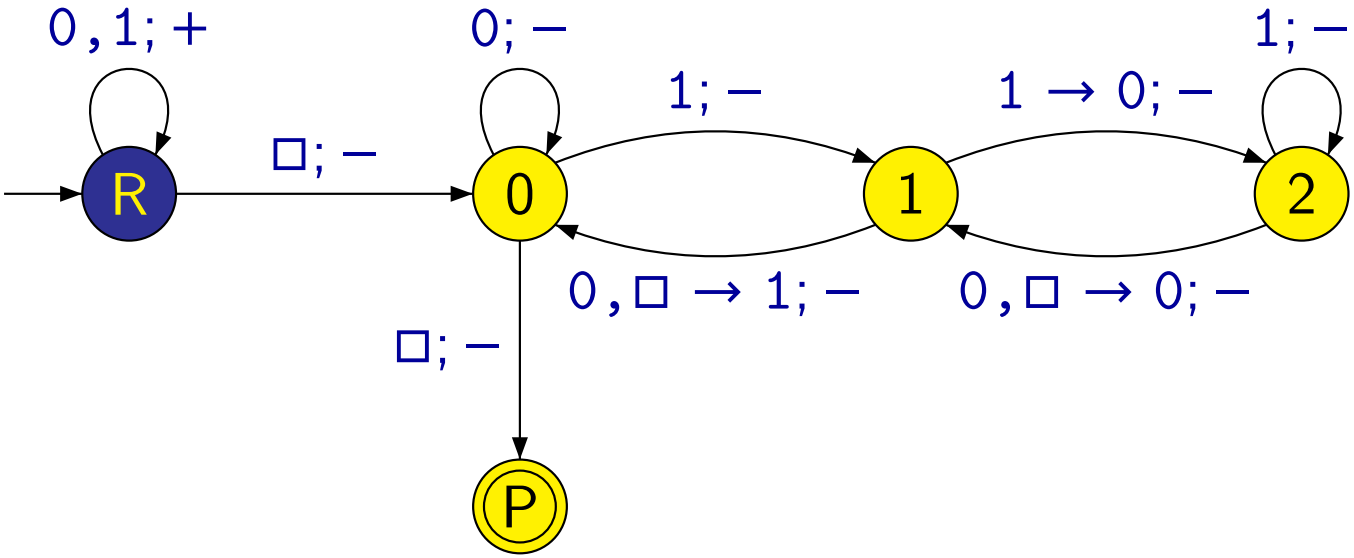
- Slovo přiřazené slovu w je slovo, které zůstane zapsáno na pásce po výpočtu nad slovem w, když odstraníme všechny znaky $\square$.

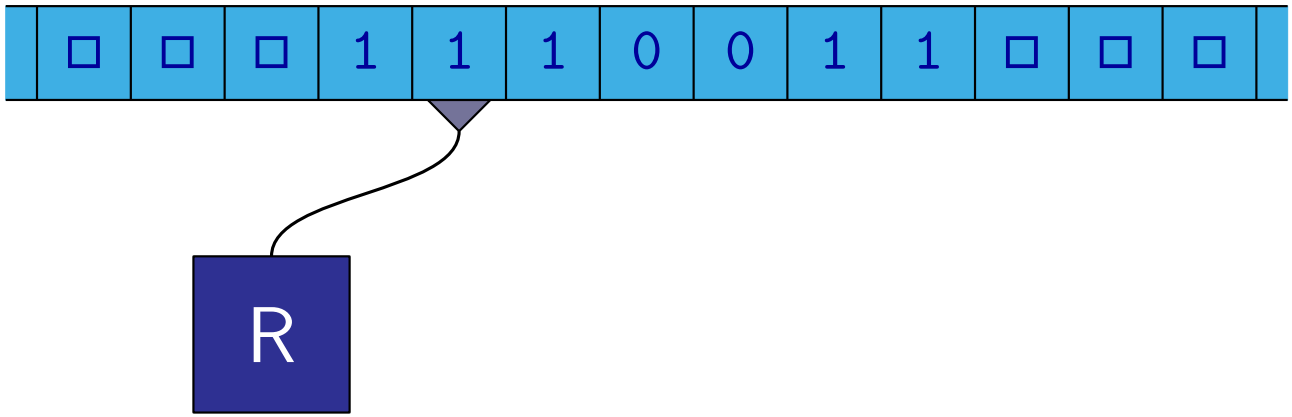
Turingův stroj – násobení třemi



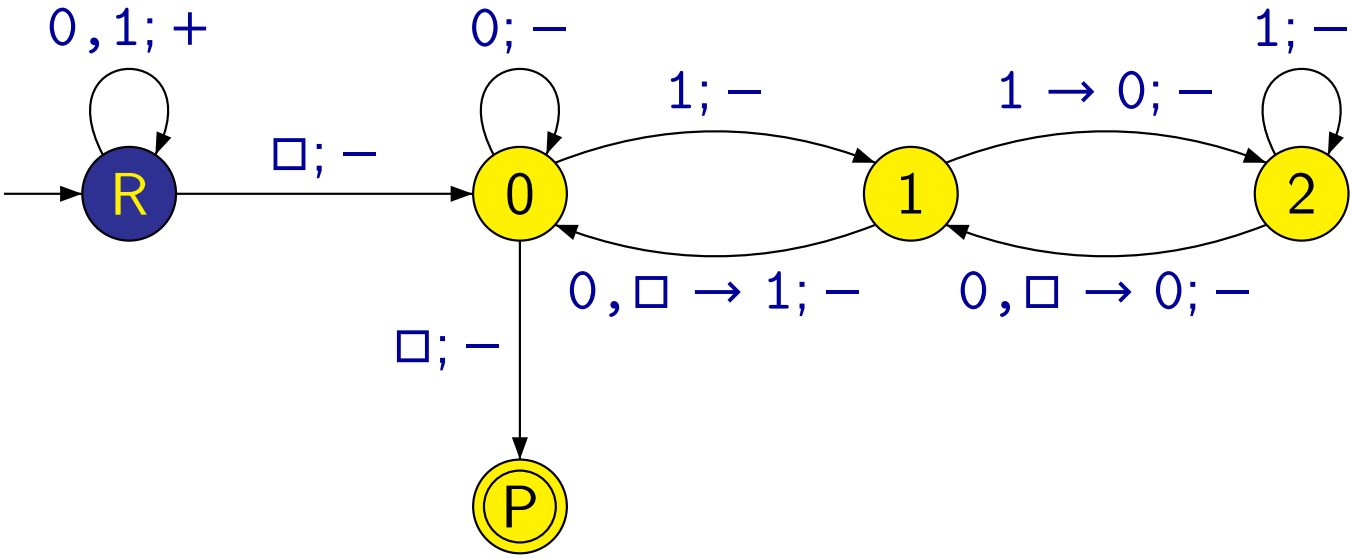


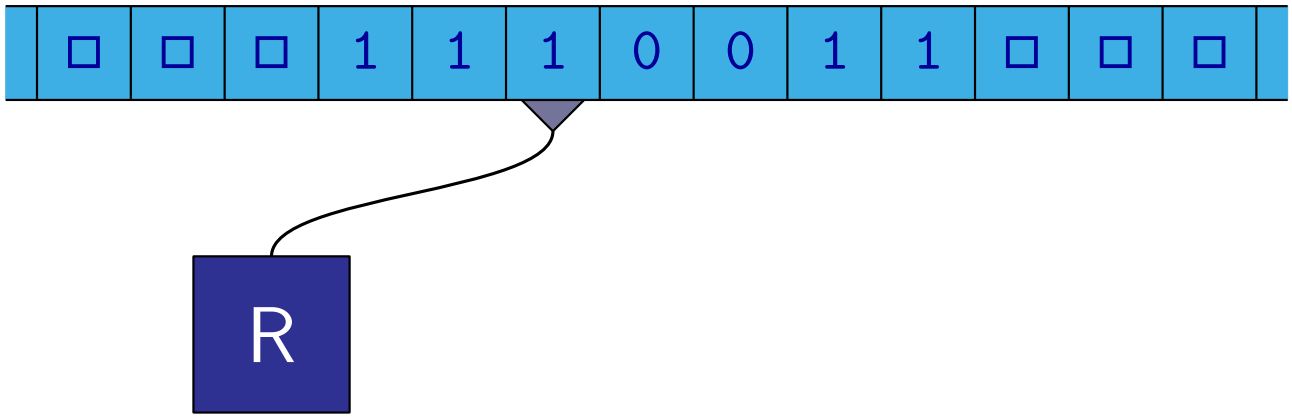
Turingův stroj – násobení třemi



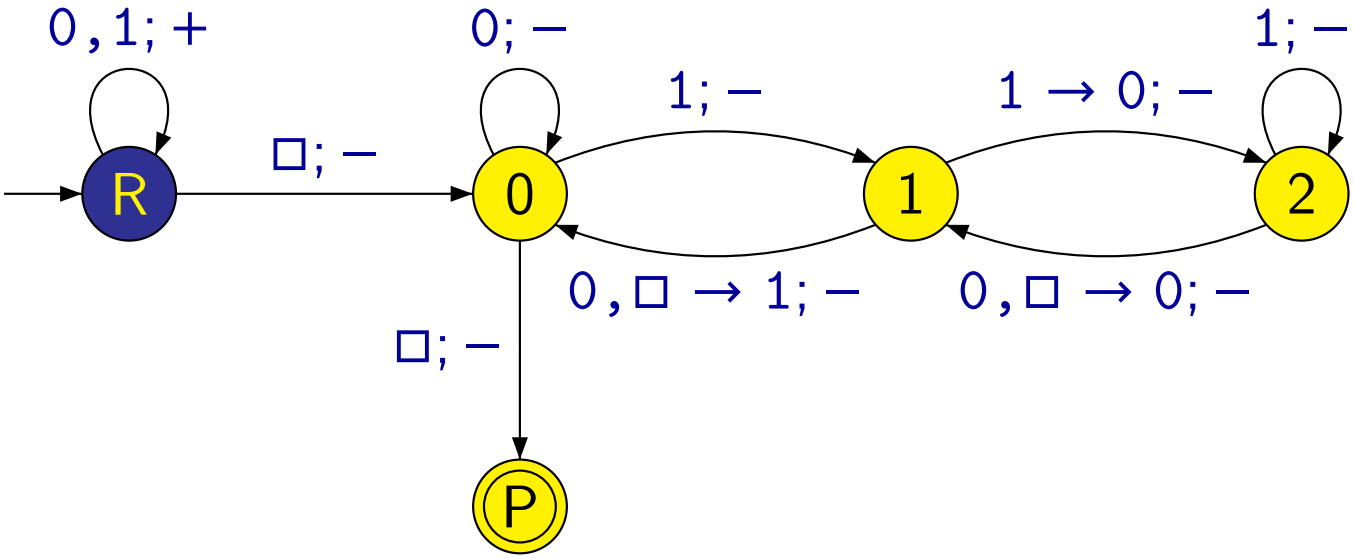


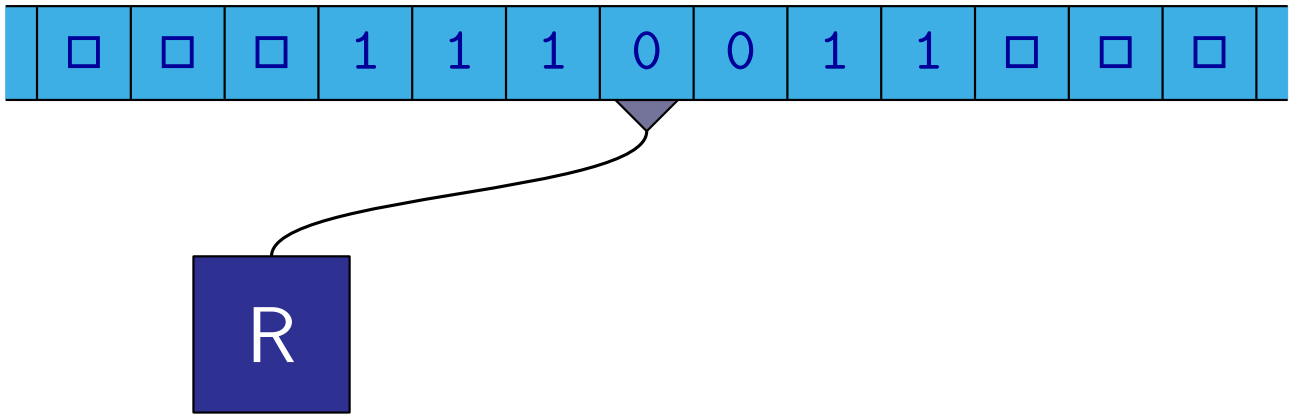
Turingův stroj – násobení třemi



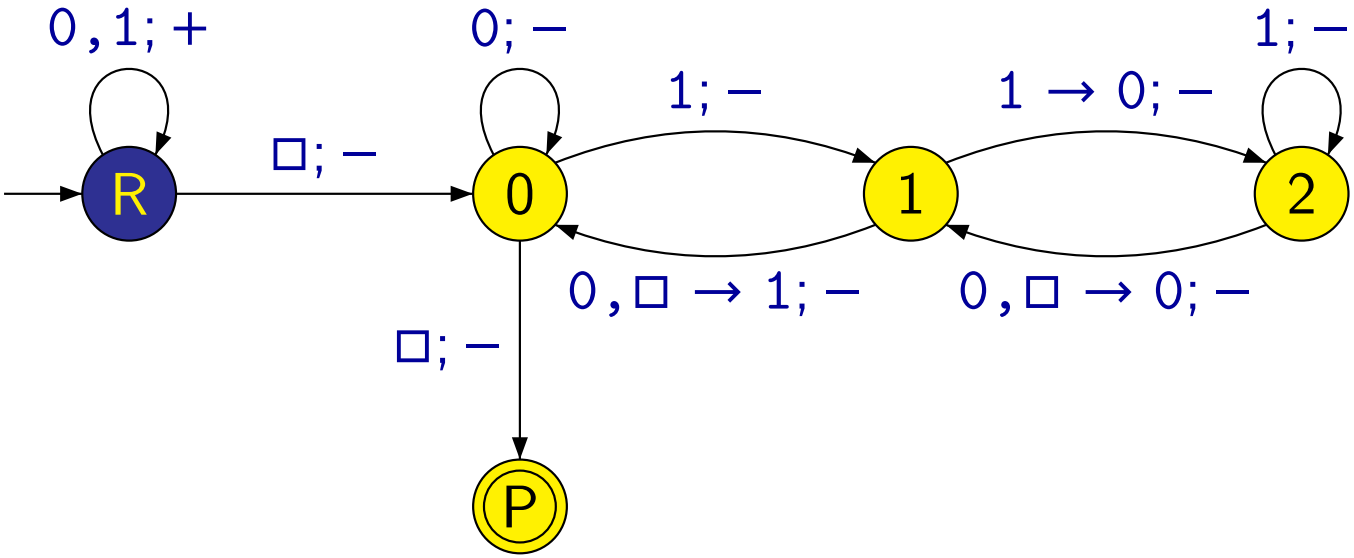


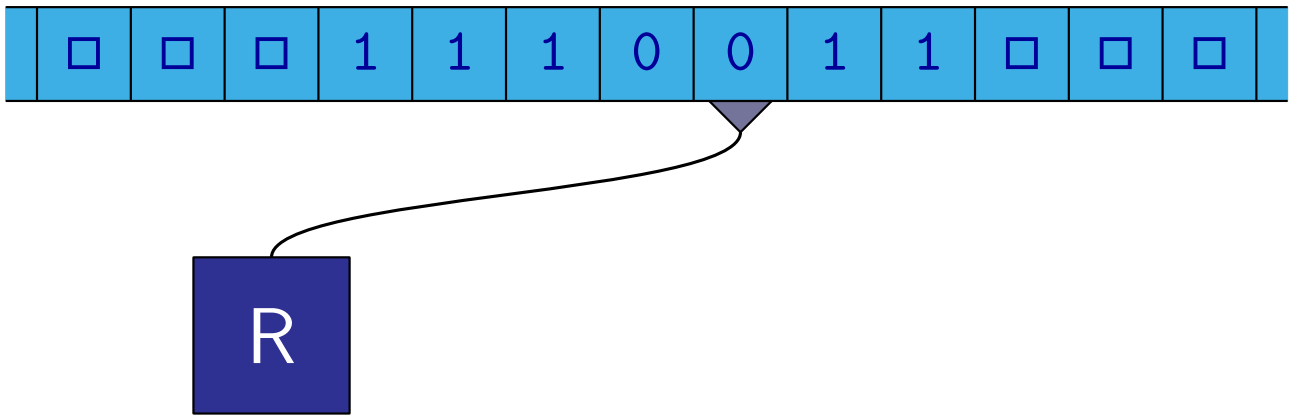
Turingův stroj – násobení třemi



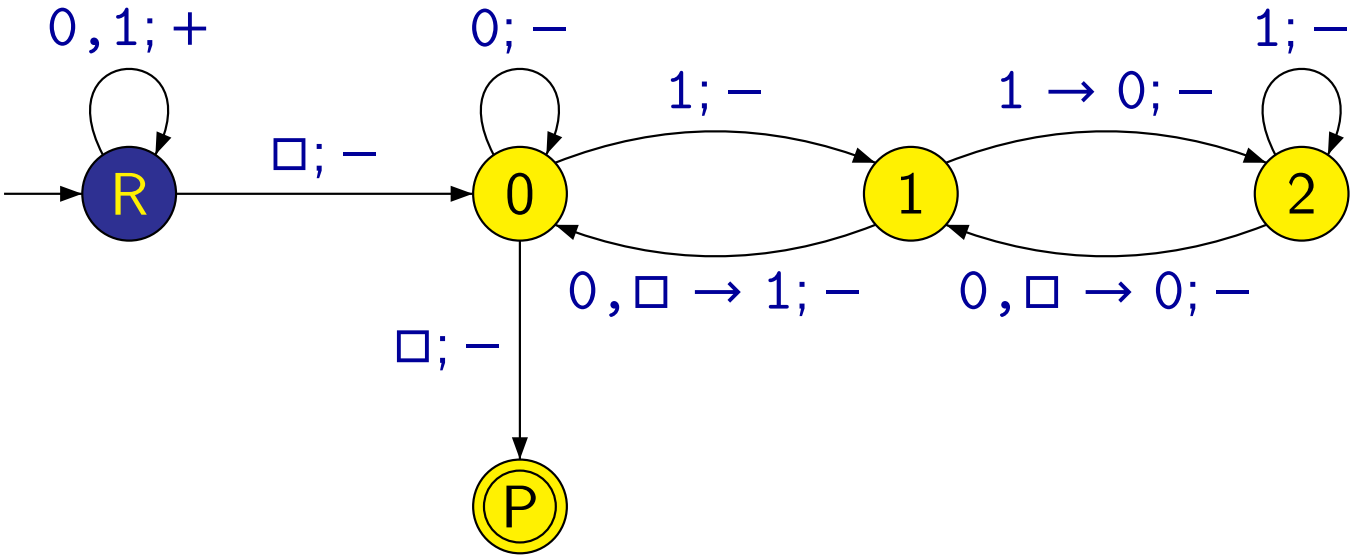


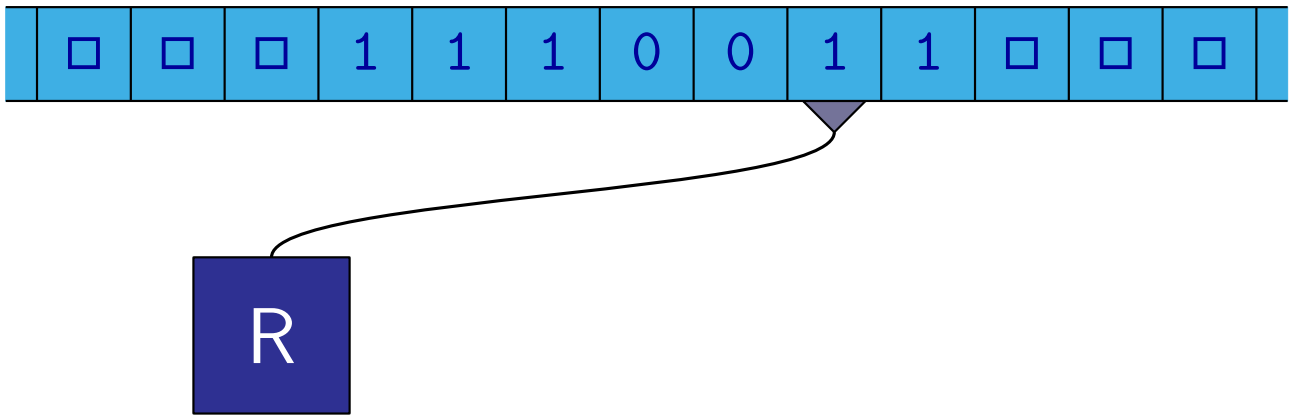
Turingův stroj – násobení třemi



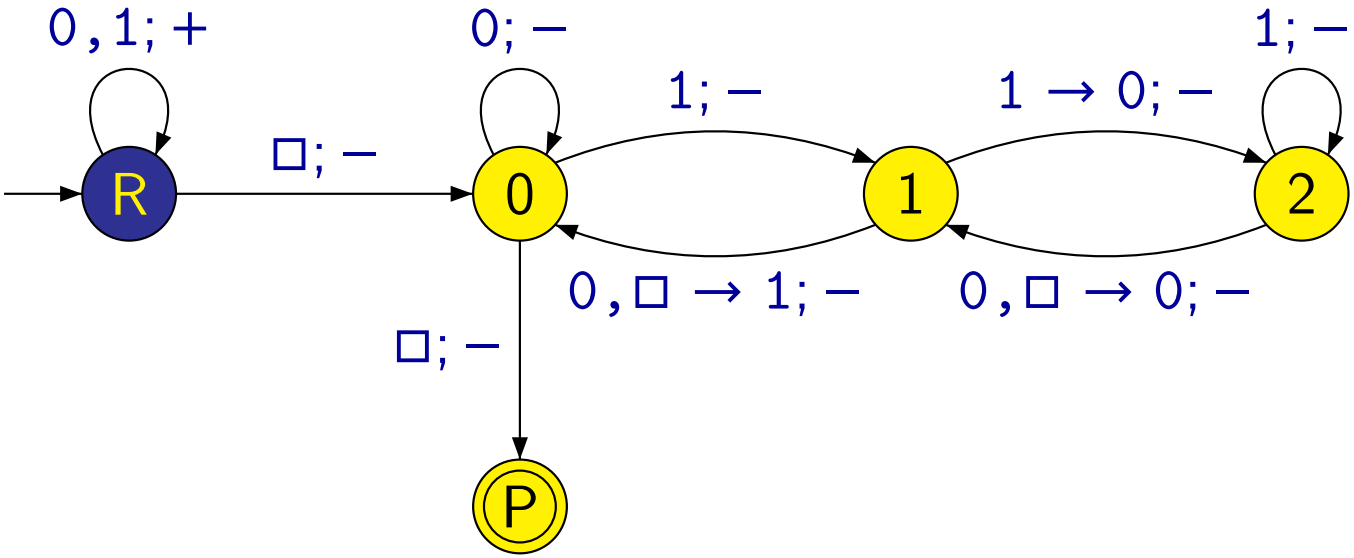


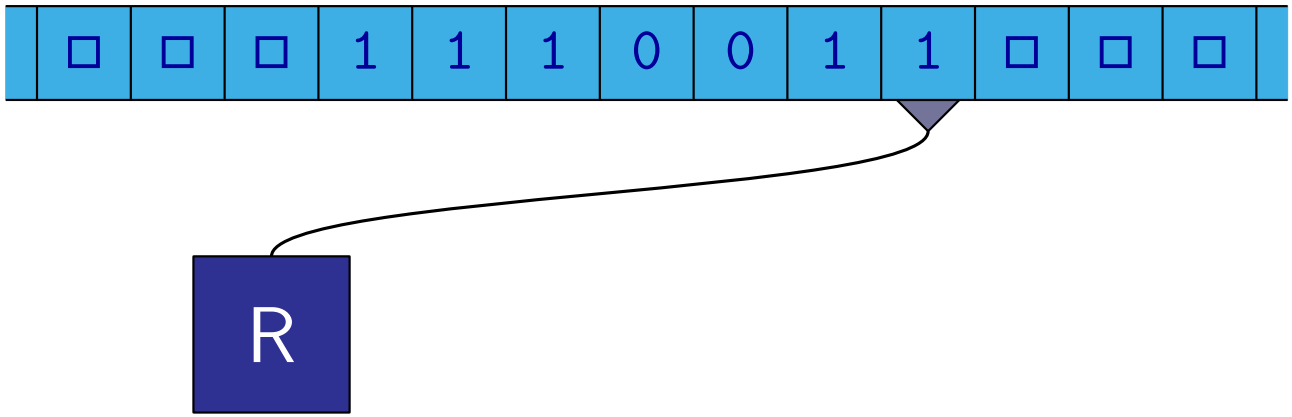
Turingův stroj – násobení třemi



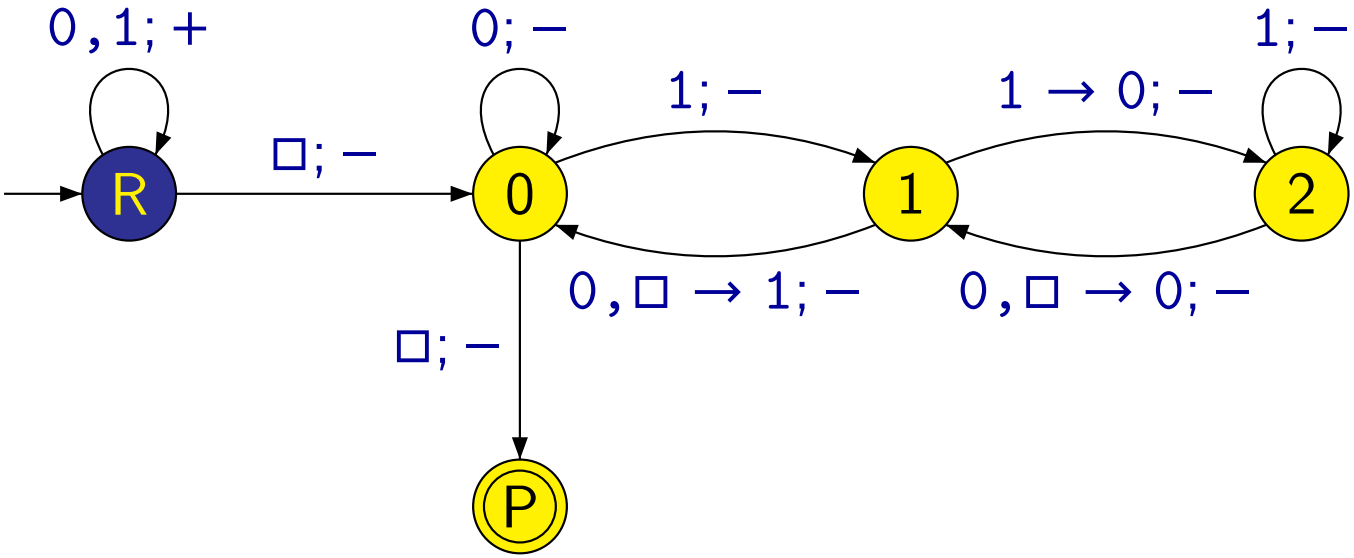


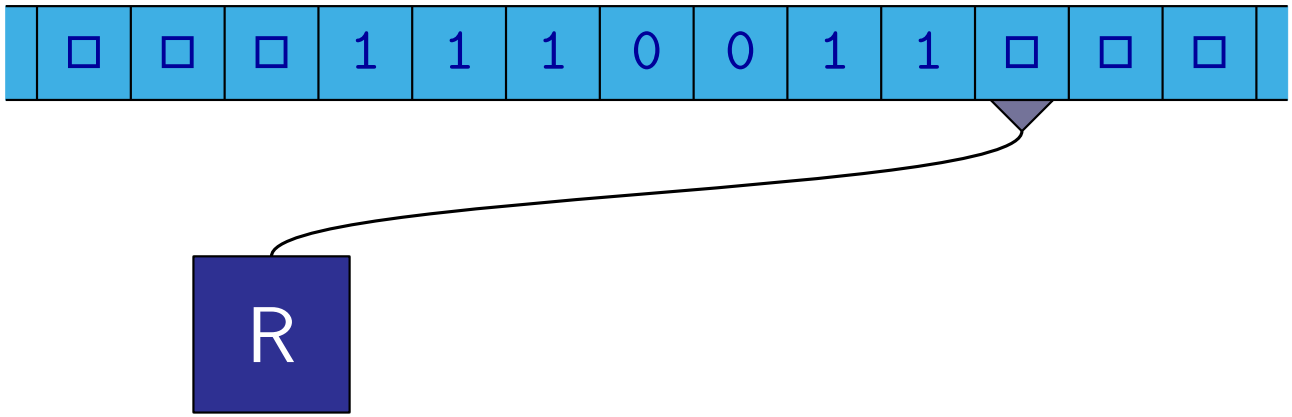
Turingův stroj – násobení třemi



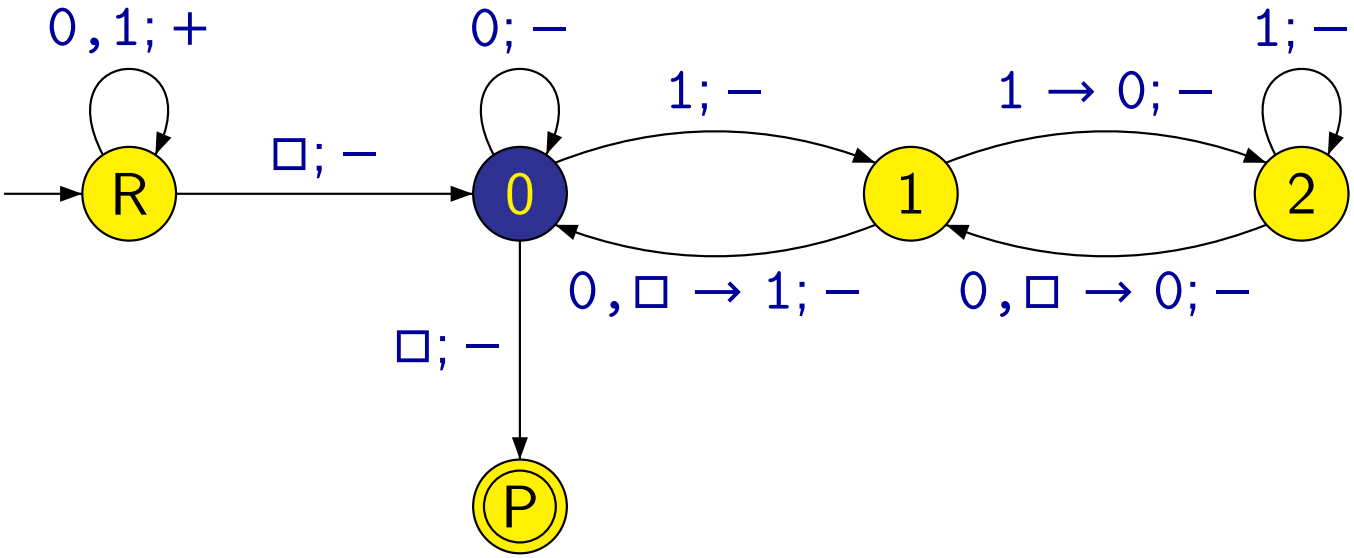


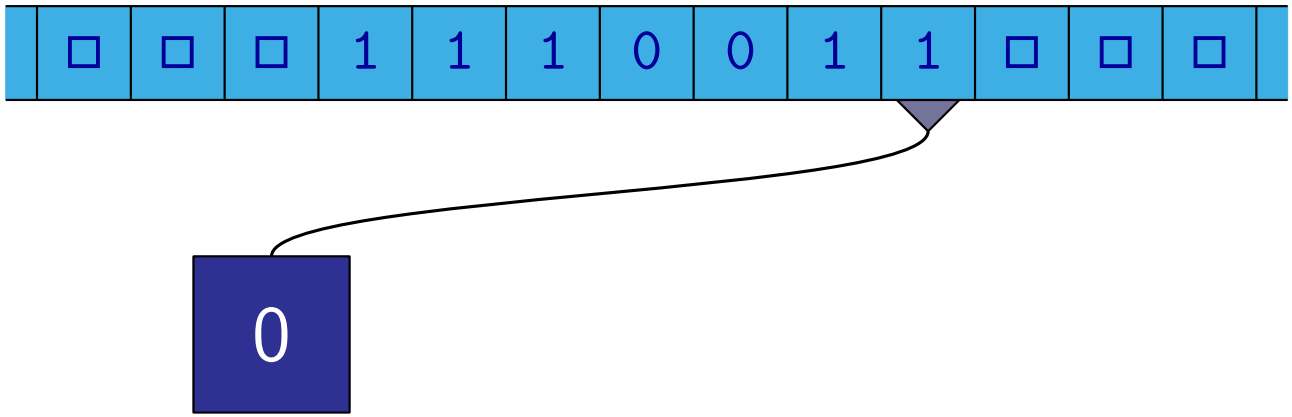
Turingův stroj – násobení třemi



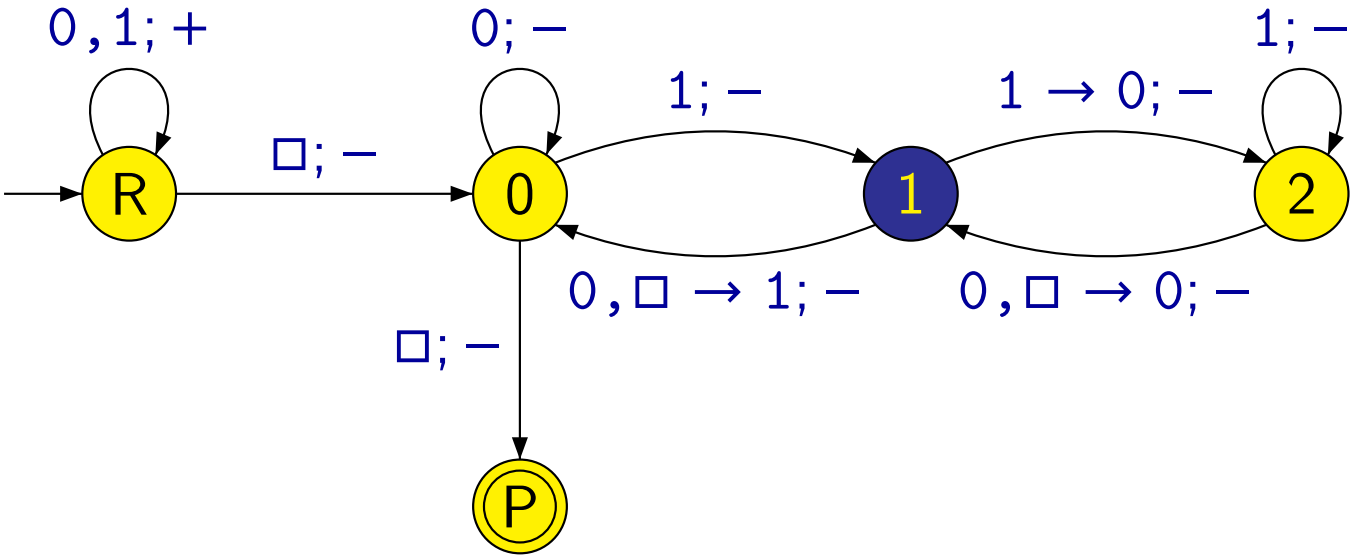


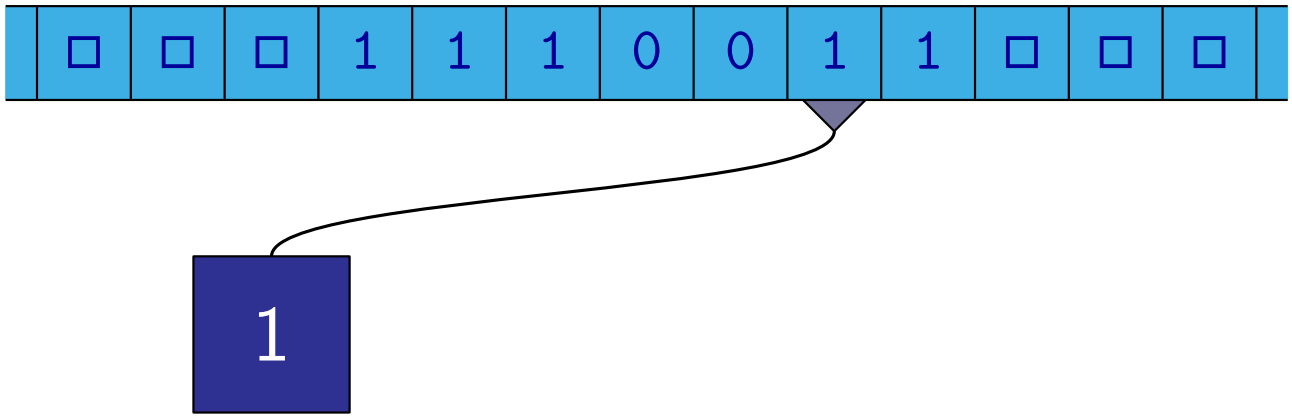
Turingův stroj – násobení třemi



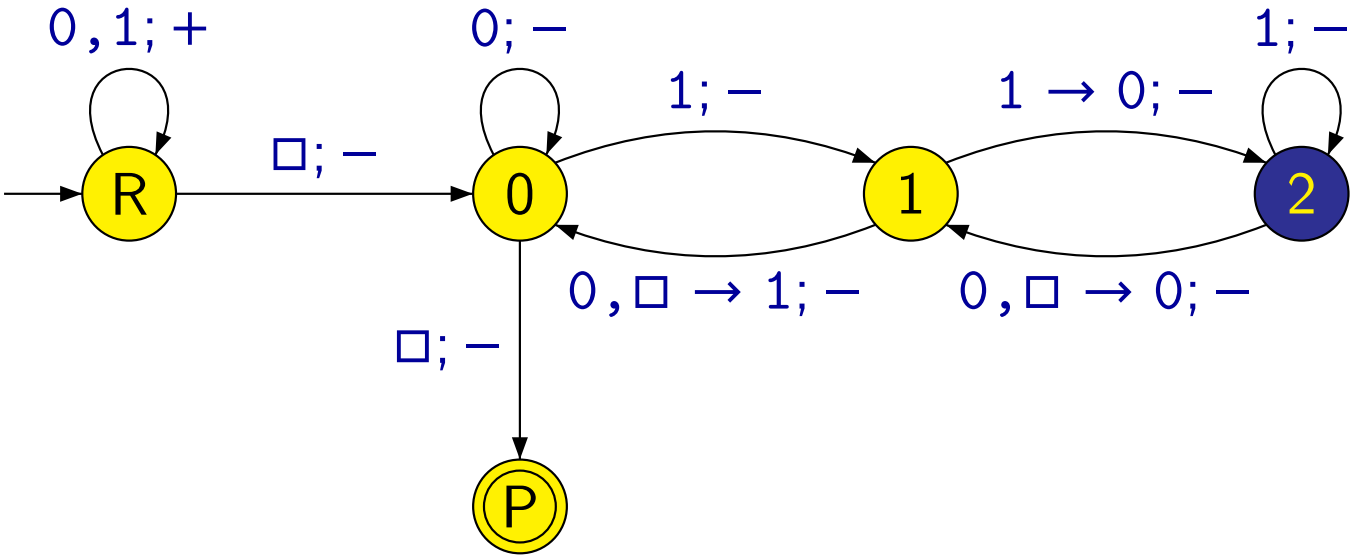


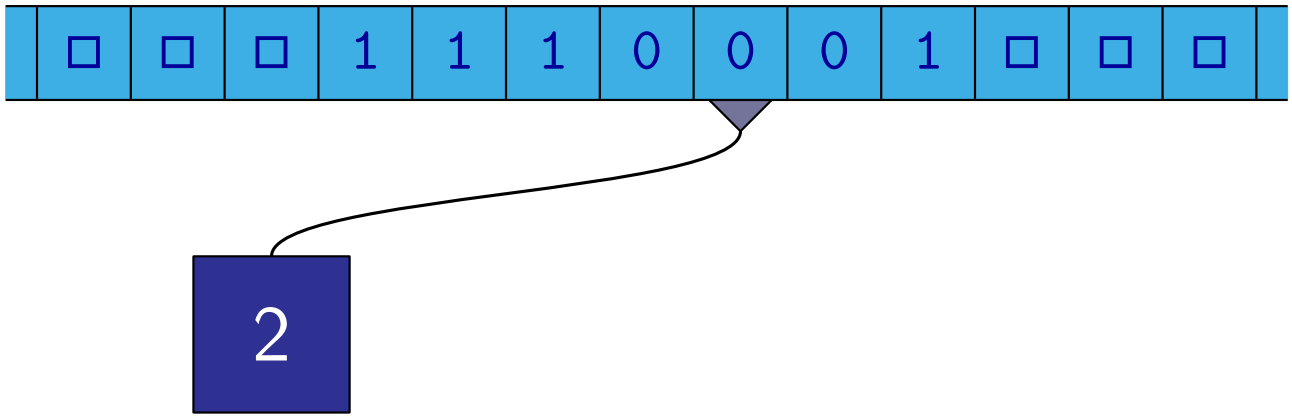
Turingův stroj – násobení třemi



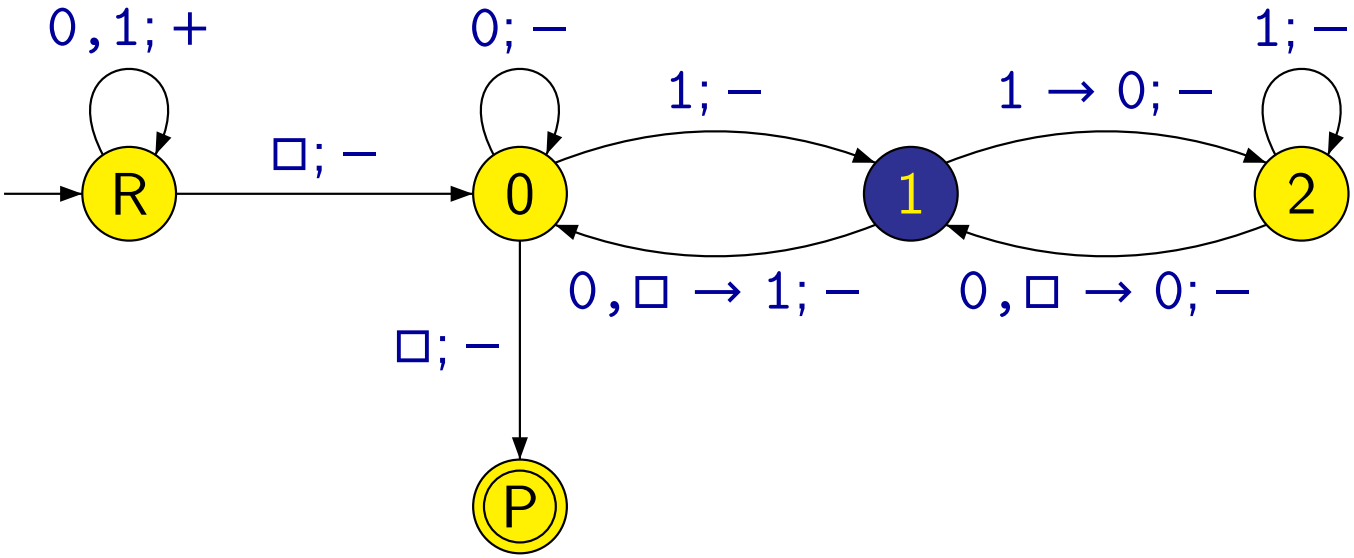


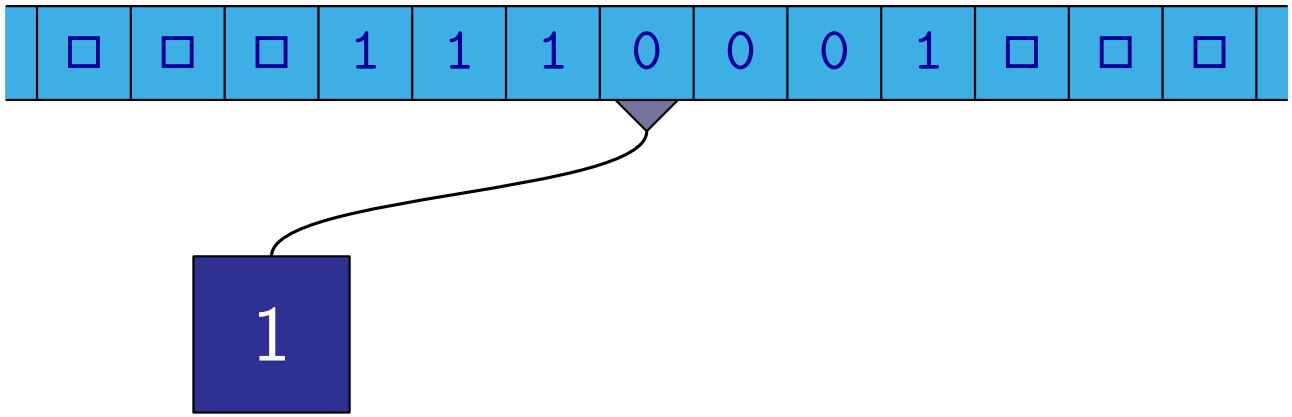
Turingův stroj – násobení třemi



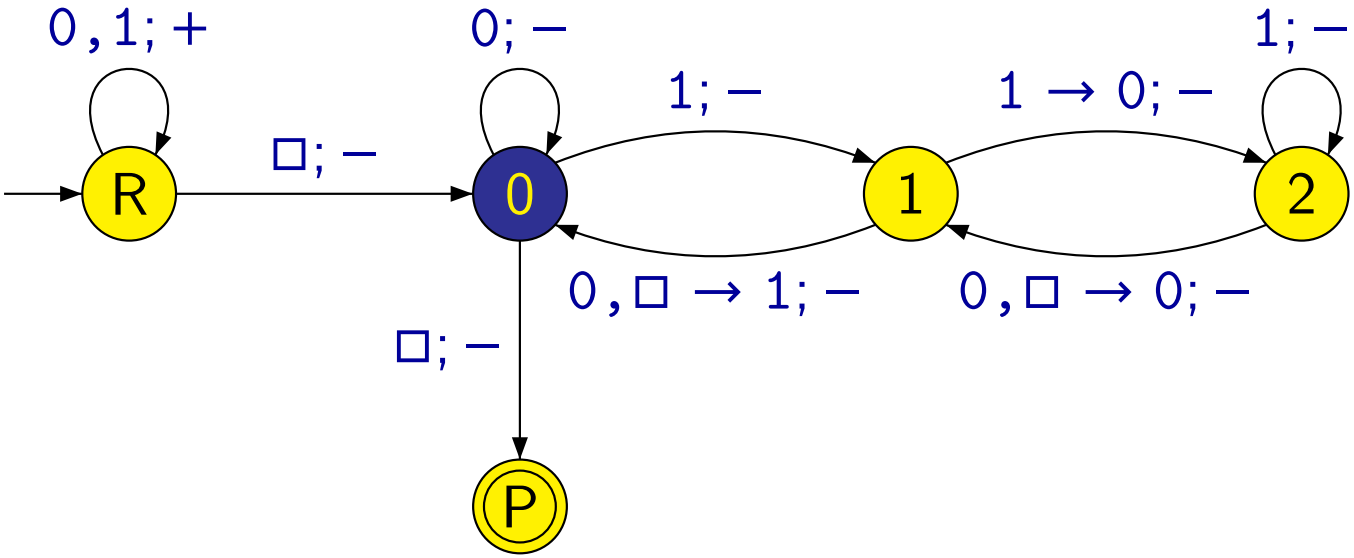


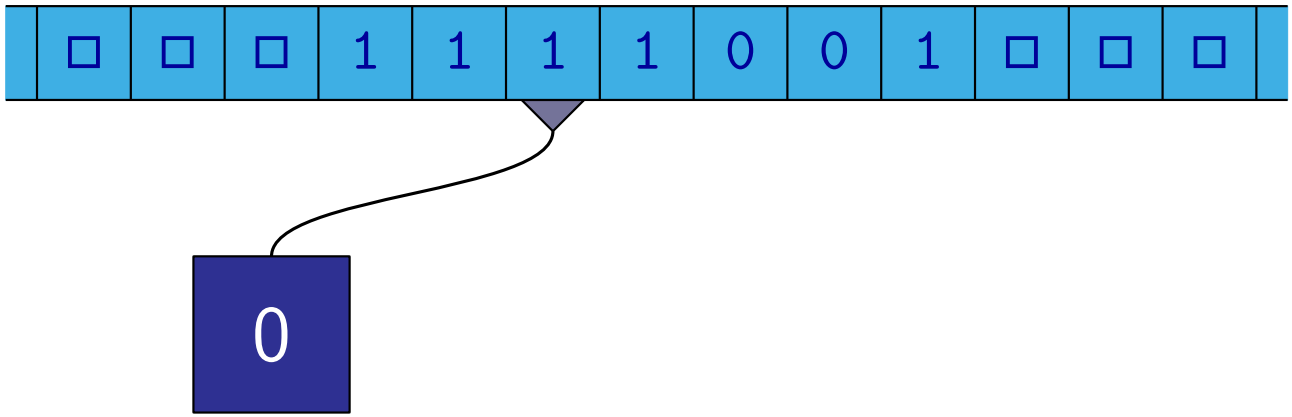
Turingův stroj – násobení třemi



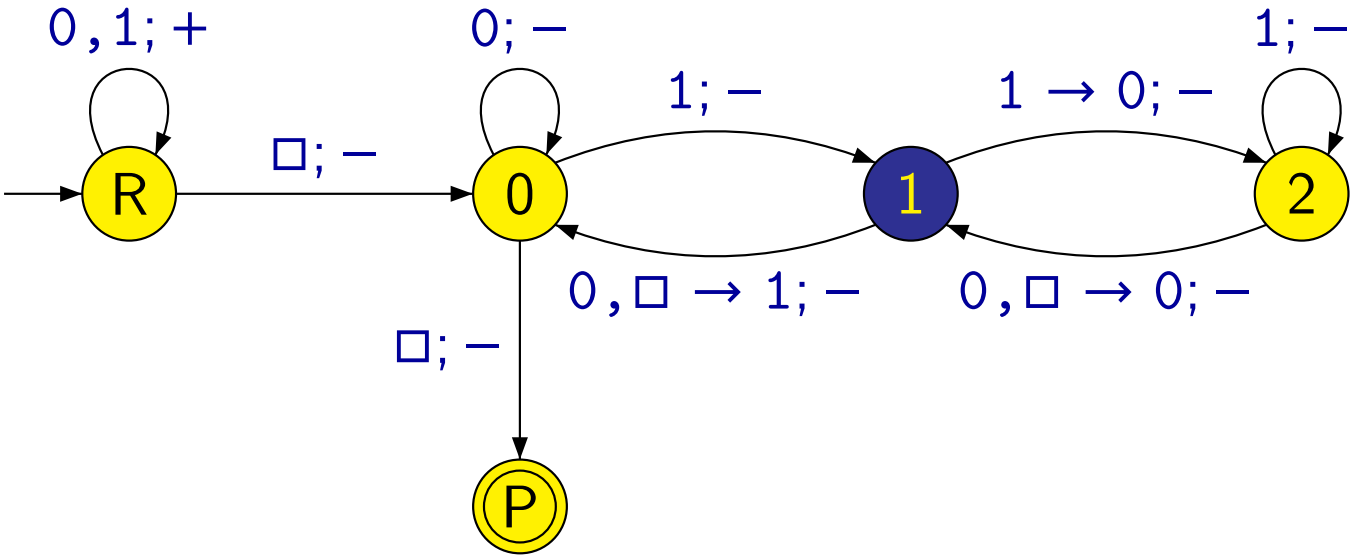


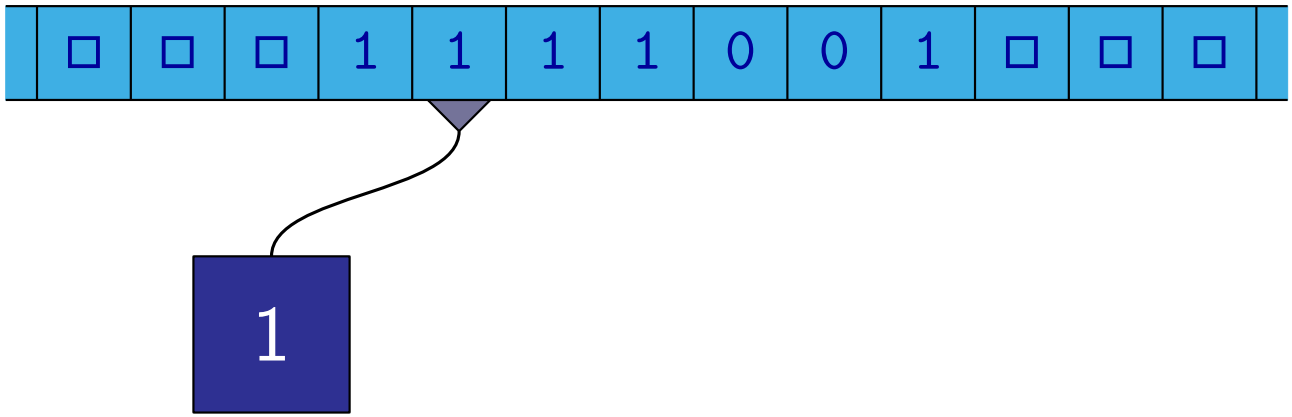
Turingův stroj – násobení třemi



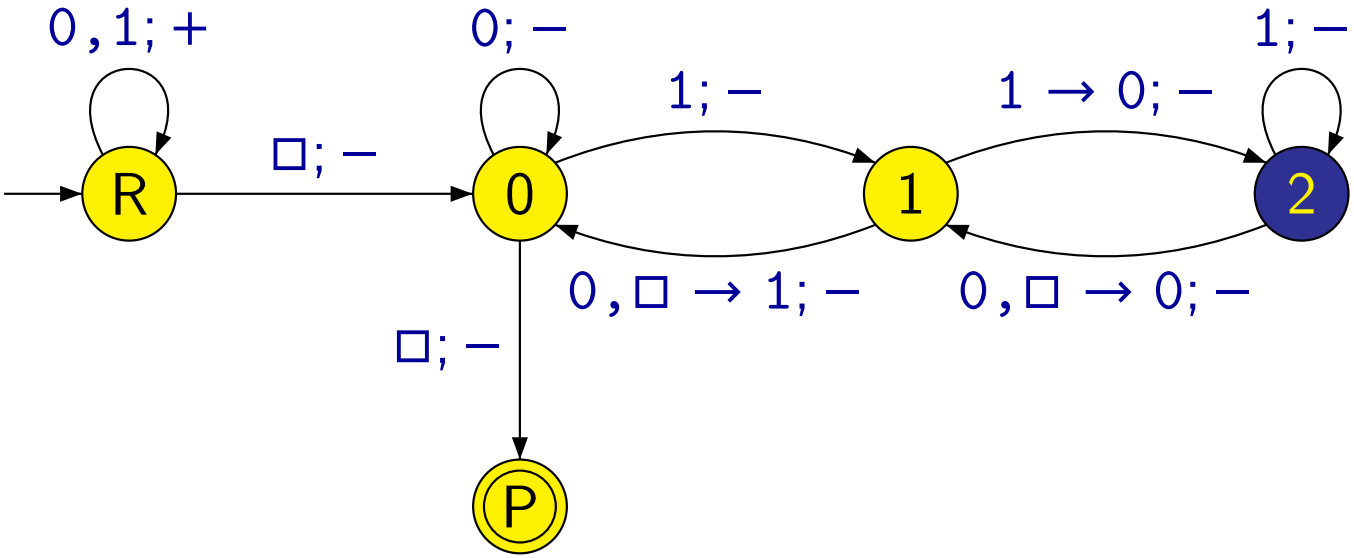


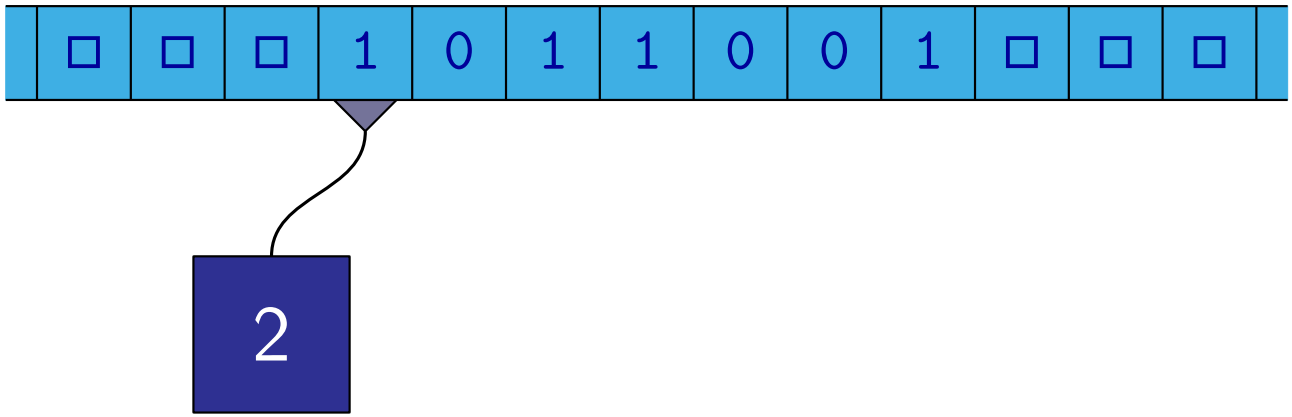
Turingův stroj – násobení třemi



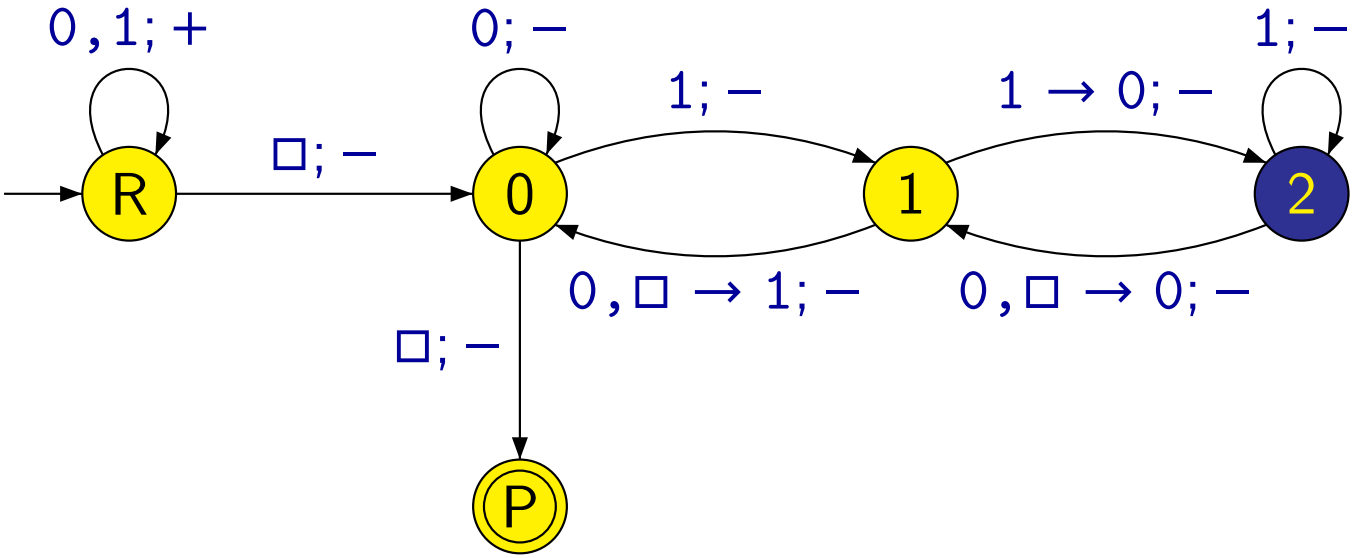


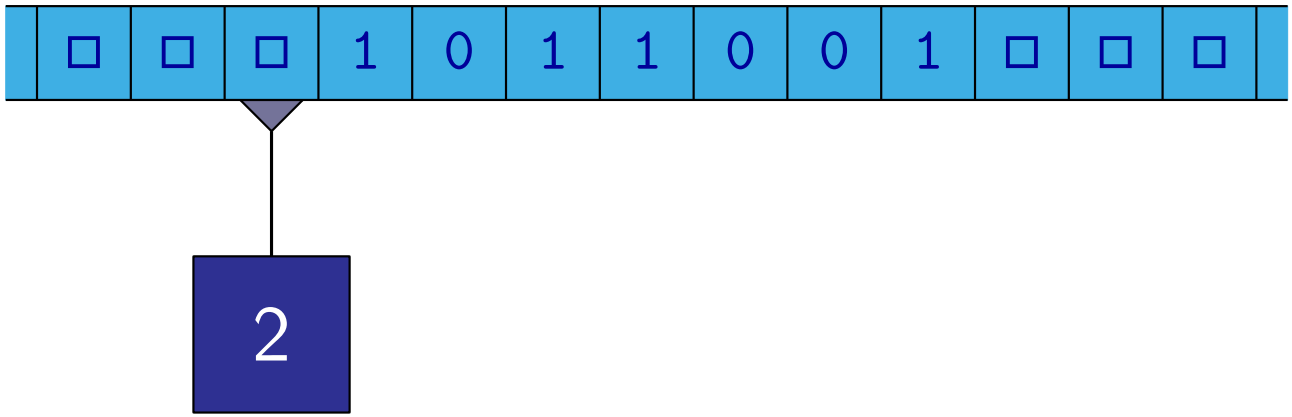
Turingův stroj – násobení třemi



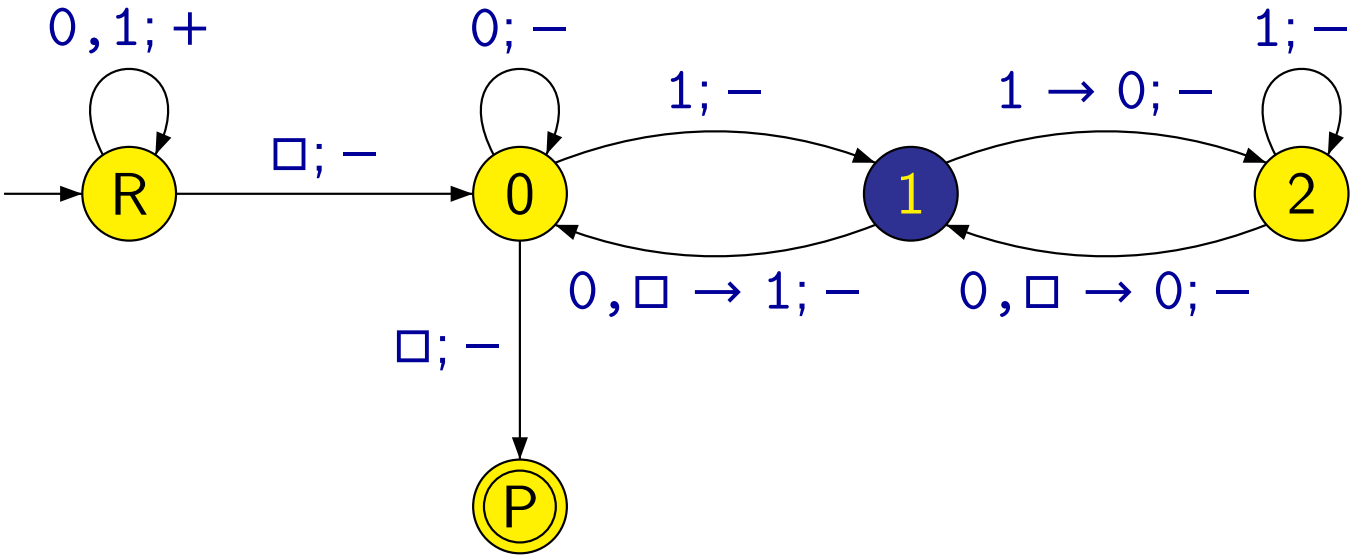


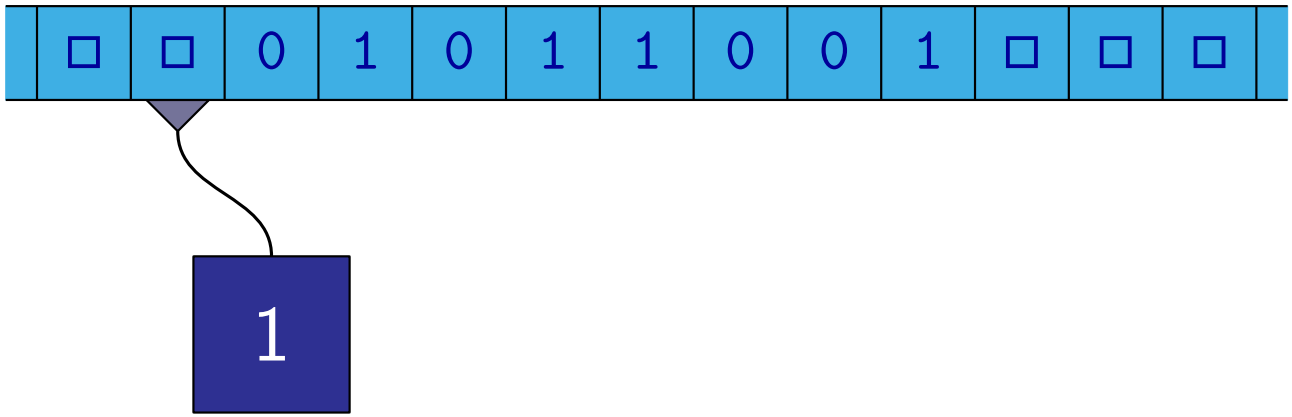
Turingův stroj – násobení třemi



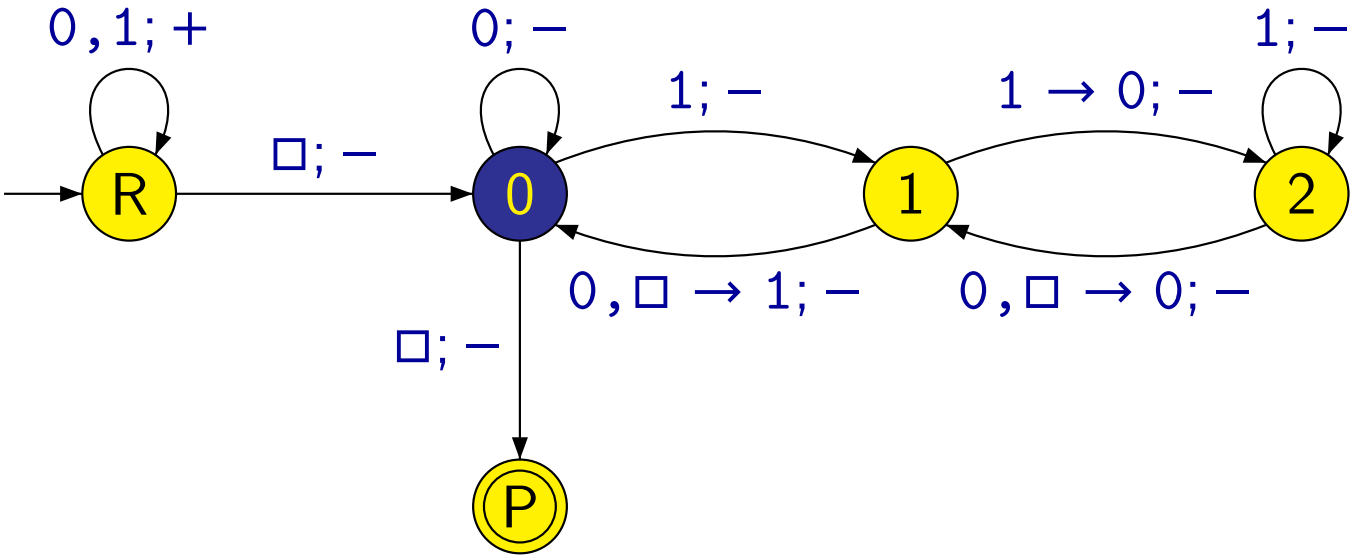


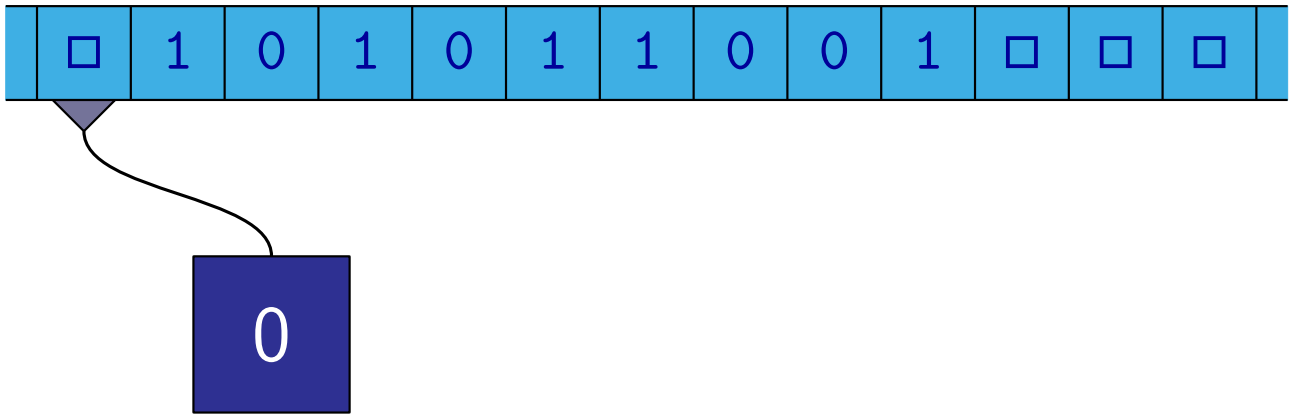
Turingův stroj – násobení třemi



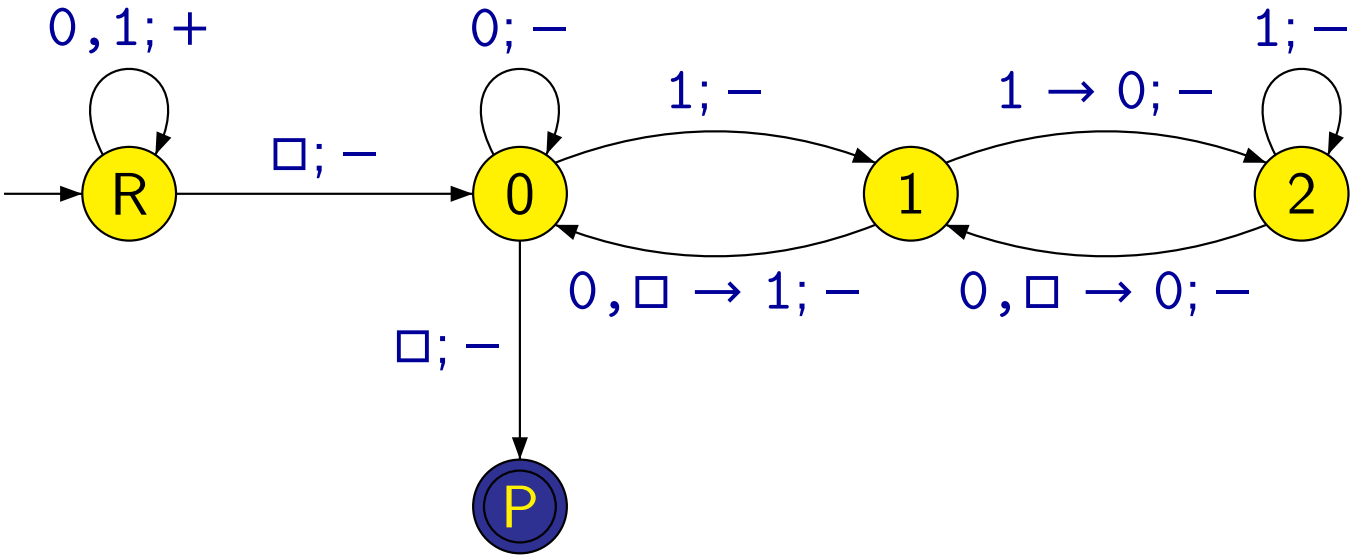


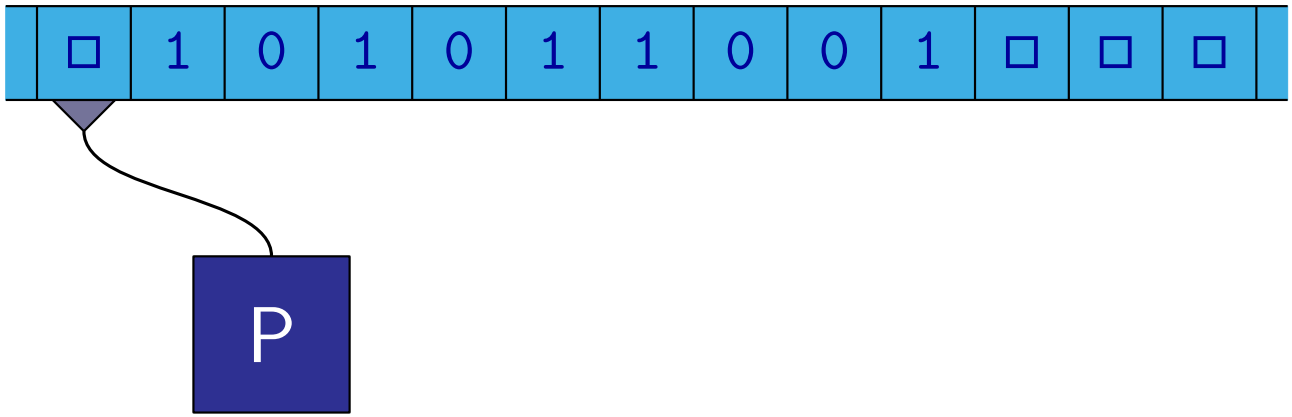
Turingův stroj – násobení třemi





Turingův stroj – násobení třemi





Nedeterministické Turingovy stroje

Můžeme uvažovat i **nedeterministické Turingovy stroje**, kde pro každý stav q a symbol b přechodová funkce $\delta(q, b)$ určuje více různých trojic (q', b', d).

Stroj si může vybrat libovolnou z nich.

Stroj přijímá slovo w, jestliže existuje alespoň jeden jeho výpočet vedoucí k přijetí slova w.

Poznámka: Ke každému nedeterministickému Turingovu stroji je možné sestrojit ekvivalentní deterministický Turingův stroj.

Nedeterministické Turingovy stroje

Formálně se v definici deterministického a nedeterministického Turingova stroje $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F)$ liší pouze definice přechodové funkce δ:

- **Deterministický** Turingův stroj:

$$\delta : (Q - F) \times \Gamma \rightarrow Q \times \Gamma \times \{-1, 0, +1\}$$

- **Nedeterministický** Turingův stroj:

$$\delta : (Q - F) \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{-1, 0, +1\})$$

Poznámka: U nedeterministických Turingových strojů také nedává příliš smysl jiná množina koncových stavů než $F = \{q_{acc}, q_{rej}\}$.

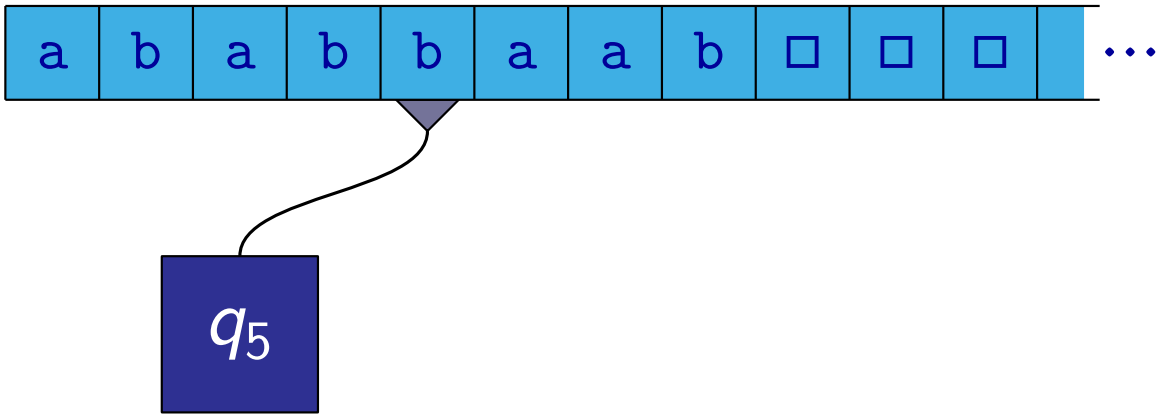
Varianty Turingových strojů

- Dříve uvedená definice Turingova stroje je jen jednou z mnoha možných variant.
- Uvedeme několik příkladů toho, v čem se mohou některé jiné varianty Turingových strojů lišit.
- Prakticky všechny tyto varianty Turingových strojů jsou schopny přijímat či rozpoznávat tytéž jazyky a počítat tytéž funkce.
- Co se týká doby výpočtu a množství použité paměti, mezi různými variantami mohou, ale nemusí být významné rozdíly.
- Všechny níže uvedené varianty můžeme uvažovat v deterministické i nedeterministické verzi.

Varianty Turingových strojů

Jednostranně či **oboustranně** nekonečná páska:

- V předchozí definici jsme uvažovali pásku, která je nekonečná jak směrem doleva, tak směrem doprava.
- Místo toho se někdy v definici Turingova stroje uvažuje páska, která je nekonečná jen směrem doprava.

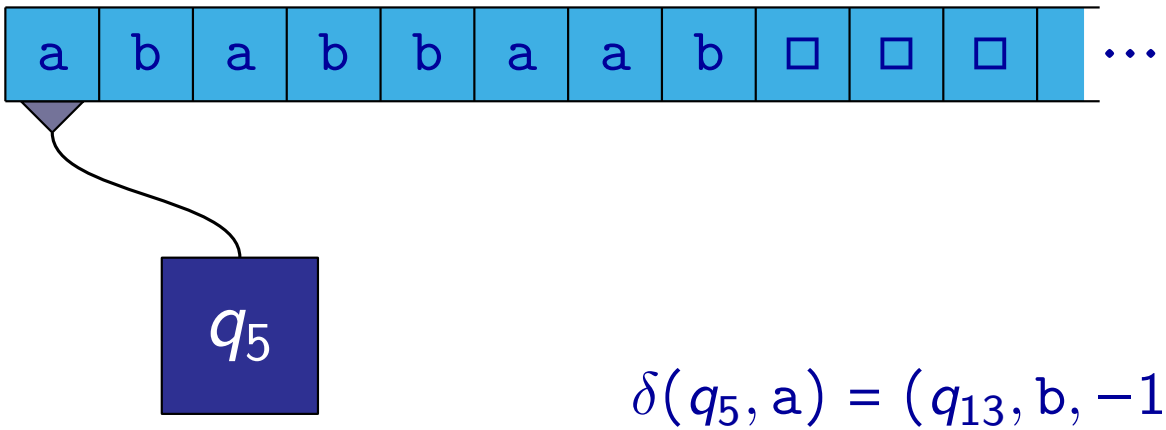


Varianty Turingových strojů

Je třeba nějak definovat, co má stát, když se hlava nachází na nejlevějším políčku pásky a má se posunout doleva.

Dvě nejběžnější možnosti:

- Nastane „chybový“ stav, kdy se výpočet (neúspěšně) ukončí:

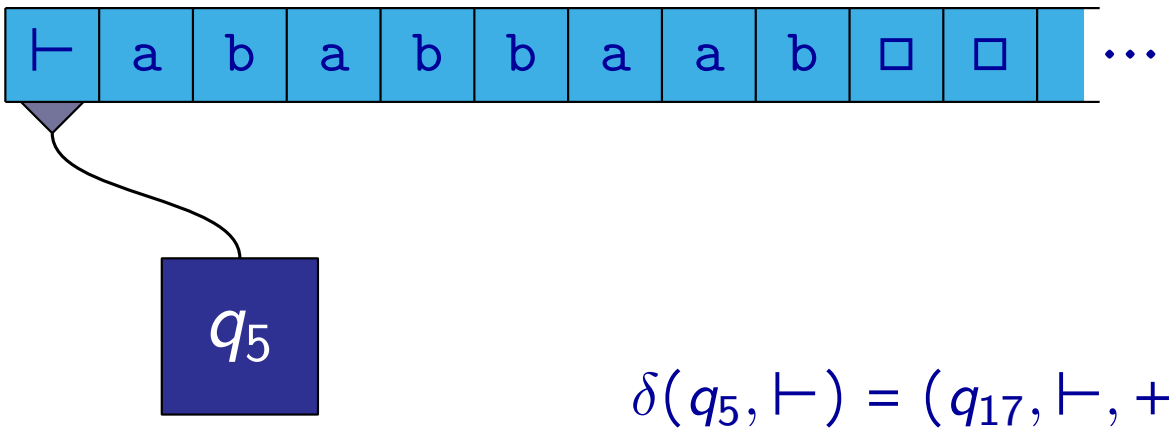


$$\delta(q_5, \mathtt{a}) = (q_{13}, \mathtt{b}, -1)$$

Varianty Turingových strojů

- Na levém konci pásky je „zarážka“ reprezentovaná speciálním symbolem $\vdash \in (\Gamma - \Sigma)$.

  Tuto zarážku není možné přepsat a není na ní možný pohyb směrem doleva, tj. pro každé $q \in Q$ platí, že pokud $\delta(q, \vdash) = (q', b, d)$, tak $b = \vdash$ a $d \in \{0, +1\}$.



$$\delta(q_5, \vdash) = (q_{17}, \vdash, +1)$$

Varianty Turingových strojů

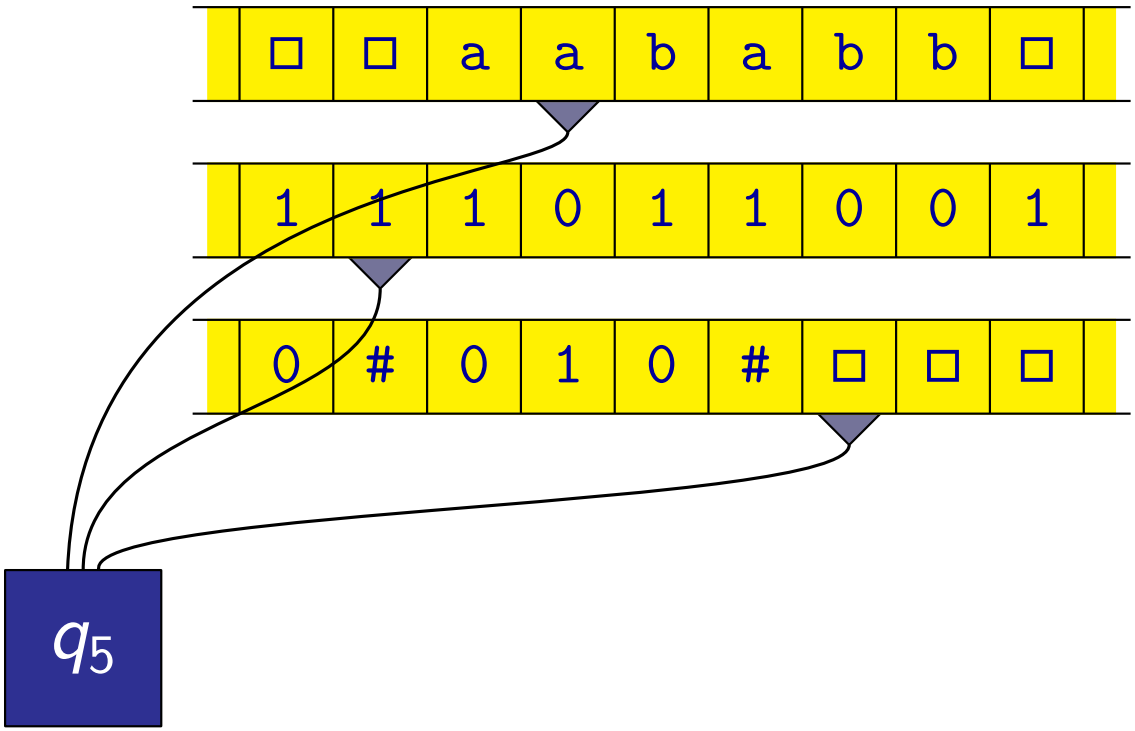
Poznámka: S možností, že výpočet může skončit neúspěšně, protože nastane nějaká chyba, kdy z dané konfigurace není možné pokračovat, ale přitom to není koncová konfigurace, se setkáme i u řady dalších strojů, kterými se budeme zabývat.

Obecně mohou při výpočtu libovolného stroje nastat následující případy:

- Výpočet skončí úspěšně v koncové konfiguraci, která odpovídá korektnímu zastavení.
- Výpočet skončí neúspěšně v konfiguraci, která není koncová, ale není v ní možné pokračovat ve výpočtu — toto chápeme tak, že výpočet skončil chybou.
- Výpočet se nikdy nezastaví.

Varianty Turingových strojů

Často se také uvažují **vícepáskové Turingovy stroje**.



Varianty Turingových strojů

V případě vícepáskového stroje:

- Každá z k pásek má svou vlastní páskovou abecedu, tj. máme páskové abecedy $\Gamma_1, \Gamma_2, \ldots, \Gamma_k$.
- Přechodová funkce δ je typu

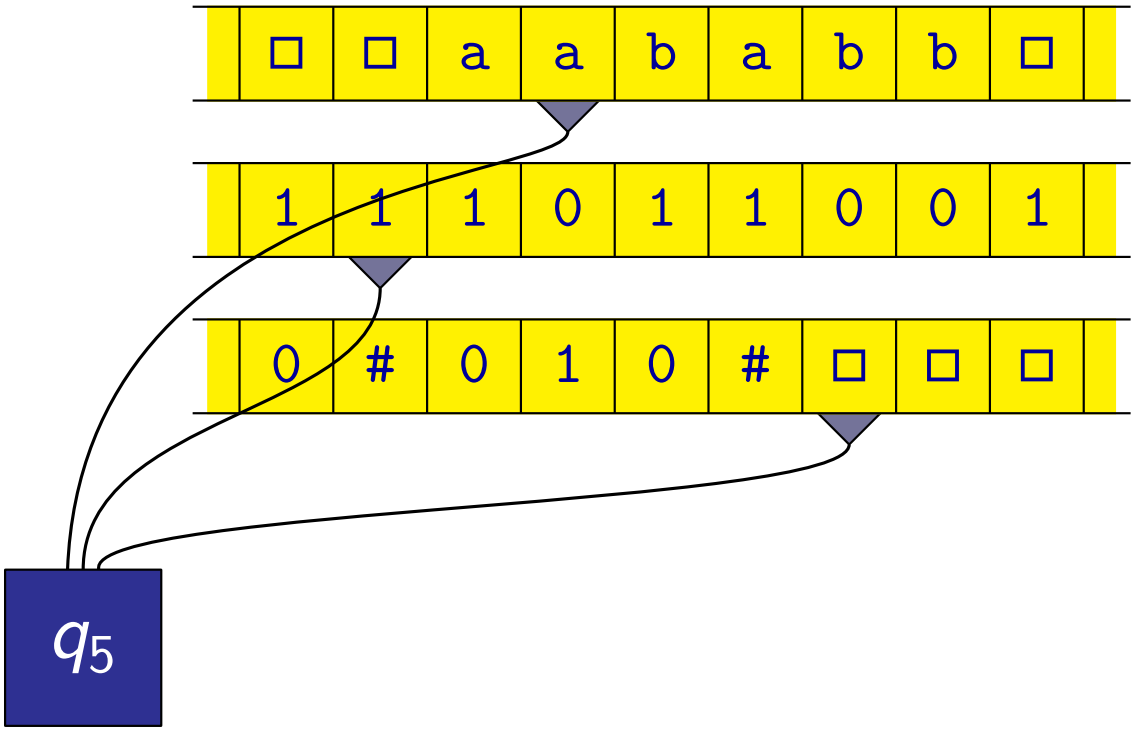
$$(Q - F) \times \Gamma_1 \times \cdots \times \Gamma_k \rightarrow Q \times \Gamma_1 \times \{-1, 0, +1\} \times \cdots \times \Gamma_k \times \{-1, 0, +1\}$$

Příklad:

$$\delta(q_5, \mathtt{a}, 1, \square) = (q_{12}, \mathtt{a}, -1, \mathtt{x}, 0, 1, +1)$$

Varianty Turingových strojů

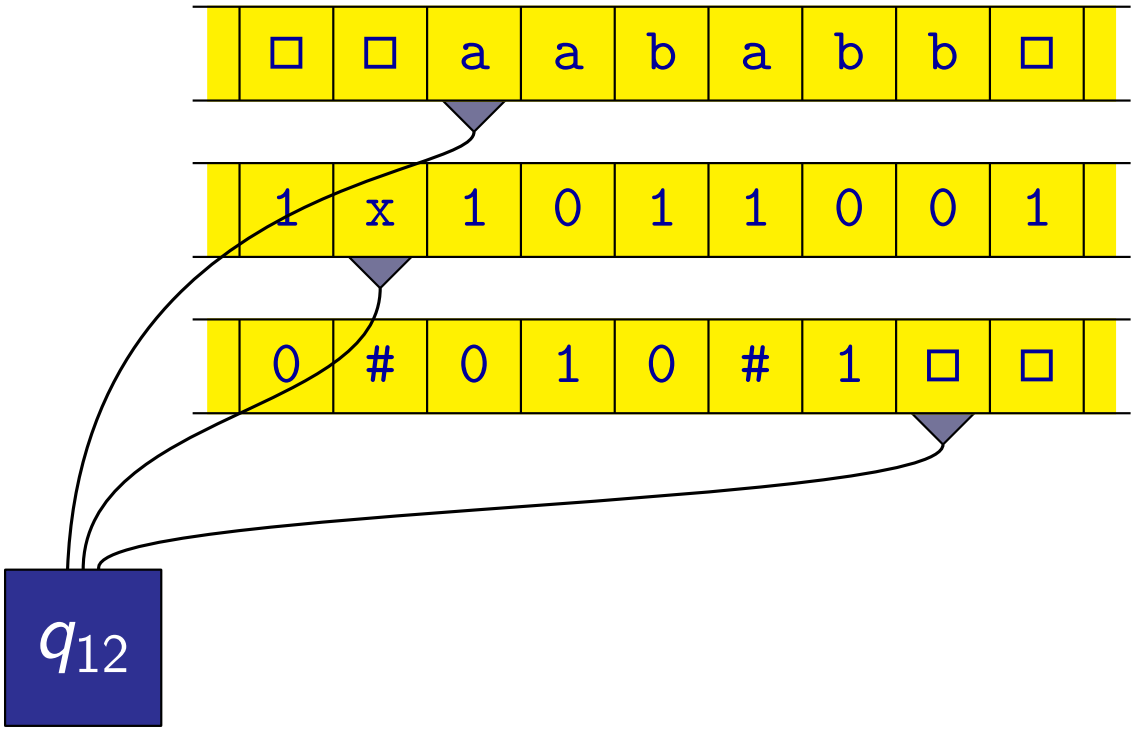
Příklad:



$$\delta(q_5, \mathtt{a}, 1, \square) = (q_{12}, \mathtt{a}, -1, \mathtt{x}, 0, 1, +1)$$

Varianty Turingových strojů

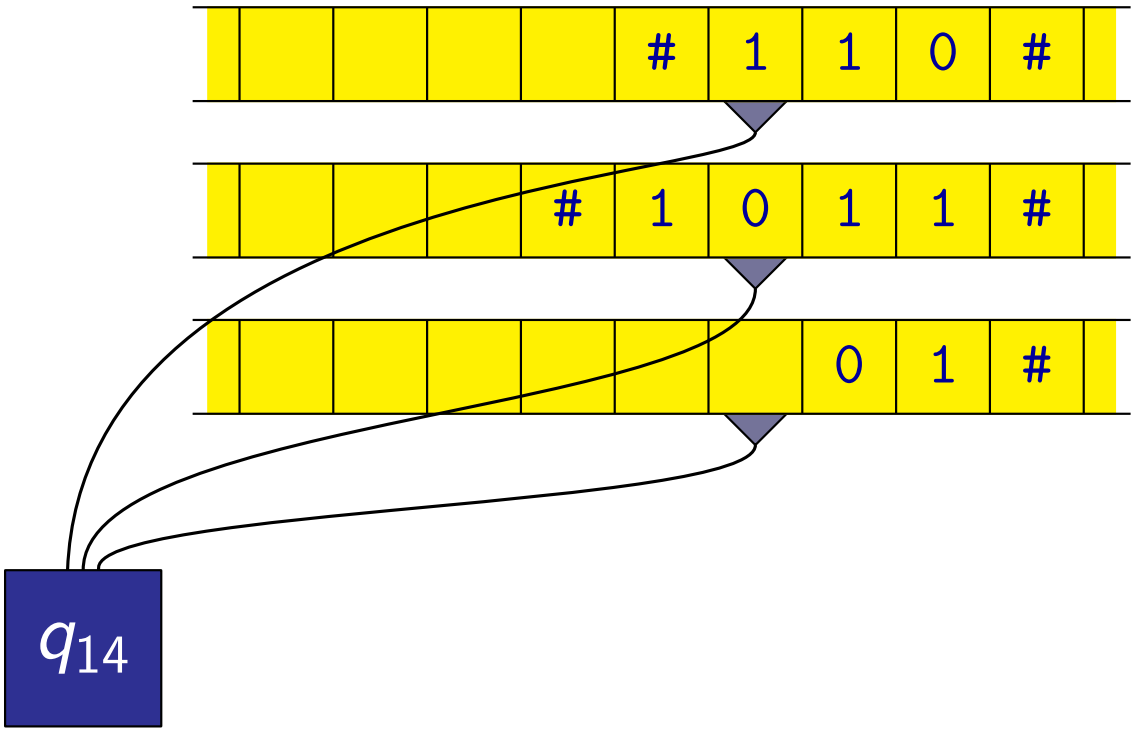
Příklad:



$$\delta(q_5, \mathtt{a}, 1, \square) = (q_{12}, \mathtt{a}, -1, \mathtt{x}, 0, 1, +1)$$

Varianty Turingových strojů

Příklad: Stroj provádějící sčítání dvou binárně zapsaných čísel ohraničených znaky # (např. čísla 6 a 11 budou zapsaná jako slova "#110#" a "#1011#").

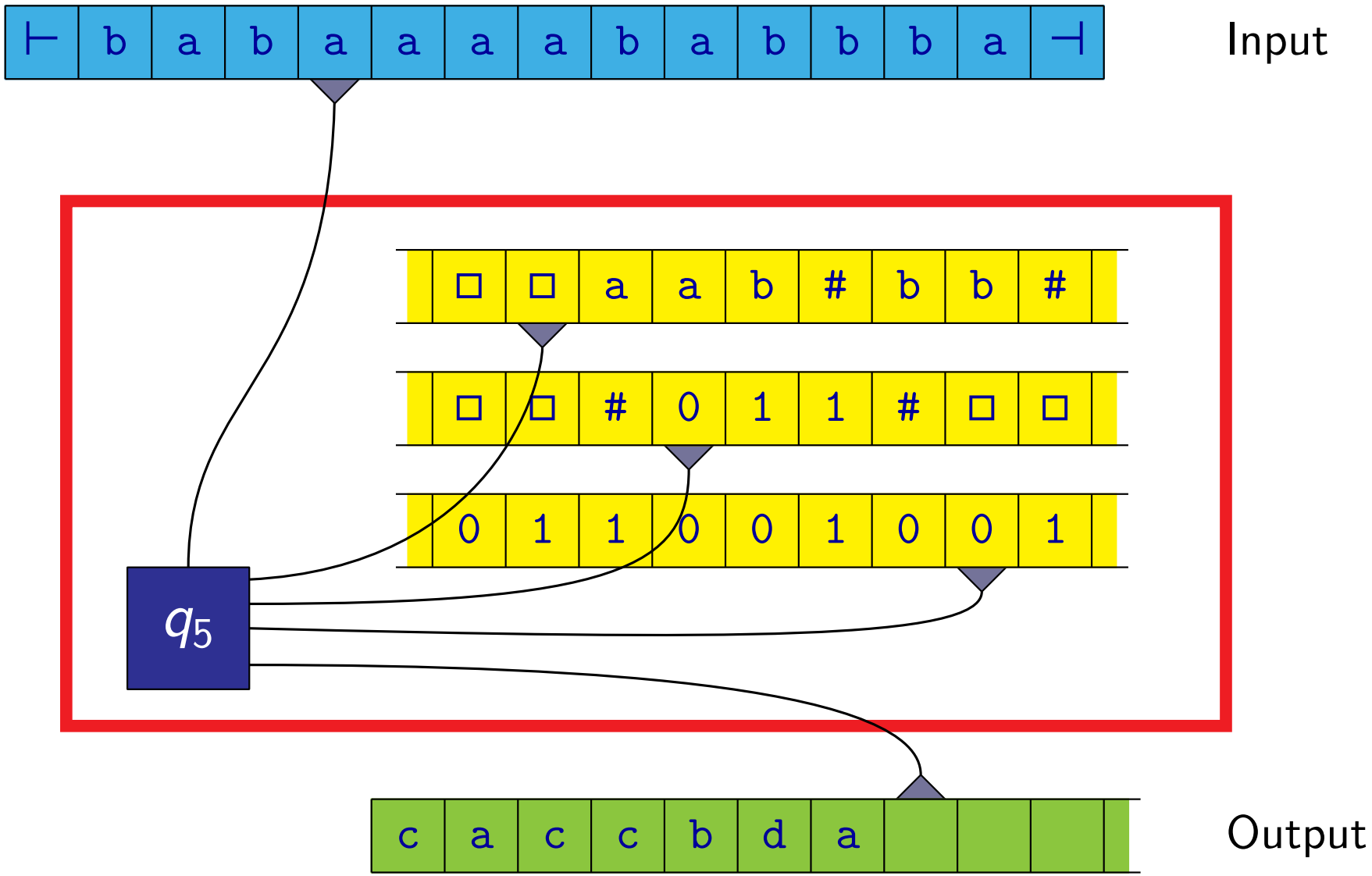


Varianty Turingových strojů

Vícepáskové stroje mají často jednu z pásek vyčleněnu jako vstupní pásku a jednu z pásek jako výstupní pásku. Ostatní pásky pak používají jako pracovní:

- **Vstupní páska** — obsahuje vstupní slovo, není možné na ni zapisovat (je read-only), není nekonečná
- **Pracovní pásky** — je možné z nich číst i na ně zapisovat (jsou typu read/write), na začátku výpočtu jsou prázdné (obsahují pouze symboly $\square$)
- **Výstupní páska** — je na ni možné pouze zapisovat (je write-only), není z ní možné číst, na začátku výpočtu je prázdná, pohyb hlavy je možný jen zleva doprava

Varianty Turingových strojů



Varianty Turingových strojů

Pokud má stroj vyčleněnou speciální vstupní pásku (která je read-only), používají se typicky dvě následující varianty:

- Na této vstupní pásce je možný pohyb hlavy doleva i doprava.

  Vstupní slovo $w \in \Sigma^*$ je v takovém případě ohraničeno zleva a zprava pomocí „zarážek“, tj. speciálních symbolů $\vdash, \dashv \in (\Gamma - \Sigma)$.

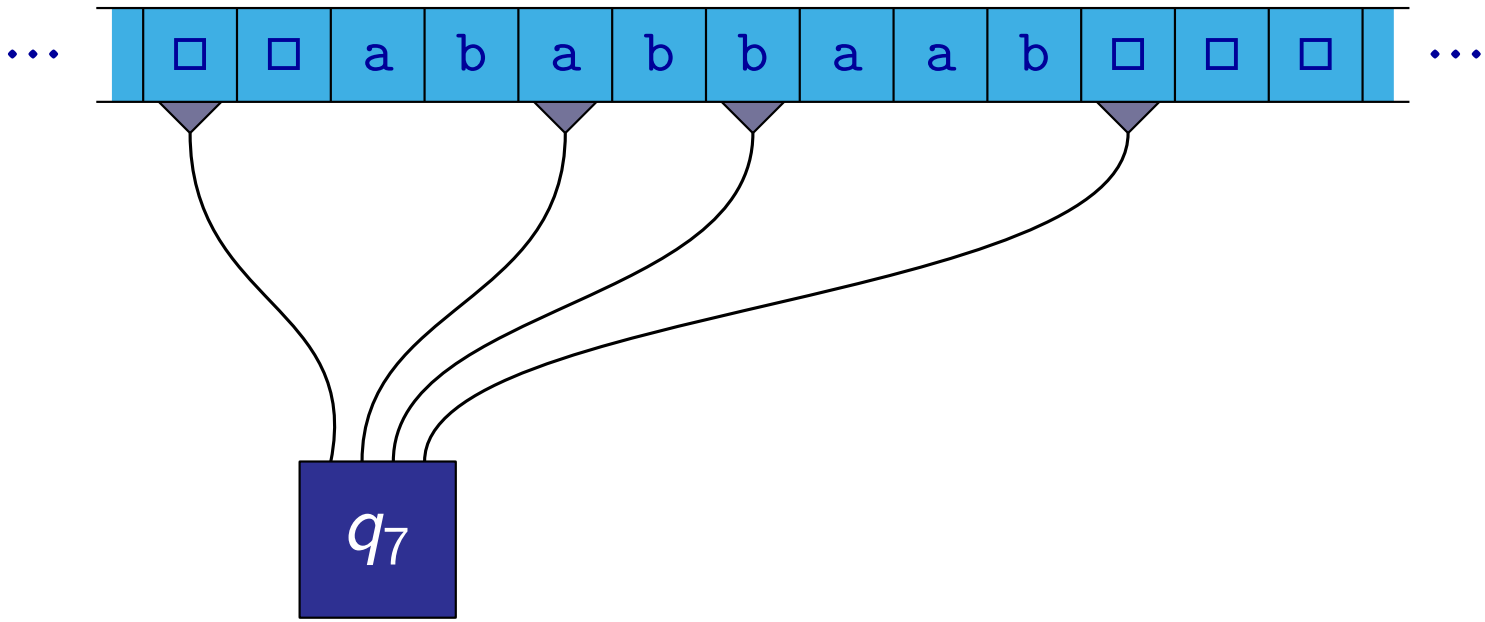
- Na vstupní pásce je možný pohyb hlavy pouze zleva doprava.

Poznámka: Varianta s možným pohybem hlavy na obě strany a se zarážkami je obvyklejší.

Pokud nebude řečeno jinak, budeme uvažovat tuto variantu.

Varianty Turingových strojů

Místo více pásek je možné též uvažovat **více hlav** na jedné pásce:



Varianty Turingových strojů

V případě více hlav na jedné pásce, je třeba specifikovat:

- Zda se může více hlav nacházet současně na jednom políčku pásky.
- A pokud ano, jak je definováno chování daného stroje v případě, že hlavy nacházející se na stejném políčku budou chtít na toto políčko zapsat rozdílné symboly.
- Zda je daný stroj schopen detekovat to, že se dvě nebo více hlav nacházejí současně na témže políčku.

Poznámka: Samozřejmě obecně můžeme uvažovat stroje s více páskami, kde každá z těchto pásek může být vybavena více hlavami.

Varianty Turingových strojů

Uvažujme stroj s více páskami a s libovolným počtem hlav na každé pásce.

Místo toho, aby stroj pracoval v každém kroku zároveň se všemi hlavami, můžeme jeho činnost popisovat jako „program“ skládající se z jednodušších instrukcí následujících typů:

- posunout danou hlavu o jedno políčko doleva
- posunout danou hlavu o jedno políčko doprava
- zapsat na pozici dané hlavy daný specifikovaný symbol
- přečíst z pozice dané hlavy jeden symbol a provést větvení programu (tj. jít do různých stavů řídící jednotky) v závislosti na tom, o jaký symbol se jedná

Varianty Turingových strojů

Zatím jsme uvažovali jen **lineární** (jednorozměrné) pásky.

Místo jednorozměrné pásky může mít paměť s políčky (kde každé políčko obsahuje jeden znak z nějaké dané abecedy) nějakou jinou strukturu.

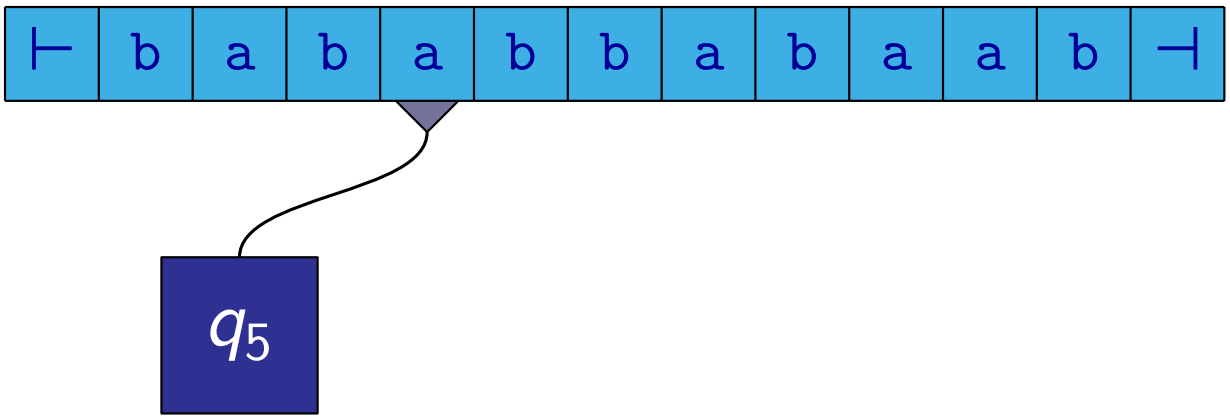
Například:

- dvourozměrná **čtverečková rovina**
  — pohyb hlavy do čtyř směrů: doleva, doprava, nahoru, dolů
- d-rozměrná paměť pro nějaké $d = 3, 4, \ldots$
  (třírozměrná, čtyřrozměrná, atd.)
- paměť organizovaná ve formě (nekonečného) stromu
- ...

Lineárně omezený automat

Lineárně omezený automat (**LBA** — linear bounded automaton):

- Nedeterministický Turingův stroj, který může využívat jen úsek pásky, kde je zapsáno vstupní slovo.
- Políčka pásky, která na začátku obsahují symboly vstupního slova, je možné během výpočtu libovolně přepisovat.
- Levá a pravá zarážka kolem slova. Tyto zarážky nemohou být přepsány.
- Z levé zarážky je možný pohyb jen vpravo, z pravé zarážky jen vlevo.



Lineárně omezený automat

- Lineárně omezené automaty je možné uvažovat v **deterministické** i **nedeterministické** verzi.
- Jako standardní (tj. pokud není uvedeno jinak) se bere nedeterministická verze.
- Otázka, zda je možné jakýkoli jazyk, který je rozpoznáván nedeterministickým LBA, rozpoznávat také deterministickým LBA, je otevřeným problémem.

Poznámka: Z hlediska jazyků, jaké jsou schopné přijímat nebo rozpoznávat, a z hlediska funkcí, jaké jsou schopné počítat, jsou lineárně omezené automaty výrazně slabší než Turingovy stroje, které mají k dispozici neomezeně velkou paměť (ve formě nekonečné pásky).

Chomského hierarchie

Generativní gramatiky

Definice

Generativní gramatika je dána čtveřicí parametrů $\mathcal{G} = (\Pi, \Sigma, S, P)$, kde

- Π je konečná množina neterminálů
- Σ je konečná množina terminálů, $\Pi \cap \Sigma = \varnothing$
- $S \in \Pi$ je počáteční neterminál
- P je konečná množina pravidel typu $\alpha \to \beta$, kde $\alpha \in (\Pi \cup \Sigma)^* \Pi (\Pi \cup \Sigma)^*$ a $\beta \in (\Pi \cup \Sigma)^*$.

Příklad pravidla:

$$CaECb \to bDFbBDaC$$

Poznámka: Tento druh gramatik bývá též označován jako gramatiky **typu 0**, **neomezené** gramatiky či **obecné** gramatiky.

Generativní gramatiky

Předpokládejme, že máme dánu generativní gramatiku $\mathcal{G} = (\Pi, \Sigma, S, P)$.

Relace $\Rightarrow \subseteq (\Pi \cup \Sigma)^* \times (\Pi \cup \Sigma)^*$:

- $\mu_1 \alpha \mu_2 \Rightarrow \mu_1 \beta \mu_2$ pokud $\alpha \to \beta$ je pravidlo v P

Příklad: Jestliže $(BcE \to DDaBb) \in P$, pak

$$CaBCBcEAccABb \Rightarrow CaBCDDaBbAccABb$$

Jazyk $\mathcal{L}(\mathcal{G})$ generovaný gramatikou $\mathcal{G} = (\Pi, \Sigma, S, P)$ je množina všech slov v abecedě Σ, která lze odvodit nějakou derivací z počátečního neterminálu S pomocí pravidel z P, tj.

$$\mathcal{L}(\mathcal{G}) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$

Generativní gramatiky

Předpokládejme, že máme dánu generativní gramatiku $\mathcal{G} = (\Pi, \Sigma, S, P)$.

Relace $\Rightarrow \subseteq (\Pi \cup \Sigma)^* \times (\Pi \cup \Sigma)^*$:

- $\mu_1 \alpha \mu_2 \Rightarrow \mu_1 \beta \mu_2$ pokud $\alpha \to \beta$ je pravidlo v P

Příklad: Jestliže $(BcE \to DDaBb) \in P$, pak

$$CaBC\underline{BcE}AccABb \Rightarrow CaBC\underline{DDaBb}AccABb$$

Jazyk $\mathcal{L}(\mathcal{G})$ generovaný gramatikou $\mathcal{G} = (\Pi, \Sigma, S, P)$ je množina všech slov v abecedě Σ, která lze odvodit nějakou derivací z počátečního neterminálu S pomocí pravidel z P, tj.

$$\mathcal{L}(\mathcal{G}) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova *aaaaabbbbbccccc*:

S

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova *aaaaabbbbbccccc*:

$$S \Rightarrow aSQ$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova *aaaaabbbbbccccc*:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \end{aligned}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova *aaaaabbbbbccccc*:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \end{aligned}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova *aaaaabbbbbccccc*:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \end{aligned}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova *aaaaabbbbbccccc*:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabcQQQQ \end{aligned}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabcQQQQ \\ &\Rightarrow aaaaabQcQQQ \end{aligned}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabcQQQQ \\ &\Rightarrow aaaaabQcQQQ \\ &\Rightarrow aaaaabbccQQQ \end{aligned}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabcQQQQ \\ &\Rightarrow aaaaabQcQQQ \\ &\Rightarrow aaaaabbccQQQ \\ &\Rightarrow aaaaabbcQcQQ \end{aligned}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabcQQQQ \\ &\Rightarrow aaaaabQcQQQ \\ &\Rightarrow aaaaabbccQQQ \\ &\Rightarrow aaaaabbcQcQQ \\ &\Rightarrow aaaaabbcQccQQ \end{aligned}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabcQQQQ \\ &\Rightarrow aaaaabQcQQQ \\ &\Rightarrow aaaaabbccQQQ \\ &\Rightarrow aaaaabbcQcQQ \\ &\Rightarrow aaaaabbQccQQ \\ &\Rightarrow aaaaabbbcccQQ \end{aligned}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabcQQQQ \\ &\Rightarrow aaaaabQcQQQ \\ &\Rightarrow aaaaabbccQQQ \\ &\Rightarrow aaaaabbcQcQQ \\ &\Rightarrow aaaaabbQccQQ \\ &\Rightarrow aaaaabbbcccQQ \\ &\Rightarrow aaaaabbbccQcQ \end{aligned}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabcQQQQ \\ &\Rightarrow aaaaabQcQQQ \\ &\Rightarrow aaaaabbccQQQ \\ &\Rightarrow aaaaabbcQcQQ \\ &\Rightarrow aaaaabbQccQQ \\ &\Rightarrow aaaaabbbcccQQ \\ &\Rightarrow aaaaabbbccQcQ \\ &\Rightarrow aaaaabbbcQccQ \end{aligned}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabcQQQQ \\ &\Rightarrow aaaaabQcQQQ \\ &\Rightarrow aaaaabbccQQQ \\ &\Rightarrow aaaaabbcQcQQ \\ &\Rightarrow aaaaabbQccQQ \\ &\Rightarrow aaaaabbbcccQQ \\ &\Rightarrow aaaaabbbccQcQ \\ &\Rightarrow aaaaabbbcQccQ \\ &\Rightarrow aaaaabbbQcccQ \end{aligned}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabcQQQQ \\ &\Rightarrow aaaaabQcQQQ \\ &\Rightarrow aaaaabbccQQQ \\ &\Rightarrow aaaaabbcQcQQ \\ &\Rightarrow aaaaabbQccQQ \\ &\Rightarrow aaaaabbbcccQQ \\ &\Rightarrow aaaaabbbccQcQ \\ &\Rightarrow aaaaabbbcQccQ \\ &\Rightarrow aaaaabbbQcccQ \\ &\Rightarrow aaaaabbbbccccQ \end{aligned}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabcQQQQ \\ &\Rightarrow aaaaabQcQQQ \\ &\Rightarrow aaaaabbccQQQ \\ &\Rightarrow aaaaabbcQcQQ \\ &\Rightarrow aaaaabbQccQQ \\ &\Rightarrow aaaaabbbcccQQ \\ &\Rightarrow aaaaabbbccQcQ \\ &\Rightarrow aaaaabbbcQccQ \\ &\Rightarrow aaaaabbbQcccQ \\ &\Rightarrow aaaaabbbbccccQ \\ &\Rightarrow aaaaabbbbcccQc \end{aligned}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabcQQQQ \\ &\Rightarrow aaaaabQcQQQ \\ &\Rightarrow aaaaabbccQQQ \\ &\Rightarrow aaaaabbcQcQQ \\ &\Rightarrow aaaaabbcQccQQ \\ &\Rightarrow aaaaabbcccQQ \\ &\Rightarrow aaaaabbbccQcQ \\ &\Rightarrow aaaaabbbcQccQ \\ &\Rightarrow aaaaabbbQcccQ \\ &\Rightarrow aaaaabbbbccccQ \\ &\Rightarrow aaaaabbbbcccQc \\ &\Rightarrow aaaaabbbbccQcc \end{aligned}$$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$\Rightarrow aaaaabbbbcQccc$

$\Rightarrow aaaaabbQccQQ$
$\Rightarrow aaaaabbbcccQQ$
$\Rightarrow aaaaabbbccQcQ$
$\Rightarrow aaaaabbbcQccQ$
$\Rightarrow aaaaabbbQcccQ$
$\Rightarrow aaaaabbbbccccQ$
$\Rightarrow aaaaabbbbcccQc$
$\Rightarrow aaaaabbbbccQcc$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$\Rightarrow aaaaabbbbcQccc$
$\Rightarrow aaaaabbbbQcccc$
$\Rightarrow aaaaabbQccQQ$
$\Rightarrow aaaaabbbcccQQ$
$\Rightarrow aaaaabbbccQcQ$
$\Rightarrow aaaaabbbcQccQ$
$\Rightarrow aaaaabbbQcccQ$
$\Rightarrow aaaaabbbbccccQ$
$\Rightarrow aaaaabbbbcccQc$
$\Rightarrow aaaaabbbbccQcc$

Generativní gramatiky

Příklad: Gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abc \\ cQ &\to Qc \\ bQc &\to bbcc \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} &\Rightarrow aaaaabbbbcQccc \\ &\Rightarrow aaaaabbbbQcccc \\ &\Rightarrow aaaaabbbbbccccc \end{aligned}$$

$$\begin{aligned} &\Rightarrow aaaaabbQccQQ \\ &\Rightarrow aaaaabbbcccQQ \\ &\Rightarrow aaaaabbbccQcQ \\ &\Rightarrow aaaaabbbcQccQ \\ &\Rightarrow aaaaabbbQcccQ \\ &\Rightarrow aaaaabbbbccccQ \\ &\Rightarrow aaaaabbbbcccQc \\ &\Rightarrow aaaaabbbbccQcc \end{aligned}$$

Kontextové gramatiky

Kontextové gramatiky, označované též jako gramatiky **typu 1**, jsou speciálním případem generativních gramatik.

Gramatika $\mathcal{G} = (\Pi, \Sigma, S, P)$ se nazývá **kontextová**, jestliže všechna její pravidla (s jednou níže uvedenou výjimkou) jsou tvaru

$$\alpha X \beta \rightarrow \alpha \gamma \beta$$

kde $X \in \Pi$, $\alpha, \beta, \gamma \in (\Pi \cup \Sigma)^*$, přičemž $|\gamma| \geq 1$.

Jedinou výjimkou je, že gramatika $\mathcal{G}$ může obsahovat pravidlo $S \rightarrow \varepsilon$.

Pokud toto pravidlo obsahuje, nesmí se počáteční nerminál S vyskytovat na pravé straně žádného pravidla.

Příklad pravidla:

$$BaEC \rightarrow BaDAcBC$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

S

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$S \Rightarrow aSQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \end{aligned}$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \end{aligned}$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \end{aligned}$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabCQQQQ \end{aligned}$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabCQQQQ \\ &\Rightarrow aaaaabXQQQQ \end{aligned}$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabCQQQQ \\ &\Rightarrow aaaaabXQQQQ \\ &\Rightarrow aaaaabXYQQQ \end{aligned}$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabCQQQQ \\ &\Rightarrow aaaaabXQQQQ \\ &\Rightarrow aaaaabXYQQQ \\ &\Rightarrow aaaaabQYQQQ \end{aligned}$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabCQQQQ \\ &\Rightarrow aaaaabXQQQQ \\ &\Rightarrow aaaaabXYQQQ \\ &\Rightarrow aaaaabQYQQQ \\ &\Rightarrow aaaaabQCQQQ \end{aligned}$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabCQQQQ \\ &\Rightarrow aaaaabXQQQQ \\ &\Rightarrow aaaaabXYQQQ \\ &\Rightarrow aaaaabQYQQQ \\ &\Rightarrow aaaaabQCQQQ \\ &\Rightarrow aaaaabbCCQQQ \end{aligned}$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabCQQQQ \\ &\Rightarrow aaaaabXQQQQ \\ &\Rightarrow aaaaabXYQQQ \\ &\Rightarrow aaaaabQYQQQ \\ &\Rightarrow aaaaabQCQQQ \\ &\Rightarrow aaaaabbCCQQQ \\ &\Rightarrow aaaaabbCXQQQ \end{aligned}$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabCQQQQ \\ &\Rightarrow aaaaabXQQQQ \\ &\Rightarrow aaaaabXYQQQ \\ &\Rightarrow aaaaabQYQQQ \\ &\Rightarrow aaaaabQCQQQ \\ &\Rightarrow aaaaabbCCQQQ \\ &\Rightarrow aaaaabbCXQQQ \\ &\Rightarrow aaaaabbCXYQQ \end{aligned}$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabCQQQQ \\ &\Rightarrow aaaaabXQQQQ \\ &\Rightarrow aaaaabXYQQQ \\ &\Rightarrow aaaaabQYQQQ \\ &\Rightarrow aaaaabQCQQQ \\ &\Rightarrow aaaaabbCCQQQ \\ &\Rightarrow aaaaabbCXQQQ \\ &\Rightarrow aaaaabbCXYQQ \\ &\Rightarrow aaaaabbCQYQQ \end{aligned}$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabCQQQQ \\ &\Rightarrow aaaaabXQQQQ \\ &\Rightarrow aaaaabXYQQQ \\ &\Rightarrow aaaaabQYQQQ \\ &\Rightarrow aaaaabQCQQQ \\ &\Rightarrow aaaaabbCCQQQ \\ &\Rightarrow aaaaabbCXQQQ \\ &\Rightarrow aaaaabbCXYQQ \\ &\Rightarrow aaaaabbCQYQQ \\ &\Rightarrow aaaaabbCQCQQ \end{aligned}$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabCQQQQ \\ &\Rightarrow aaaaabXQQQQ \\ &\Rightarrow aaaaabXYQQQ \\ &\Rightarrow aaaaabQYQQQ \\ &\Rightarrow aaaaabQCQQQ \\ &\Rightarrow aaaaabbCCQQQ \\ &\Rightarrow aaaaabbCXQQQ \\ &\Rightarrow aaaaabbCXYQQ \\ &\Rightarrow aaaaabbCQYQQ \\ &\Rightarrow aaaaabbCQCQQ \\ &\Rightarrow aaaaabbXQCQQ \end{aligned}$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$\begin{aligned} S &\to aSQ \\ S &\to abC \\ bQC &\to bbCC \\ C &\to c \end{aligned} \qquad \begin{aligned} CQ &\to XQ \\ XQ &\to XY \\ XY &\to QY \\ QY &\to QC \end{aligned}$$

Derivace slova $aaaaabbbbbccccc$:

$$\begin{aligned} S &\Rightarrow aSQ \\ &\Rightarrow aaSQQ \\ &\Rightarrow aaaSQQQ \\ &\Rightarrow aaaaSQQQQ \\ &\Rightarrow aaaaabCQQQQ \\ &\Rightarrow aaaaabXQQQQ \\ &\Rightarrow aaaaabXYQQQ \\ &\Rightarrow aaaaabQYQQQ \\ &\Rightarrow aaaaabQCQQQ \\ &\Rightarrow aaaaabbCCQQQ \\ &\Rightarrow aaaaabbCXQQQ \\ &\Rightarrow aaaaabbCXYQQ \\ &\Rightarrow aaaaabbCQYQQ \\ &\Rightarrow aaaaabbCQCQQ \\ &\Rightarrow aaaaabbXQCQQ \\ &\Rightarrow aaaaabbXYCQQ \end{aligned}$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbQYCQQ$$
$$\Rightarrow aaaaabQCQQQ$$
$$\Rightarrow aaaaabbCCQQQ$$
$$\Rightarrow aaaaabbCXQQQ$$
$$\Rightarrow aaaaabbCXYQQ$$
$$\Rightarrow aaaaabbCQYQQ$$
$$\Rightarrow aaaaabbCQCQQ$$
$$\Rightarrow aaaaabbXQCQQ$$
$$\Rightarrow aaaaabbXYCQQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbQYCQQ$$
$$\Rightarrow aaaaabbQCCQQ$$
$$\Rightarrow aaaaabQCQQQ$$
$$\Rightarrow aaaaabbCCQQQ$$
$$\Rightarrow aaaaabbCXQQQ$$
$$\Rightarrow aaaaabbCXYQQ$$
$$\Rightarrow aaaaabbCQYQQ$$
$$\Rightarrow aaaaabbCQCQQ$$
$$\Rightarrow aaaaabbXQCQQ$$
$$\Rightarrow aaaaabbXYCQQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$\Rightarrow aaaaabbQYCQQ$
$\Rightarrow aaaaabbQCCQQ$
$\Rightarrow aaaaabbbCCCQQ$
$\Rightarrow aaaaabQCQQQ$
$\Rightarrow aaaaabbCCQQQ$
$\Rightarrow aaaaabbCXQQQ$
$\Rightarrow aaaaabbCXYQQ$
$\Rightarrow aaaaabbCQYQQ$
$\Rightarrow aaaaabbCQCQQ$
$\Rightarrow aaaaabbXQCQQ$
$\Rightarrow aaaaabbXYCQQ$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbQYCQQ$$
$$\Rightarrow aaaaabbQCCQQ$$
$$\Rightarrow aaaaabbbCCCQQ$$
$$\Rightarrow aaaaabbbCCXQQ$$
$$\Rightarrow aaaaabQCQQQ$$
$$\Rightarrow aaaaabbCCQQQ$$
$$\Rightarrow aaaaabbCXQQQ$$
$$\Rightarrow aaaaabbCXYQQ$$
$$\Rightarrow aaaaabbCQYQQ$$
$$\Rightarrow aaaaabbCQCQQ$$
$$\Rightarrow aaaaabbXQCQQ$$
$$\Rightarrow aaaaabbXYCQQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbQYCQQ$$
$$\Rightarrow aaaaabbQCCQQ$$
$$\Rightarrow aaaaabbbCCCQQ$$
$$\Rightarrow aaaaabbbCCXQQ$$
$$\Rightarrow aaaaabbbCCXYQ$$
$$\Rightarrow aaaaabQCQQQ$$
$$\Rightarrow aaaaabbCCQQQ$$
$$\Rightarrow aaaaabbCXQQQ$$
$$\Rightarrow aaaaabbCXYQQ$$
$$\Rightarrow aaaaabbCQYQQ$$
$$\Rightarrow aaaaabbCQCQQ$$
$$\Rightarrow aaaaabbXQCQQ$$
$$\Rightarrow aaaaabbXYCQQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbQYCQQ$$
$$\Rightarrow aaaaabbQCCQQ$$
$$\Rightarrow aaaaabbbCCCQQ$$
$$\Rightarrow aaaaabbbCCXQQ$$
$$\Rightarrow aaaaabbbCCXYQ$$
$$\Rightarrow aaaaabbbCCQYQ$$
$$\Rightarrow aaaaabQCQQQ$$
$$\Rightarrow aaaaabbCCQQQ$$
$$\Rightarrow aaaaabbCXQQQ$$
$$\Rightarrow aaaaabbCXYQQ$$
$$\Rightarrow aaaaabbCQYQQ$$
$$\Rightarrow aaaaabbCQCQQ$$
$$\Rightarrow aaaaabbXQCQQ$$
$$\Rightarrow aaaaabbXYCQQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbQYCQQ$$
$$\Rightarrow aaaaabbQCCQQ$$
$$\Rightarrow aaaaabbbCCCQQ$$
$$\Rightarrow aaaaabbbCCXQQ$$
$$\Rightarrow aaaaabbbCCXYQ$$
$$\Rightarrow aaaaabbbCCQYQ$$
$$\Rightarrow aaaaabbbCCQCQ$$
$$\Rightarrow aaaaabQCQQQ$$
$$\Rightarrow aaaaabbCCQQQ$$
$$\Rightarrow aaaaabbCXQQQ$$
$$\Rightarrow aaaaabbCXYQQ$$
$$\Rightarrow aaaaabbCQYQQ$$
$$\Rightarrow aaaaabbCQCQQ$$
$$\Rightarrow aaaaabbXQCQQ$$
$$\Rightarrow aaaaabbXYCQQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbQYCQQ$$
$$\Rightarrow aaaaabbQCCQQ$$
$$\Rightarrow aaaaabbbCCCQQ$$
$$\Rightarrow aaaaabbbCCXQQ$$
$$\Rightarrow aaaaabbbCCXYQ$$
$$\Rightarrow aaaaabbbCCQYQ$$
$$\Rightarrow aaaaabbbCCQCQ$$
$$\Rightarrow aaaaabbbCXQCQ$$
$$\Rightarrow aaaaabQCQQQ$$
$$\Rightarrow aaaaabbCCQQQ$$
$$\Rightarrow aaaaabbCXQQQ$$
$$\Rightarrow aaaaabbCXYQQ$$
$$\Rightarrow aaaaabbCQYQQ$$
$$\Rightarrow aaaaabbCQCQQ$$
$$\Rightarrow aaaaabbXQCQQ$$
$$\Rightarrow aaaaabbXYCQQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$\Rightarrow aaaaabbQYCQQ$
$\Rightarrow aaaaabbQCCQQ$
$\Rightarrow aaaaabbbCCCQQ$
$\Rightarrow aaaaabbbCCXQQ$
$\Rightarrow aaaaabbbCCXYQ$
$\Rightarrow aaaaabbbCCQYQ$
$\Rightarrow aaaaabbbCCQCQ$
$\Rightarrow aaaaabbbCXQCQ$
$\Rightarrow aaaaabbbCXYCQ$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$\Rightarrow aaaaabbQYCQQ$
$\Rightarrow aaaaabbQCCQQ$
$\Rightarrow aaaaabbbCCCQQ$
$\Rightarrow aaaaabbbCCXQQ$
$\Rightarrow aaaaabbbCCXYQ$
$\Rightarrow aaaaabbbCCQYQ$
$\Rightarrow aaaaabbbCCQCQ$
$\Rightarrow aaaaabbbCXQCQ$
$\Rightarrow aaaaabbbCXYCQ$
$\Rightarrow aaaaabbbCQYCQ$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$\Rightarrow aaaaabbQYCQQ$
$\Rightarrow aaaaabbQCCQQ$
$\Rightarrow aaaaabbbCCCQQ$
$\Rightarrow aaaaabbbCCXQQ$
$\Rightarrow aaaaabbbCCXYQ$
$\Rightarrow aaaaabbbCCQYQ$
$\Rightarrow aaaaabbbCCQCQ$
$\Rightarrow aaaaabbbCXQCQ$
$\Rightarrow aaaaabbbCXYCQ$
$\Rightarrow aaaaabbbCQYCQ$
$\Rightarrow aaaaabbbCQCCQ$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$\Rightarrow aaaaabbQYCQQ$
$\Rightarrow aaaaabbQCCQQ$
$\Rightarrow aaaaabbbCCCQQ$
$\Rightarrow aaaaabbbCCXQQ$
$\Rightarrow aaaaabbbCCXYQ$
$\Rightarrow aaaaabbbCCQYQ$
$\Rightarrow aaaaabbbCCQCQ$
$\Rightarrow aaaaabbbCXQCQ$
$\Rightarrow aaaaabbbCXYCQ$
$\Rightarrow aaaaabbbCQYCQ$
$\Rightarrow aaaaabbbCQCCQ$
$\Rightarrow aaaaabbbXQCCQ$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbQYCQQ$$
$$\Rightarrow aaaaabbQCCQQ$$
$$\Rightarrow aaaaabbbCCCQQ$$
$$\Rightarrow aaaaabbbCCXQQ$$
$$\Rightarrow aaaaabbbCCXYQ$$
$$\Rightarrow aaaaabbbCCQYQ$$
$$\Rightarrow aaaaabbbCCQCQ$$
$$\Rightarrow aaaaabbbCXQCQ$$
$$\Rightarrow aaaaabbbCXYCQ$$
$$\Rightarrow aaaaabbbCQYCQ$$
$$\Rightarrow aaaaabbbCQCCQ$$
$$\Rightarrow aaaaabbbXQCCQ$$
$$\Rightarrow aaaaabbbXYCCQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbQYCQQ$$
$$\Rightarrow aaaaabbQCCQQ$$
$$\Rightarrow aaaaabbbCCCQQ$$
$$\Rightarrow aaaaabbbCCXQQ$$
$$\Rightarrow aaaaabbbCCXYQ$$
$$\Rightarrow aaaaabbbCCQYQ$$
$$\Rightarrow aaaaabbbCCQCQ$$
$$\Rightarrow aaaaabbbCXQCQ$$
$$\Rightarrow aaaaabbbCXYCQ$$
$$\Rightarrow aaaaabbbCQYCQ$$
$$\Rightarrow aaaaabbbCQCCQ$$
$$\Rightarrow aaaaabbbXQCCQ$$
$$\Rightarrow aaaaabbbXYCCQ$$
$$\Rightarrow aaaaabbbQYCCQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbQYCQQ$$
$$\Rightarrow aaaaabbQCCQQ$$
$$\Rightarrow aaaaabbbCCCQQ$$
$$\Rightarrow aaaaabbbCCXQQ$$
$$\Rightarrow aaaaabbbCCXYQ$$
$$\Rightarrow aaaaabbbCCQYQ$$
$$\Rightarrow aaaaabbbCCQCQ$$
$$\Rightarrow aaaaabbbCXQCQ$$
$$\Rightarrow aaaaabbbCXYCQ$$
$$\Rightarrow aaaaabbbCQYCQ$$
$$\Rightarrow aaaaabbbCQCCQ$$
$$\Rightarrow aaaaabbbXQCCQ$$
$$\Rightarrow aaaaabbbXYCCQ$$
$$\Rightarrow aaaaabbbQYCCQ$$
$$\Rightarrow aaaaabbbQCCCQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ \qquad CQ \to XQ$$
$$S \to abC \qquad XQ \to XY$$
$$bQC \to bbCC \qquad XY \to QY$$
$$C \to c \qquad QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbQYCQQ \qquad \Rightarrow aaaaabbbCXYCQ$$
$$\Rightarrow aaaaabbQCCQQ \qquad \Rightarrow aaaaabbbCQYCQ$$
$$\Rightarrow aaaaabbbCCCQQ \qquad \Rightarrow aaaaabbbCQCCQ$$
$$\Rightarrow aaaaabbbCCXQQ \qquad \Rightarrow aaaaabbbXQCCQ$$
$$\Rightarrow aaaaabbbCCXYQ \qquad \Rightarrow aaaaabbbXYCCQ$$
$$\Rightarrow aaaaabbbCCQYQ \qquad \Rightarrow aaaaabbbQYCCQ$$
$$\Rightarrow aaaaabbbCCQCQ \qquad \Rightarrow aaaaabbbQCCCQ$$
$$\Rightarrow aaaaabbbCXQCQ \qquad \Rightarrow aaaaabbbbCCCCQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbCCCXQ$$
$$\Rightarrow aaaaabbbCXYCQ$$
$$\Rightarrow aaaaabbbCQYCQ$$
$$\Rightarrow aaaaabbbCQCCQ$$
$$\Rightarrow aaaaabbbXQCCQ$$
$$\Rightarrow aaaaabbbXYCCQ$$
$$\Rightarrow aaaaabbbQYCCQ$$
$$\Rightarrow aaaaabbbQCCCQ$$
$$\Rightarrow aaaaabbbbCCCCQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbCCCXQ$$
$$\Rightarrow aaaaabbbbCCCXY$$
$$\Rightarrow aaaaabbbCXYCQ$$
$$\Rightarrow aaaaabbbCQYCQ$$
$$\Rightarrow aaaaabbbCQCCQ$$
$$\Rightarrow aaaaabbbXQCCQ$$
$$\Rightarrow aaaaabbbXYCCQ$$
$$\Rightarrow aaaaabbbQYCCQ$$
$$\Rightarrow aaaaabbbQCCCQ$$
$$\Rightarrow aaaaabbbbCCCCQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbCCCXQ$$
$$\Rightarrow aaaaabbbbCCCXY$$
$$\Rightarrow aaaaabbbbCCCQY$$
$$\Rightarrow aaaaabbbCXYCQ$$
$$\Rightarrow aaaaabbbCQYCQ$$
$$\Rightarrow aaaaabbbCQCCQ$$
$$\Rightarrow aaaaabbbXQCCQ$$
$$\Rightarrow aaaaabbbXYCCQ$$
$$\Rightarrow aaaaabbbQYCCQ$$
$$\Rightarrow aaaaabbbQCCCQ$$
$$\Rightarrow aaaaabbbbCCCCQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbCCCXQ$$
$$\Rightarrow aaaaabbbbCCCXY$$
$$\Rightarrow aaaaabbbbCCCQY$$
$$\Rightarrow aaaaabbbbCCCQC$$
$$\Rightarrow aaaaabbbCXYCQ$$
$$\Rightarrow aaaaabbbCQYCQ$$
$$\Rightarrow aaaaabbbCQCCQ$$
$$\Rightarrow aaaaabbbXQCCQ$$
$$\Rightarrow aaaaabbbXYCCQ$$
$$\Rightarrow aaaaabbbQYCCQ$$
$$\Rightarrow aaaaabbbQCCCQ$$
$$\Rightarrow aaaaabbbbCCCCQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbCCCXQ$$
$$\Rightarrow aaaaabbbbCCCXY$$
$$\Rightarrow aaaaabbbbCCCQY$$
$$\Rightarrow aaaaabbbbCCCQC$$
$$\Rightarrow aaaaabbbbCCXQC$$
$$\Rightarrow aaaaabbbCXYCQ$$
$$\Rightarrow aaaaabbbCQYCQ$$
$$\Rightarrow aaaaabbbCQCCQ$$
$$\Rightarrow aaaaabbbXQCCQ$$
$$\Rightarrow aaaaabbbXYCCQ$$
$$\Rightarrow aaaaabbbQYCCQ$$
$$\Rightarrow aaaaabbbQCCCQ$$
$$\Rightarrow aaaaabbbbCCCCQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbCCCXQ$$
$$\Rightarrow aaaaabbbbCCCXY$$
$$\Rightarrow aaaaabbbbCCCQY$$
$$\Rightarrow aaaaabbbbCCCQC$$
$$\Rightarrow aaaaabbbbCCXQC$$
$$\Rightarrow aaaaabbbbCCXYC$$
$$\Rightarrow aaaaabbbCXYCQ$$
$$\Rightarrow aaaaabbbCQYCQ$$
$$\Rightarrow aaaaabbbCQCCQ$$
$$\Rightarrow aaaaabbbXQCCQ$$
$$\Rightarrow aaaaabbbXYCCQ$$
$$\Rightarrow aaaaabbbQYCCQ$$
$$\Rightarrow aaaaabbbQCCCQ$$
$$\Rightarrow aaaaabbbbCCCCQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbCCCXQ$$
$$\Rightarrow aaaaabbbbCCCXY$$
$$\Rightarrow aaaaabbbbCCCQY$$
$$\Rightarrow aaaaabbbbCCCQC$$
$$\Rightarrow aaaaabbbbCCXQC$$
$$\Rightarrow aaaaabbbbCCXYC$$
$$\Rightarrow aaaaabbbbCCQYC$$
$$\Rightarrow aaaaabbbCXYCQ$$
$$\Rightarrow aaaaabbbCQYCQ$$
$$\Rightarrow aaaaabbbCQCCQ$$
$$\Rightarrow aaaaabbbXQCCQ$$
$$\Rightarrow aaaaabbbXYCCQ$$
$$\Rightarrow aaaaabbbQYCCQ$$
$$\Rightarrow aaaaabbbQCCCQ$$
$$\Rightarrow aaaaabbbbCCCCQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ \qquad CQ \to XQ$$
$$S \to abC \qquad XQ \to XY$$
$$bQC \to bbCC \qquad XY \to QY$$
$$C \to c \qquad QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbCCCXQ \qquad \Rightarrow aaaaabbbCXYCQ$$
$$\Rightarrow aaaaabbbbCCCXY \qquad \Rightarrow aaaaabbbCQYCQ$$
$$\Rightarrow aaaaabbbbCCCQY \qquad \Rightarrow aaaaabbbCQCCQ$$
$$\Rightarrow aaaaabbbbCCCQC \qquad \Rightarrow aaaaabbbXQCCQ$$
$$\Rightarrow aaaaabbbbCCXQC \qquad \Rightarrow aaaaabbbXYCCQ$$
$$\Rightarrow aaaaabbbbCCXYC \qquad \Rightarrow aaaaabbbQYCCQ$$
$$\Rightarrow aaaaabbbbCCQYC \qquad \Rightarrow aaaaabbbQCCCQ$$
$$\Rightarrow aaaaabbbbCCQCC \qquad \Rightarrow aaaaabbbbCCCCQ$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ \qquad CQ \to XQ$$
$$S \to abC \qquad XQ \to XY$$
$$bQC \to bbCC \qquad XY \to QY$$
$$C \to c \qquad QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$\Rightarrow aaaaabbbbCCCXQ$
$\Rightarrow aaaaabbbbCCCXY$
$\Rightarrow aaaaabbbbCCCQY$
$\Rightarrow aaaaabbbbCCCQC$
$\Rightarrow aaaaabbbbCCXQC$
$\Rightarrow aaaaabbbbCCXYC$
$\Rightarrow aaaaabbbbCCQYC$
$\Rightarrow aaaaabbbbCCQCC$
$\Rightarrow aaaaabbbbCXQCC$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbCCCXQ$$
$$\Rightarrow aaaaabbbbCCCXY$$
$$\Rightarrow aaaaabbbbCCCQY$$
$$\Rightarrow aaaaabbbbCCCQC$$
$$\Rightarrow aaaaabbbbCCXQC$$
$$\Rightarrow aaaaabbbbCCXYC$$
$$\Rightarrow aaaaabbbbCCQYC$$
$$\Rightarrow aaaaabbbbCCQCC$$
$$\Rightarrow aaaaabbbbCXQCC$$
$$\Rightarrow aaaaabbbbCXYCC$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$\Rightarrow aaaaabbbbCCCXQ$
$\Rightarrow aaaaabbbbCCCXY$
$\Rightarrow aaaaabbbbCCCQY$
$\Rightarrow aaaaabbbbCCCQC$
$\Rightarrow aaaaabbbbCCXQC$
$\Rightarrow aaaaabbbbCCXYC$
$\Rightarrow aaaaabbbbCCQYC$
$\Rightarrow aaaaabbbbCCQCC$
$\Rightarrow aaaaabbbbCXQCC$
$\Rightarrow aaaaabbbbCXYCC$
$\Rightarrow aaaaabbbbCQYCC$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbCCCXQ$$
$$\Rightarrow aaaaabbbbCCCXY$$
$$\Rightarrow aaaaabbbbCCCQY$$
$$\Rightarrow aaaaabbbbCCCQC$$
$$\Rightarrow aaaaabbbbCCXQC$$
$$\Rightarrow aaaaabbbbCCXYC$$
$$\Rightarrow aaaaabbbbCCQYC$$
$$\Rightarrow aaaaabbbbCCQCC$$
$$\Rightarrow aaaaabbbbCXQCC$$
$$\Rightarrow aaaaabbbbCXYCC$$
$$\Rightarrow aaaaabbbbCQYCC$$
$$\Rightarrow aaaaabbbbCQCCC$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbCCCXQ$$
$$\Rightarrow aaaaabbbbCCCXY$$
$$\Rightarrow aaaaabbbbCCCQY$$
$$\Rightarrow aaaaabbbbCCCQC$$
$$\Rightarrow aaaaabbbbCCXQC$$
$$\Rightarrow aaaaabbbbCCXYC$$
$$\Rightarrow aaaaabbbbCCQYC$$
$$\Rightarrow aaaaabbbbCCQCC$$
$$\Rightarrow aaaaabbbbCXQCC$$
$$\Rightarrow aaaaabbbbCXYCC$$
$$\Rightarrow aaaaabbbbCQYCC$$
$$\Rightarrow aaaaabbbbCQCCC$$
$$\Rightarrow aaaaabbbbXQCCC$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbCCCXQ$$
$$\Rightarrow aaaaabbbbCCCXY$$
$$\Rightarrow aaaaabbbbCCCQY$$
$$\Rightarrow aaaaabbbbCCCQC$$
$$\Rightarrow aaaaabbbbCCXQC$$
$$\Rightarrow aaaaabbbbCCXYC$$
$$\Rightarrow aaaaabbbbCCQYC$$
$$\Rightarrow aaaaabbbbCCQCC$$
$$\Rightarrow aaaaabbbbCXQCC$$
$$\Rightarrow aaaaabbbbCXYCC$$
$$\Rightarrow aaaaabbbbCQYCC$$
$$\Rightarrow aaaaabbbbCQCCC$$
$$\Rightarrow aaaaabbbbXQCCC$$
$$\Rightarrow aaaaabbbbXYCCC$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ \qquad CQ \to XQ$$
$$S \to abC \qquad XQ \to XY$$
$$bQC \to bbCC \qquad XY \to QY$$
$$C \to c \qquad QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$\Rightarrow aaaaabbbbCCCXQ$
$\Rightarrow aaaaabbbbCCCXY$
$\Rightarrow aaaaabbbbCCCQY$
$\Rightarrow aaaaabbbbCCCQC$
$\Rightarrow aaaaabbbbCCXQC$
$\Rightarrow aaaaabbbbCCXYC$
$\Rightarrow aaaaabbbbCCQYC$
$\Rightarrow aaaaabbbbCCQCC$
$\Rightarrow aaaaabbbbCXQCC$
$\Rightarrow aaaaabbbbCXYCC$
$\Rightarrow aaaaabbbbCQYCC$
$\Rightarrow aaaaabbbbCQCCC$
$\Rightarrow aaaaabbbbXQCCC$
$\Rightarrow aaaaabbbbXYCCC$
$\Rightarrow aaaaabbbbQYCCC$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ \qquad CQ \to XQ$$
$$S \to abC \qquad XQ \to XY$$
$$bQC \to bbCC \qquad XY \to QY$$
$$C \to c \qquad QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbCCCXQ \qquad \Rightarrow aaaaabbbbCXQCC$$
$$\Rightarrow aaaaabbbbCCCXY \qquad \Rightarrow aaaaabbbbCXYCC$$
$$\Rightarrow aaaaabbbbCCCQY \qquad \Rightarrow aaaaabbbbCQYCC$$
$$\Rightarrow aaaaabbbbCCCQC \qquad \Rightarrow aaaaabbbbCQCCC$$
$$\Rightarrow aaaaabbbbCCXQC \qquad \Rightarrow aaaaabbbbXQCCC$$
$$\Rightarrow aaaaabbbbCCXYC \qquad \Rightarrow aaaaabbbbXYCCC$$
$$\Rightarrow aaaaabbbbCCQYC \qquad \Rightarrow aaaaabbbbQYCCC$$
$$\Rightarrow aaaaabbbbCCQCC \qquad \Rightarrow aaaaabbbbQCCCC$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbbCCCCC$$
$$\Rightarrow aaaaabbbbCXQCC$$
$$\Rightarrow aaaaabbbbCXYCC$$
$$\Rightarrow aaaaabbbbCQYCC$$
$$\Rightarrow aaaaabbbbCQCCC$$
$$\Rightarrow aaaaabbbbXQCCC$$
$$\Rightarrow aaaaabbbbXYCCC$$
$$\Rightarrow aaaaabbbbQYCCC$$
$$\Rightarrow aaaaabbbbQCCCC$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbbCCCCC$$
$$\Rightarrow aaaaabbbbbcCCCC$$
$$\Rightarrow aaaaabbbbCXQCC$$
$$\Rightarrow aaaaabbbbCXYCC$$
$$\Rightarrow aaaaabbbbCQYCC$$
$$\Rightarrow aaaaabbbbCQCCC$$
$$\Rightarrow aaaaabbbbXQCCC$$
$$\Rightarrow aaaaabbbbXYCCC$$
$$\Rightarrow aaaaabbbbQYCCC$$
$$\Rightarrow aaaaabbbbQCCCC$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$
\begin{aligned}
S &\to aSQ \\
S &\to abC \\
bQC &\to bbCC \\
C &\to c
\end{aligned}
\qquad\qquad
\begin{aligned}
CQ &\to XQ \\
XQ &\to XY \\
XY &\to QY \\
QY &\to QC
\end{aligned}
$$

Derivace slova $aaaaabbbbbccccc$:

$$
\begin{aligned}
&\Rightarrow aaaaabbbbbCCCCC \\
&\Rightarrow aaaaabbbbbcCCCC \\
&\Rightarrow aaaaabbbbbccCCC
\end{aligned}
\qquad
\begin{aligned}
&\Rightarrow aaaaabbbbCXQCC \\
&\Rightarrow aaaaabbbbCXYCC \\
&\Rightarrow aaaaabbbbCQYCC \\
&\Rightarrow aaaaabbbbCQCCC \\
&\Rightarrow aaaaabbbbXQCCC \\
&\Rightarrow aaaaabbbbXYCCC \\
&\Rightarrow aaaaabbbbQYCCC \\
&\Rightarrow aaaaabbbbQCCCC
\end{aligned}
$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$
\begin{aligned}
S &\to aSQ \\
S &\to abC \\
bQC &\to bbCC \\
C &\to c
\end{aligned}
\qquad\qquad
\begin{aligned}
CQ &\to XQ \\
XQ &\to XY \\
XY &\to QY \\
QY &\to QC
\end{aligned}
$$

Derivace slova $aaaaabbbbbccccc$:

$$
\begin{aligned}
&\Rightarrow aaaaabbbbbCCCCC \\
&\Rightarrow aaaaabbbbbcCCCC \\
&\Rightarrow aaaaabbbbbccCCC \\
&\Rightarrow aaaaabbbbbcccCC
\end{aligned}
\qquad
\begin{aligned}
&\Rightarrow aaaaabbbbCXQCC \\
&\Rightarrow aaaaabbbbCXYCC \\
&\Rightarrow aaaaabbbbCQYCC \\
&\Rightarrow aaaaabbbbCQCCC \\
&\Rightarrow aaaaabbbbXQCCC \\
&\Rightarrow aaaaabbbbXYCCC \\
&\Rightarrow aaaaabbbbQYCCC \\
&\Rightarrow aaaaabbbbQCCCC
\end{aligned}
$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbbCCCCC$$
$$\Rightarrow aaaaabbbbbcCCCC$$
$$\Rightarrow aaaaabbbbbccCCC$$
$$\Rightarrow aaaaabbbbbcccCC$$
$$\Rightarrow aaaaabbbbbccccC$$
$$\Rightarrow aaaaabbbbCXQCC$$
$$\Rightarrow aaaaabbbbCXYCC$$
$$\Rightarrow aaaaabbbbCQYCC$$
$$\Rightarrow aaaaabbbbCQCCC$$
$$\Rightarrow aaaaabbbbXQCCC$$
$$\Rightarrow aaaaabbbbXYCCC$$
$$\Rightarrow aaaaabbbbQYCCC$$
$$\Rightarrow aaaaabbbbQCCCC$$

Kontextové gramatiky

Příklad: Kontextová gramatika generující jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to aSQ$$
$$S \to abC$$
$$bQC \to bbCC$$
$$C \to c$$
$$CQ \to XQ$$
$$XQ \to XY$$
$$XY \to QY$$
$$QY \to QC$$

Derivace slova $aaaaabbbbbccccc$:

$$\Rightarrow aaaaabbbbbCCCCC$$
$$\Rightarrow aaaaabbbbbcCCCC$$
$$\Rightarrow aaaaabbbbbccCCC$$
$$\Rightarrow aaaaabbbbbcccCC$$
$$\Rightarrow aaaaabbbbbccccC$$
$$\Rightarrow aaaaabbbbbccccc$$
$$\Rightarrow aaaaabbbbCXQCC$$
$$\Rightarrow aaaaabbbbCXYCC$$
$$\Rightarrow aaaaabbbbCQYCC$$
$$\Rightarrow aaaaabbbbCQCCC$$
$$\Rightarrow aaaaabbbbXQCCC$$
$$\Rightarrow aaaaabbbbXYCCC$$
$$\Rightarrow aaaaabbbbQYCCC$$
$$\Rightarrow aaaaabbbbQCCCC$$

Bezkontextové gramatiky

Dalším speciálním typem generativních gramatik jsou **bezkontextové gramatiky**.

Bezkontextové gramatiky jsou označovány též jako gramatiky **typu 2**.

Gramatika $\mathcal{G} = (\Pi, \Sigma, S, P)$ se nazývá **bezkontextová**, jestliže všechna její pravidla jsou tvaru

$$X \to \gamma$$

kde $X \in \Pi$, $\gamma \in (\Pi \cup \Sigma)^*$.

Příklad pravidla:

$$C \to DaBBc$$

Bezkontextové gramatiky

Poznámka: Ne každá bezkontextová gramatika je kontextová, protože bezkontextová gramatika může obsahovat i jiná ε-pravidla (tj. pravidla tvaru $X \to \varepsilon$) než $S \to \varepsilon$.

Libovolná bezkontextová gramatika bez ε-pravidel (resp. nanejvýš s jedním ε-pravidlem $S \to \varepsilon$, přičemž se neterminál S nenachází na pravé straně žádného pravidla) je speciálním případem kontextové gramatiky.

Ke každé bezkontextové gramatice $\mathcal{G}$ je možné sestrojit ekvivalentní bezkontextovou gramatiku bez ε-pravidel.

Ke každé bezkontextové gramatice tedy existuje ekvivalentní kontextová gramatika.

Regulární gramatiky

Připomeňme, že gramatika je **pravá** (resp. **levá**) **regulární** gramatika, jestliže všechna její pravidla jsou následujících dvou tvarů:

- $A \to wB$ (resp. $A \to Bw$)
- $A \to w$

kde $A, B \in \Pi$, $w \in \Sigma^*$.

Gramatika je **regulární**, jestliže se jedná o pravou nebo levou regulární gramatiku.

Regularní gramatiky jsou označovány jako gramatiky **typu 3**.

Je zjevné, že regulární gramatiky jsou speciálním případem bezkontextových gramatik.

Chomského hierarchie

Podle tvaru pravidel, která jsou v gramatice povolena, je tedy možné rozdělit gramatiky na následující čtyři typy:

- **Typ 0** — obecné **generativní gramatiky**
  pravidla bez omezení
- **Typ 1** — **kontextové gramatiky**
  pravidla tvaru $\alpha X \beta \to \alpha \gamma \beta$, kde $|\gamma| \geq 1$
  (Výjimka $S \to \varepsilon$, ale S pak není na pravé straně žádného pravidla.)
- **Typ 2** — **bezkontextové gramatiky**
  pravidla tvaru $X \to \gamma$
- **Typ 3** — **regulární gramatiky**
  pravidla tvaru $X \to wY$ (resp. $X \to Yw$) nebo $X \to w$

kde $\alpha, \beta, \gamma \in (\Pi \cup \Sigma)^*$, $X \in \Pi$ a $w \in \Sigma^*$

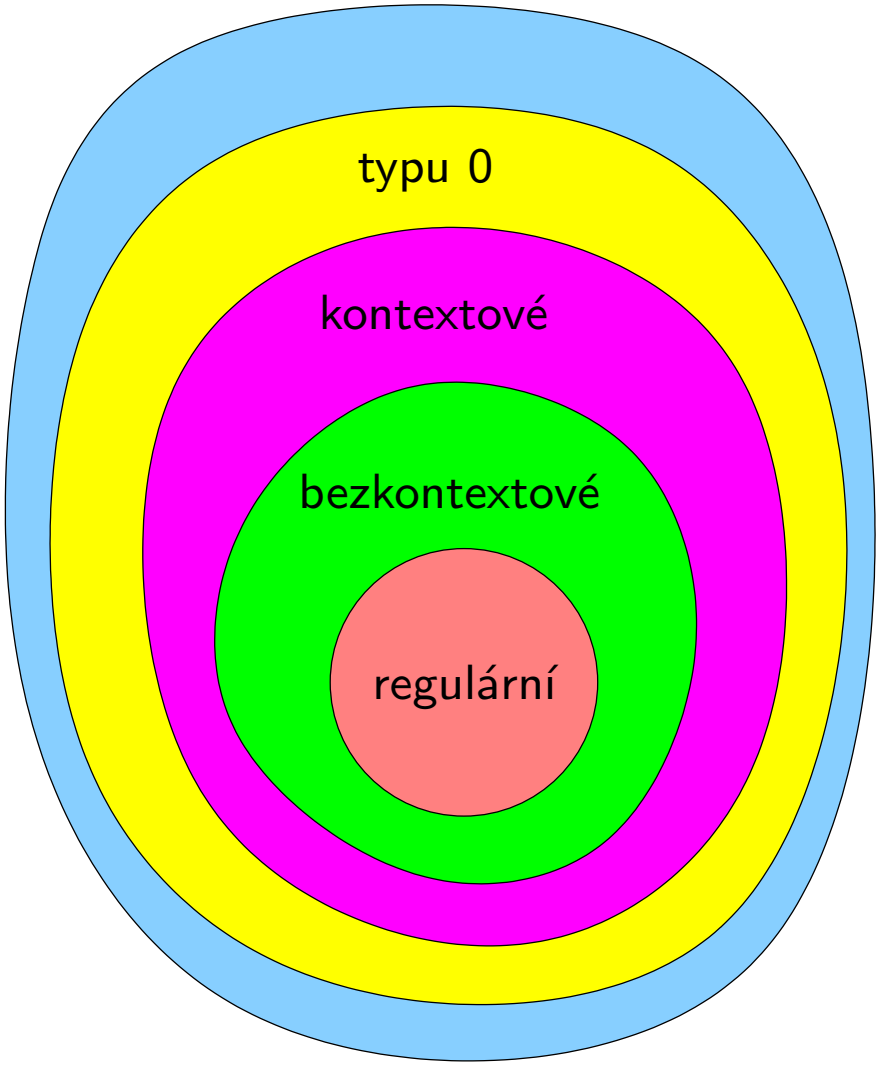
Chomského hierarchie

Jednotlivým typům gramatik odpovídají jednotlivé typy jazyků:

- **Typ 0**: Jazyk L je **rekurzivně spočetný** (či **typu 0**),
  jestliže existuje generativní gramatika, která tento jazyk generuje.
- **Typ 1**: Jazyk L je **kontextový** (či **typu 1**),
  jestliže existuje kontextová gramatika, která tento jazyk generuje.
- **Typ 2**: Jazyk L je **bezkontextový** (či **typu 2**),
  jestliže existuje bezkontextová gramatika, která tento jazyk generuje.
- **Typ 3**: Jazyk L je **regulární** (či **typu 3**),
  jestliže existuje regulární gramatika, která tento jazyk generuje.

Chomského hierarchie

Třídy jazyků:



Chomského hierarchie

- Příklad jazyka, který je bezkontextový, ale není regulární:

$$\{a^n b^n \mid n \geq 1\}$$

- Příklad jazyka, který je kontextový, ale není bezkontextový:

$$\{a^n b^n c^n \mid n \geq 1\}$$

Chomského hierarchie

- Příklady jazyků, které jsou typu 0, ale nejsou kontextové:
  - Jazyk tvořený slovy, která reprezentují logicky platné formule predikátové logiky.
  - Jazyk tvořený slovy, která reprezentují kódy těch Turingových strojů, které při výpočtu nad prázdným slovem po konečném počtu kroků zastaví.
- Příklady jazyků, které nejsou typu 0:
  - Jazyk tvořený slovy, která reprezentují právě ty formule predikátové logiky, které nejsou logicky platné.
  - Jazyk tvořený slovy, která reprezentují kódy těch Turingových strojů, které při výpočtu nad prázdným slovem nikdy nezastaví.
  - Jazyk tvořený slovy, která reprezentují kódy těch Turingových strojů, které při výpočtu nad libovolným slovem vždy po konečném počtu kroků zastaví.

Chomského hierarchie

- Další možné charakterizace **regulárních** jazyků:
  - jazyky přijímané konečnými automaty (deterministickými, nedeterministickými, zobecněnými nedeterministickými)
  - jazyky, které je možné popsat pomocí regulárních výrazů
- Další možná charakterizace **bezkontextových** jazyků:
  - jazyky přijímané nedeterministickými zásobníkovými automaty
- Další možná charakterizace **kontextových** jazyků:
  - jazyky přijímané nedeterministickými lineárně omezenými automaty
- Další možná charakterizace jazyků **typu 0**:
  - jazyky přijímané (deterministickými či nedeterministickými) Turingovými stroji

Chomského hierarchie

Chomského hierarchie — shrnutí:

- **Typ 0** — **rekurzivně spočetné** jazyky:
  - obecné generativní gramatiky
  - Turingovy stroje (deterministické, nedeterministické)
- **Typ 1** — **kontextové** jazyky:
  - kontextové gramatiky
  - nedeterministické lineárně omezené automaty
- **Typ 2** — **bezkontextové** jazyky:
  - bezkontextové gramatiky
  - nedeterministické zásobníkové automaty
- **Typ 3** — **regulární** jazyky:
  - regulární gramatiky
  - konečné automaty (deterministické, nedeterministické)
  - regulární výrazy

Výpočetní modely

Výpočet algoritmu

Algoritmy jsou vykonávány stroji — může to být například:

- skutečný počítač — vykonává instrukce strojového kódu
- virtuální stroj — vykonává instrukce bytekódu
- nějaký idealizovaný matematický model počítače
- ...

Stroj může být:

- jednoúčelový — vykonává jen jeden algoritmus
- obecnější — algoritmus dostává ve formě **programu**

Stroj pracuje po **krocích**.

Algoritmus během výpočtu zpracovává konkrétní **vstup** a produkuje příslušný **výstup**.

Výpočetní modely

Výpočetní model — nějaký idealizovaný matematický model počítače

- abstrahujeme od různých nepodstatných implementačních detailů
- chceme analyzovat ty vlastnosti algoritmů, které pokud možno co nejméně závisí na detailech stroje, který bude daný algoritmus vykonávat

Příklady některých výpočetních modelů:

- konečné automaty
- zásobníkové automaty
- Turingovy stroje
- stroje RAM
- ...

Výpočetní modely

Během výpočtu si stroj typicky musí pamatovat:

- která instrukce se právě provádí
- obsah své pracovní paměti

Podle typu stroje je určeno:

- s jakým typem dat stroj pracuje
- jak jsou tato data v paměti organizována
- jaké operace s těmito daty může stroj vykonávat

Podle typu algoritmu a typu analýzy, kterou chceme provádět, se můžeme rozhodnout, zda má smysl mezi obsah paměti zahranout i místa

- odkud se čtou vstupní data
- kam se zapisují výstupní data

Výpočetní modely

Jedním z využití výpočetních modelů je to, že mohou sloužit pro přesné definování pojmů, důležitých pro stanovení **výpočetní složitosti** daného algoritmu:

- **doby výpočtu** daného algoritmu $\mathcal{A}$ pro daný vstup w
  (pozn.: většinou je to počet kroků vykonaných strojem během výpočtu)
- **množství použité paměti** během tohoto výpočtu

Obecně je pro různé výpočetní modely také důležité

- zda je daný typ stroje schopen **simulovat** výpočty nějakého jiného typu stroje
- jak se při této simulaci liší doba výpočtu či množství použité paměti oproti původnímu stroji

Simulace výpočtu

Vysvětlení toho, co to znamená, že stroj $\mathcal{M}$ je **simulován** strojem $\mathcal{M}'$:

- Výpočet stroje $\mathcal{M}$ pro vstup w je (konečná nebo nekonečná) posloupnost konfigurací stroje $\mathcal{M}$

$$\alpha_0 \longrightarrow \alpha_1 \longrightarrow \alpha_2 \longrightarrow \cdots$$

- Tomuto výpočtu odpovídá výpočet stroje $\mathcal{M}'$ tvořený konfiguracemi

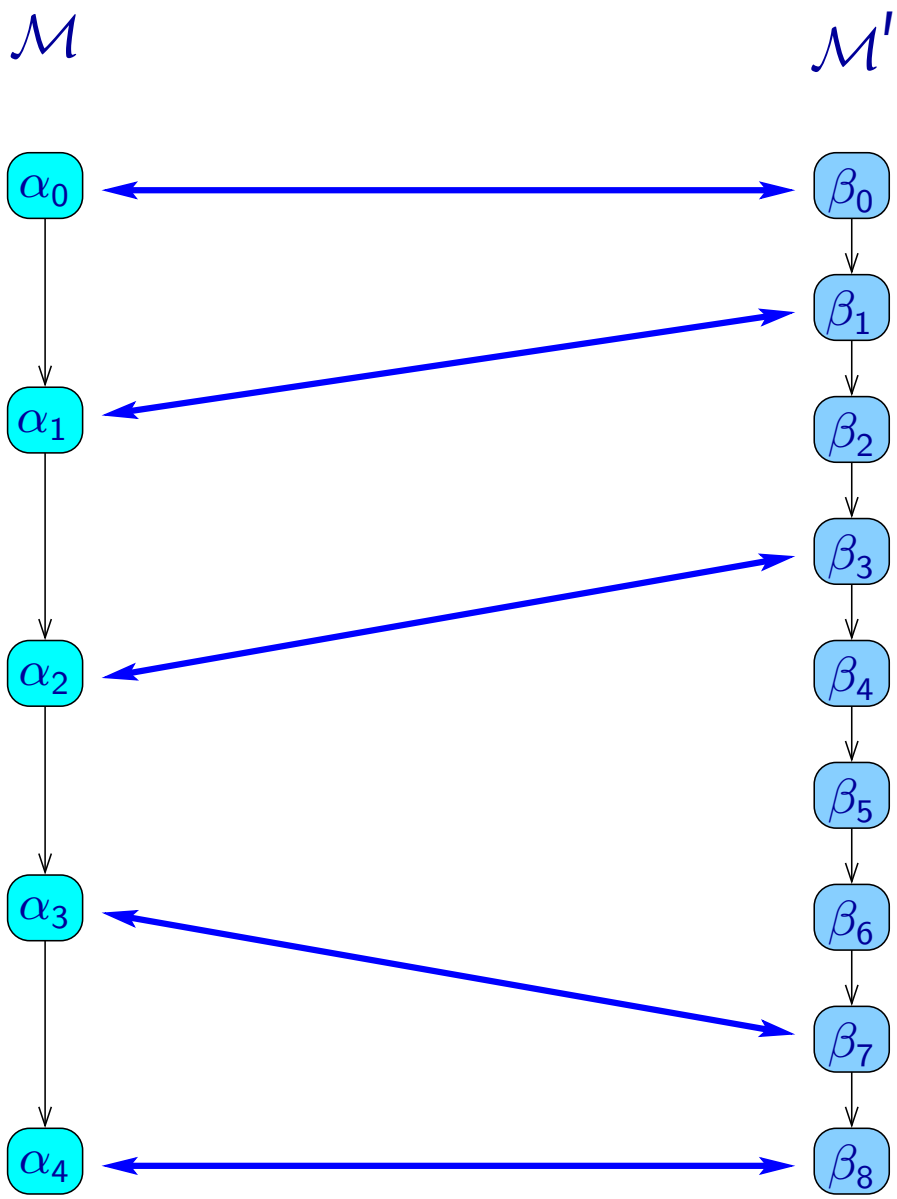
$$\beta_0 \longrightarrow \beta_1 \longrightarrow \beta_2 \longrightarrow \cdots$$

  kde každé konfiguraci α_i odpovídá nějaká konfigurace $\beta_{f(i)}$, kde $f : \mathbb{N} \to \mathbb{N}$ je funkce, pro kterou platí $f(i) \leq f(j)$ pro každé i a j, kde $i < j$.

- Existuje relace mezi vzájemně si odpovídajícími konfiguracemi stroje $\mathcal{M}$ a jim odpovídajícími konfiguracemi stroje $\mathcal{M}'$.

- Existují funkce mapující vstup w na odpovídající počáteční konfigurace α_0 a β_0 a analogicky funkce mapující koncové konfigurace na výsledek výpočtu.

Simulace výpočtu



Výpočetní modely

Některé výpočetní modely jsou slabší (konečné automaty, zásobníkové automaty, . . .) a není pomocí nich možné implementovat libovolný algoritmus.

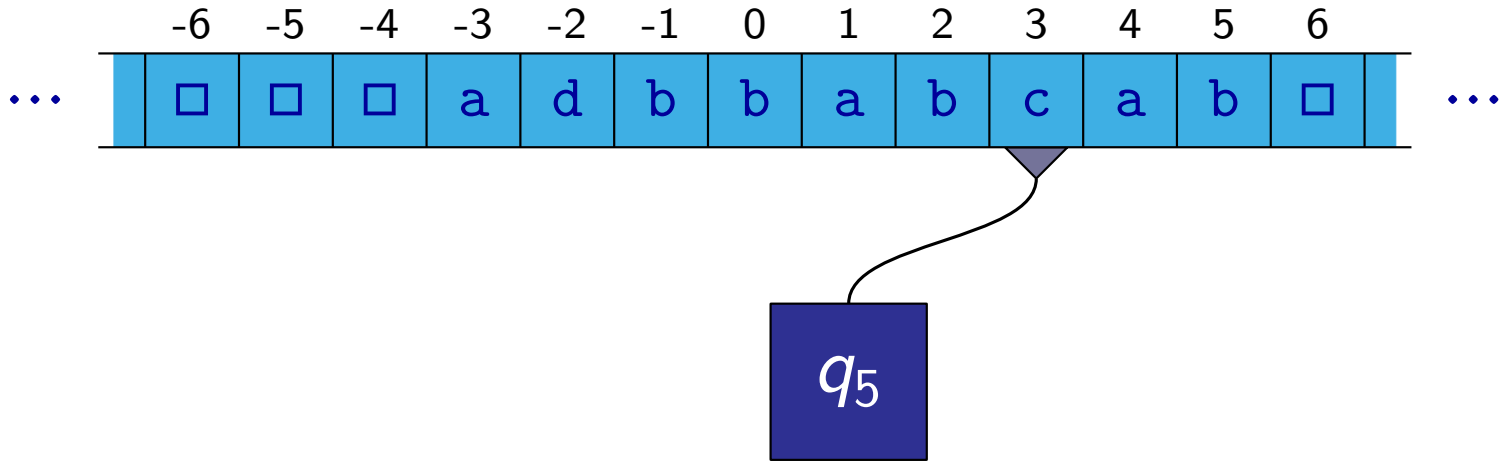
My se teď zaměříme na výpočetní modely, které jsou dostatečně silné na to, aby byly schopny vykonávat libovolný algoritmus (např. takový, jaký je možné zapsat jako program v nějakém programovacím jazyce).

Takovým výpočetním modelům se říká **Turingovsky úplné**:

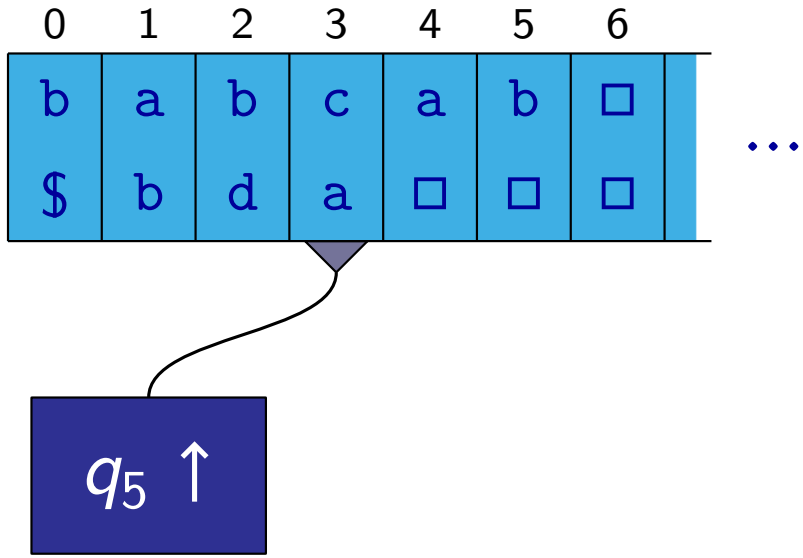
- samy jsou schopny simulovat činnost libovolného Turingova stroje
- jejich činnost může být simulována Turingovým strojem

Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

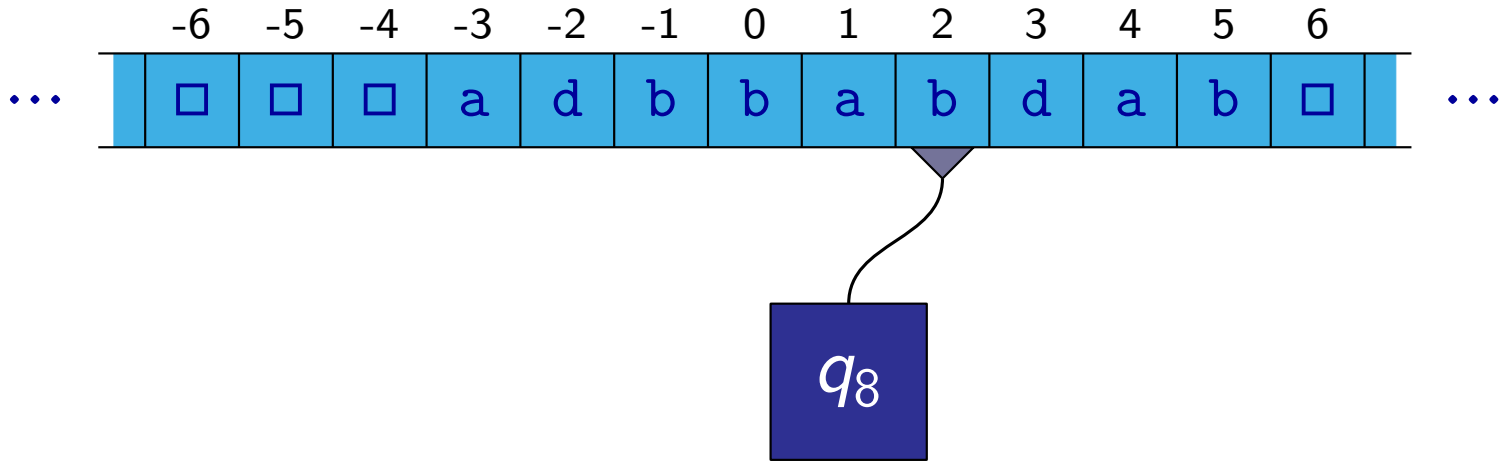


Jednostranně nekonečná páska:

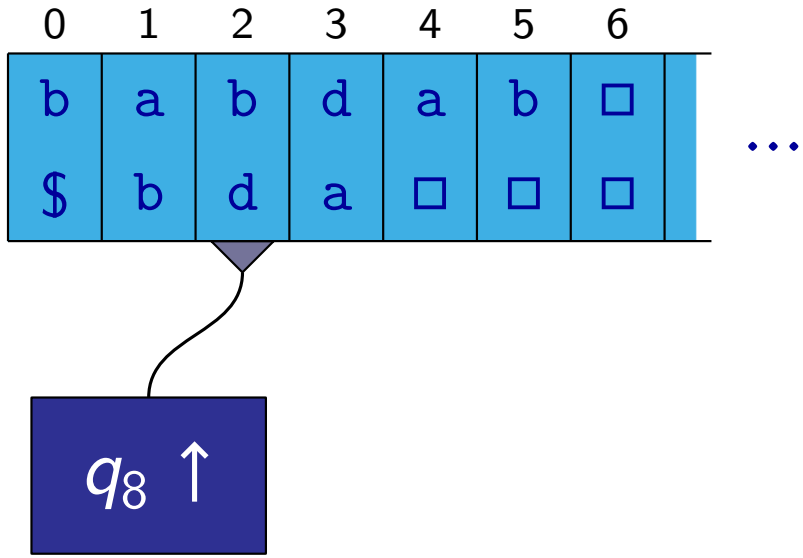


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

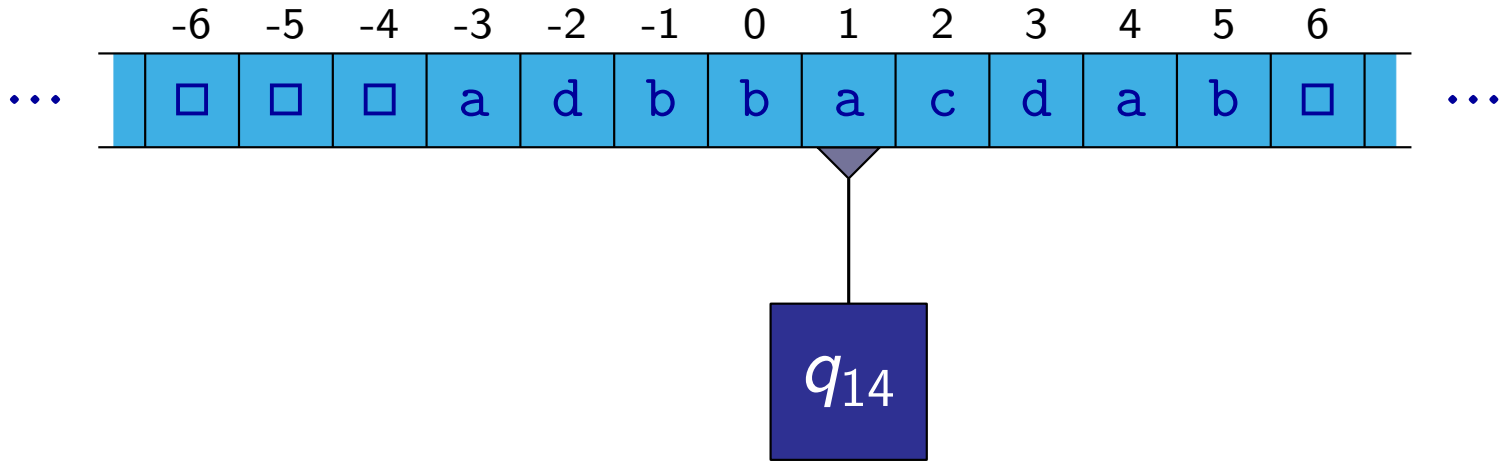


Jednostranně nekonečná páska:

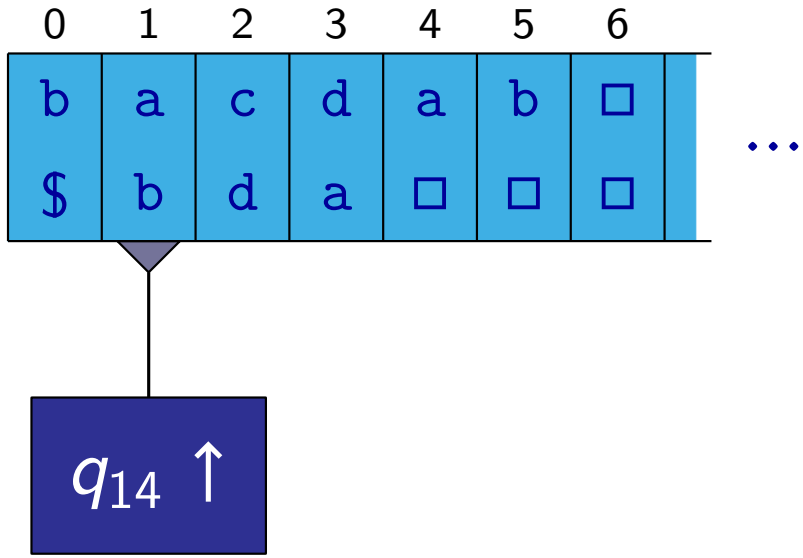


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

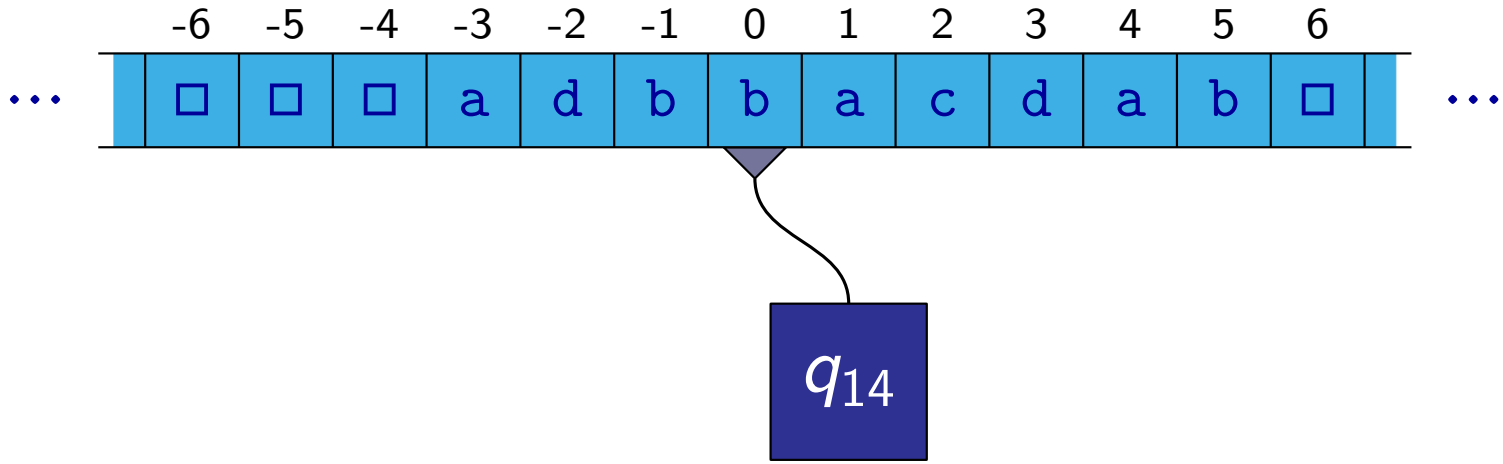


Jednostranně nekonečná páska:

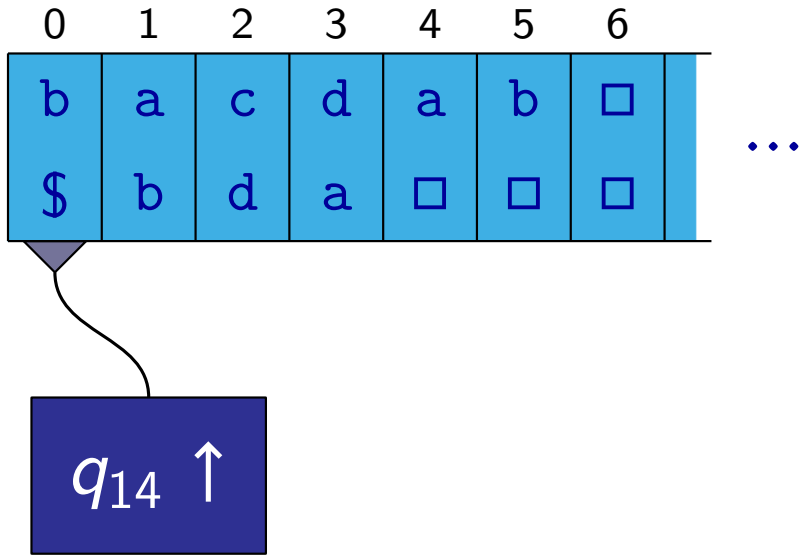


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

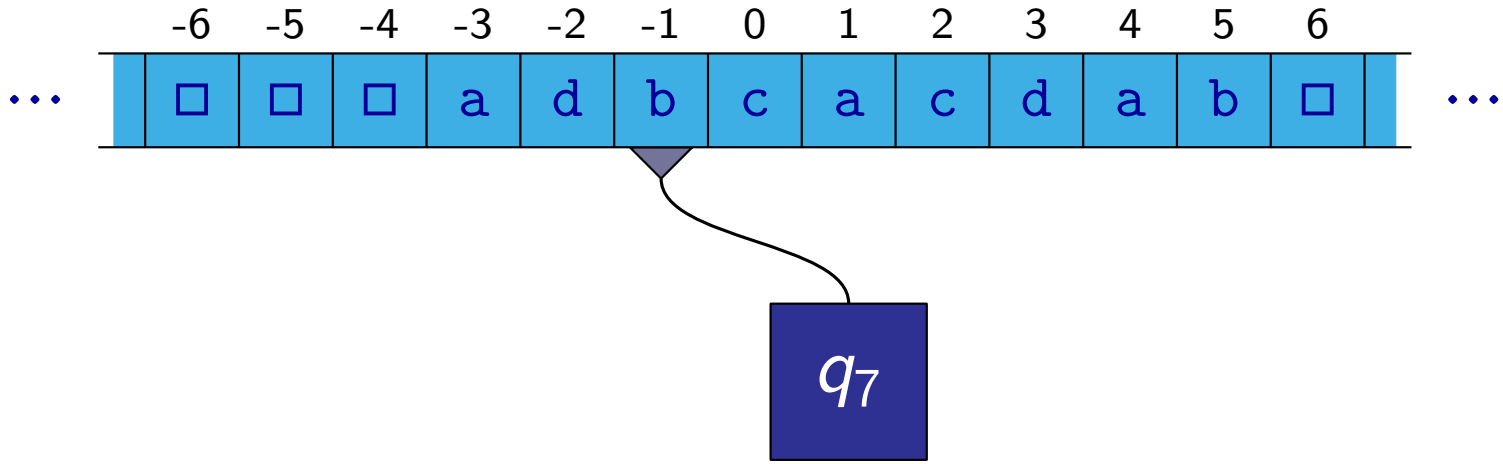


Jednostranně nekonečná páska:

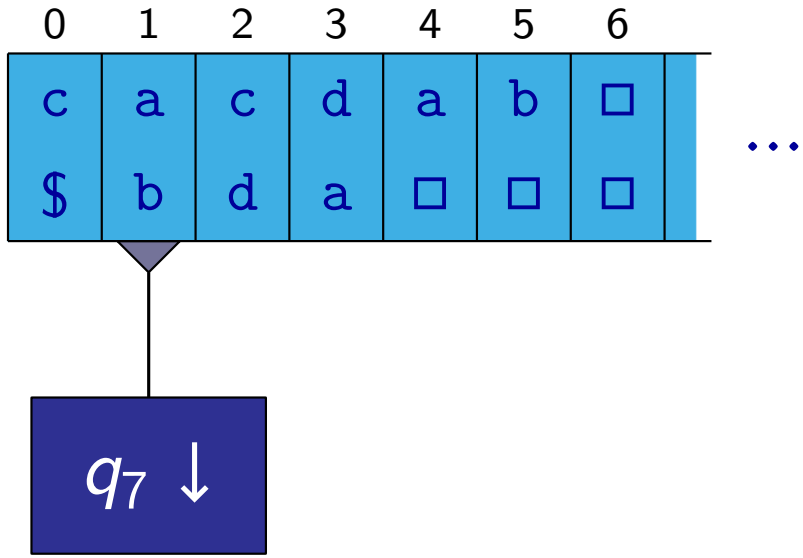


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

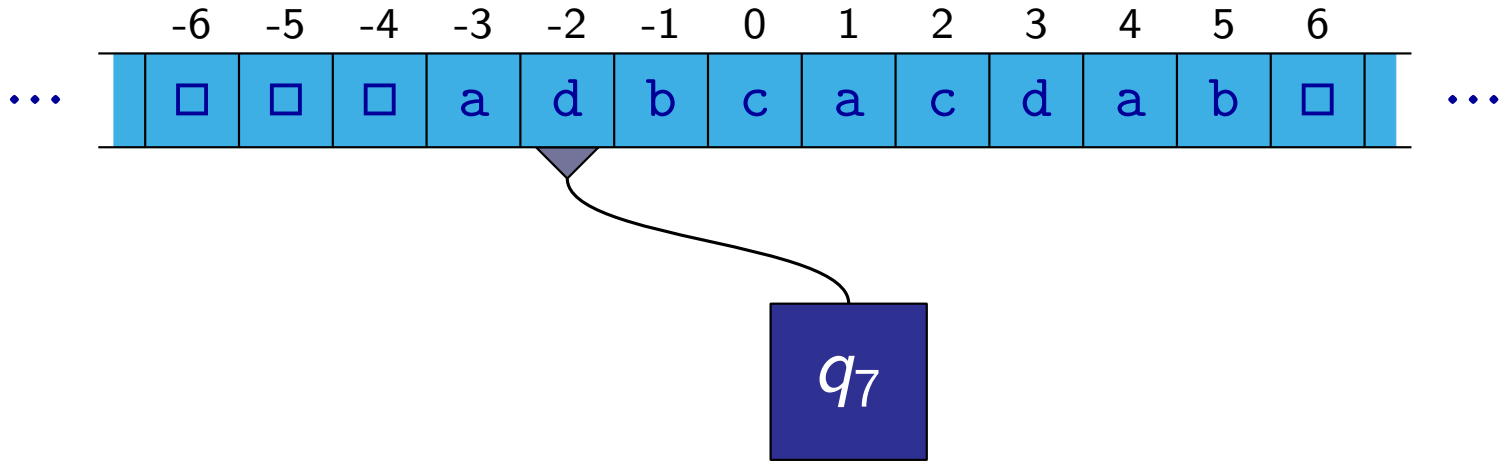


Jednostranně nekonečná páska:

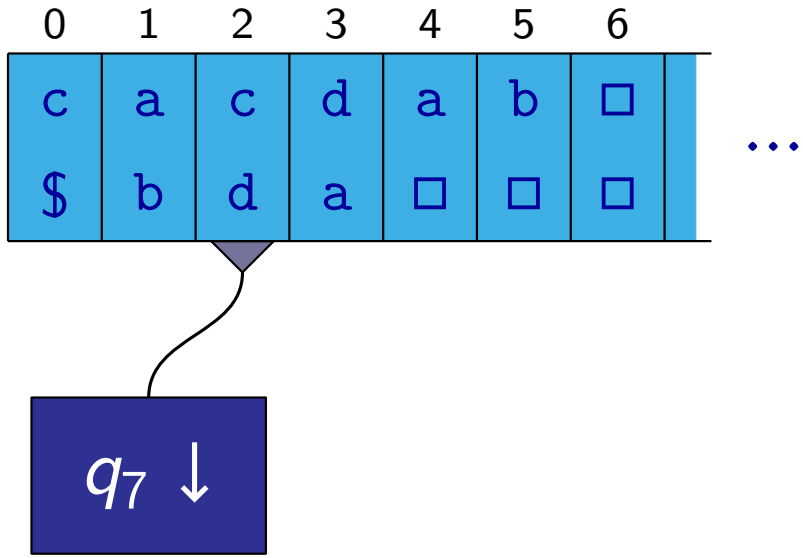


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

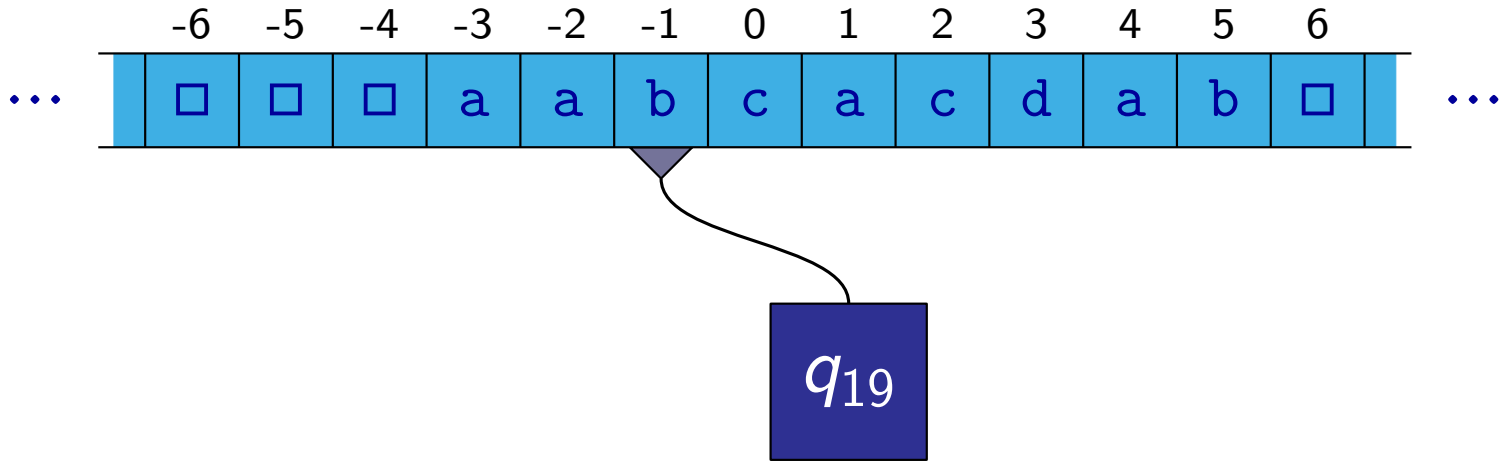


Jednostranně nekonečná páska:

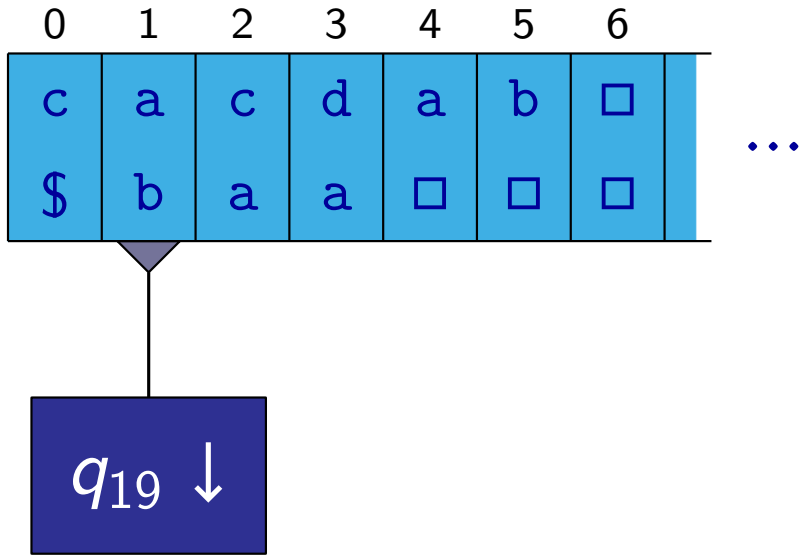


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

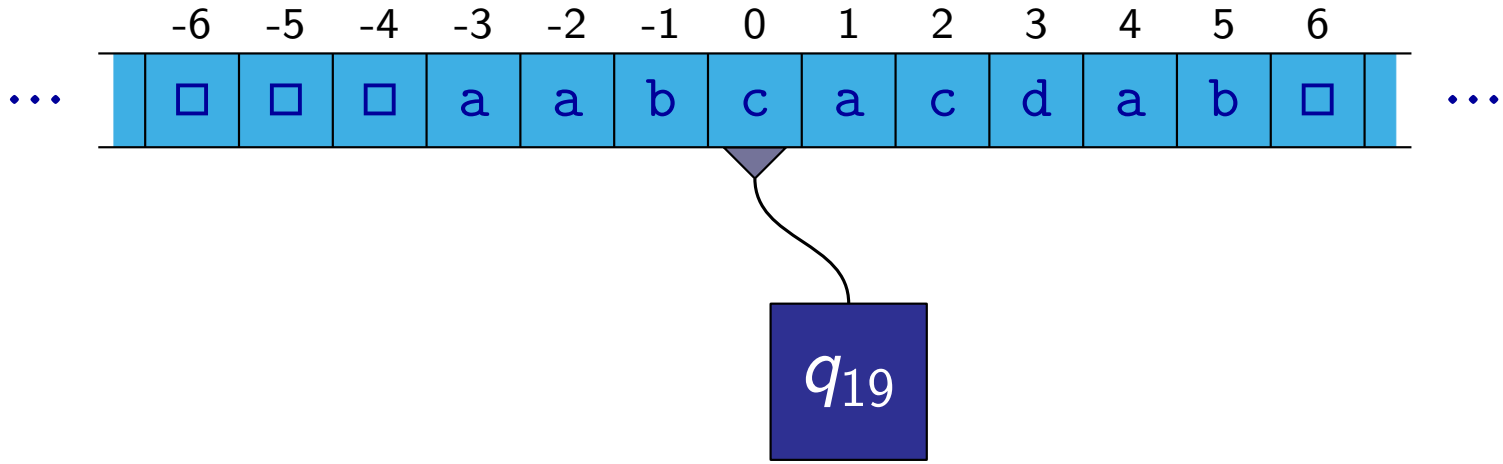


Jednostranně nekonečná páska:

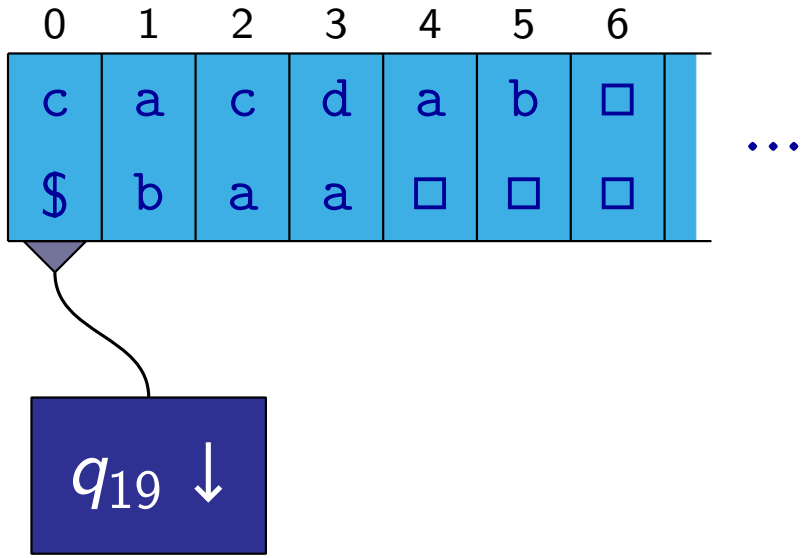


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

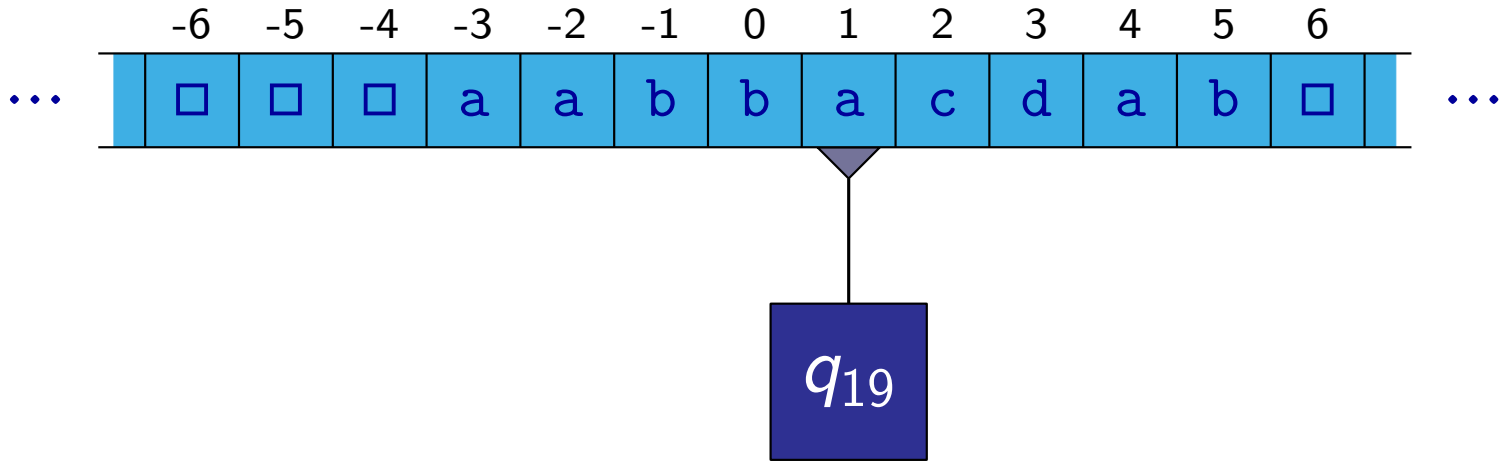


Jednostranně nekonečná páska:

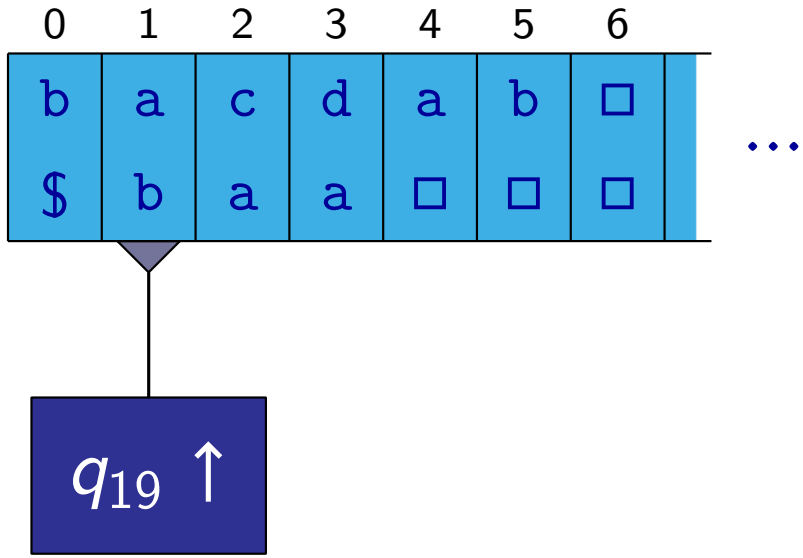


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

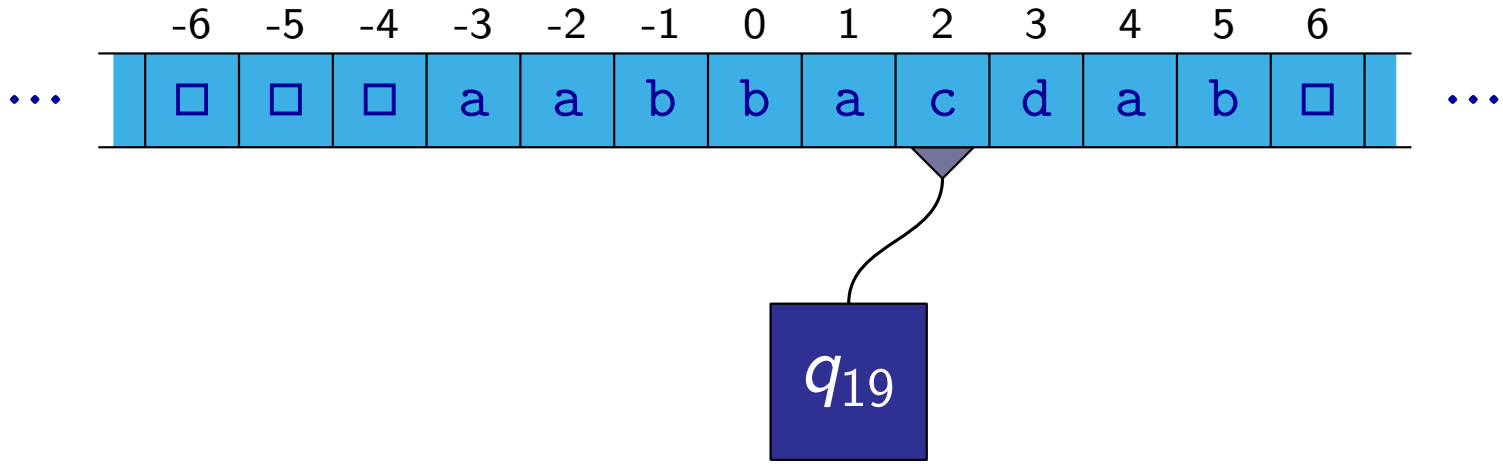


Jednostranně nekonečná páska:

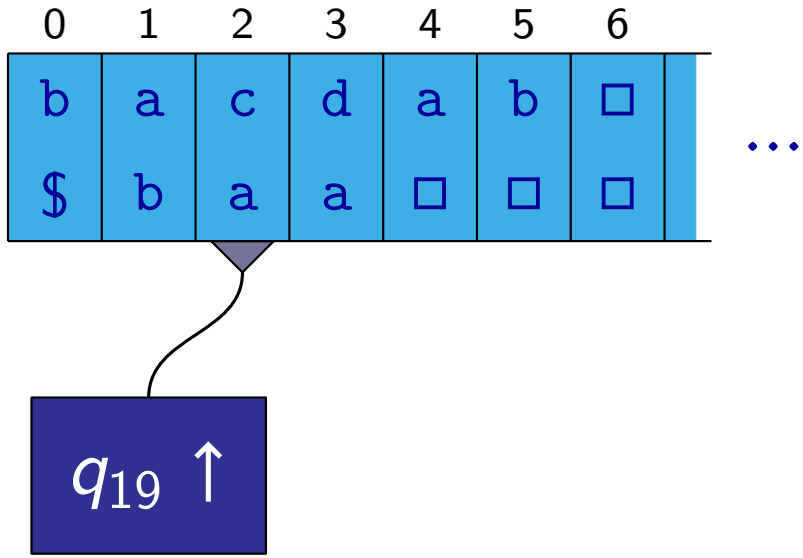


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:



Jednostranně nekonečná páska:



Abeceda $\{0, 1\}$

Činnost stroje s libovolnou páskovou abecedou Γ může být simulována strojem s páskovou abecedou $\{0, 1\}$.

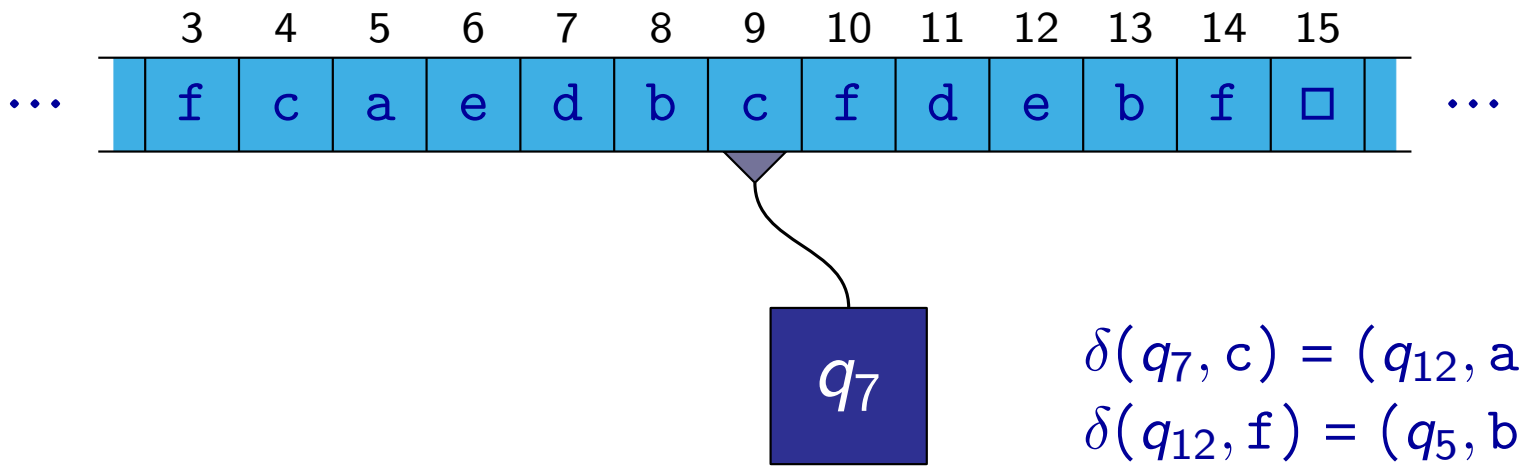
Stačí zvolit nějaké vhodné kódování symbolů abecedy Γ pomocí k-bitových sekvencí.

Příklad: Pásková abeceda $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}\}$

$$
\begin{array}{ccc}
\square & \leftrightarrow & 000 \\
\mathtt{a} & \leftrightarrow & 001 \\
\mathtt{b} & \leftrightarrow & 010 \\
\mathtt{c} & \leftrightarrow & 011 \\
\mathtt{d} & \leftrightarrow & 100 \\
\mathtt{e} & \leftrightarrow & 101 \\
\mathtt{f} & \leftrightarrow & 110 \\
\mathtt{g} & \leftrightarrow & 111
\end{array}
$$

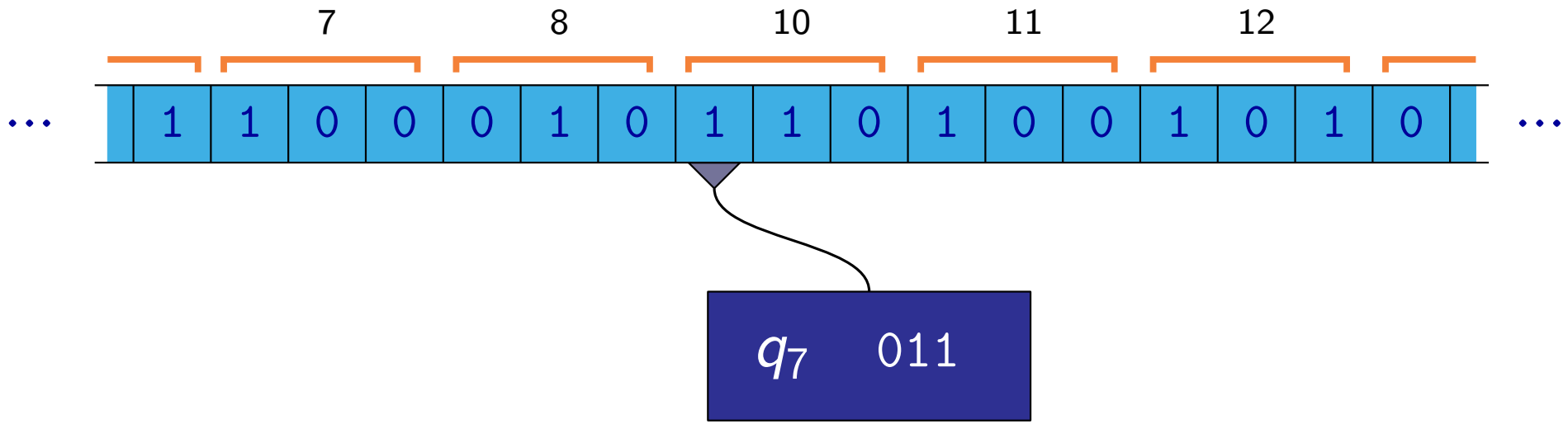
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



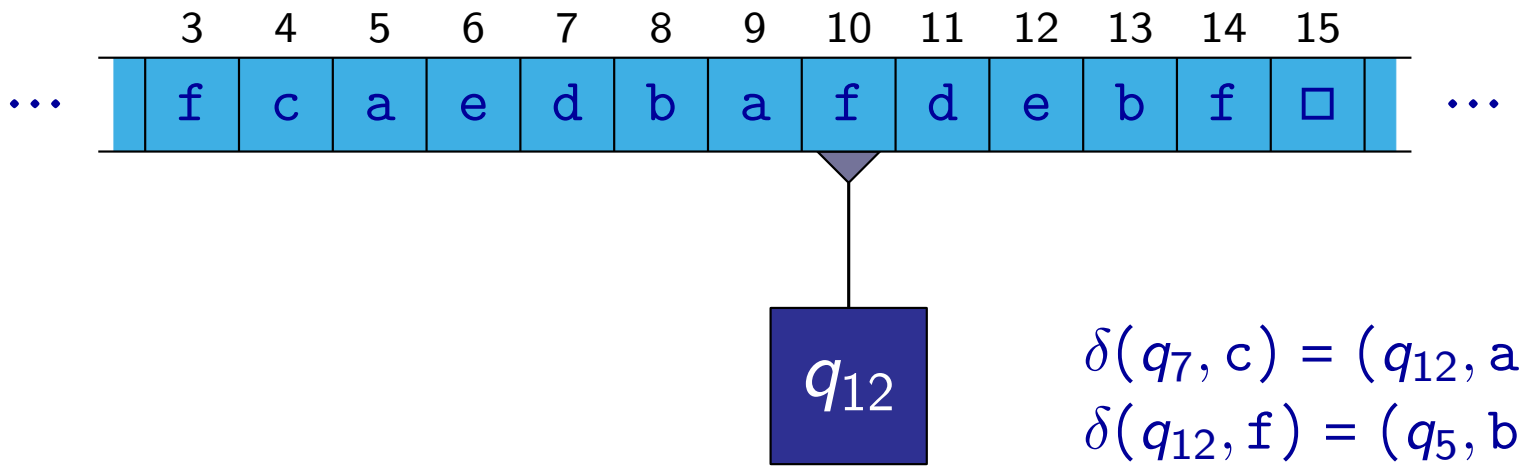
$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



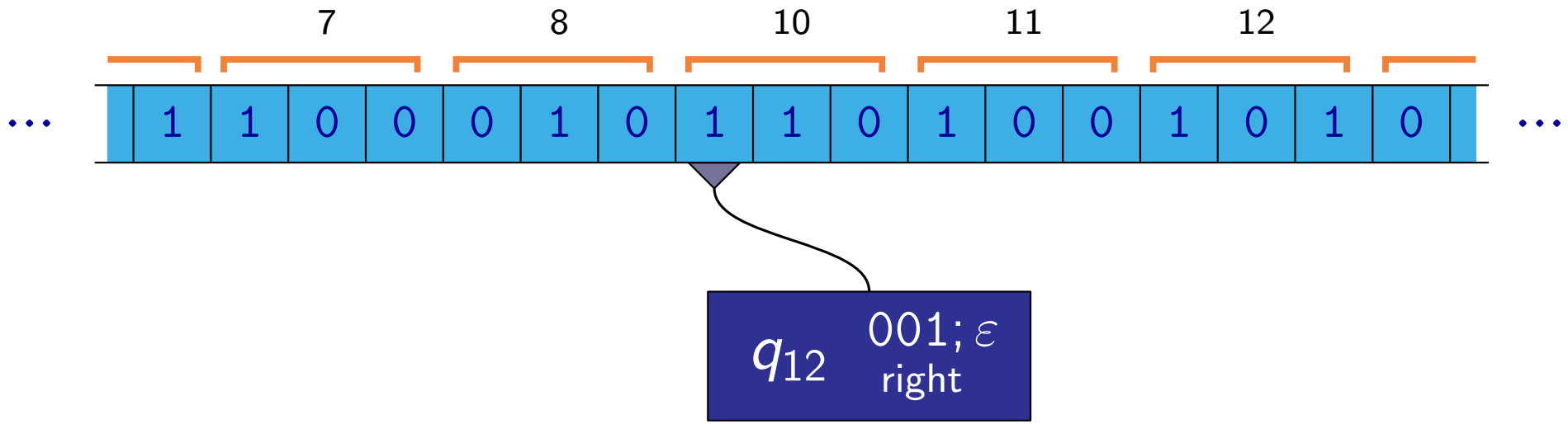
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



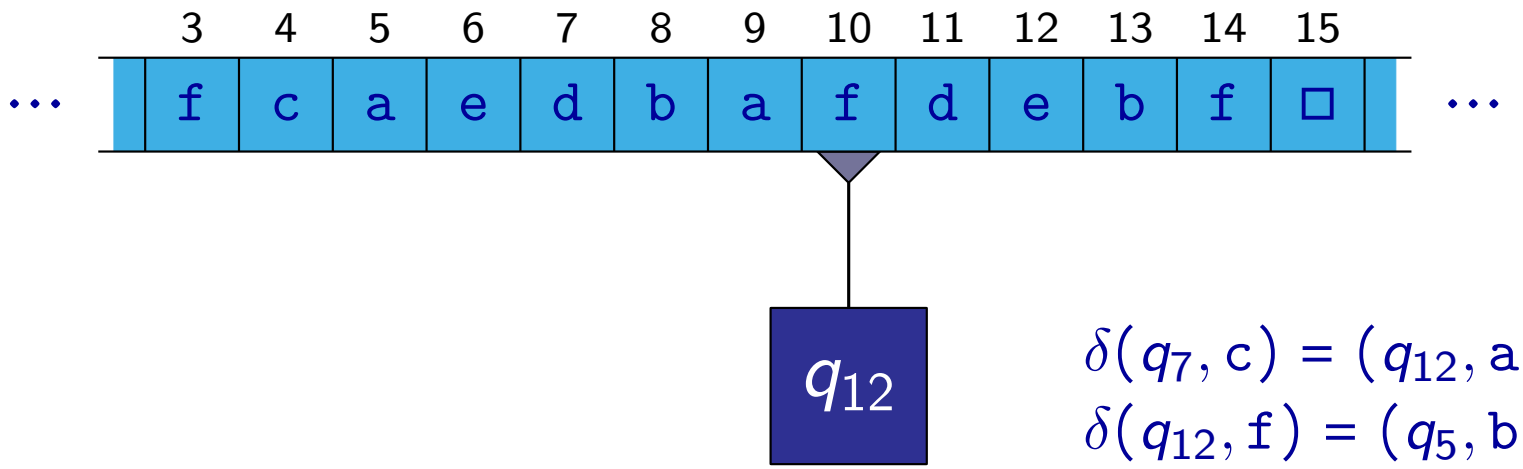
$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



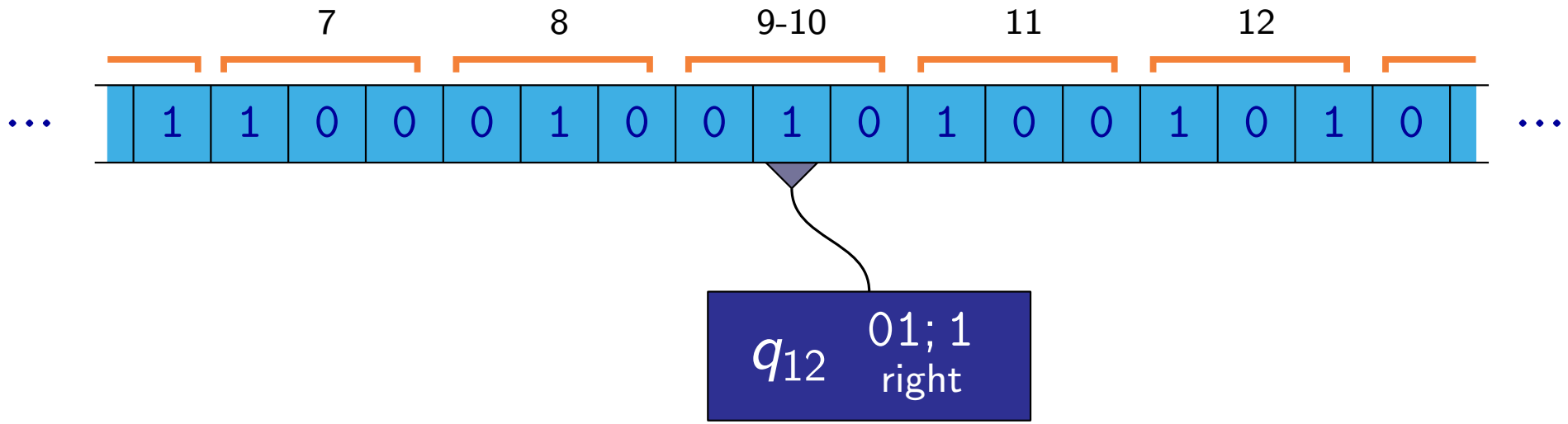
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



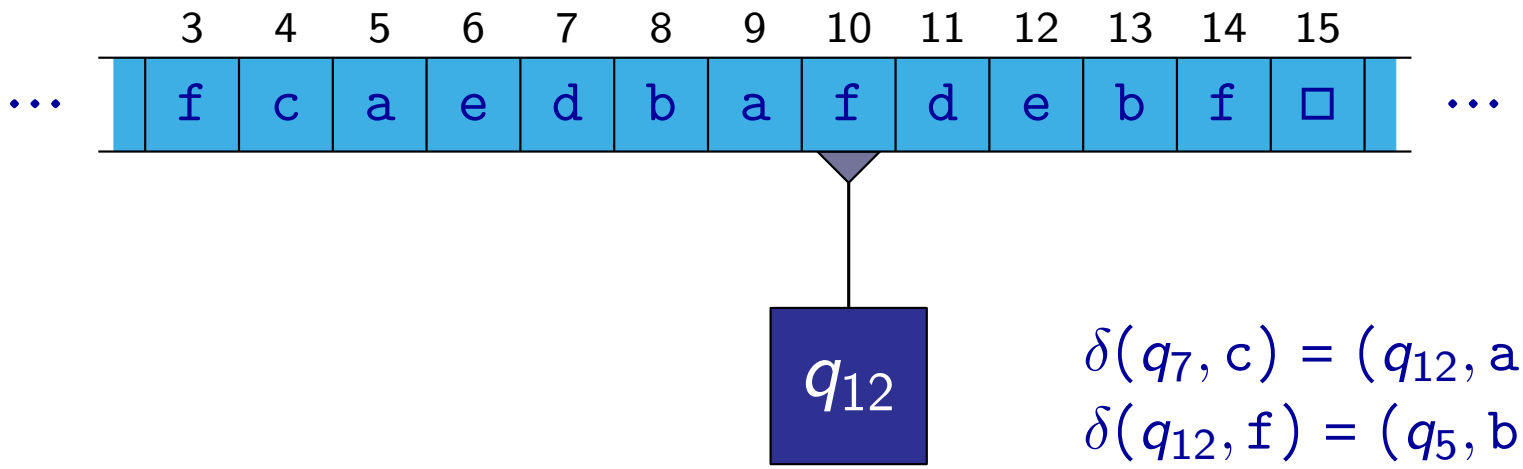
$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



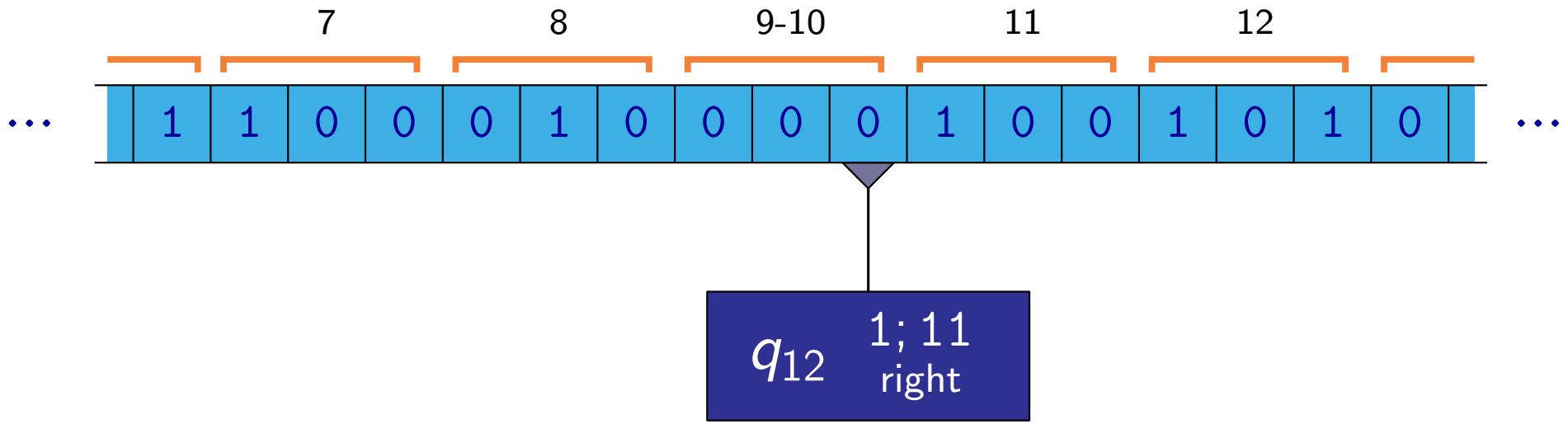
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



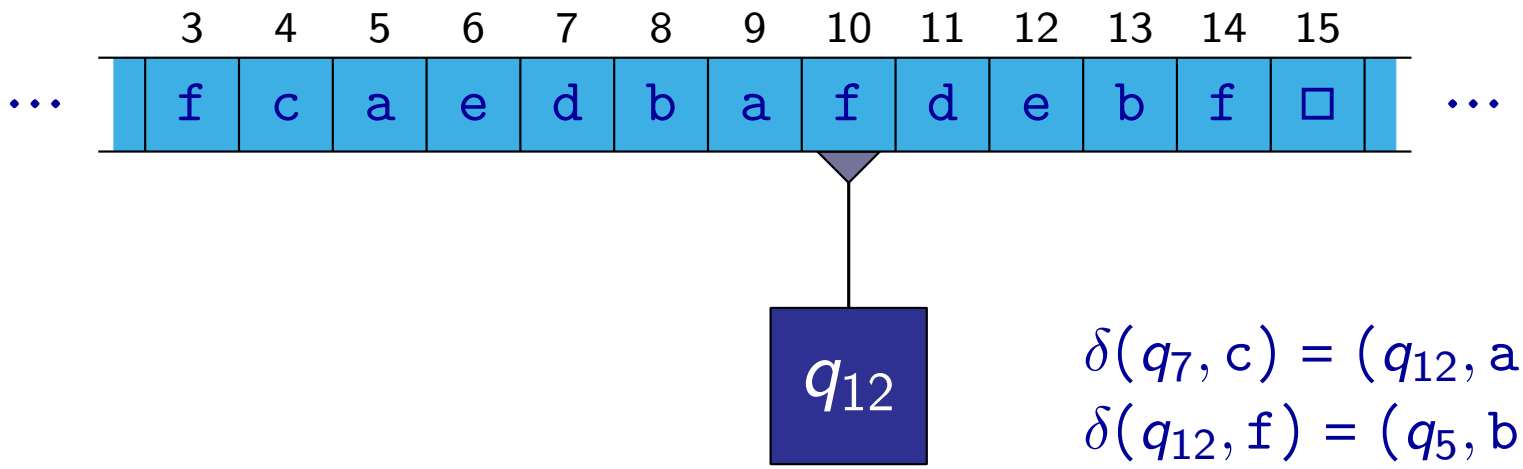
$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



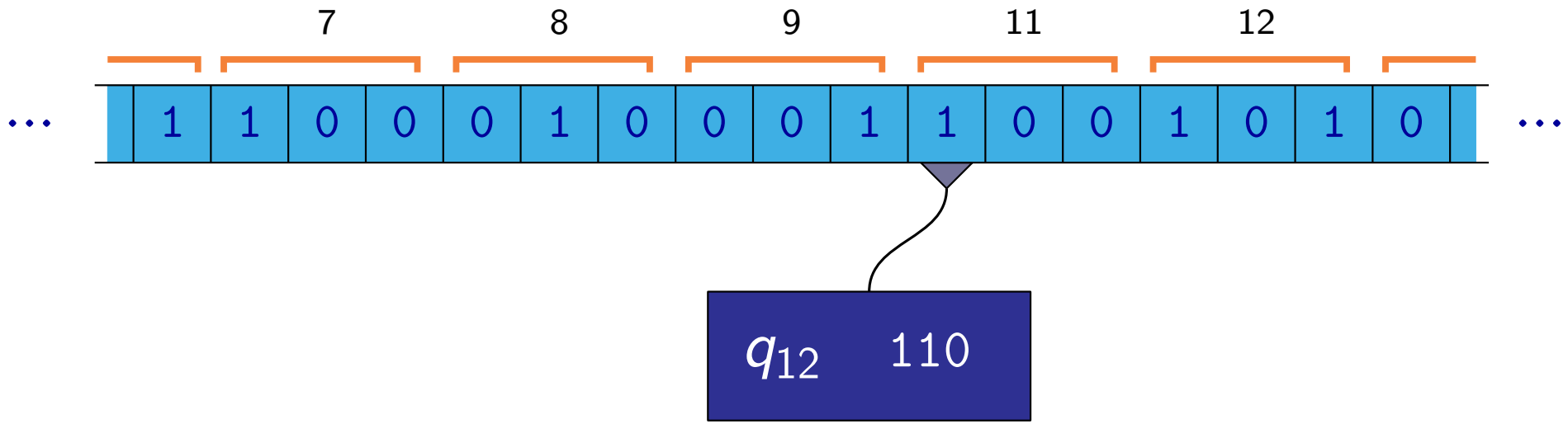
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



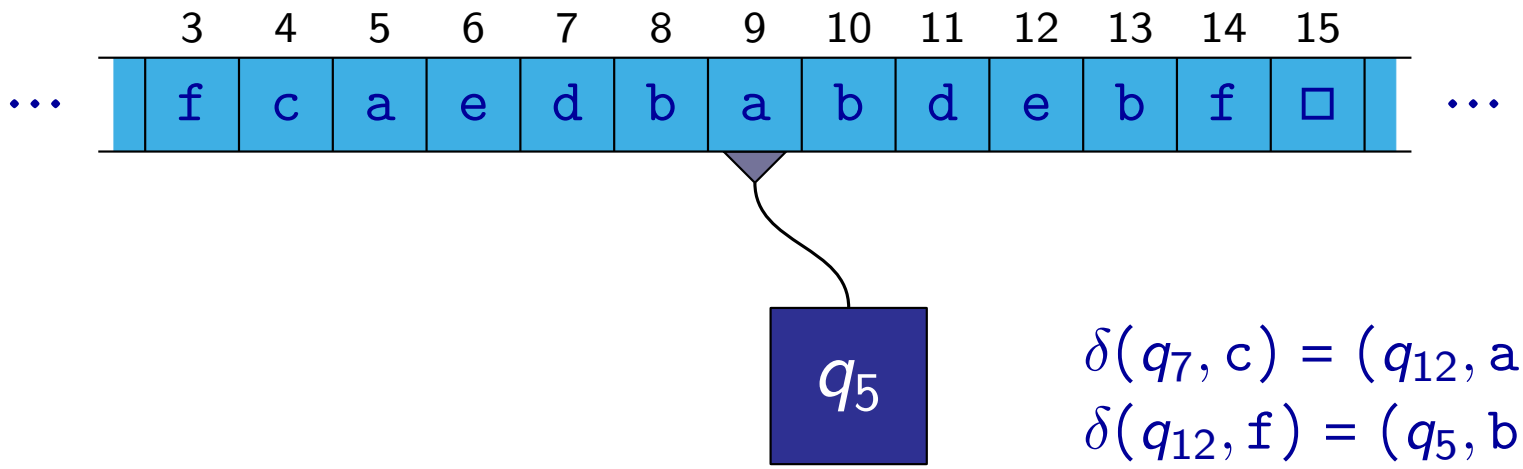
$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



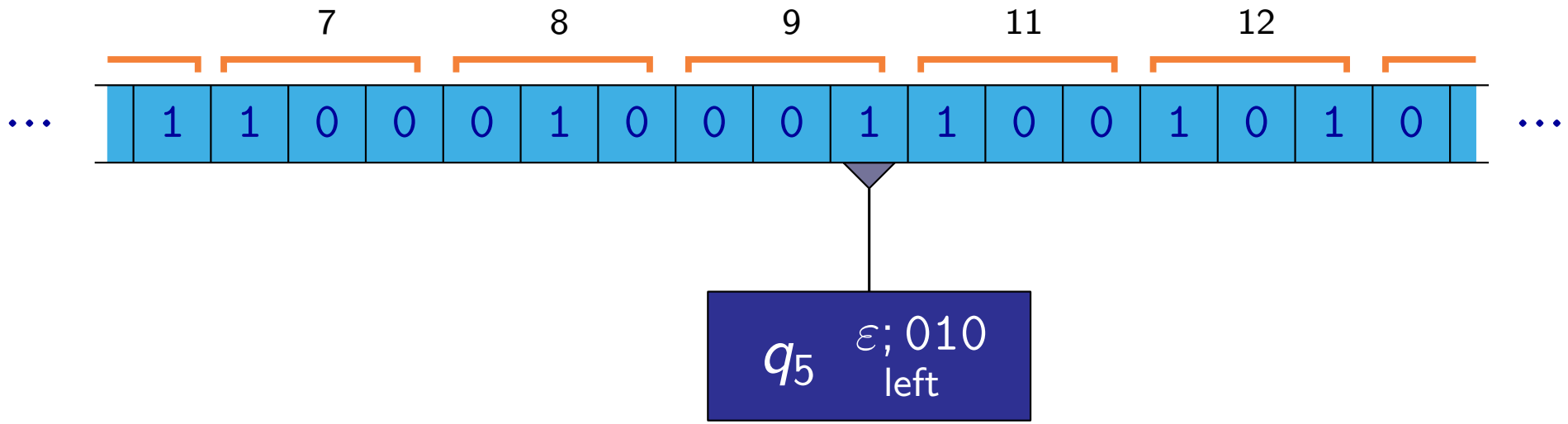
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



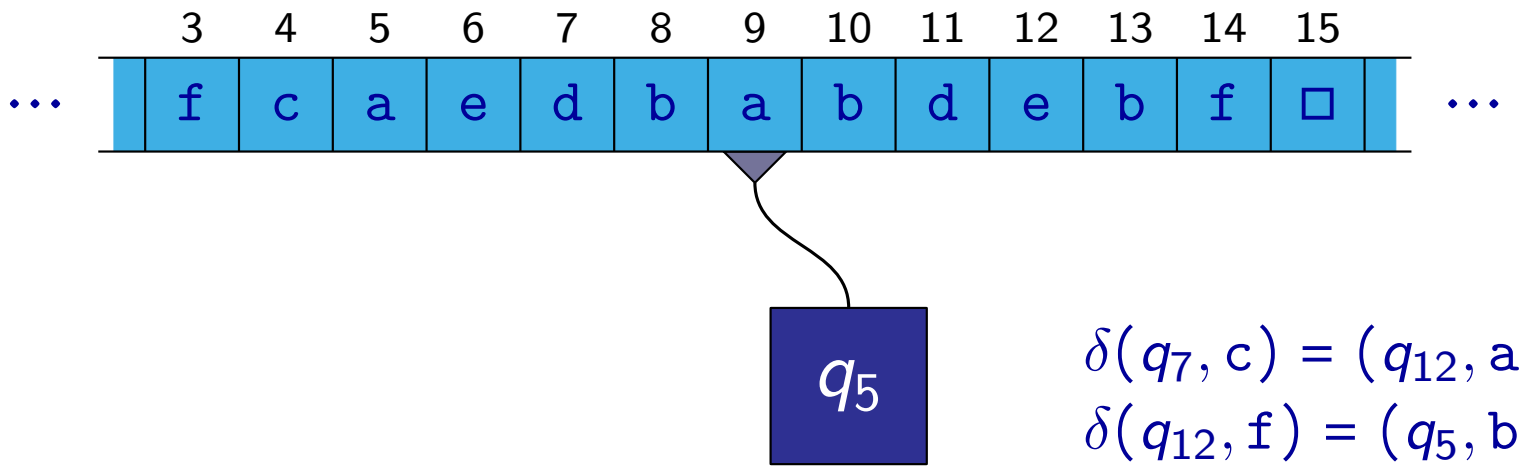
$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



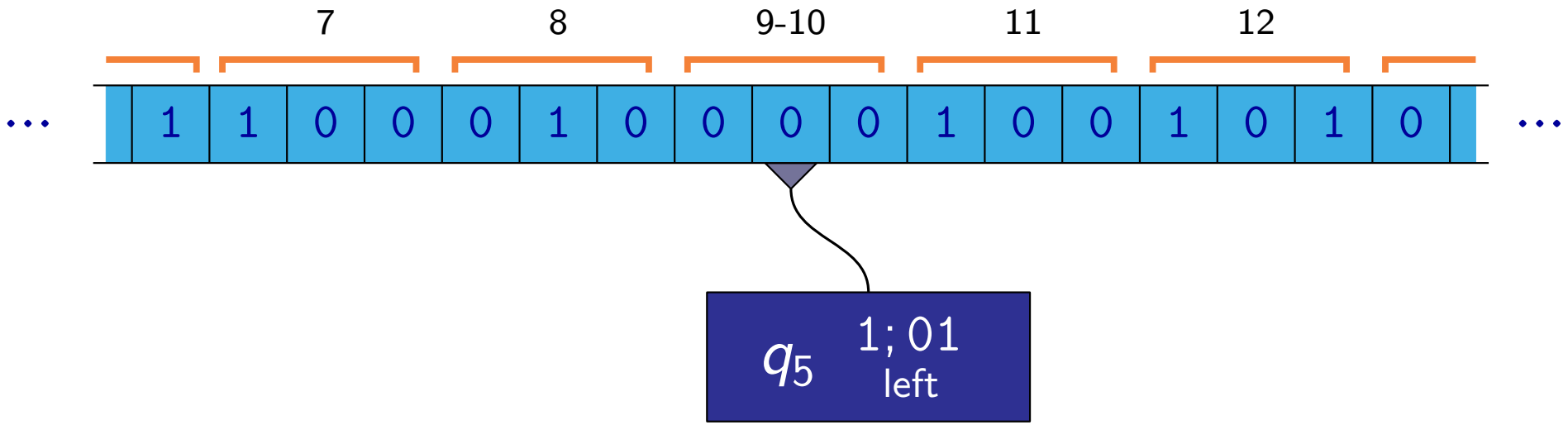
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



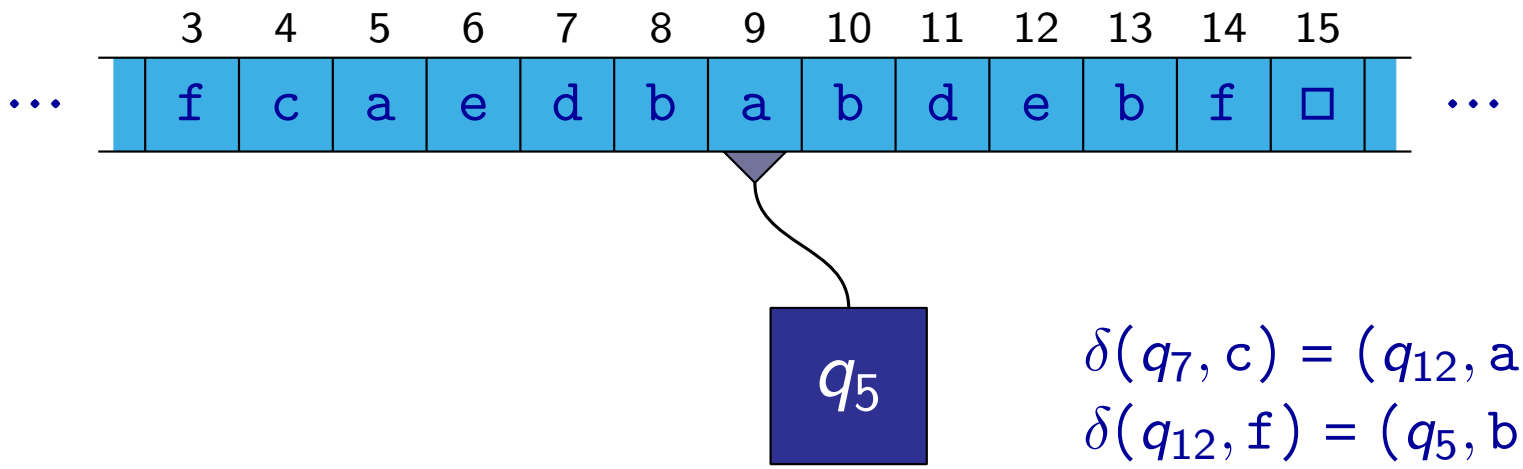
$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



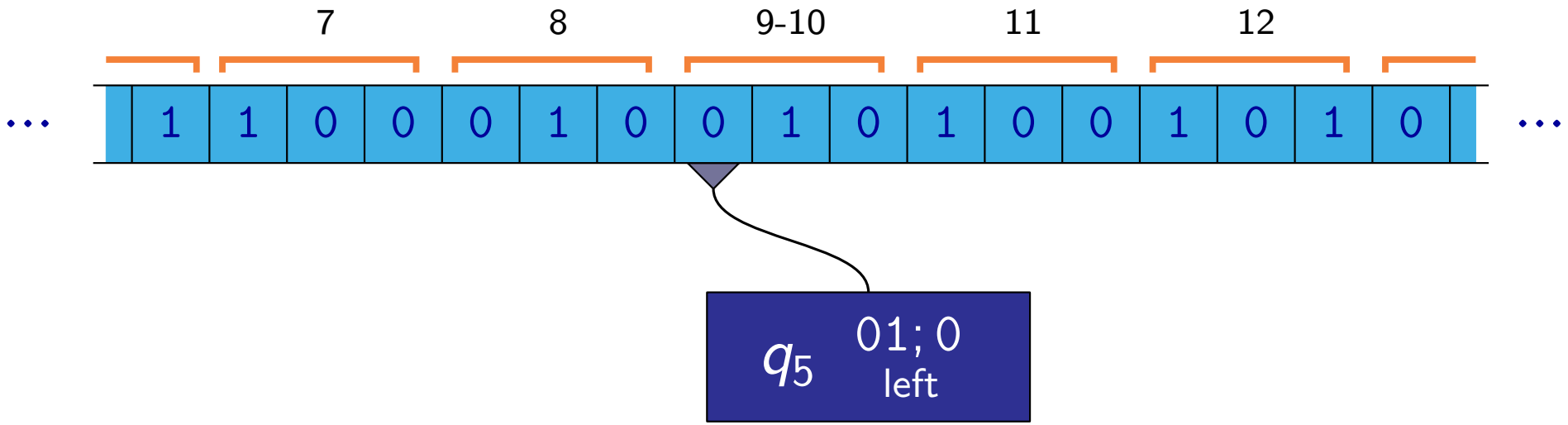
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



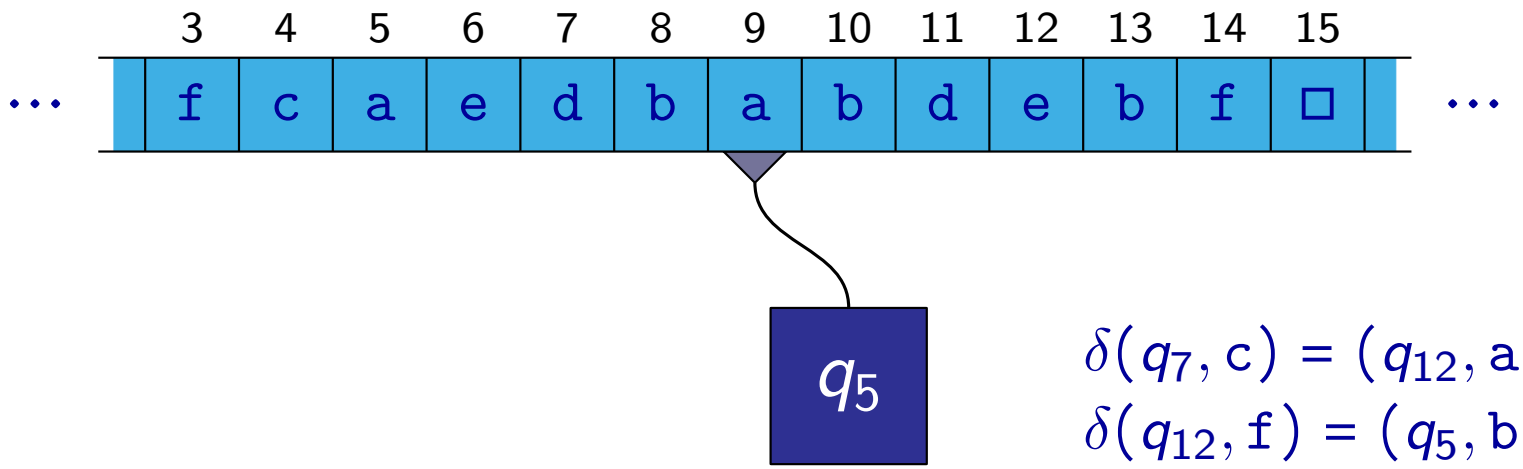
$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



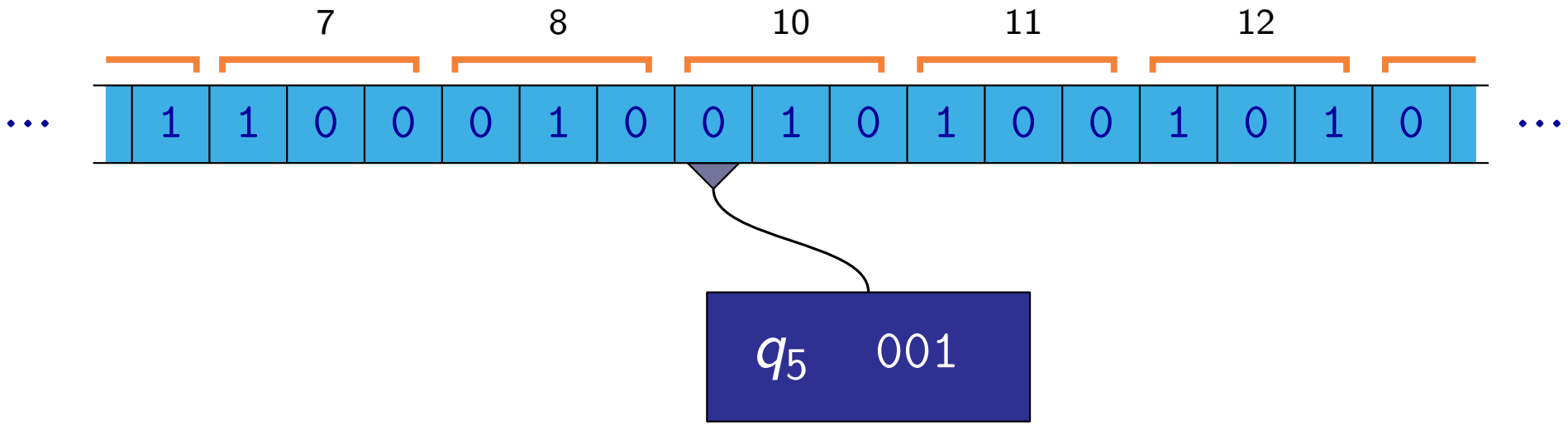
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



Abeceda $\{0, 1\}$

Při výše uvedené simulaci je jeden krok původního stroje simulován $k + 1$ kroky, kde k je počet bitů kódující jeden symbol abecedy Γ.

Pokud tedy původní stroj provede během výpočtu t kroků, simulující stroj provede $O(t)$ kroků.

Zmenšení počtu stavů řídící jednotky

Poznámka: Tak, jako je možné zmenšit páskovou abecedu na pouhé dva symboly za cenu nárůstu velikosti počtu stavů řídící jednotky, je rovněž možné snížit počet stavů řídící jednotky:

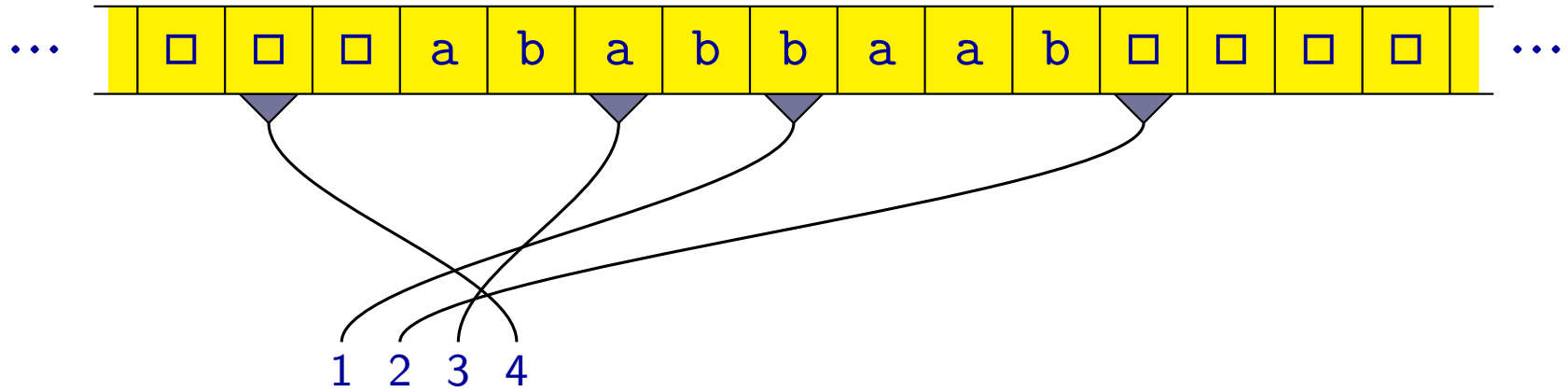
- Činnost libovolného Turingova stroje je možné simulovat Turingovým strojem, který má pouze dva nekoncové stavy řídící jednotky (a případně nějaké koncové stavy), ovšem za cenu nárůstu velikosti páskové abecedy.

Podobně jako v předchozím případě je jeden krok původního stroje simulován s kroky, kde s je konstanta závisející pouze na počtu stavů řídící jednotky původního stroje (tj. na velikosti množiny Q).

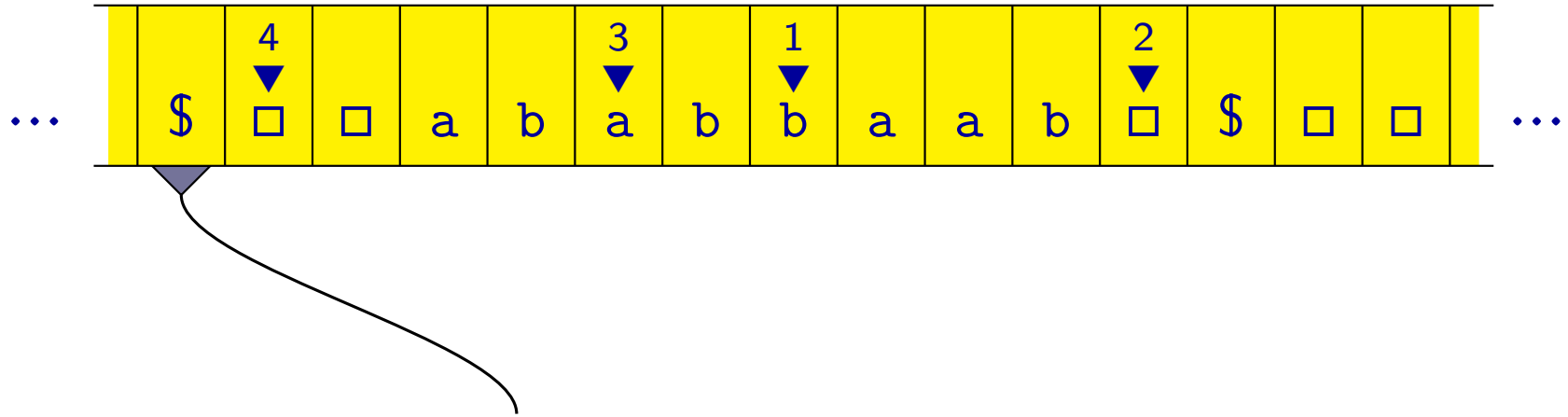
Opět zde tedy platí, že pokud původní stroj provede během výpočtu t kroků, simulující stroj provede $O(t)$ kroků.

Simulace více hlav na pásce pomocí jedné

Více hlav na pásce:

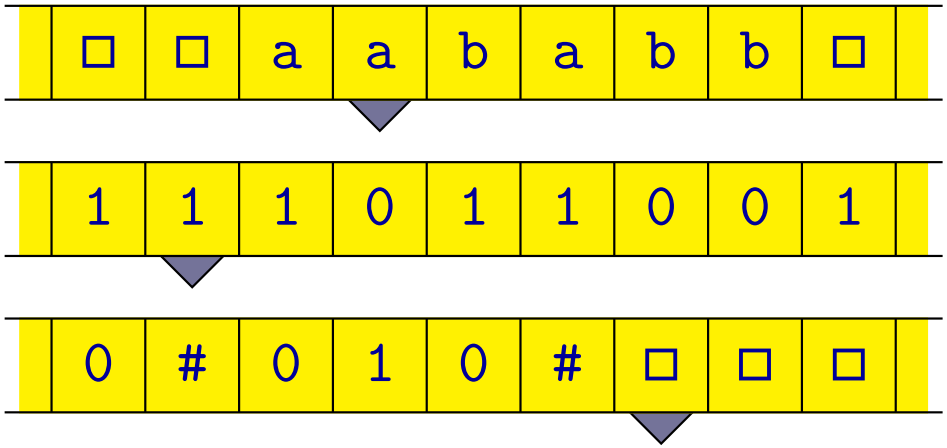


Páska s jednou hlavou:

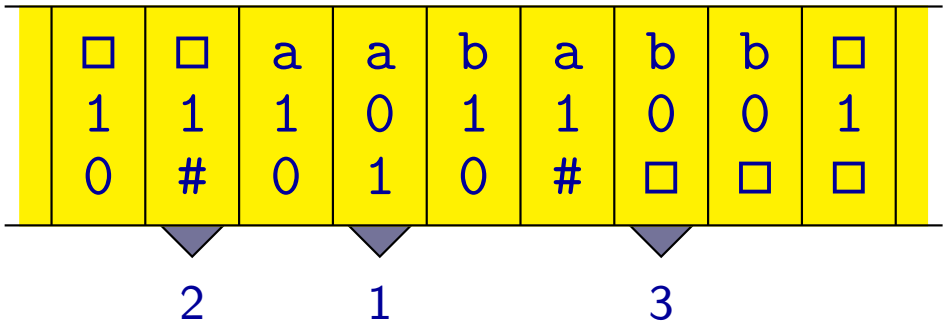


Simulace více pásek pomocí jedné

Více pásek:

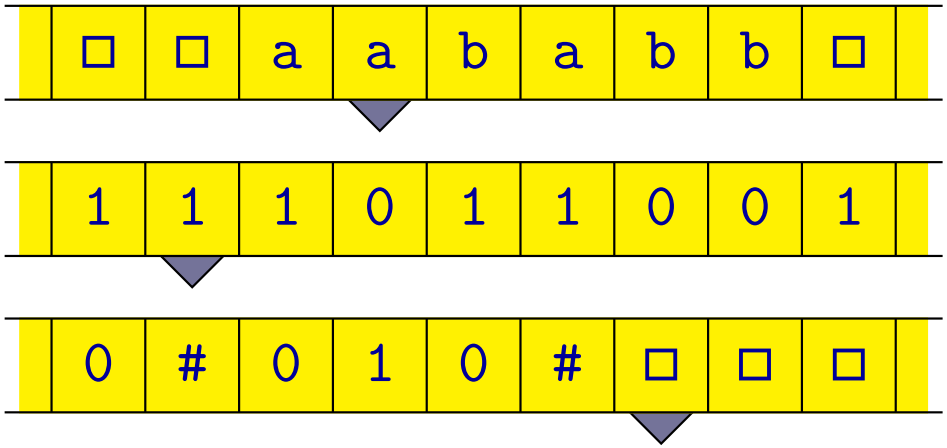


Jedna páska s více hlavami:

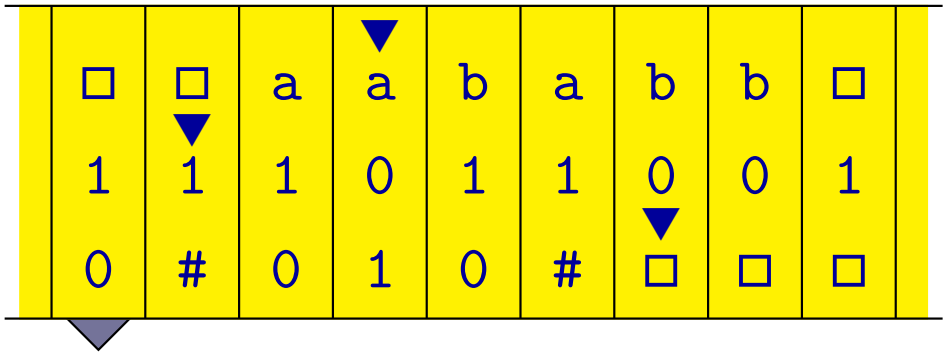


Simulace více pásek pomocí jedné

Více pásek:

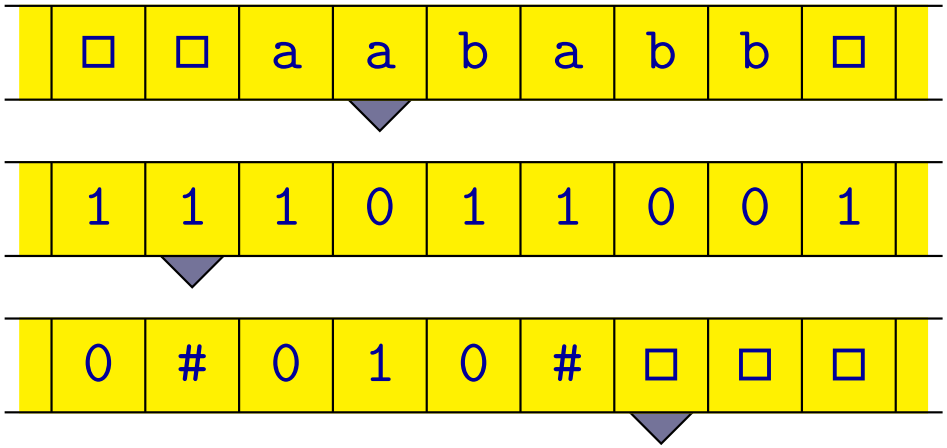


Jedna páska s jednou hlavou: varianta, kde se posunují značky hlav

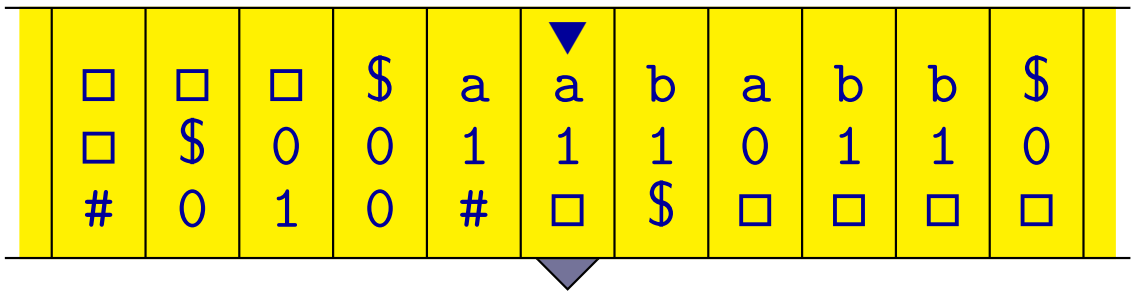


Simulace více pásek pomocí jedné

Více pásek:



Jedna páska s jednou hlavou: varianta, kde se posunují obsahy pásek



Pásky, zásobníky a čítače

Můžeme uvažovat různé stroje, které mají konečnou řídící jednotku doplněnou o nějaký druh neomezeně velké paměti.

Tato paměť může být tvořena jednou nebo více strukturami, jako jsou třeba:

- **Páska** — čtení a zápis symbolu na aktuální pozici, posun hlavy doleva a doprava

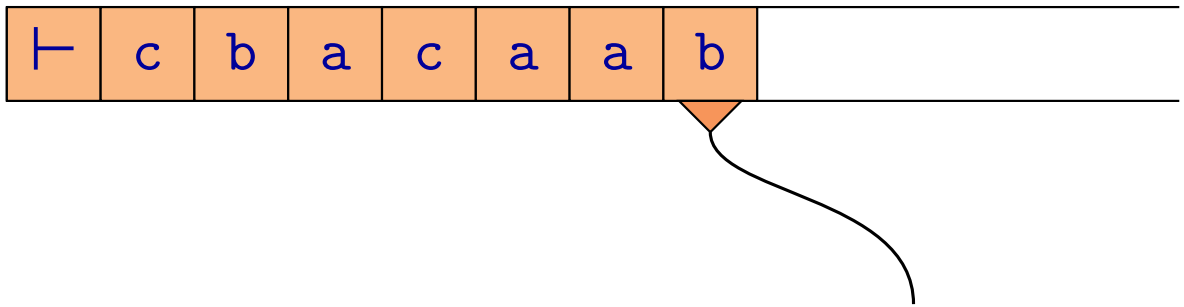
  Poznámka: Páska může být jednostranně nebo oboustranně nekonečná.

- **Zásobník** — push, pop, test prázdnosti zásobníku
- **Čítač** — hodnotou je přirozené číslo, operace přičtení nebo odečtení hodnoty jedna, test rovnosti nule

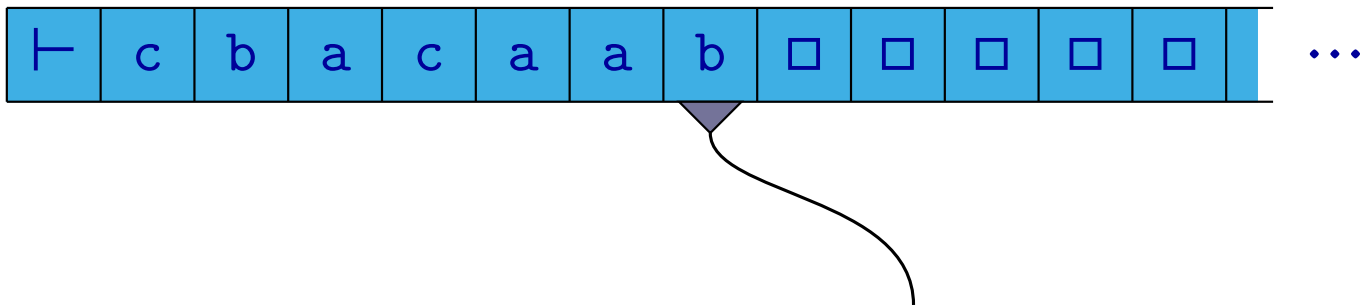
Zásobník

Na zásobník je možné se dívat jako na speciální případ jednostranně nekonečné pásky.

Zásobník:



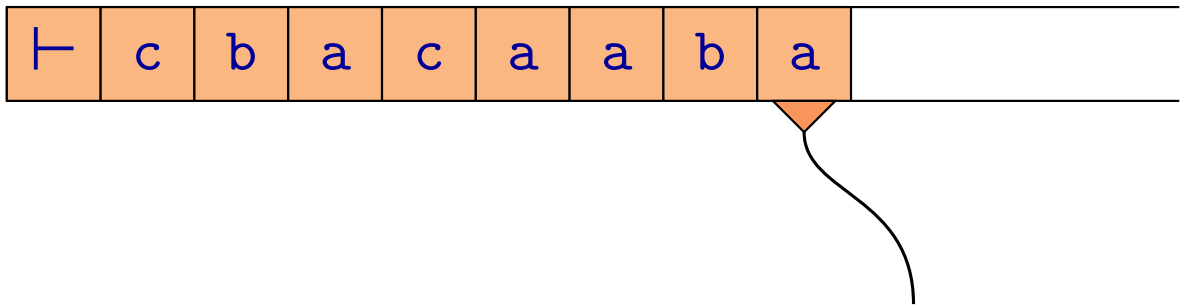
Páska:



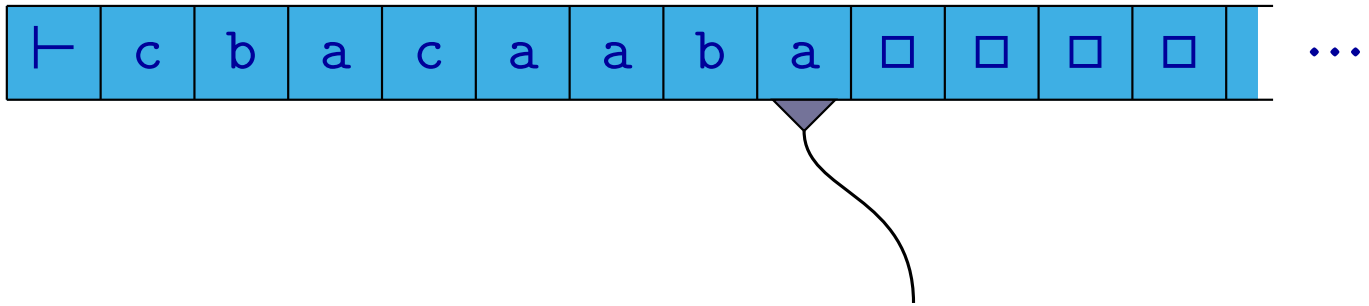
Zásobník

Na zásobník je možné se dívat jako na speciální případ jednostranně nekonečné pásky.

Zásobník:



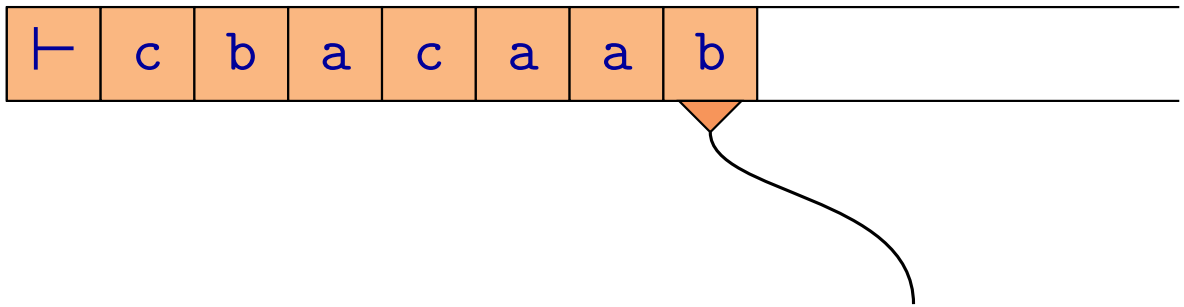
Páska:



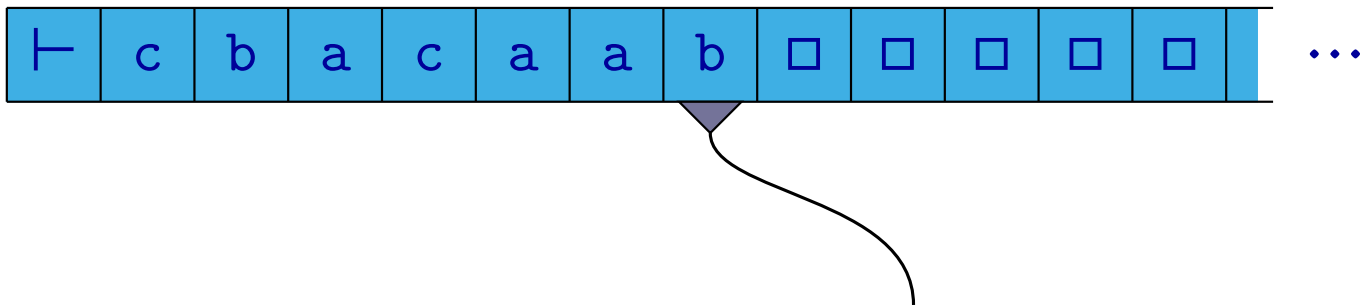
Zásobník

Na zásobník je možné se dívat jako na speciální případ jednostranně nekonečné pásky.

Zásobník:



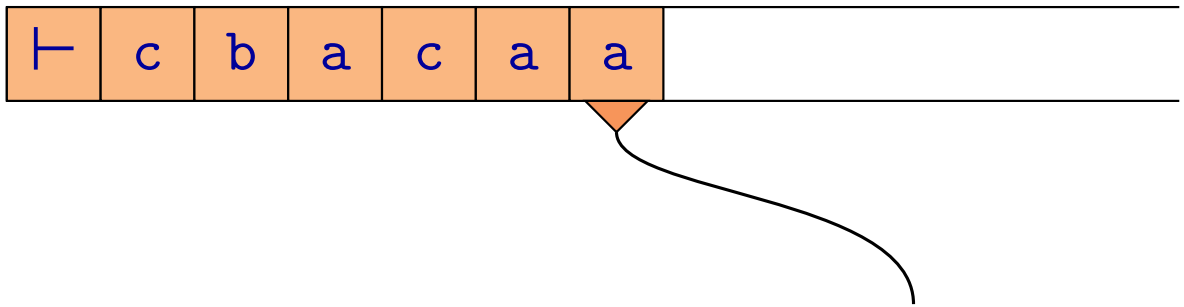
Páska:



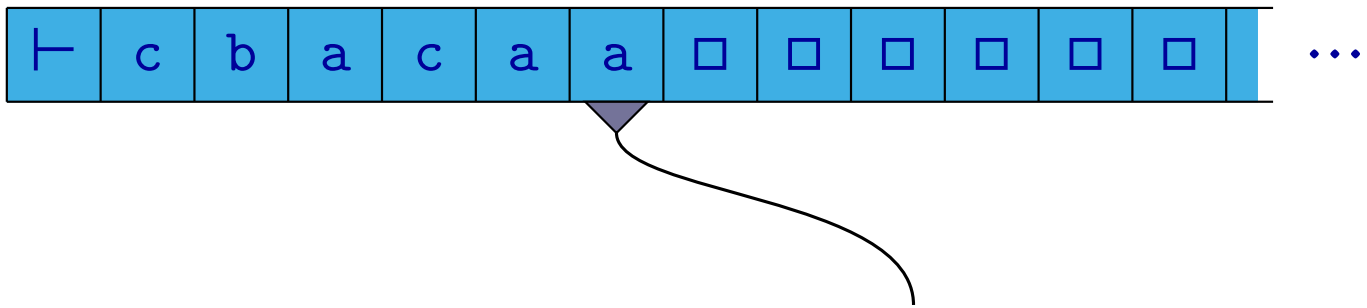
Zásobník

Na zásobník je možné se dívat jako na speciální případ jednostranně nekonečné pásky.

Zásobník:



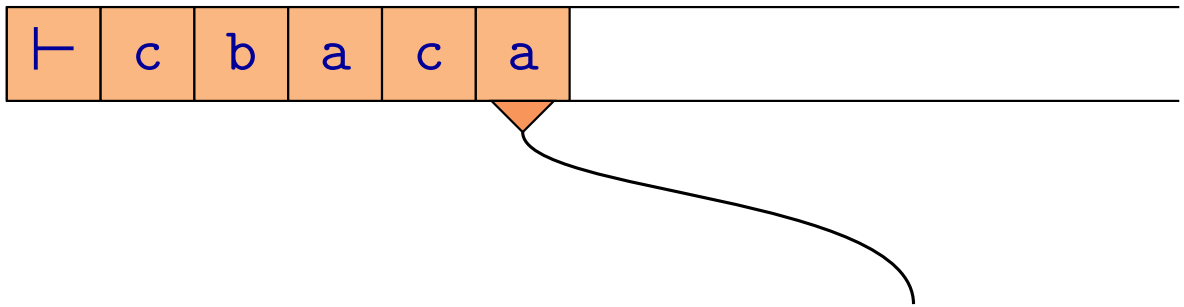
Páska:



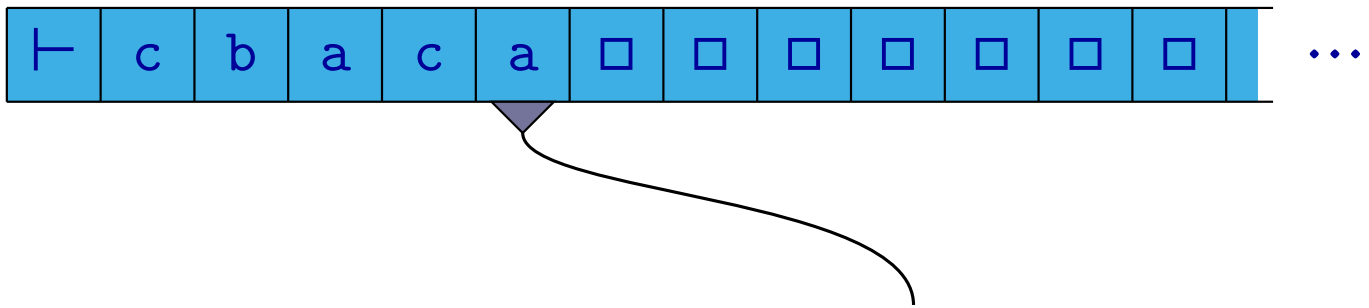
Zásobník

Na zásobník je možné se dívat jako na speciální případ jednostranně nekonečné pásky.

Zásobník:

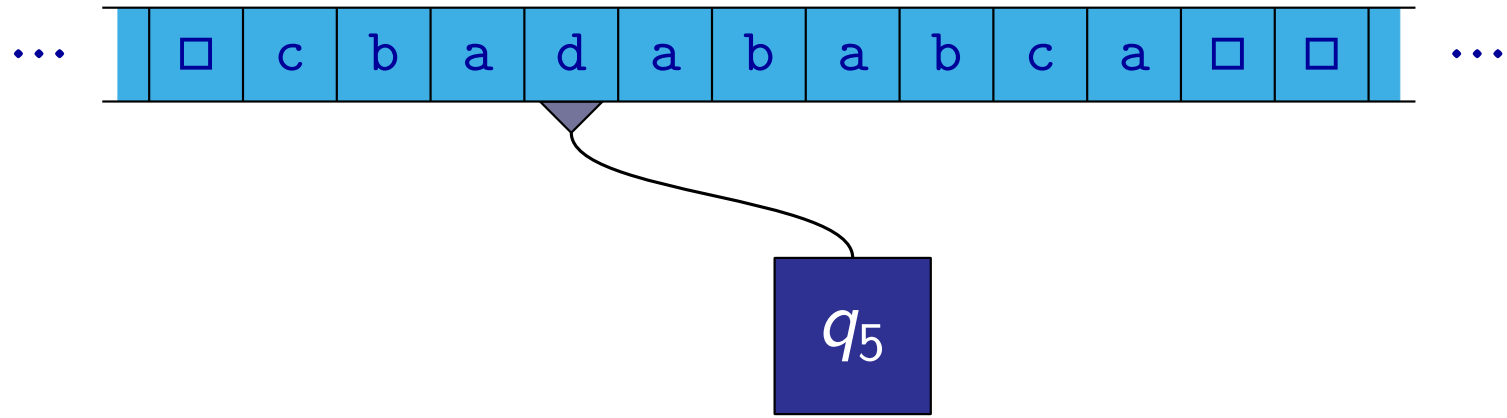


Páska:

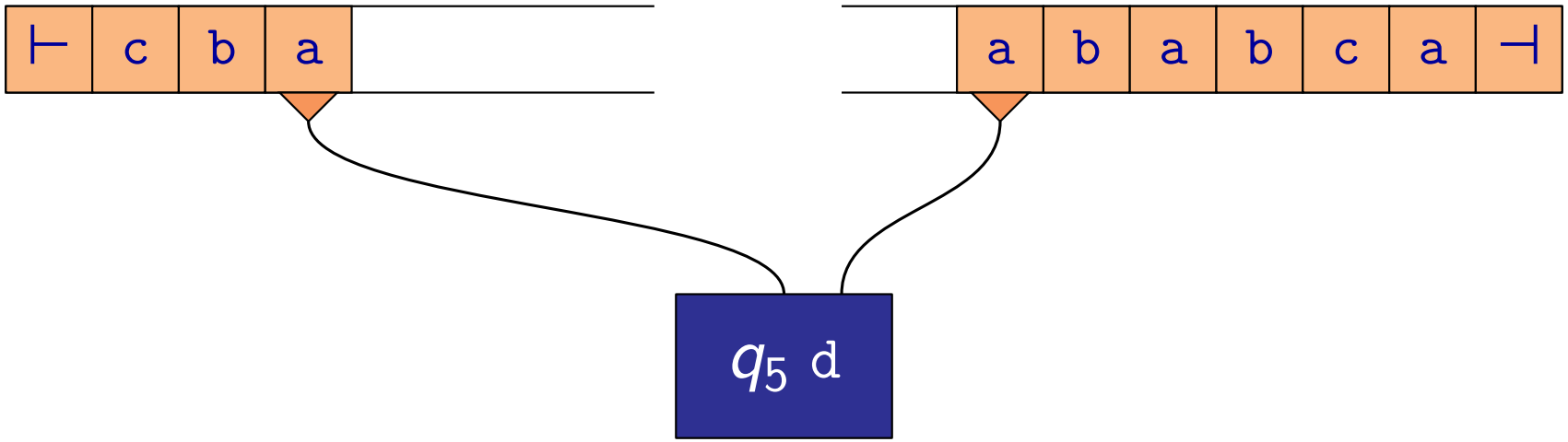


Zásobník

Oboustranně nekonečnou pásku je možné simulovat pomocí dvou zásobníků:

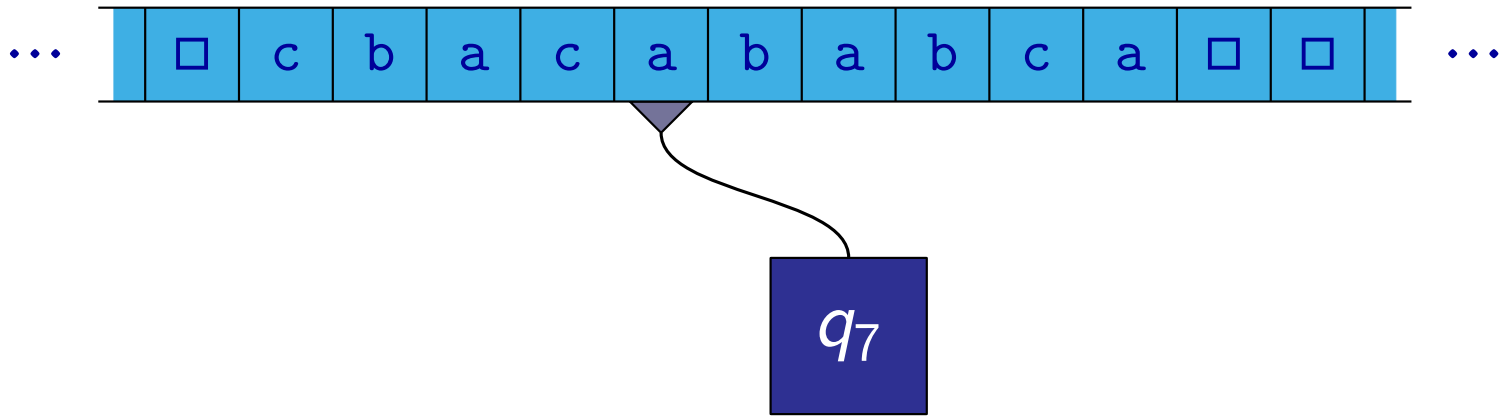


Stroj se dvěma zásobníky:

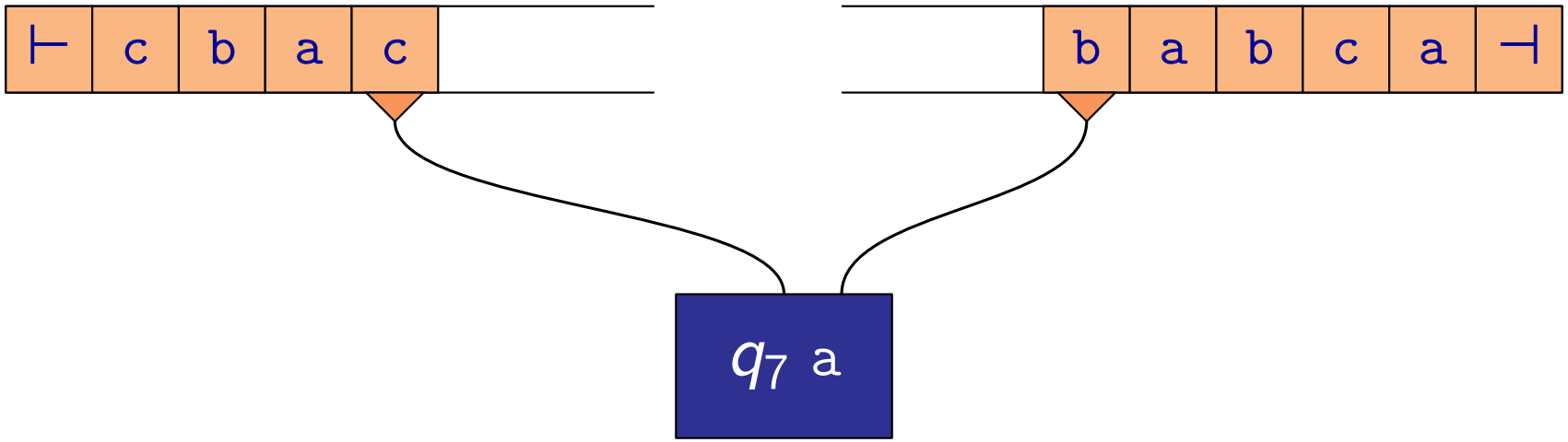


Zásobník

Oboustranně nekonečnou pásku je možné simulovat pomocí dvou zásobníků:



Stroj se dvěma zásobníky:



Čítač

Čítač — hodnotou čítače může být libovolně velké přirozené číslo, tj. prvek množiny $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.

Základní operace:

- zvýšení hodnoty o jedna:

$$x := x + 1$$

- snížení hodnoty o jedna:

$$x := x - 1$$

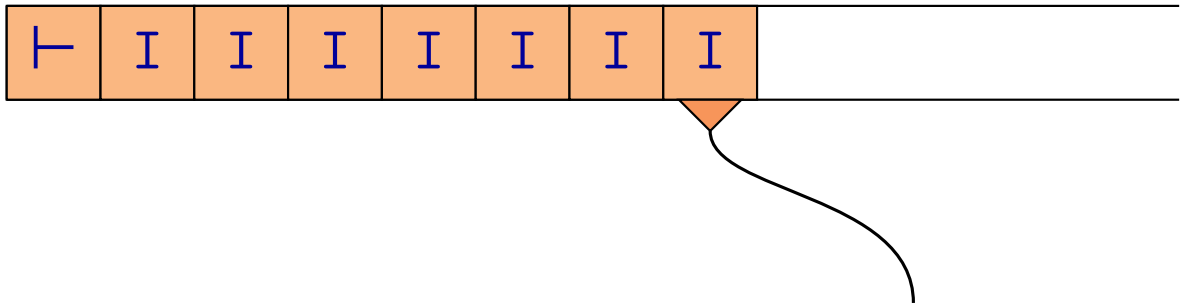
- test, jestli je hodnota čítače nula:

$$\textbf{if } (x = 0) \textbf{ goto } \ell$$

Čítač

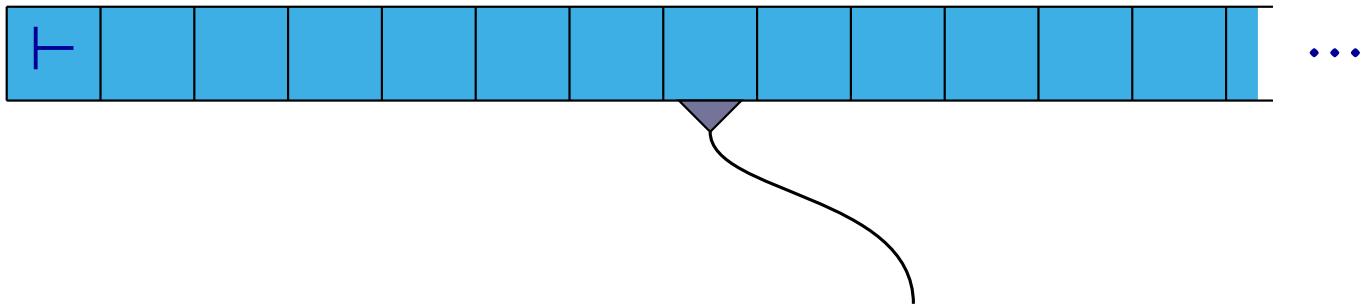
Na čítač je možné se dívat jako na speciální případ zásobníku či pásky.

Zásobník:



Čítač:

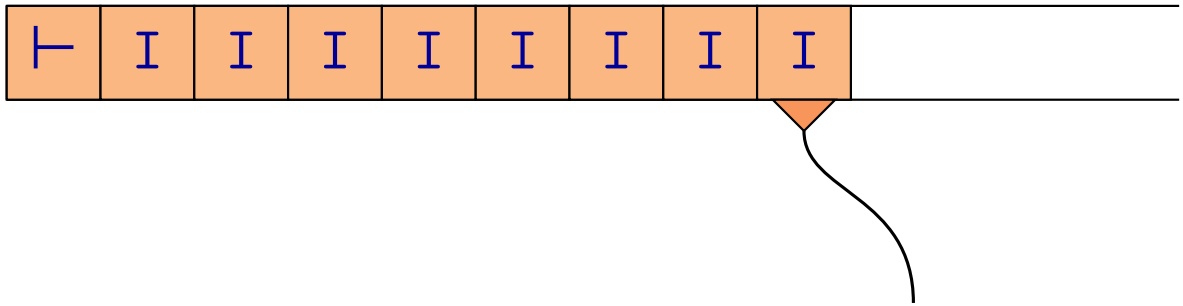
Páska:



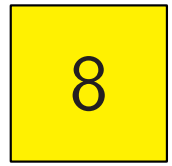
Čítač

Na čítač je možné se dívat jako na speciální případ zásobníku či pásky.

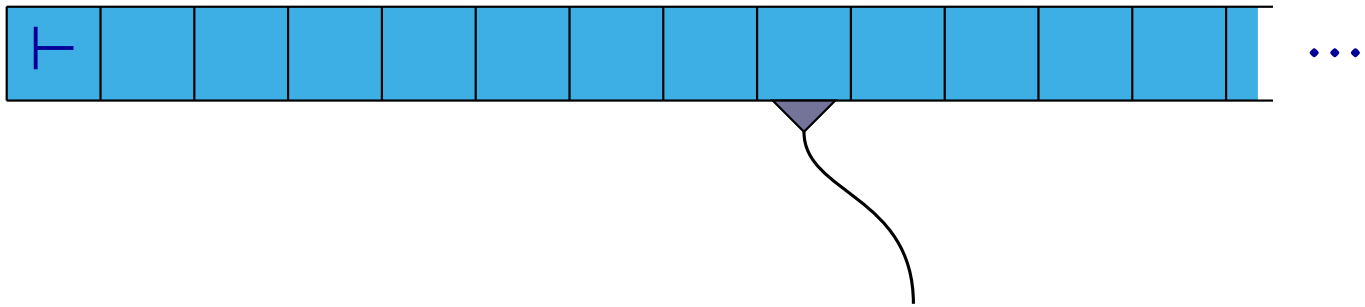
Zásobník:



Čítač:



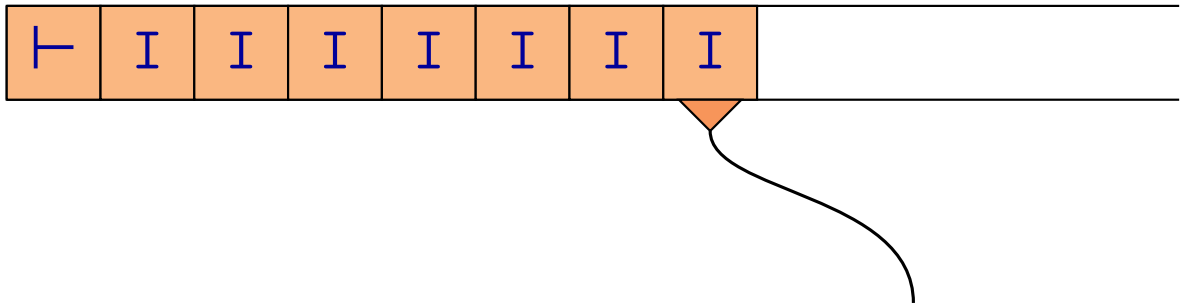
Páska:



Čítač

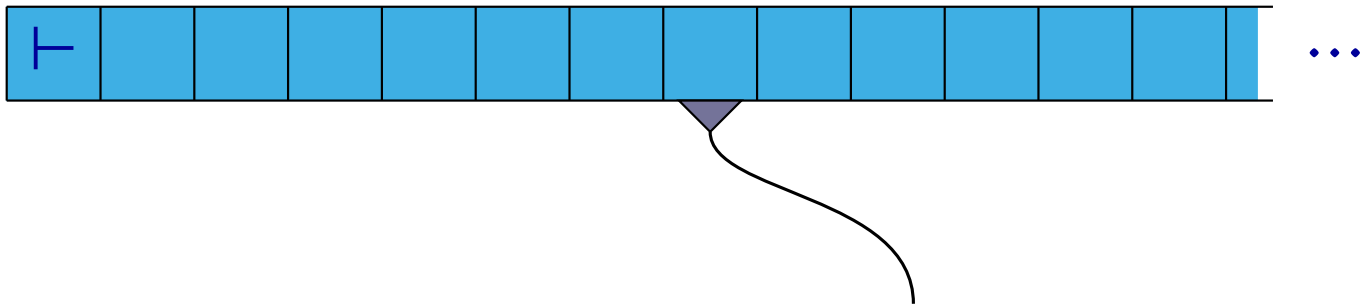
Na čítač je možné se dívat jako na speciální případ zásobníku či pásky.

Zásobník:



Čítač:

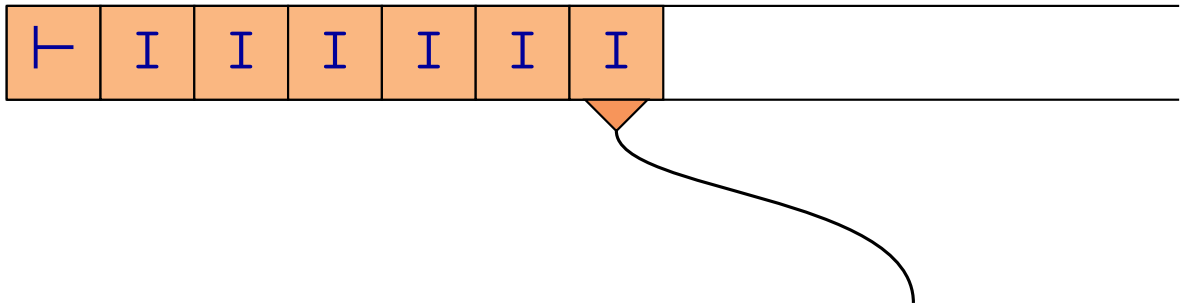
Páska:



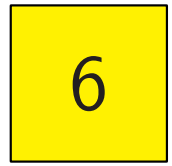
Čítač

Na čítač je možné se dívat jako na speciální případ zásobníku či pásky.

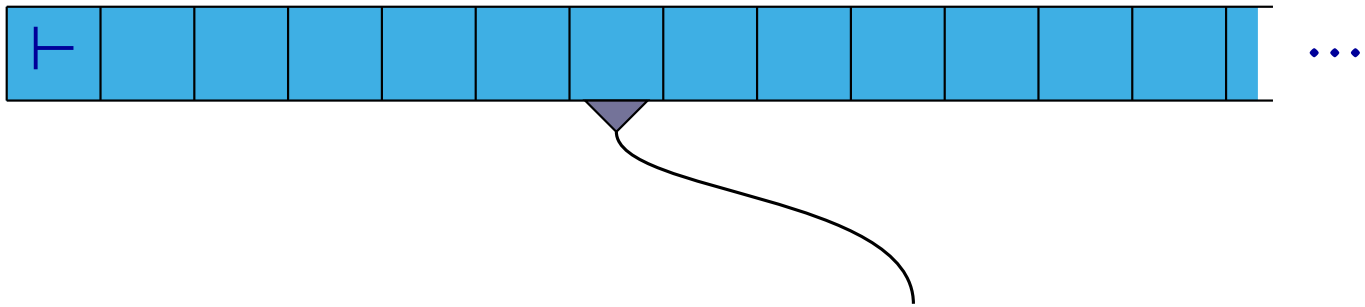
Zásobník:



Čítač:



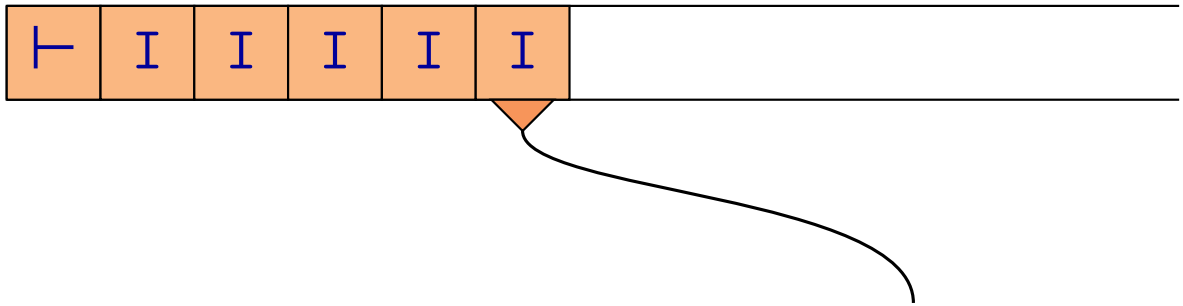
Páska:



Čítač

Na čítač je možné se dívat jako na speciální případ zásobníku či pásky.

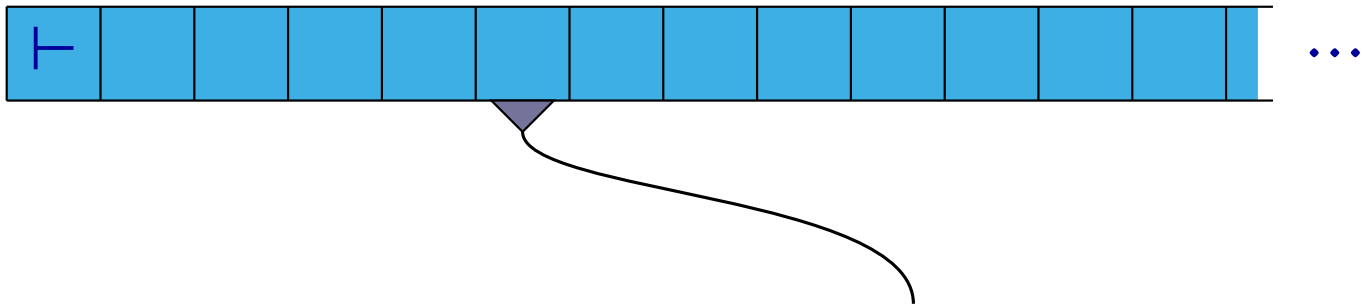
Zásobník:



Čítač:

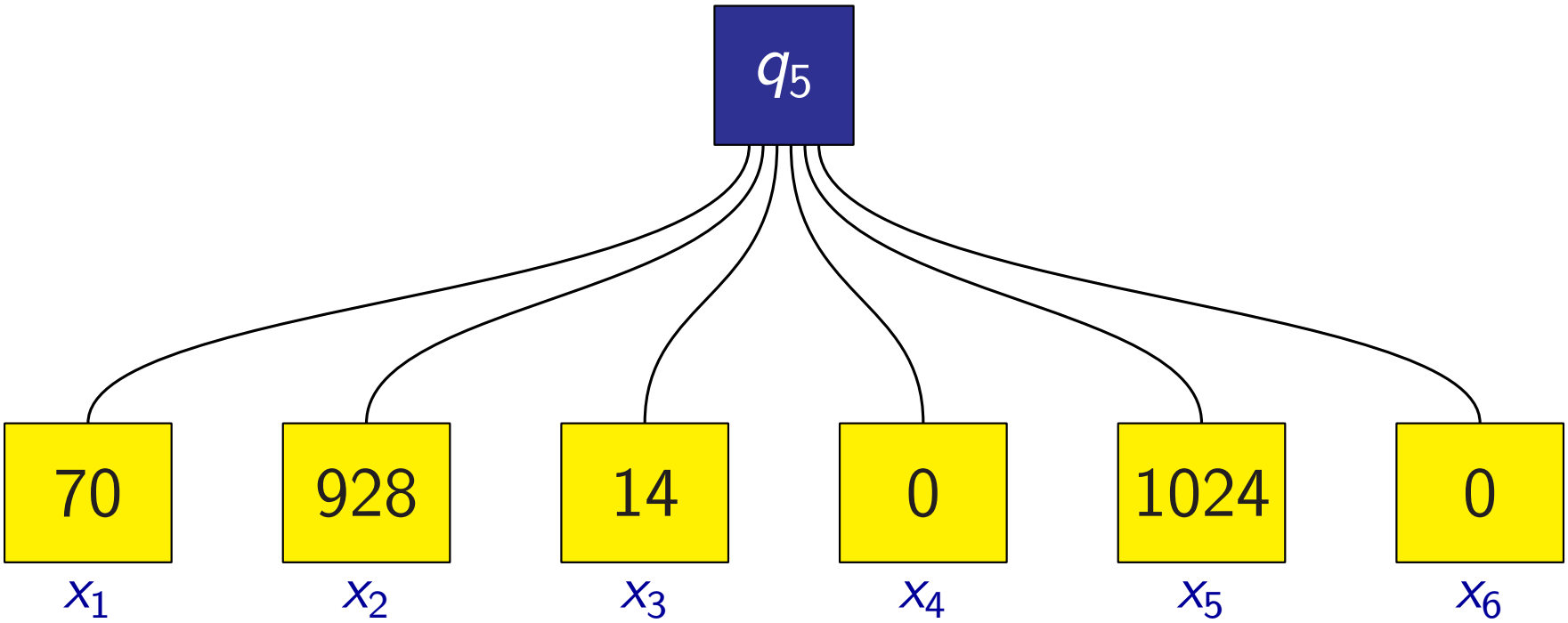
5

Páska:



Minského stroj

Minského stroj — stroj, který má konečnou řídící jednotku a konečný počet čítačů $x_1, x_2, \ldots, x_k$:



Poznámka: Pro označení čítačů budeme kromě symbolů $x_1, x_2, \ldots$ používat také symboly jako $x, y, z, \ldots$

Minského stroj

Na Minského stroj se můžeme dívat jako na program tvořený posloupností instrukcí následujících pěti typů:

- zvýšení hodnoty daného čítače o jedna:

$$x_i := x_i + 1$$

- snížení hodnoty daného čítače o jedna:

$$x_i := x_i - 1$$

- test, jestli je hodnota daného čítače nula:

$$\textbf{if } (x_i = 0) \textbf{ goto } \ell$$

- nepodmíněný skok:

$$\textbf{goto } \ell$$

- zastavení programu:

$$\textbf{halt}$$

Minského stroj

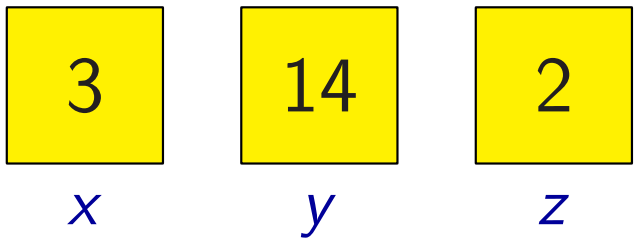
Vynulování čítače x:

L_1 : **if** $(x = 0)$ **goto** L_2

$x := x - 1$

goto L_1

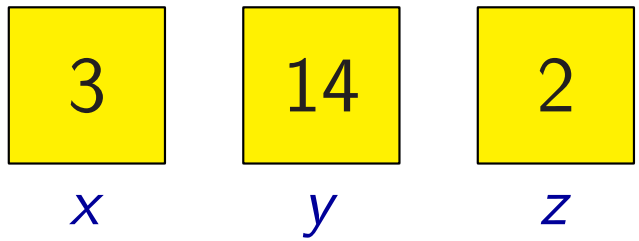
L_2 : ...



Minského stroj

Vynulování čítače x:

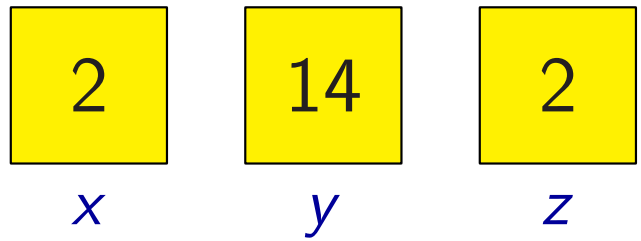
L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : ...



Minského stroj

Vynulování čítače x:

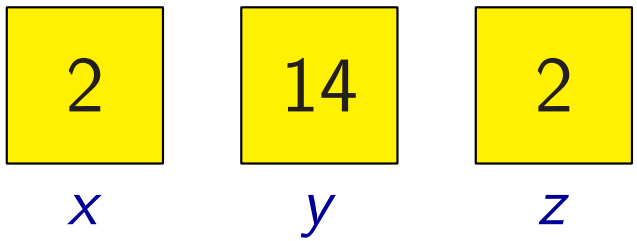
L_1 : **if** $(x = 0)$ **goto** L_2
$\quad x := x - 1$
$\quad$ **goto** L_1
L_2 : $\quad \ldots$



Minského stroj

Vynulování čítače x:

L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : ...



Minského stroj

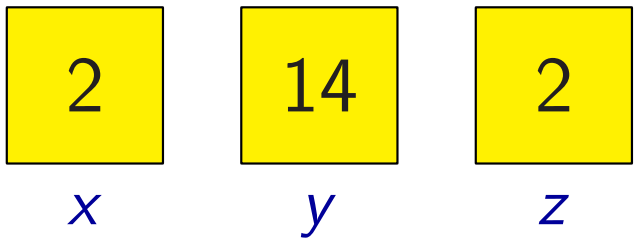
Vynulování čítače x:

L_1 : **if** $(x = 0)$ **goto** L_2

→ $x := x - 1$

goto L_1

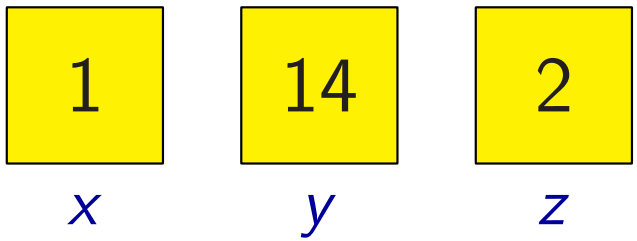
L_2 : ...



Minského stroj

Vynulování čítače x:

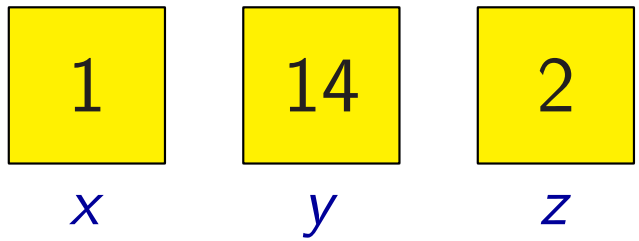
L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : ...



Minského stroj

Vynulování čítače x:

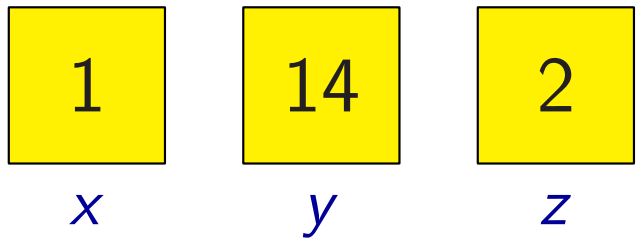
L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : ...



Minského stroj

Vynulování čítače x:

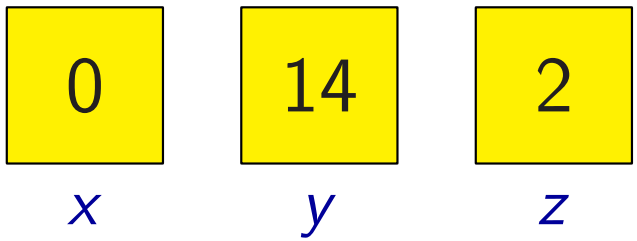
L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : ...



Minského stroj

Vynulování čítače x:

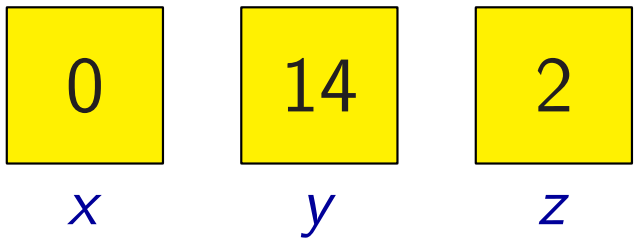
L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : ...



Minského stroj

Vynulování čítače x:

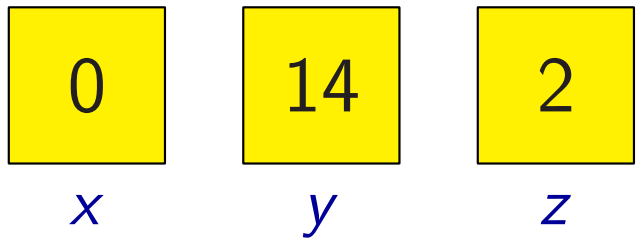
L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : ...



Minského stroj

Vynulování čítače x:

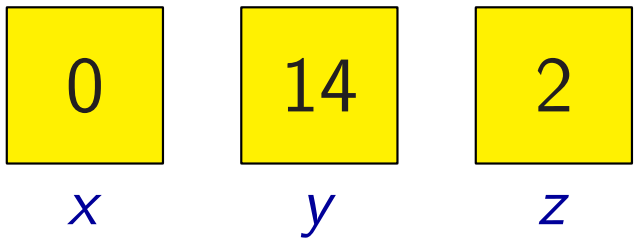
L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

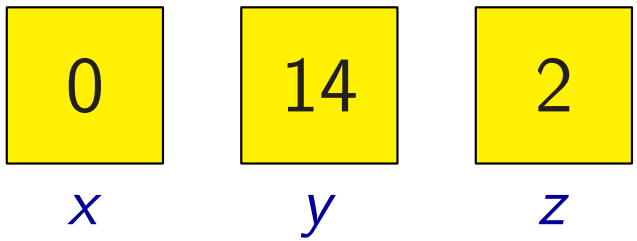
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

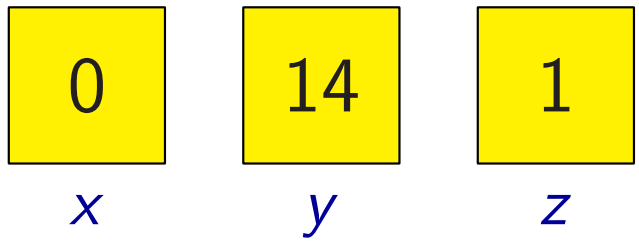
L_2 : **if** $(z = 0)$ **goto** L_3
→ $z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

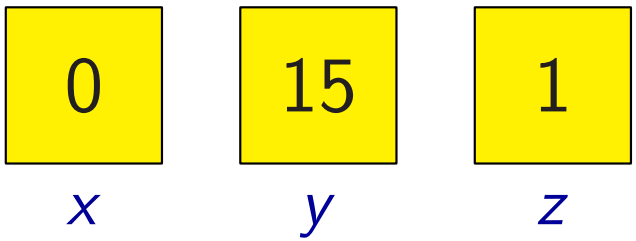
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

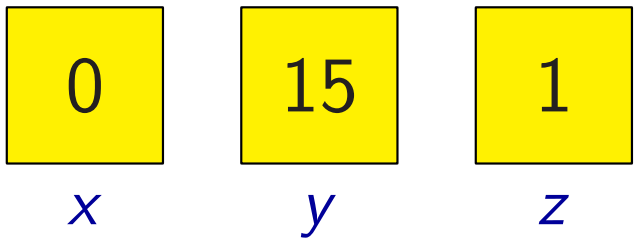
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

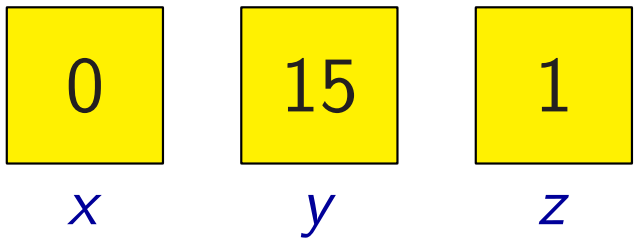
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

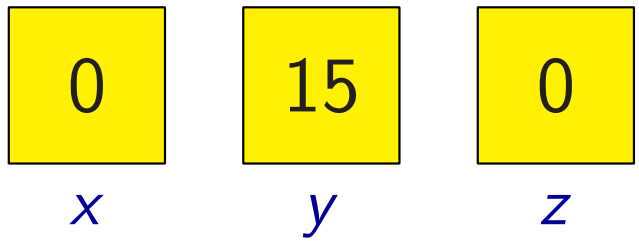
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

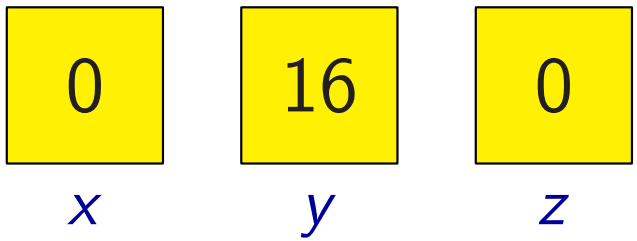
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

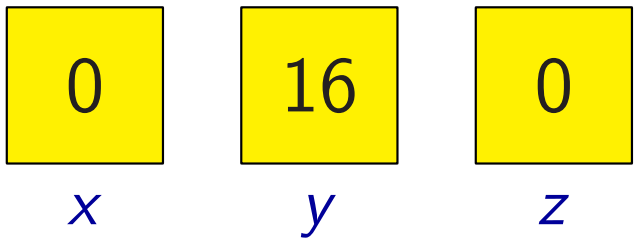
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

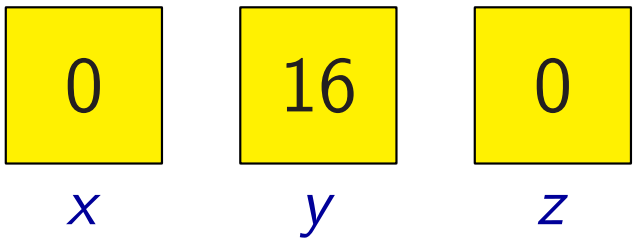
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Vynásobení hodnoty čítače x číslem 5:

$$
\begin{aligned}
L_1 &: \textbf{if } (x = 0) \textbf{ goto } L_2 \\
&\quad x := x - 1 \\
&\quad y := y + 1 \\
&\quad y := y + 1 \\
&\quad y := y + 1 \\
&\quad y := y + 1 \\
&\quad y := y + 1 \\
&\quad \textbf{goto } L_1 \\
L_2 &: \textbf{if } (y = 0) \textbf{ goto } L_3 \\
&\quad y := y - 1 \\
&\quad x := x + 1 \\
&\quad \textbf{goto } L_2 \\
L_3 &: \quad \ldots
\end{aligned}
$$

Minského stroj

Vydělení hodnoty čítače x číslem 5 a zjištění zbytku po dělení:

$$
\begin{aligned}
L_1 : \ & \textbf{if } (x = 0) \textbf{ goto } M_0 \\
& x := x - 1 \\
& \textbf{if } (x = 0) \textbf{ goto } M_1 \\
& x := x - 1 \\
& \textbf{if } (x = 0) \textbf{ goto } M_2 \\
& x := x - 1 \\
& \textbf{if } (x = 0) \textbf{ goto } M_3 \\
& x := x - 1 \\
& \textbf{if } (x = 0) \textbf{ goto } M_4 \\
& x := x - 1 \\
& y := y + 1 \\
& \textbf{goto } L_1
\end{aligned}
$$

Minského stroj

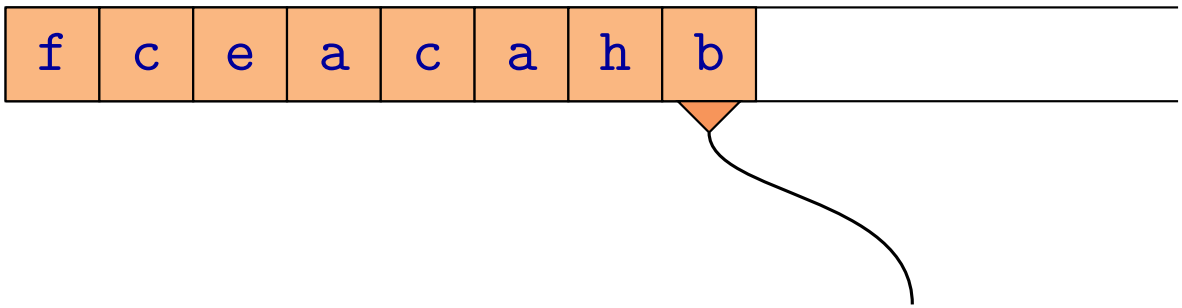
Zásobník je možné simulovat pomocí dvou čítačů — hodnota jednoho čítače reprezentuje obsah zásobníku jako číslo, jehož zápis v číselné soustavě o základu $k = |\Gamma| + 1$ (kde Γ je zásobníková abeceda) odpovídá obsahu zásobníku.

- Symbol na vrcholu zásobníku — zbytek po dělení číslem k
- Pop — vydělit číslem k
- Push — vynásobit číslem k a přičíst kód příslušného symbolu

Druhý čítač slouží jako pomocný při provádění výše uvedených operací.

Příklad:

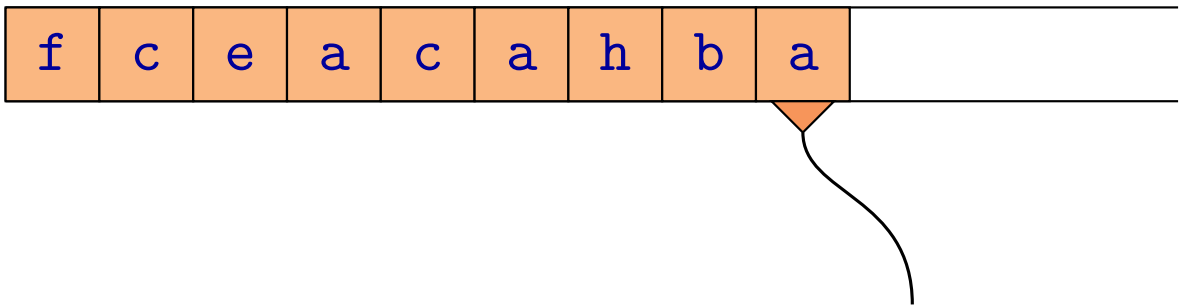
a ↔ 1
b ↔ 2
c ↔ 3
d ↔ 4
e ↔ 5
f ↔ 6
g ↔ 7
h ↔ 8
i ↔ 9



63513182

Příklad:

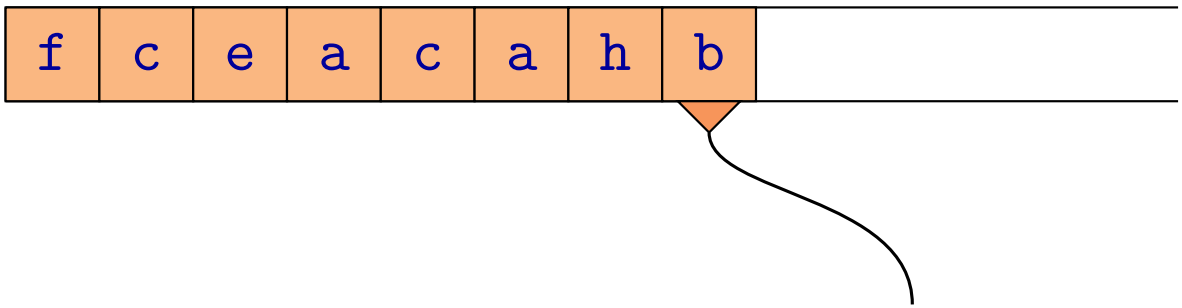
a ↔ 1
b ↔ 2
c ↔ 3
d ↔ 4
e ↔ 5
f ↔ 6
g ↔ 7
h ↔ 8
i ↔ 9



635131821

Příklad:

a ↔ 1
b ↔ 2
c ↔ 3
d ↔ 4
e ↔ 5
f ↔ 6
g ↔ 7
h ↔ 8
i ↔ 9

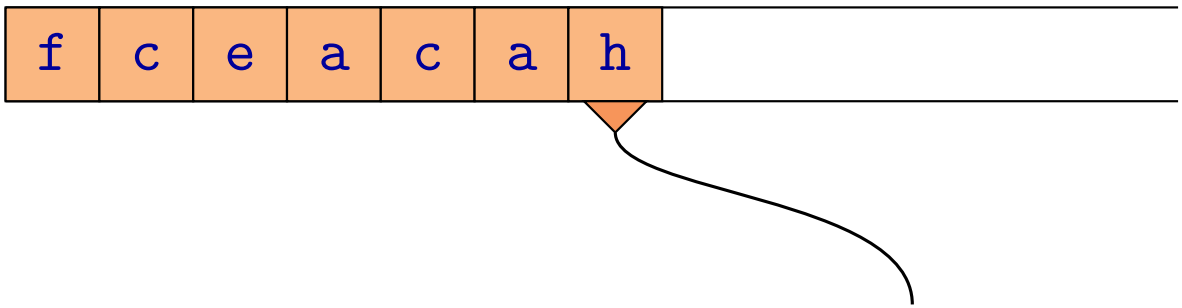


63513182

Minského stroj

Příklad:

a ↔ 1
b ↔ 2
c ↔ 3
d ↔ 4
e ↔ 5
f ↔ 6
g ↔ 7
h ↔ 8
i ↔ 9

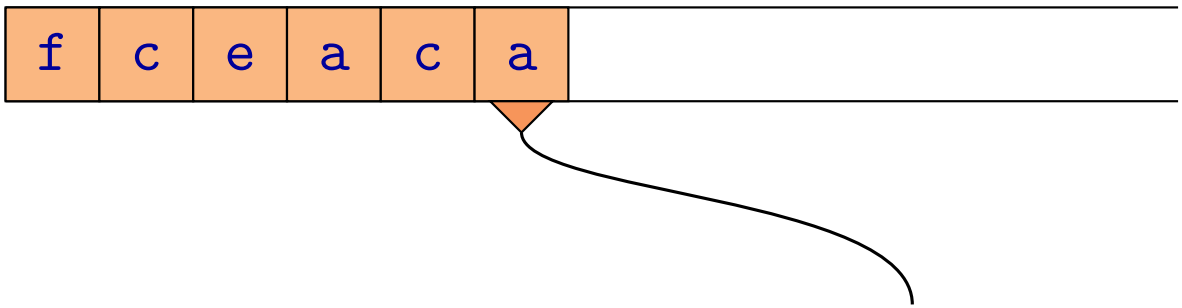


6351318

Minského stroj

Příklad:

a ↔ 1
b ↔ 2
c ↔ 3
d ↔ 4
e ↔ 5
f ↔ 6
g ↔ 7
h ↔ 8
i ↔ 9



635131

Minského stroj

Připomeňme, že oboustranně nekonečnou pásku je možné simulovat pomocí dvou zásobníků.

V Minského stroji může být obsah každého z těchto zásobníků reprezentován jemu odpovídajícím čítačem.

Navíc potřebujeme ještě jeden pomocný čítač pro implementaci operací násobení a dělení na těchto čítačích reprezentujících obsahy zásobníků.

Vidíme, že Turingův stroj s k páskami je možné simulovat Minského strojem s $2k + 1$ čítači.

Minského stroj

Libovolný konečný počet čítačů je možné simulovat pomocí dvou čítačů.

- Jeden čítač (označme jej C) reprezentuje hodnoty všech čítačů — např. hodnoty tří čítačů x, y, z mohou být v čítači C reprezentovány jako číslo $2^x 3^y 5^z$.
- Druhý čítač je používán jako pomocný při provádění operací násobení a dělení na čítači C.
- Přičtení jedničky k čítači x je simulováno jako vynásobení čítače C hodnotou 2, přičtení jedničky k čítači y jako vynásobení hodnotou 3, atd.
- Analogicky je odečtení jedničky od čítače x simulováno pomocí vydělení čítače C hodnotou 2, odečtení jedničky od čítače y vydělením hodnotou 3, atd.
- Test podmínky $x = 0$ odpovídá testu, zda je hodnota C dělitelná dvěmi, atd.

Minského stroj

Vidíme, že činnost libovolného Turingova stroje je možné simulovat Minského strojem s dvěma čítači.

Tato simulace je však mimořádně neefektivní:

- Už simulace pásky Turingova stroje pomocí tří čítačů vyžaduje exponenciálně větší počet kroků, než kolik by jich vykonal tento Turingův stroj.
- Simulace činnosti těchto tří čítačů pomocí dvou čítačů tento počet kroků dále exponenciálně zvyšuje.

Stroje RAM

Stroj RAM

Stroj RAM (Random Access Machine) je idealizovaný model počítače.

Skládá se z těchto částí:

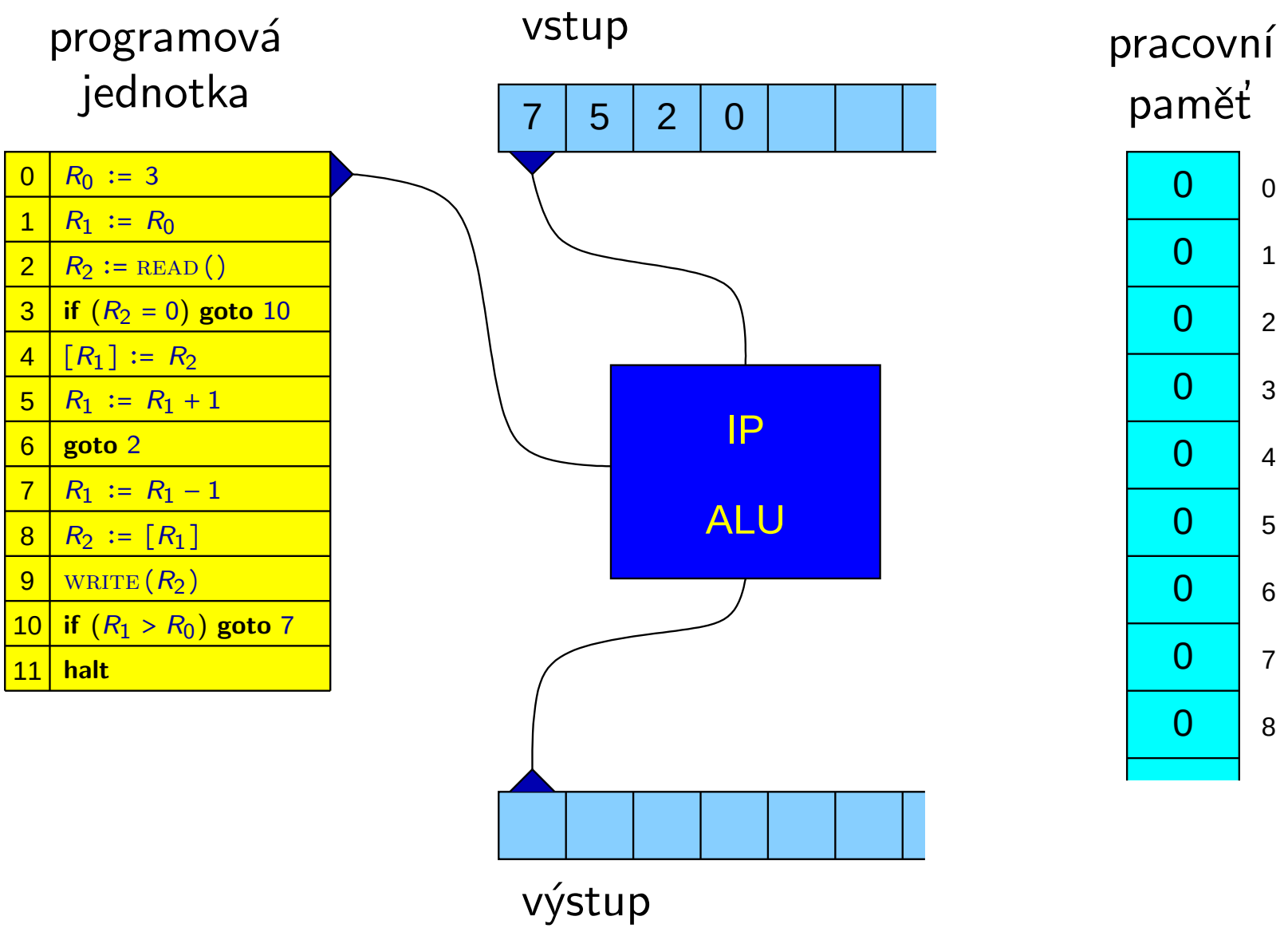
- **Programová jednotka** – obsahuje program stroje RAM a ukazatel na právě prováděnou instrukci
- **Pracovní paměť** tvořená buňkami očíslovanými $0, 1, 2, \ldots$

  Tyto buňky buňky budeme označovat $R_0, R_1, R_2, \ldots$

  Obsah buněk je možno číst i do nich zapisovat.
- **Vstupní páska** – je z ní možné pouze číst
- **Výstupní páska** – je na ni možno pouze zapisovat

Buňky paměti i vstupní a výstupní pásky obsahují jako hodnoty celá čísla (tj. prvky množiny $\mathbb{Z}$).

Stroj RAM


programová jednotka
vstup
pracovní paměť
0 $R_0 := 3$
1 $R_1 := R_0$
2 $R_2 := \text{READ}()$
3 **if** $(R_2 = 0)$ **goto** 10
4 $[R_1] := R_2$
5 $R_1 := R_1 + 1$
6 **goto** 2
7 $R_1 := R_1 - 1$
8 $R_2 := [R_1]$
9 $\text{WRITE}(R_2)$
10 **if** $(R_1 > R_0)$ **goto** 7
11 **halt**
7 5 2 0
IP
ALU
0 0 0 0 0 0 0 0
0 1 2 3 4 5 6 7 8
výstup

Stroj RAM

Přehled instrukcí:

$R_i := c$	– přiřazení konstanty
$R_i := R_j$	– přiřazení
$R_i := [R_j]$	– load (čtení z paměti)
$[R_i] := R_j$	– store (zápis do paměti)
$R_i := R_j \ op \ R_k$	– aritmetické instrukce, $op \in \{+, -, *, /\}$
nebo $R_i := R_j \ op \ c$	
if $(R_i \ rel \ R_j)$ **goto** ℓ	– podmíněný skok, $rel \in \{=, \neq, \leq, \geq, <, >\}$
nebo **if** $(R_i \ rel \ c)$ **goto** ℓ	
goto ℓ	– nepodmíněný skok
$R_i :=$ READ $()$	– čtení ze vstupu
WRITE (R_i)	– zápis na výstup
halt	– zastavení programu

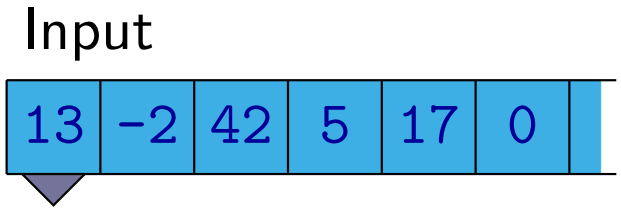
Stroj RAM

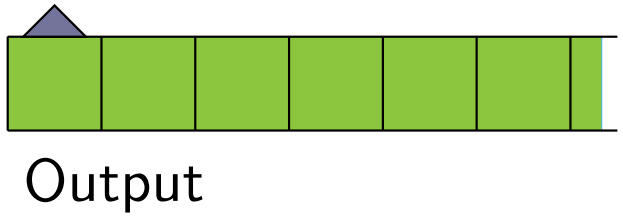
Příklady instrukcí:

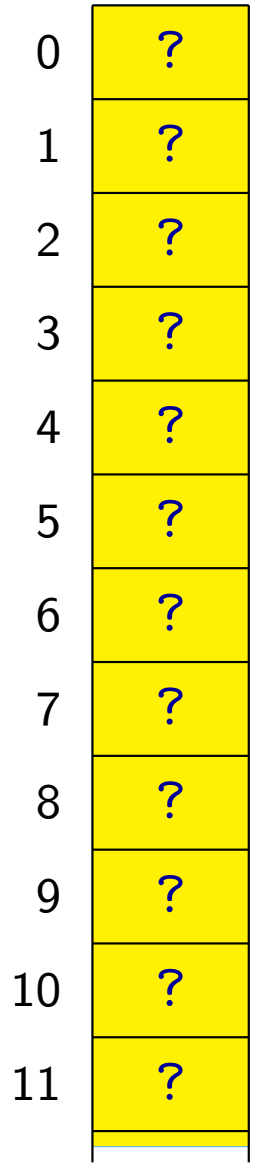
$R_5 := 42$	– přiřazení konstanty
$R_{12} := R_3$	– přiřazení
$R_8 := [R_2]$	– load (čtení z paměti)
$[R_{15}] := R_9$	– store (zápis do paměti)
$R_7 := R_3 + R_6$	– aritmetická instrukce
$R_{18} := R_{18} - 1$	– aritmetická instrukce
if $(R_4 \geq R_1)$ **goto** 2801	– podmíněný skok
if $(R_2 \neq 0)$ **goto** 3581	– podmíněný skok
goto 537	– nepodmíněný skok
$R_{23} :=$ READ $()$	– čtení ze vstupu
WRITE (R_{17})	– zápis na výstup
halt	– zastavení programu

Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

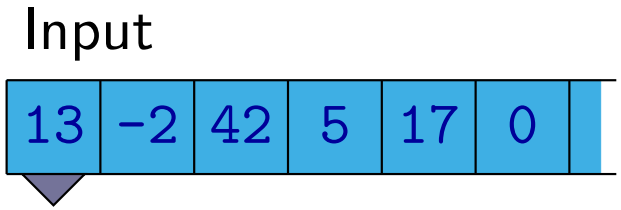


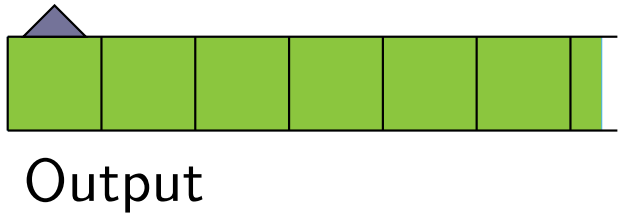


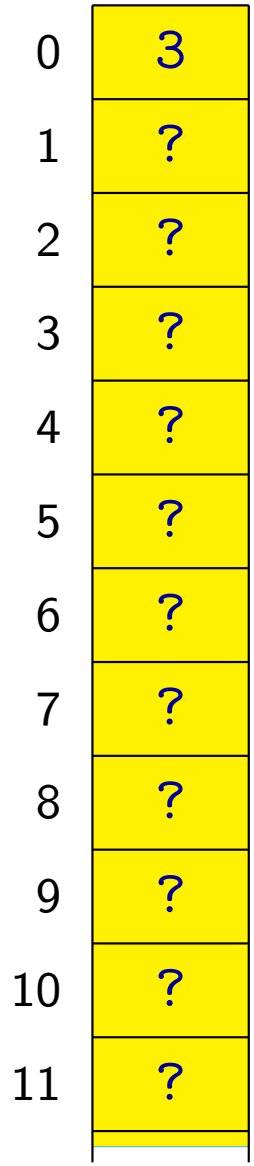


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

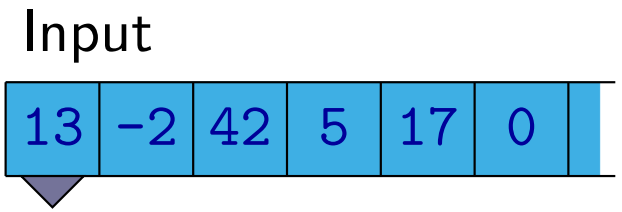


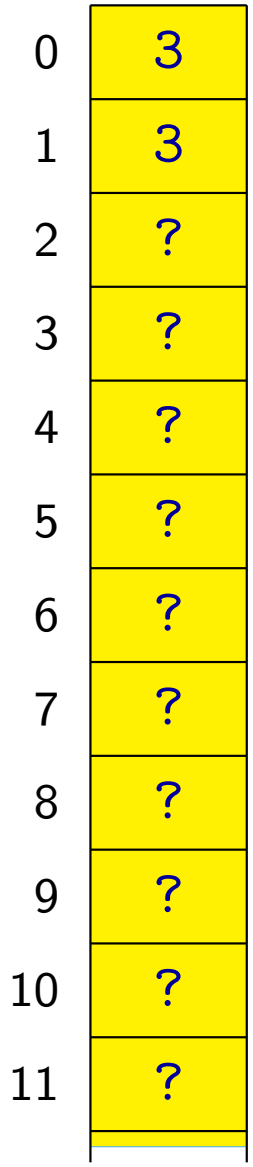


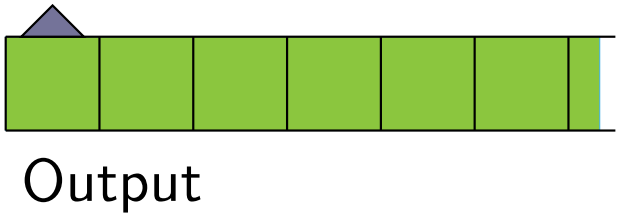


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

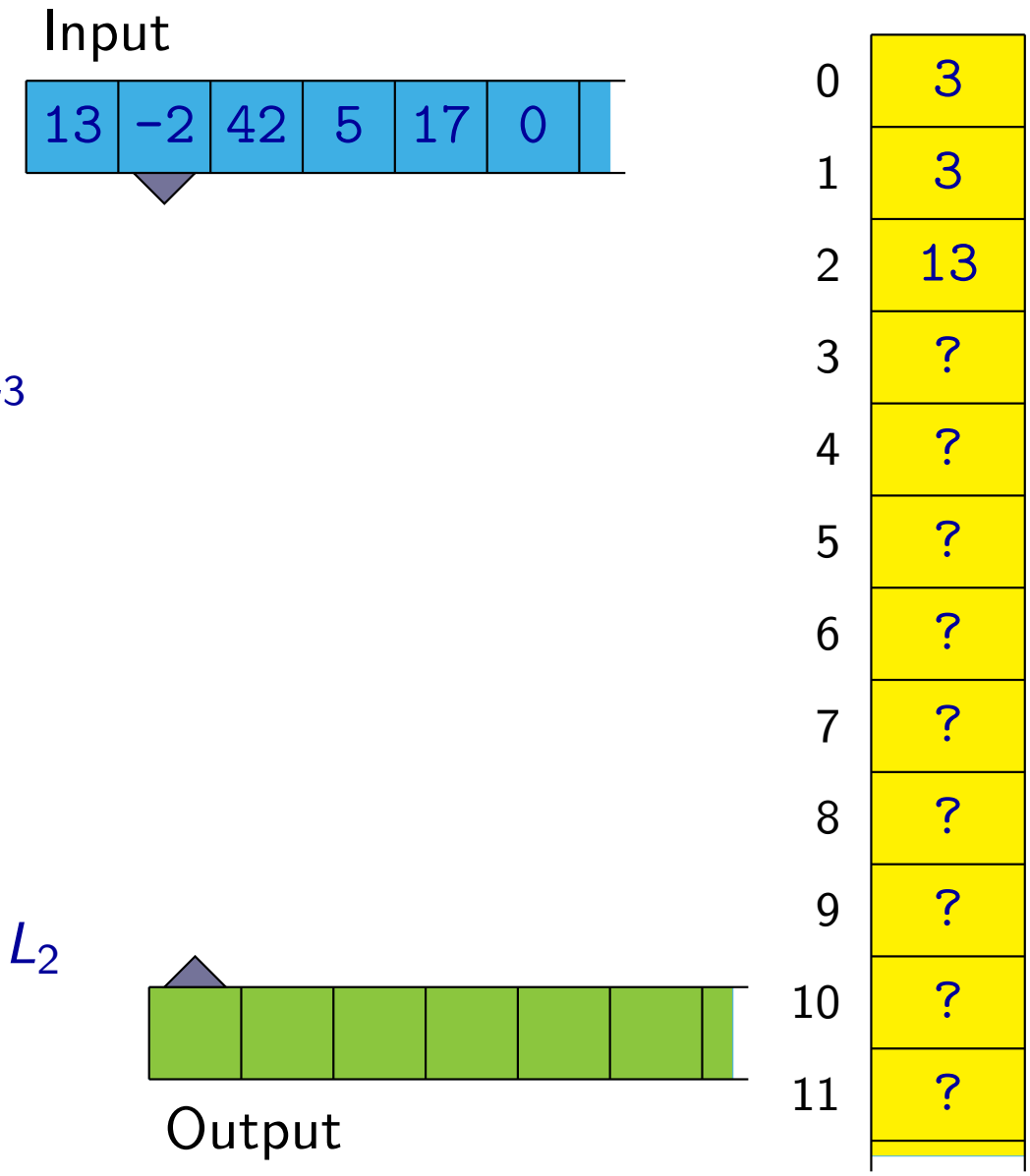






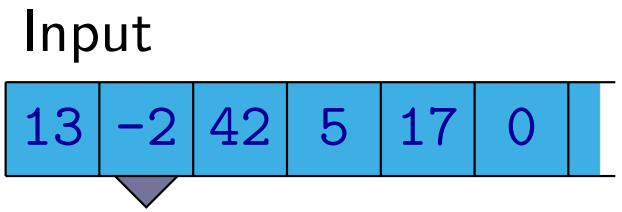
Stroj RAM

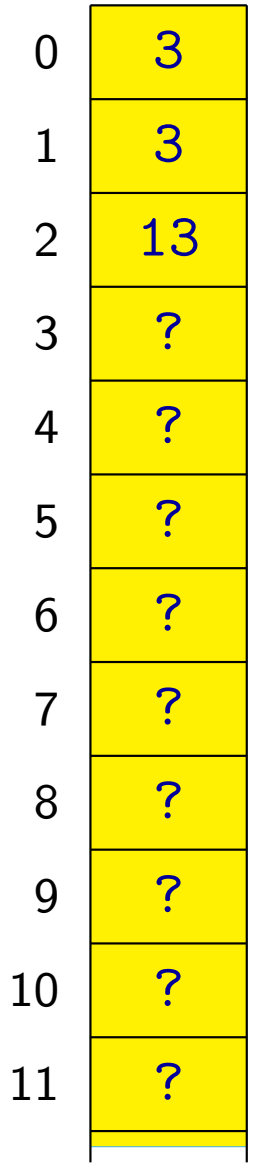
$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

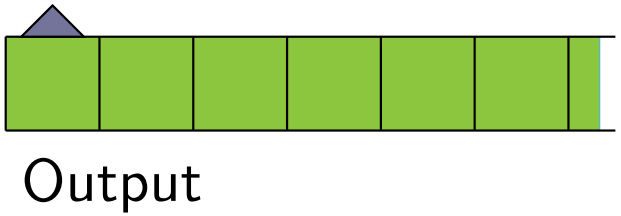


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

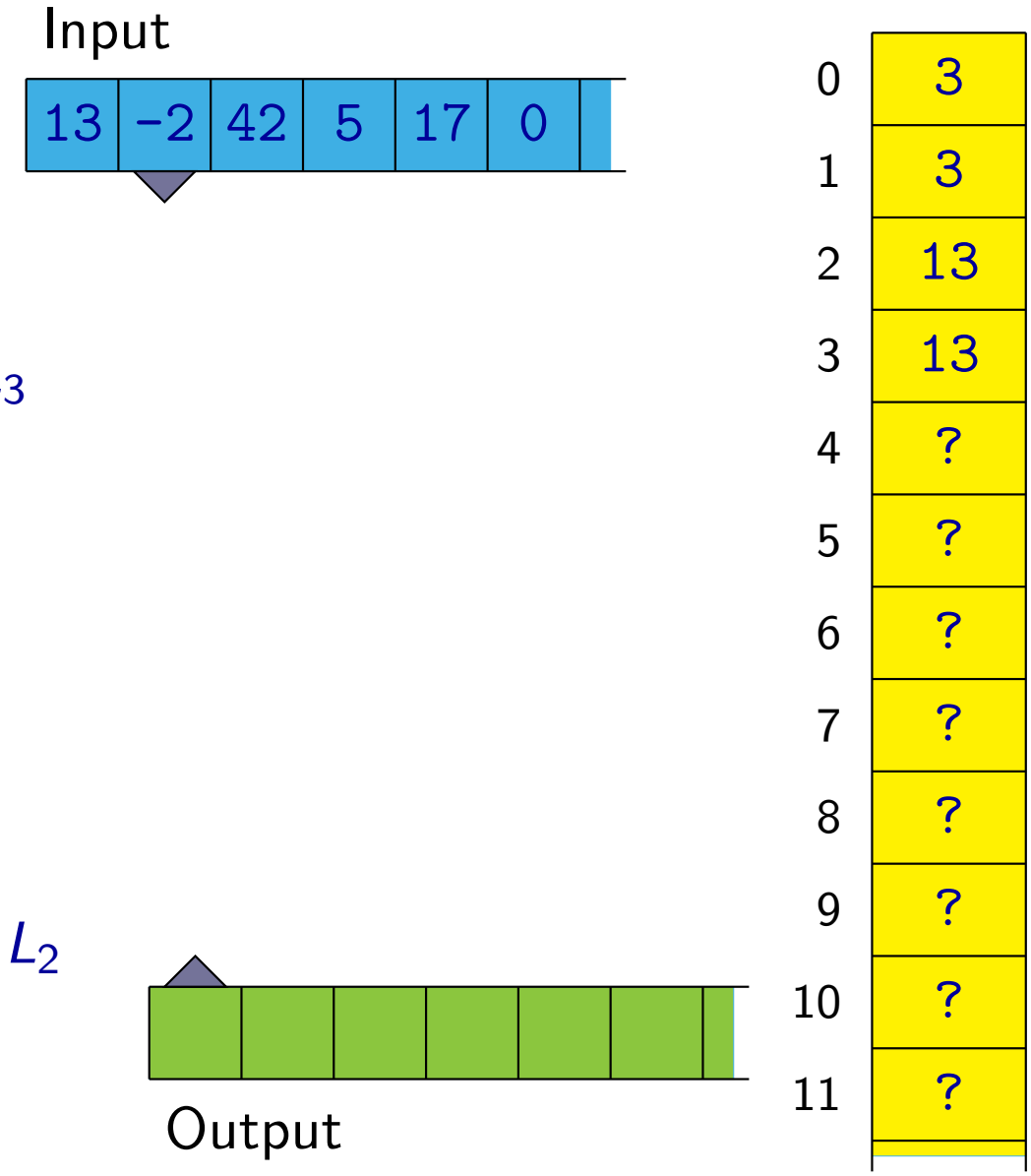






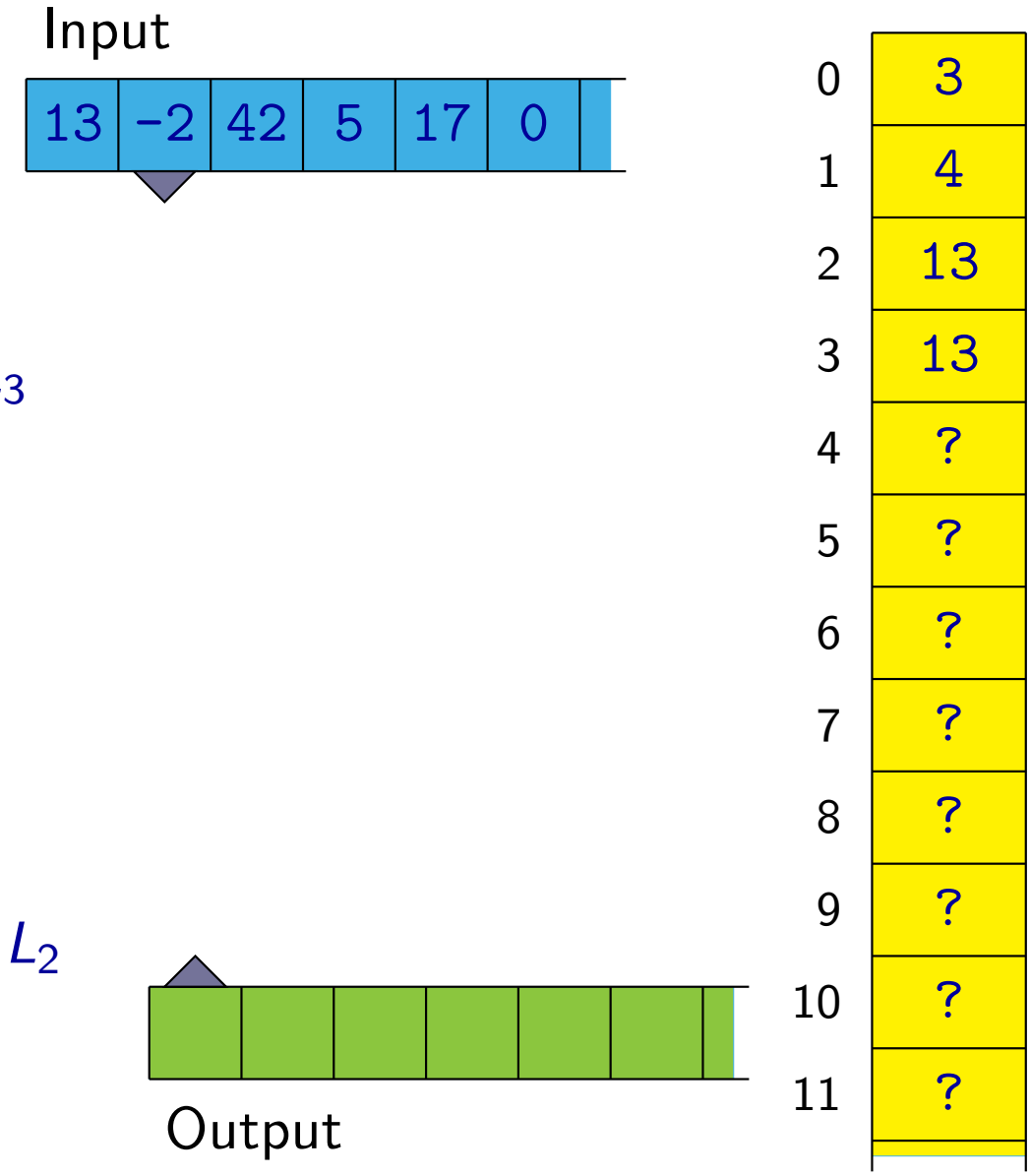
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



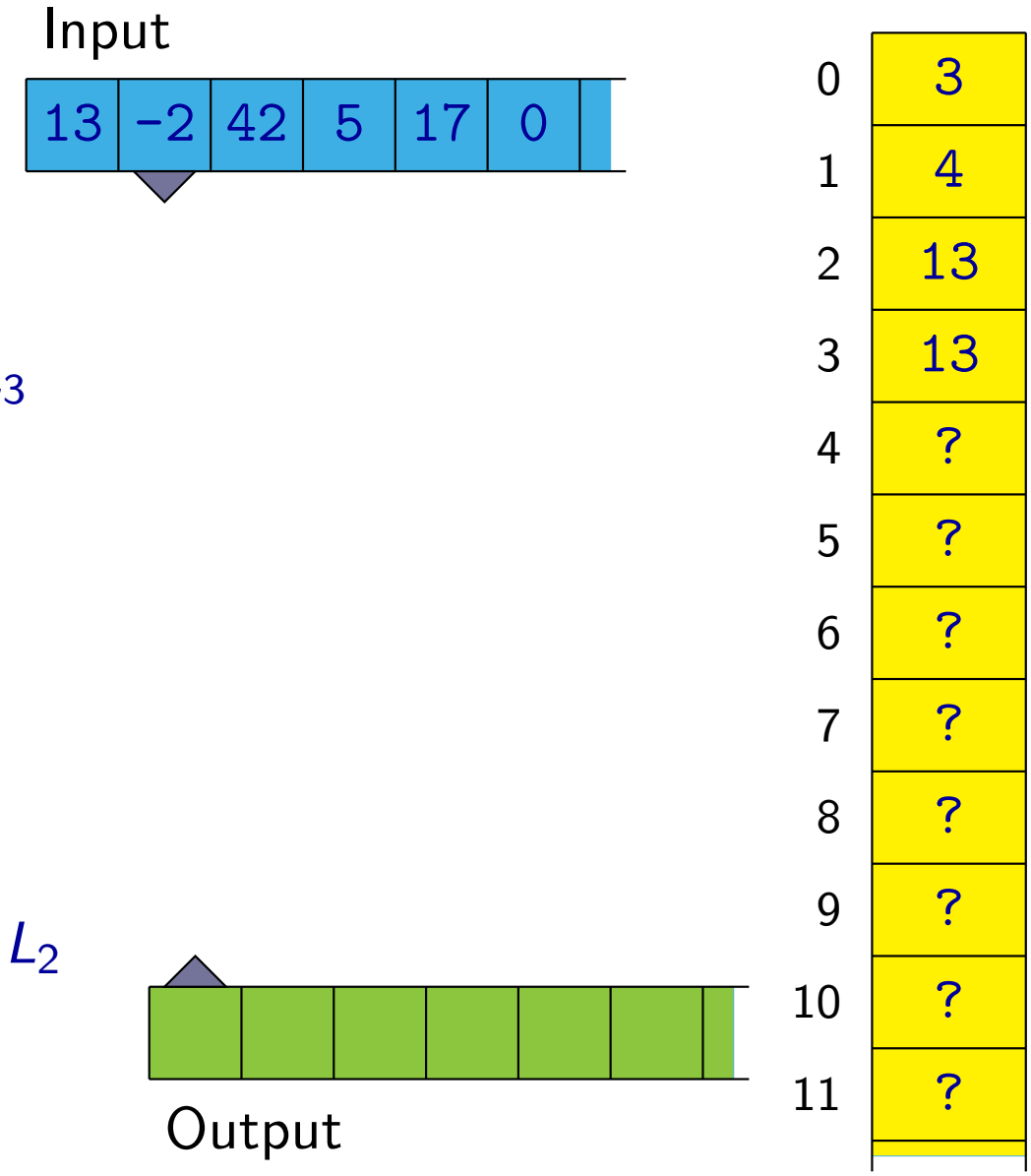
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



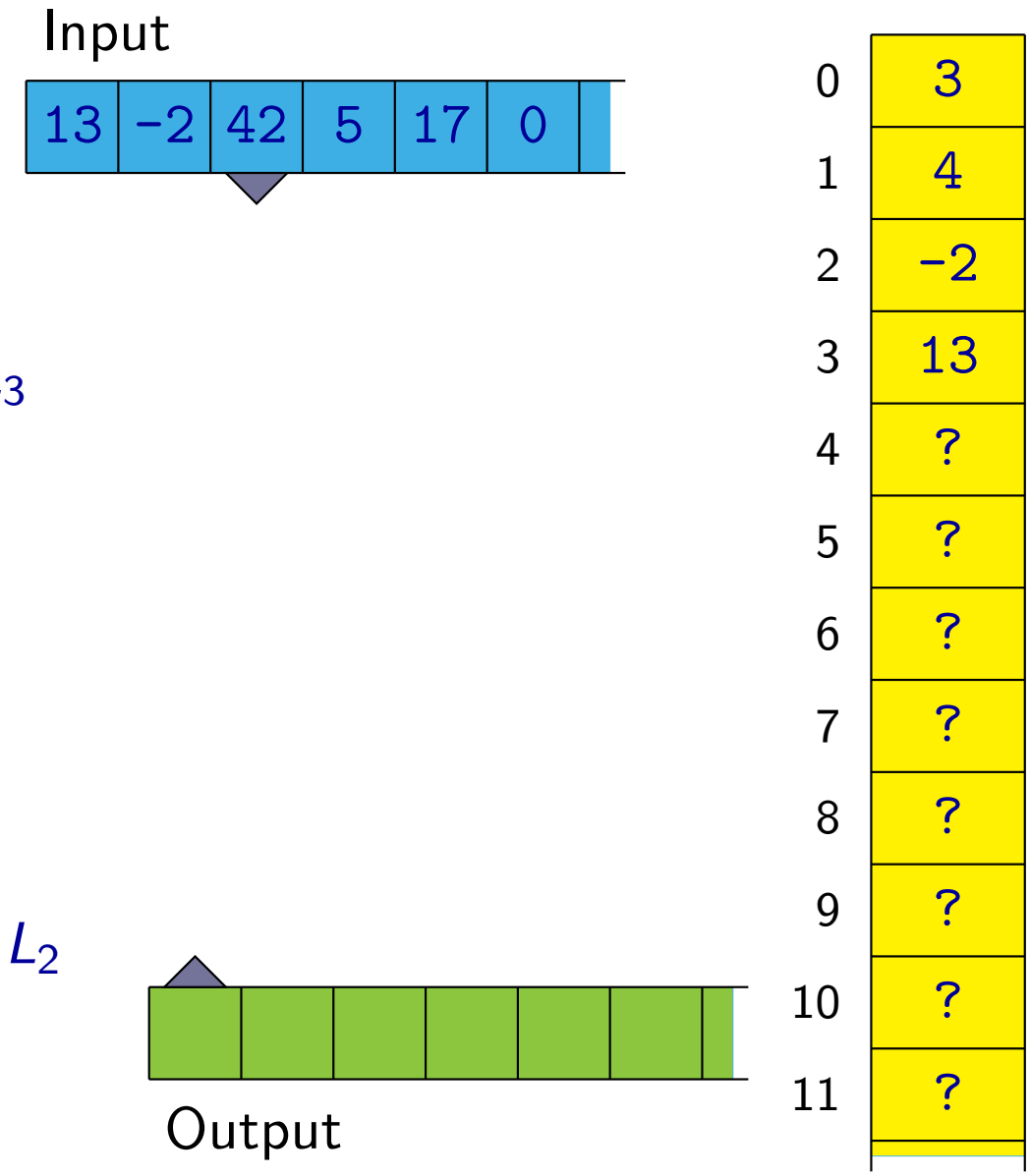
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



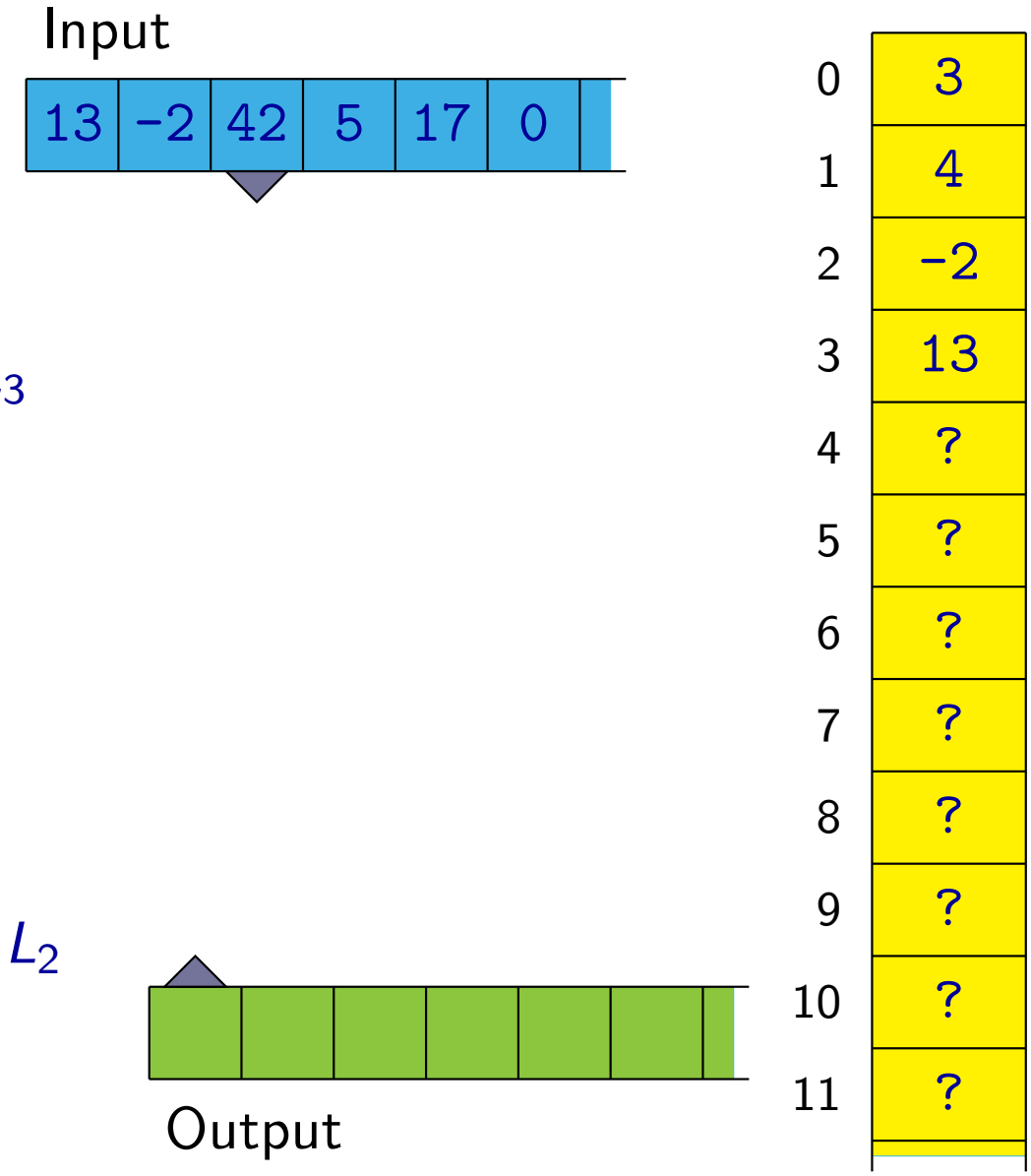
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



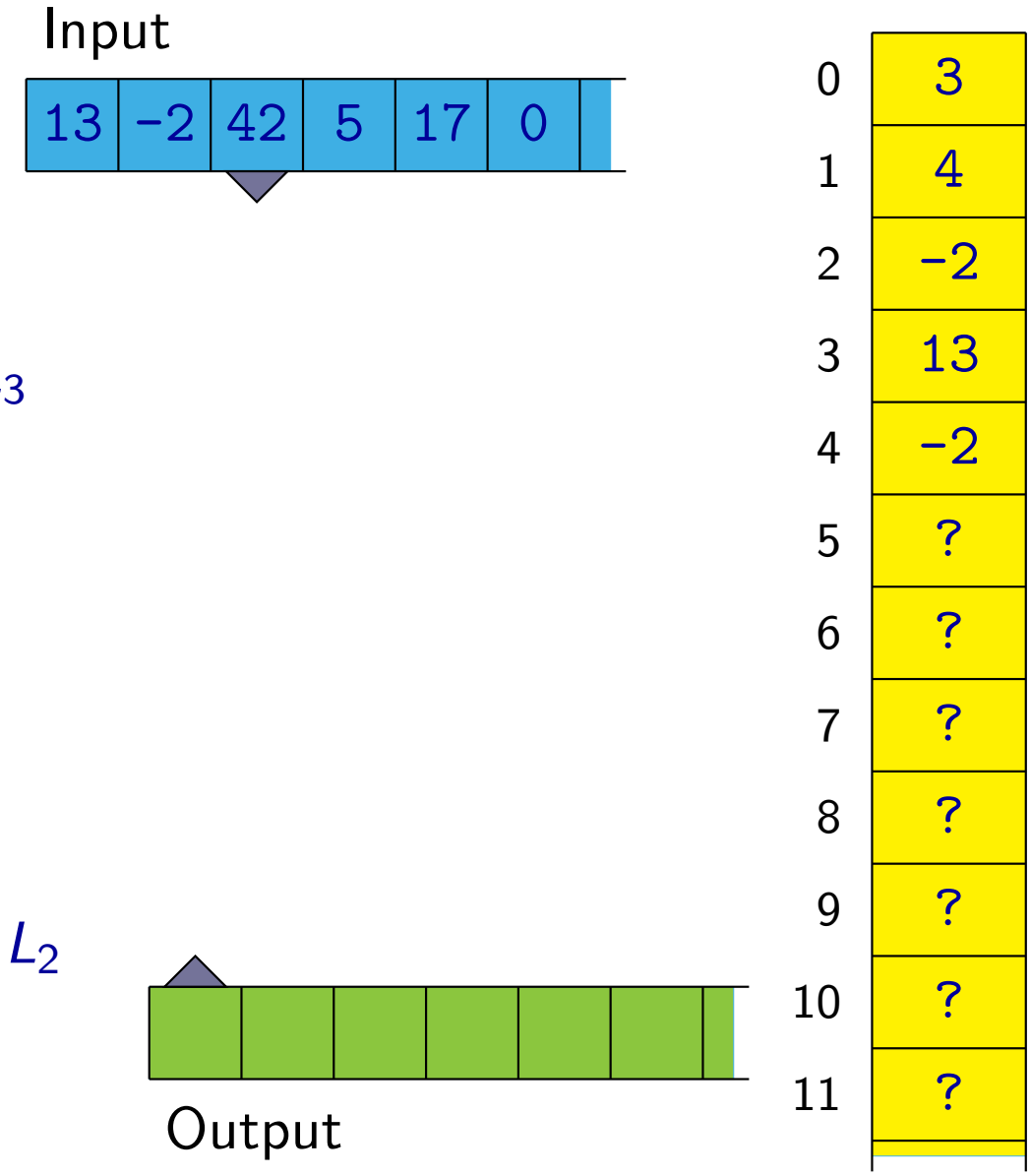
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



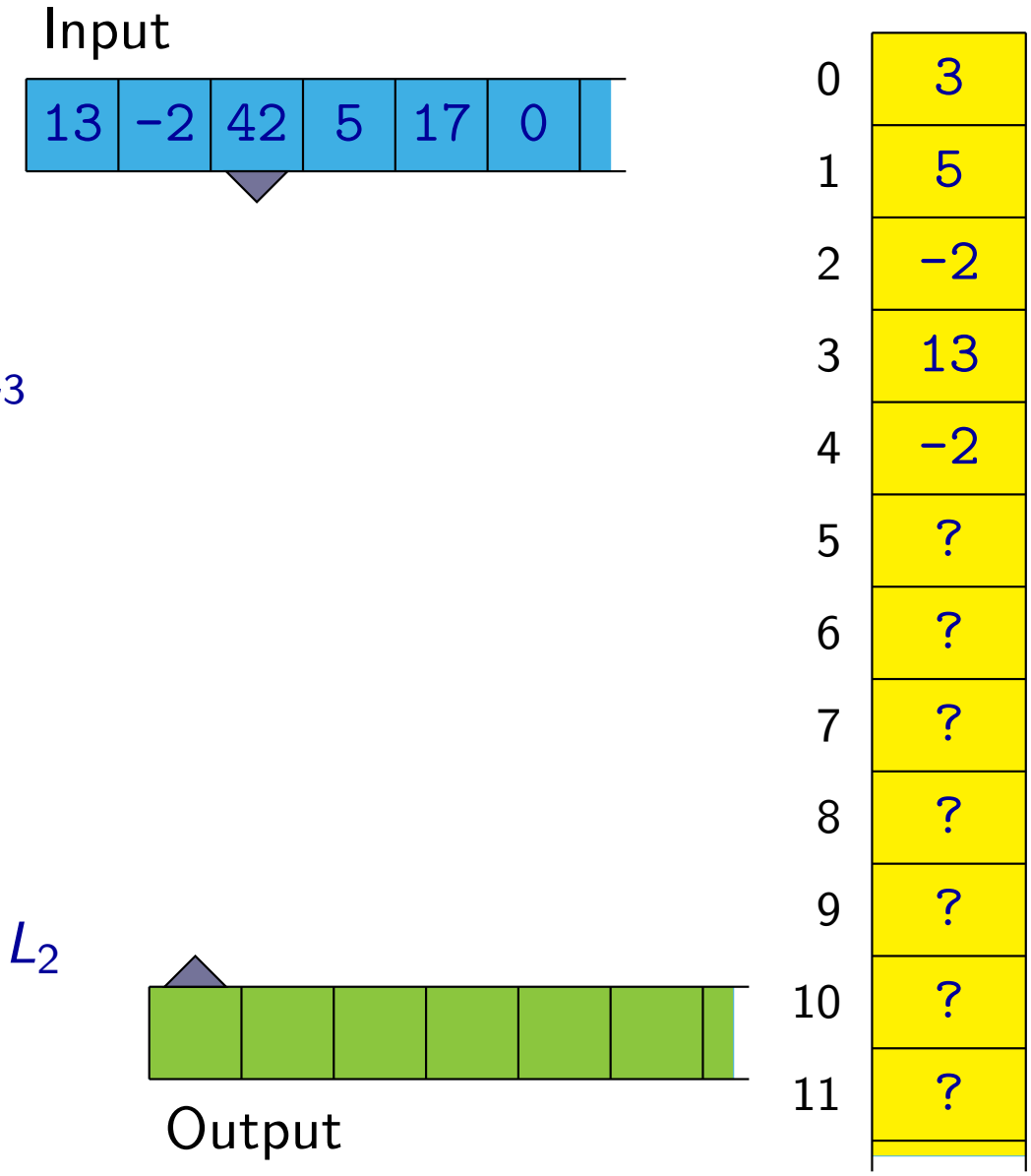
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



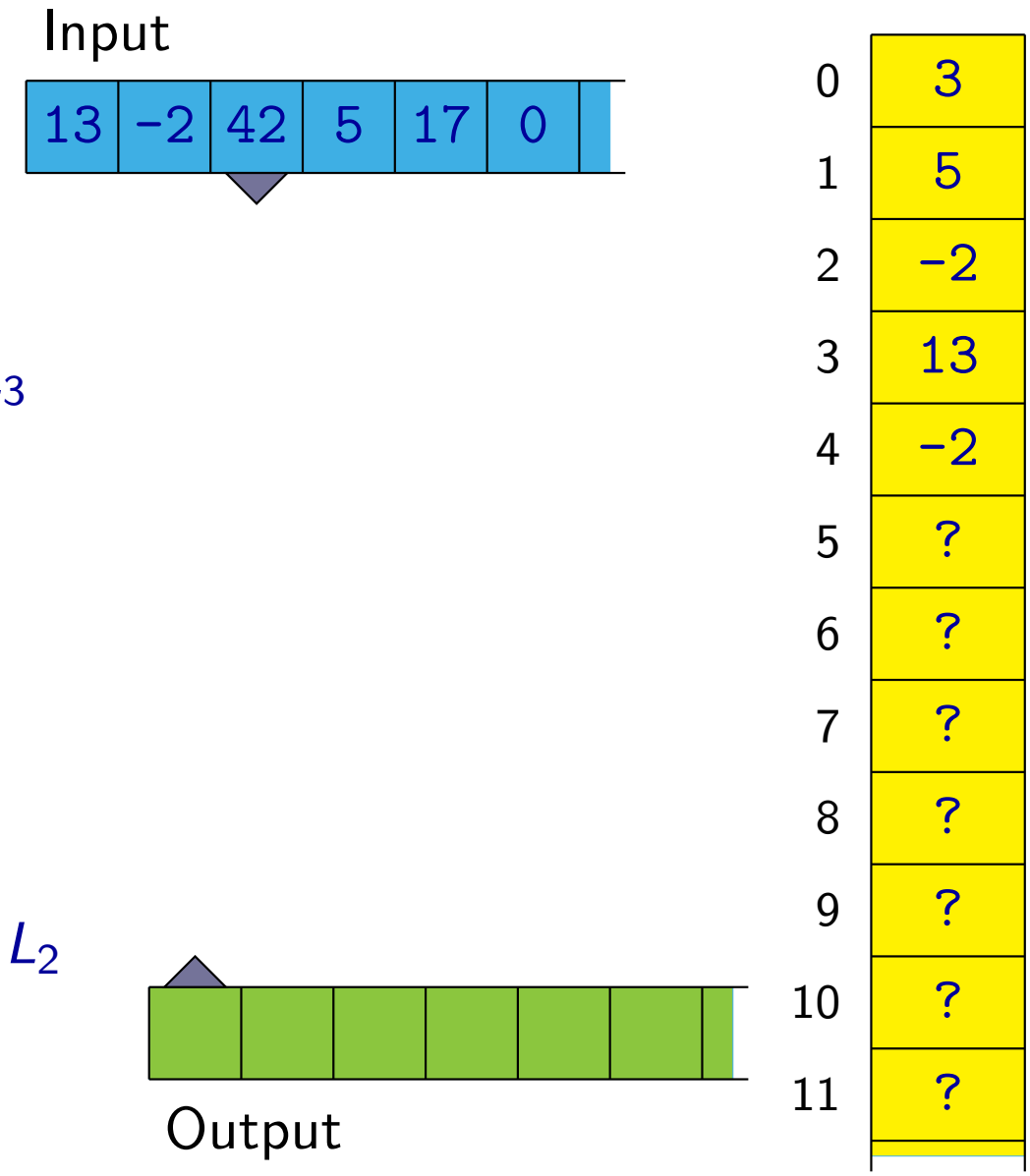
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



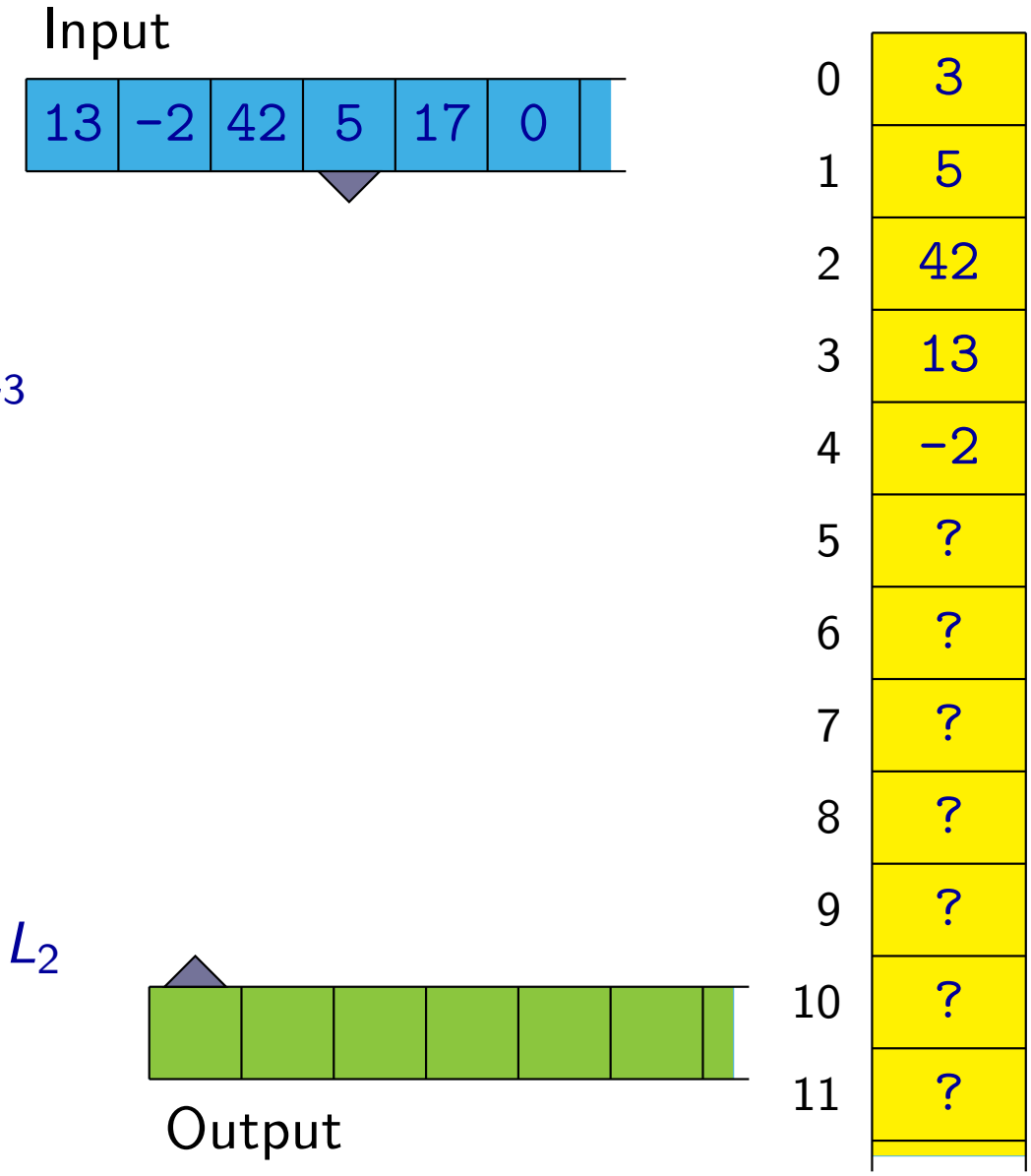
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



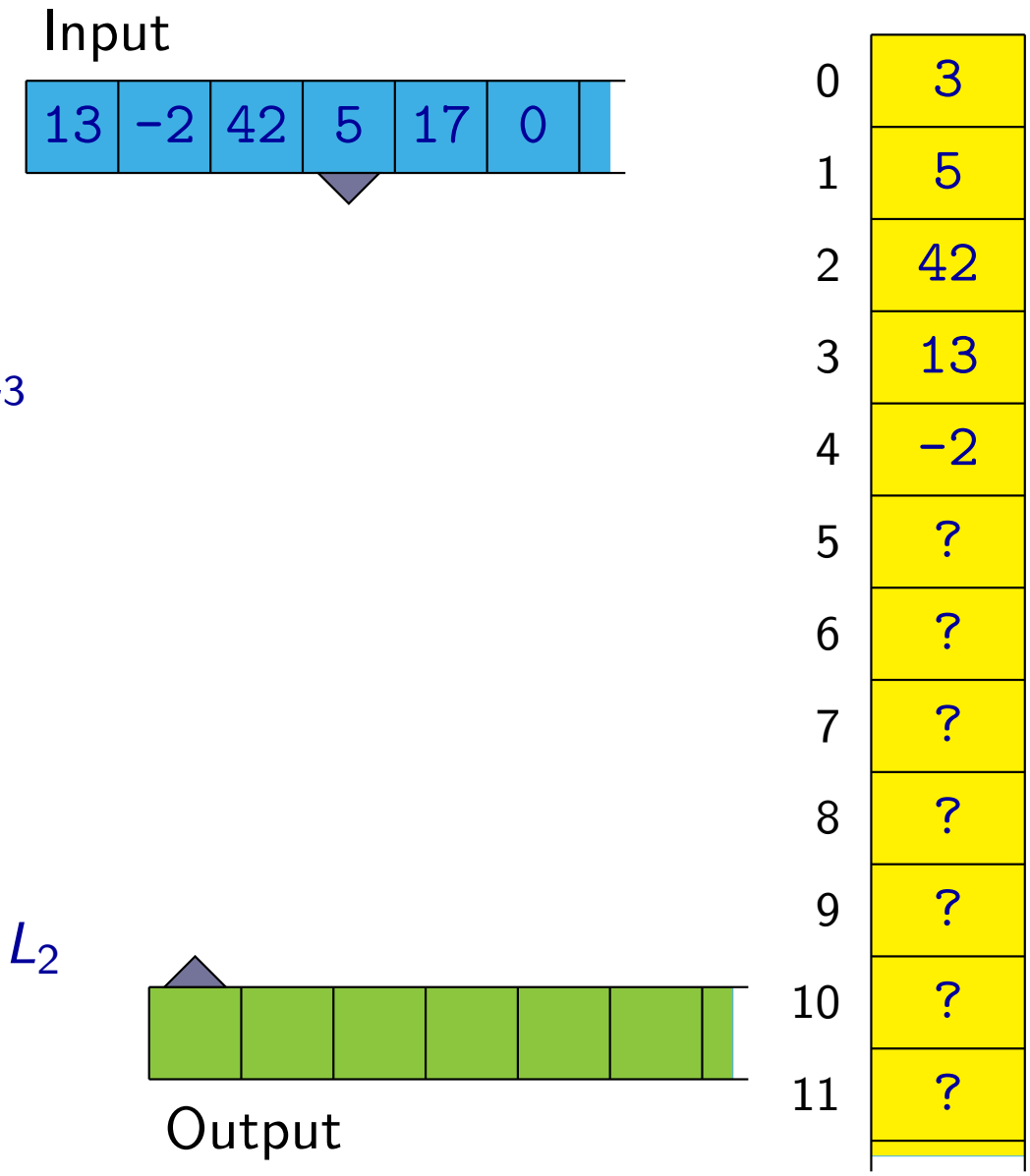
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



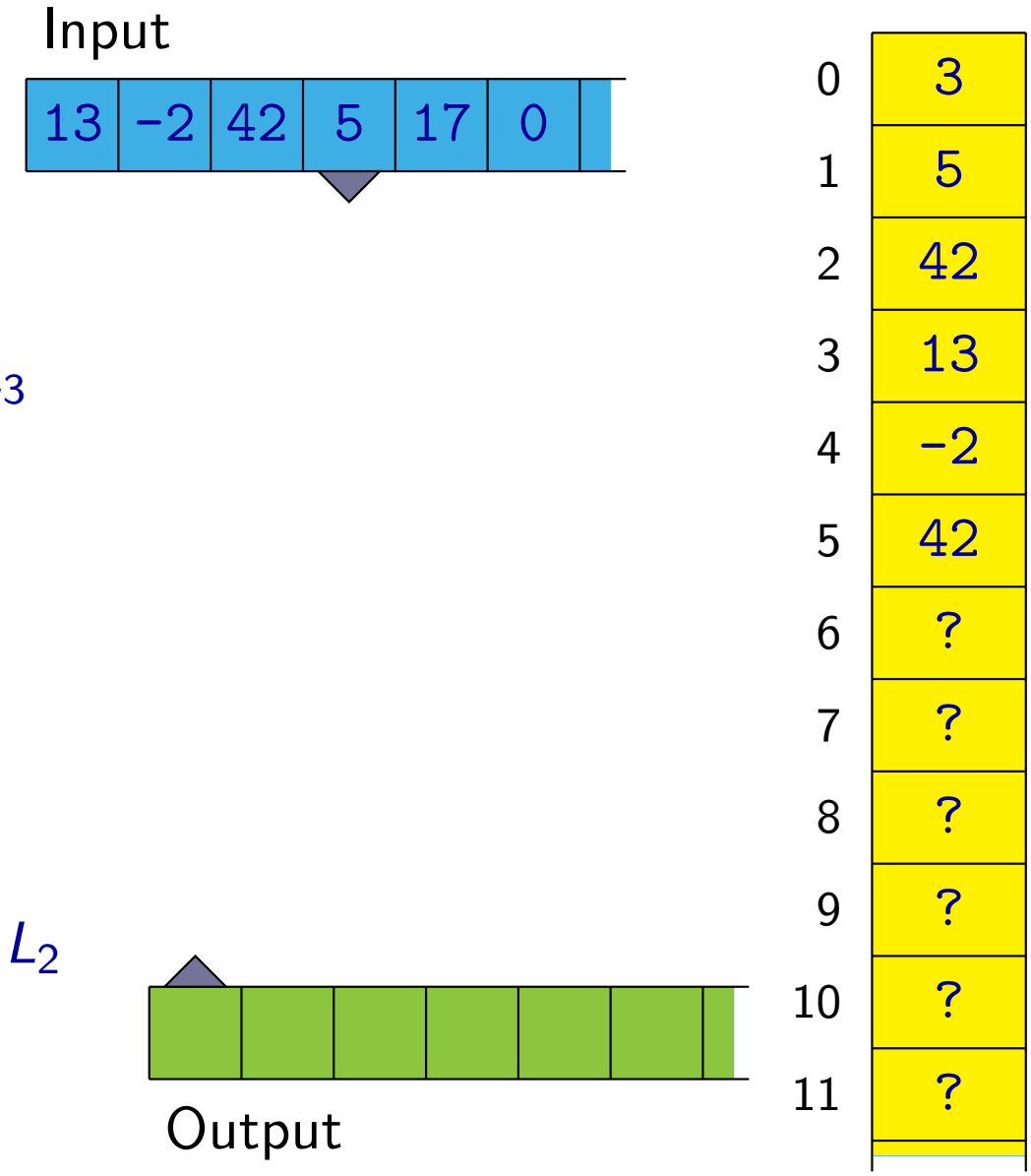
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



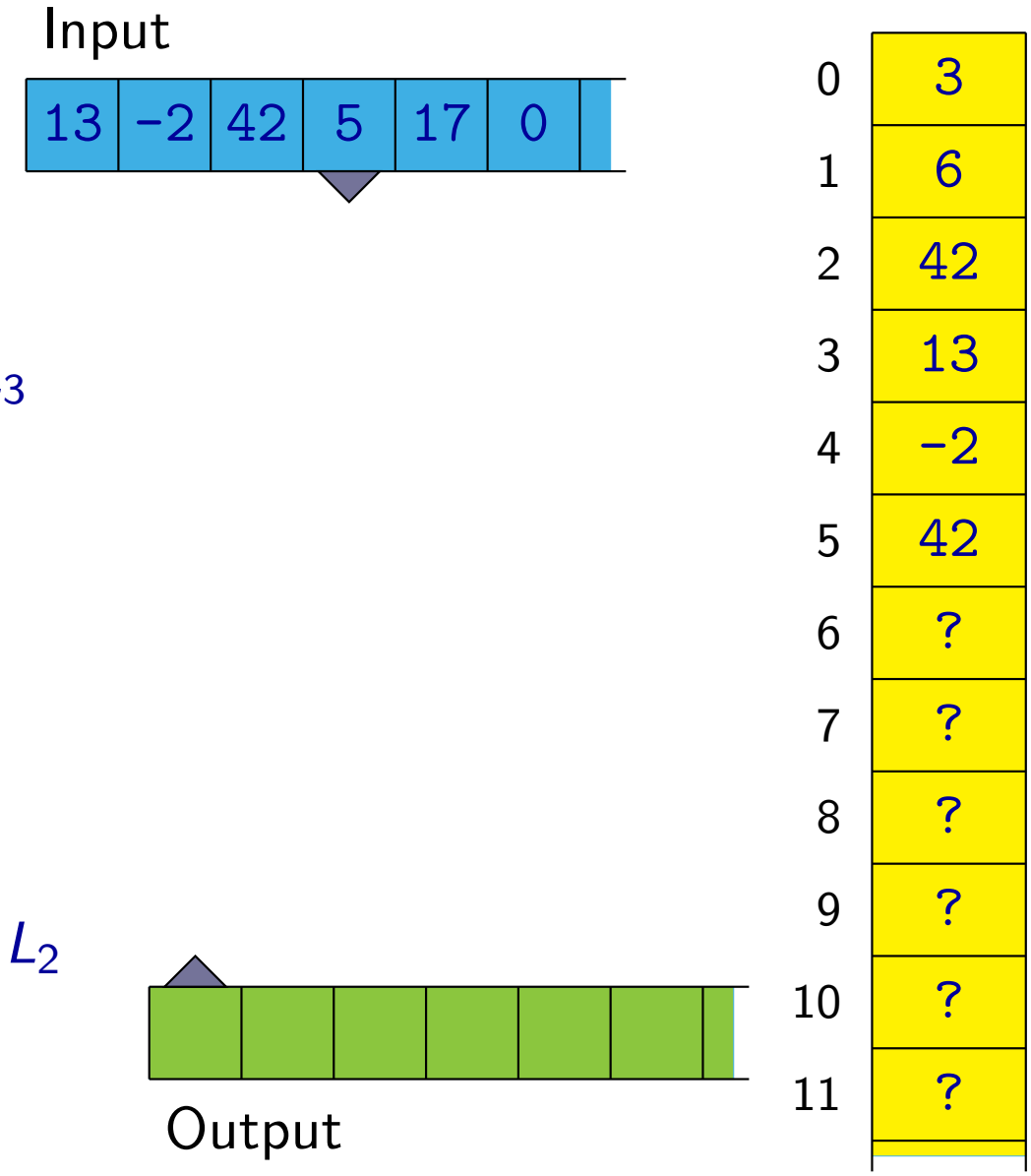
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



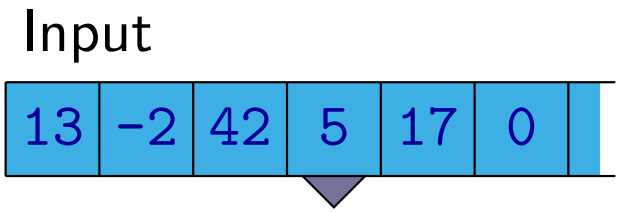
Stroj RAM

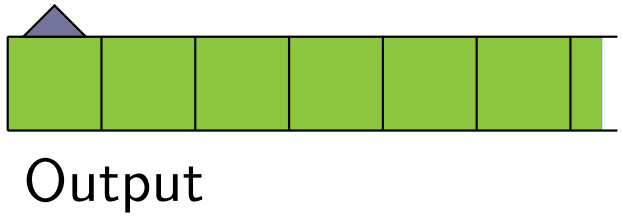
$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

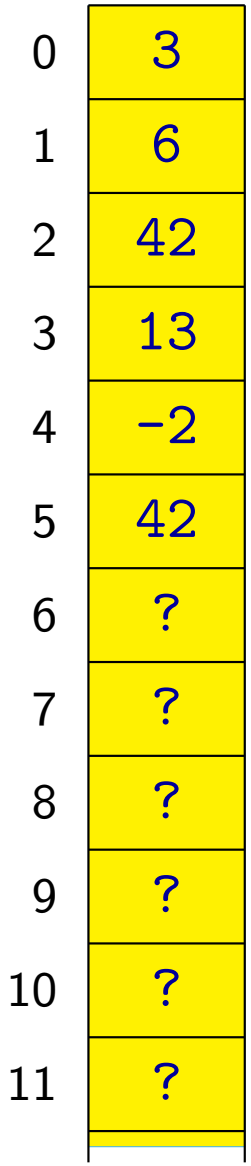


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

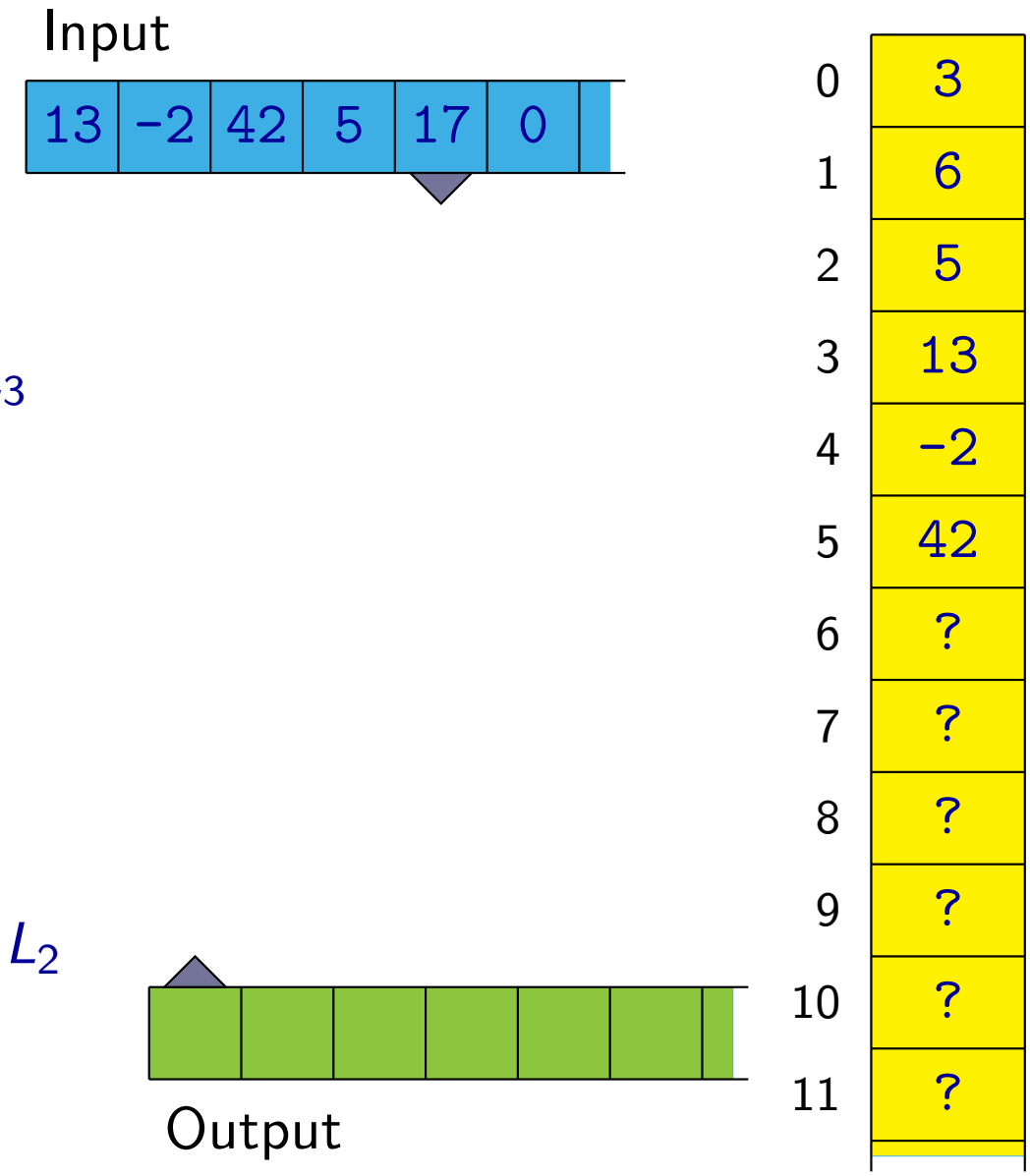






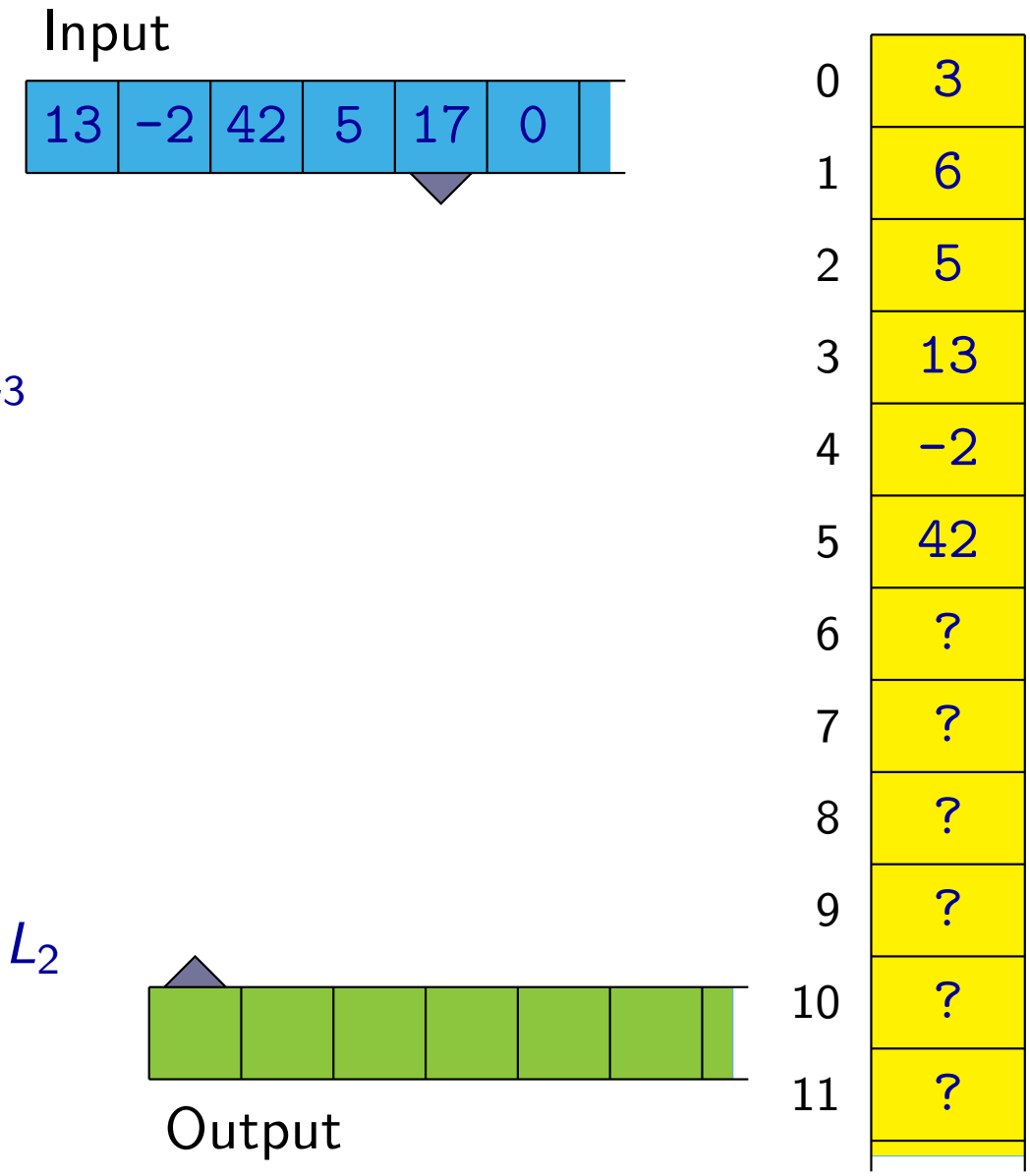
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



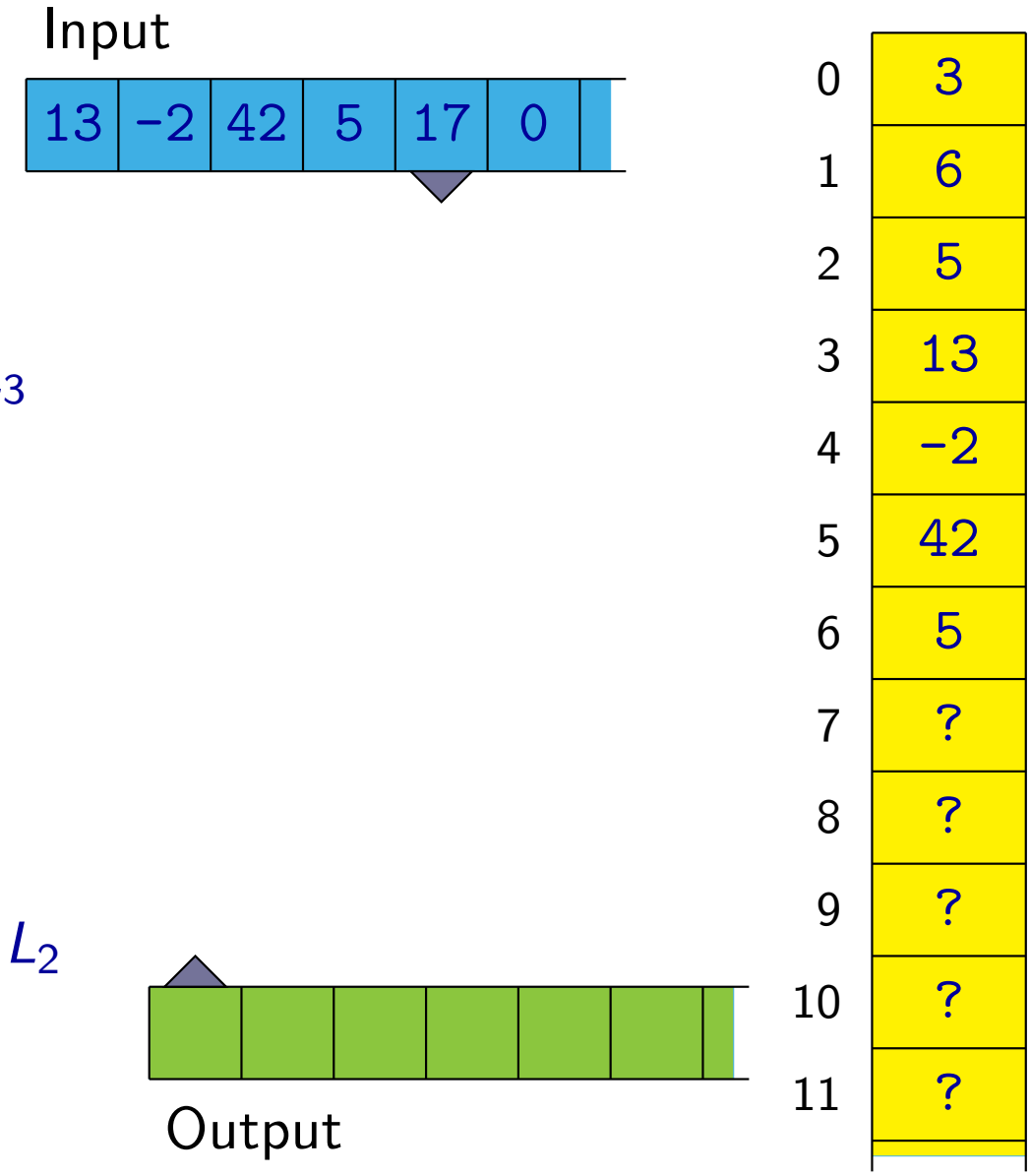
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



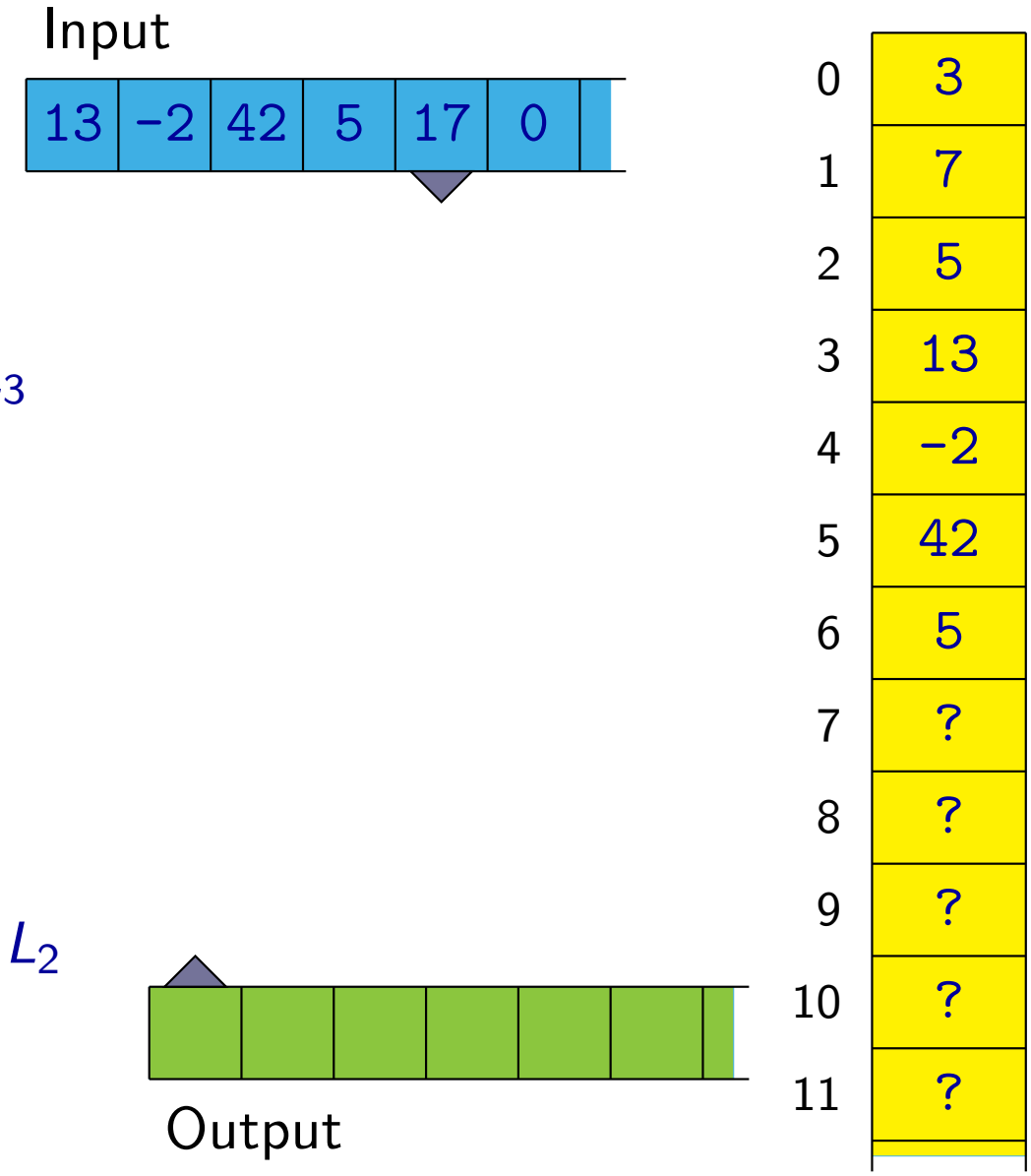
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



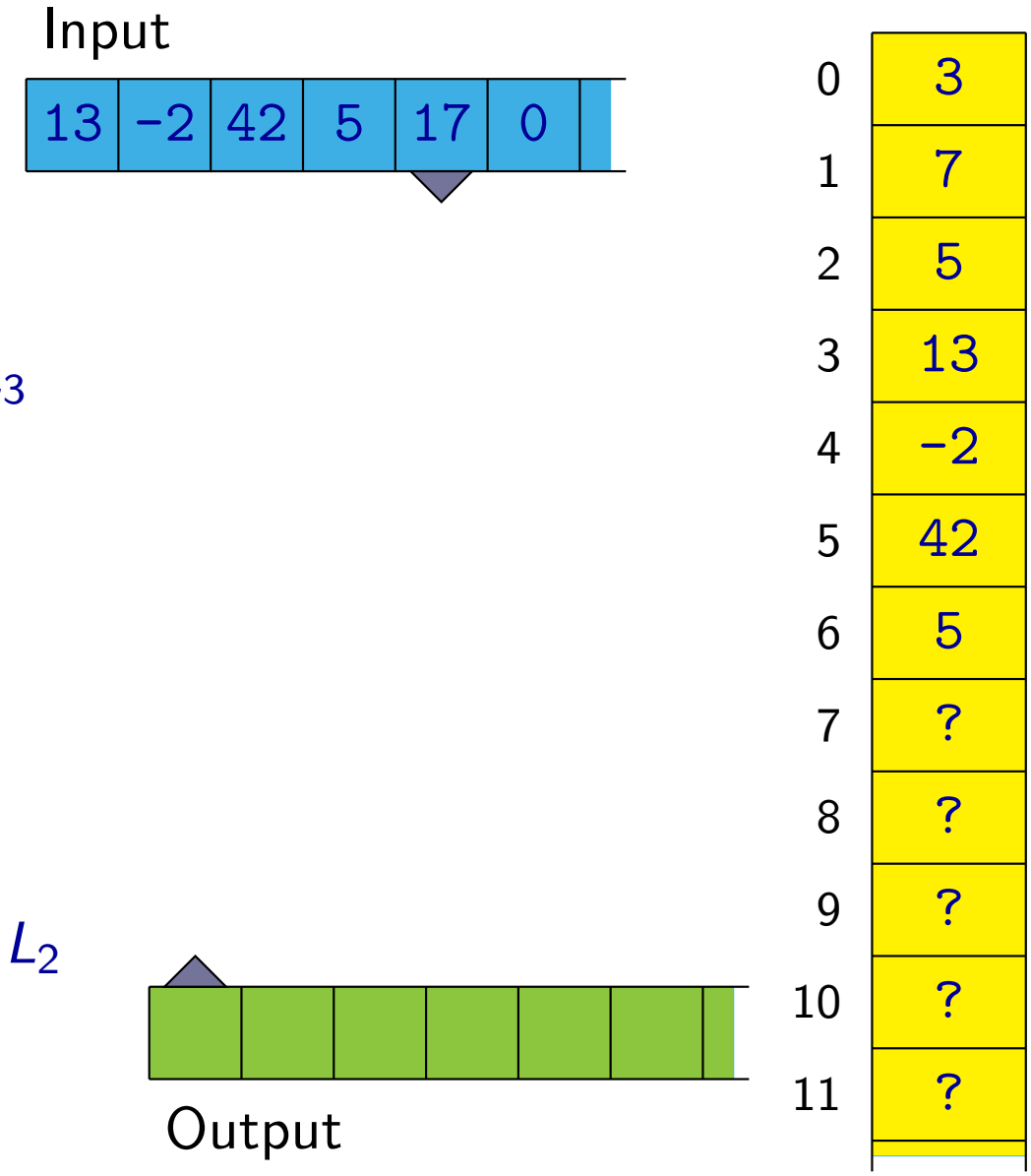
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



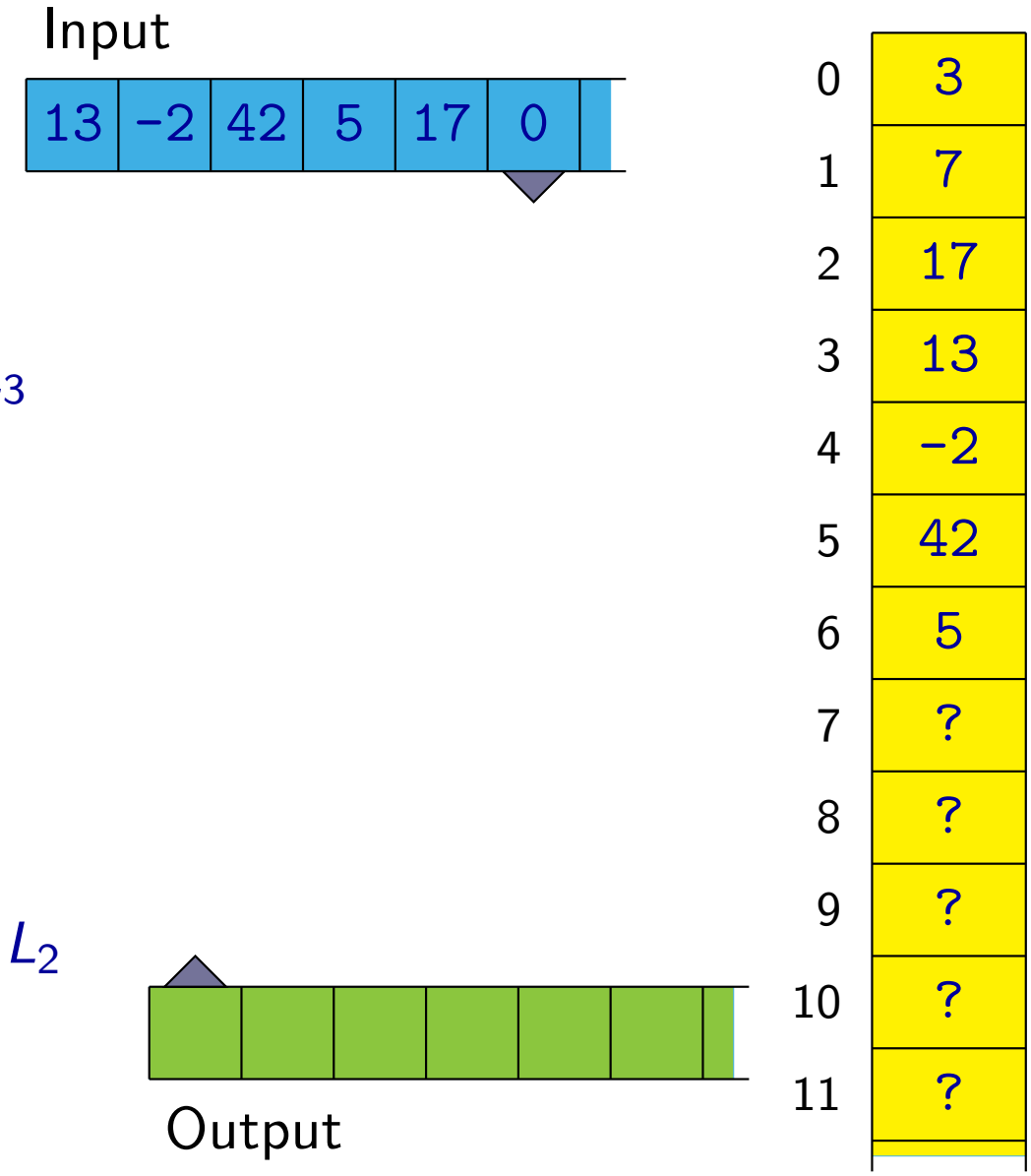
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



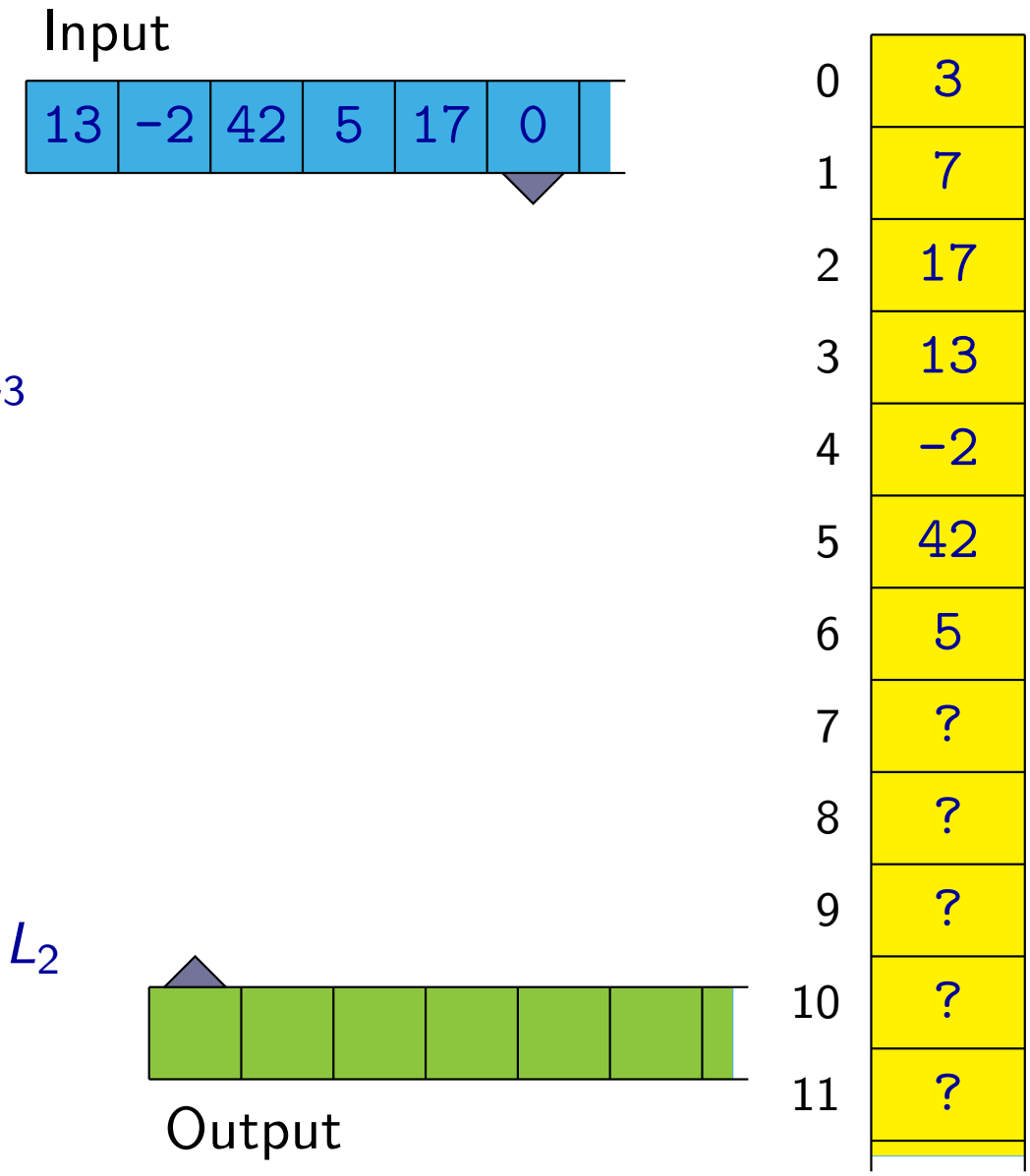
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



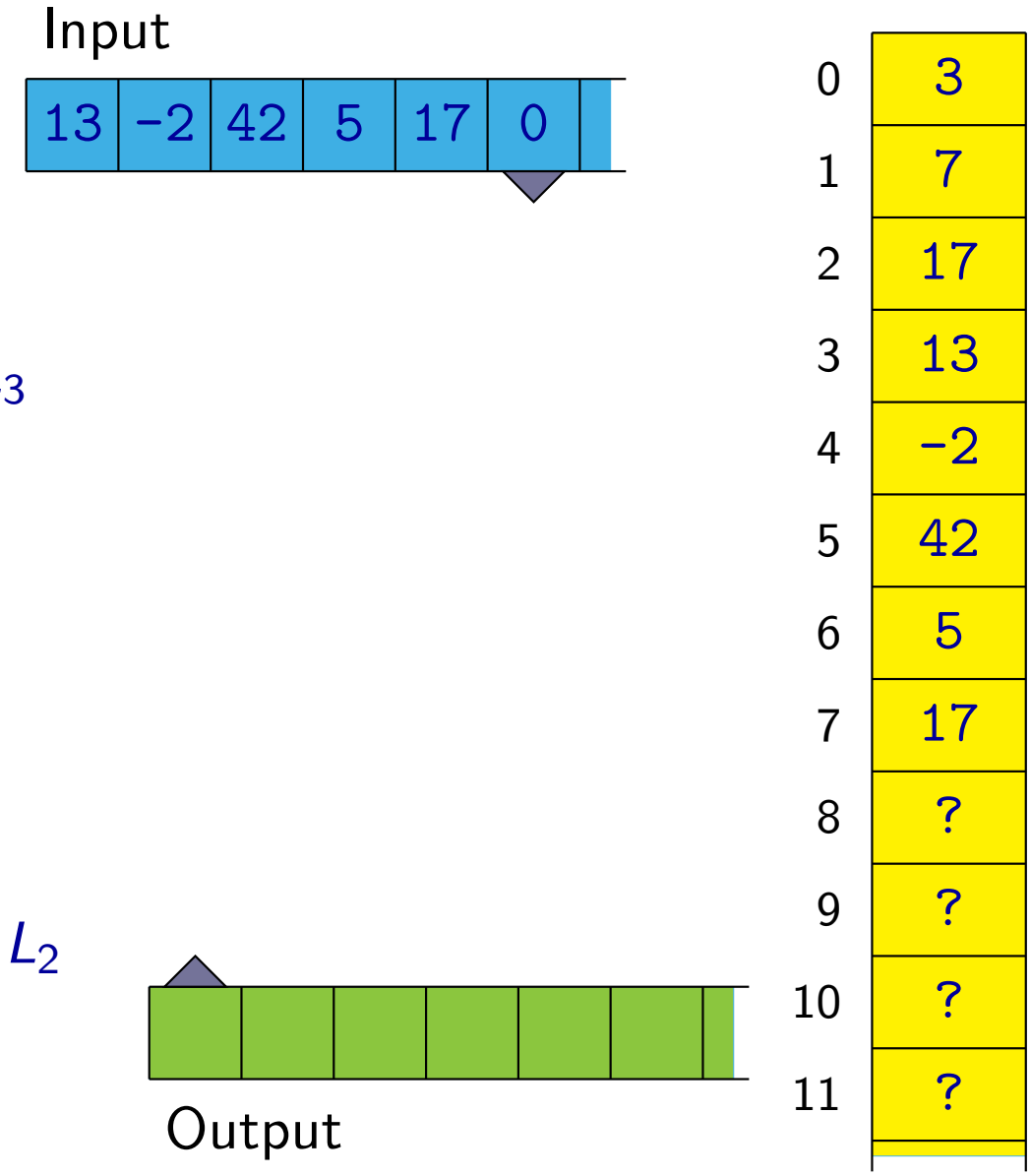
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



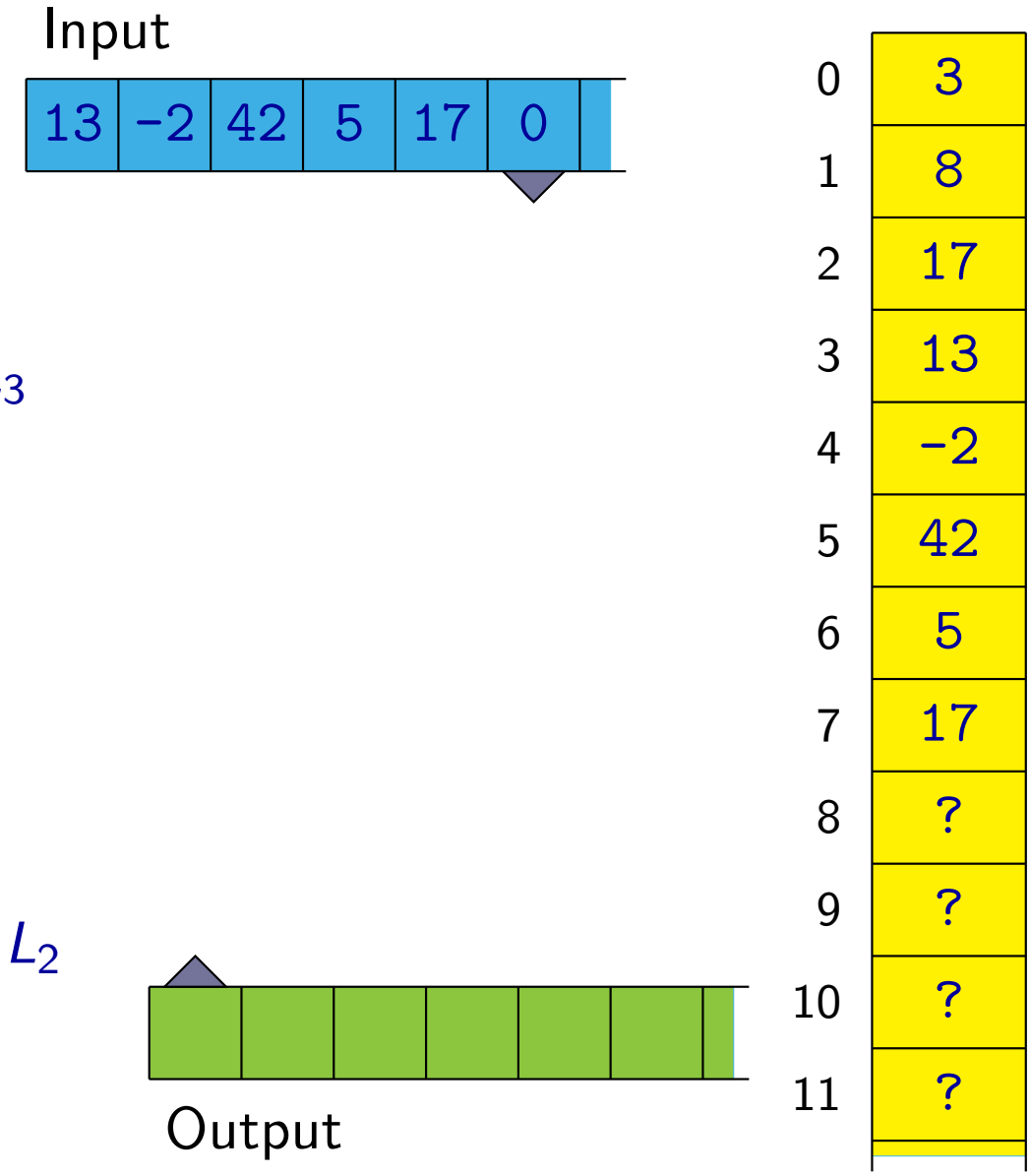
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



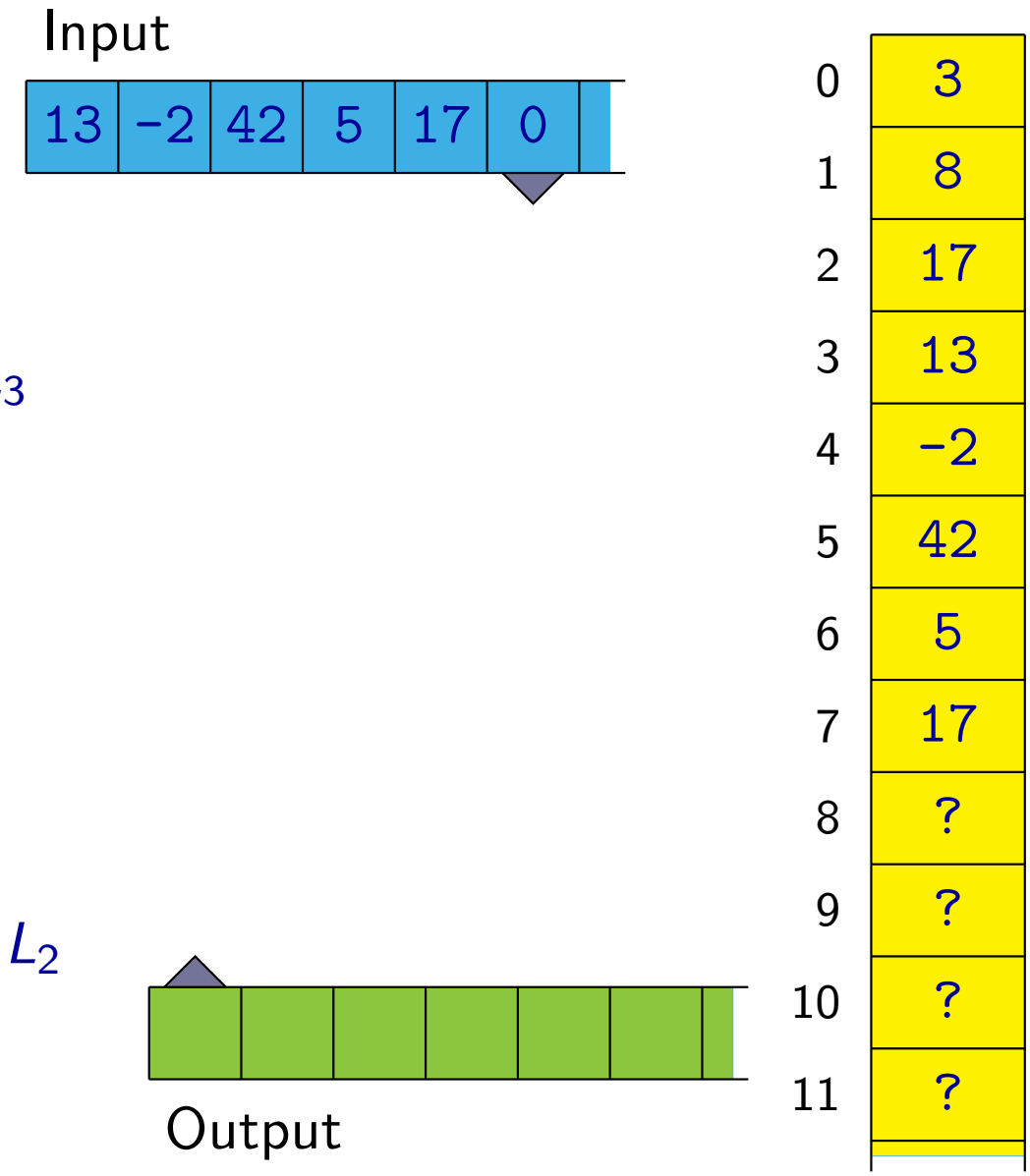
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



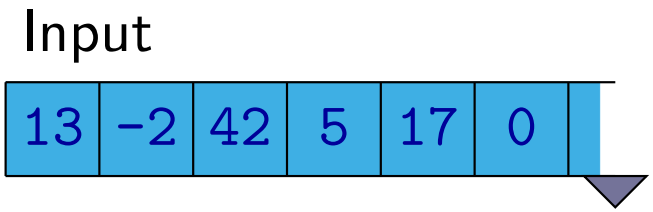
Stroj RAM

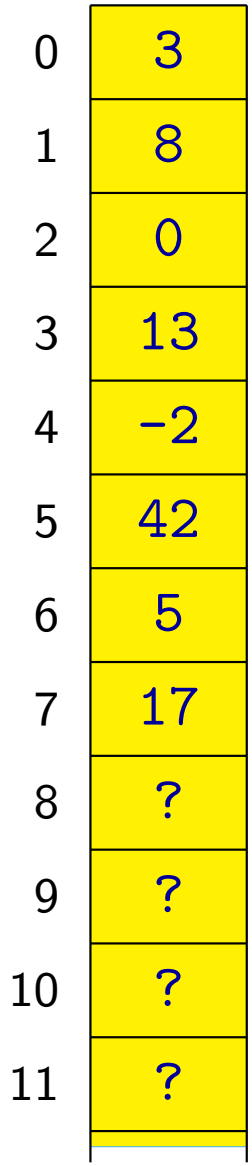
$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

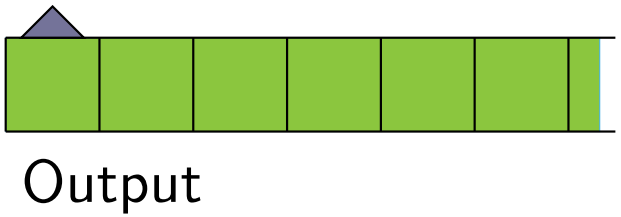


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

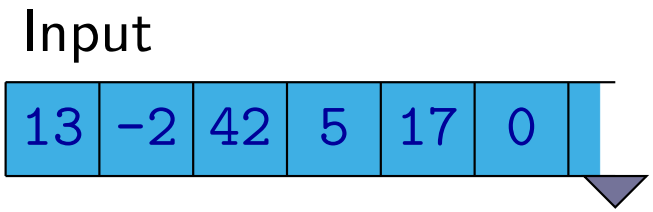


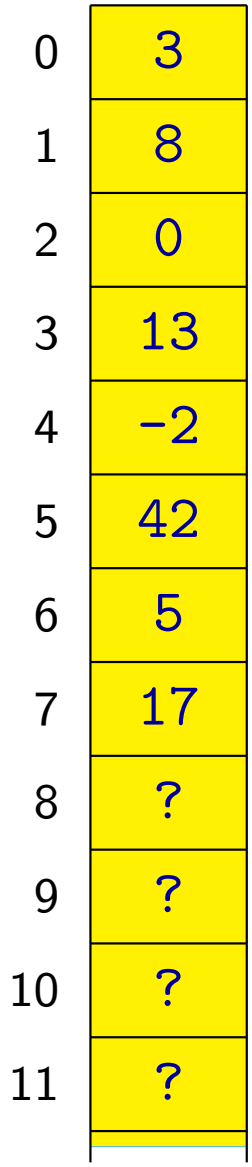


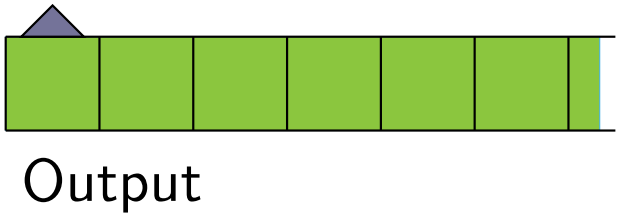


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

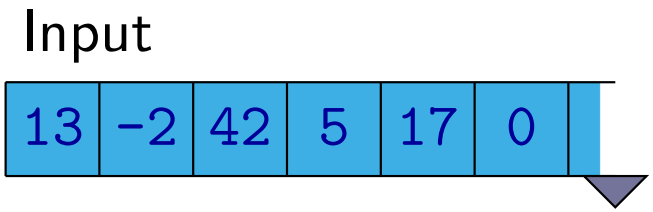


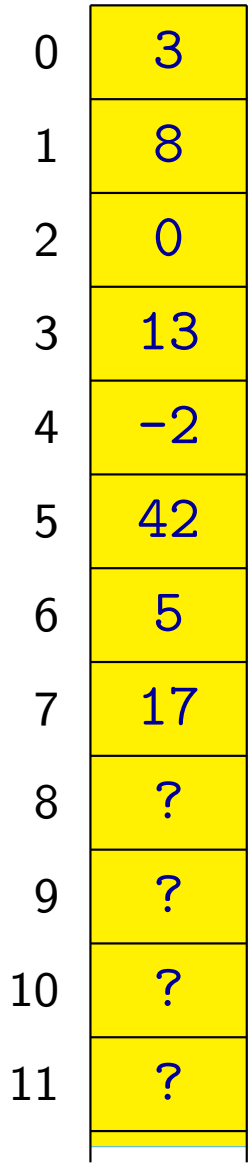


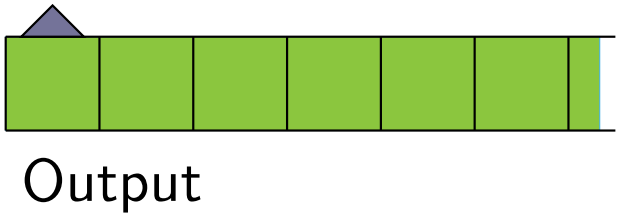


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

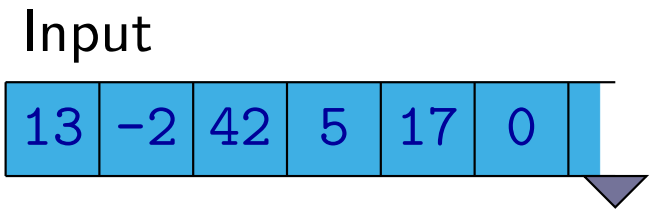


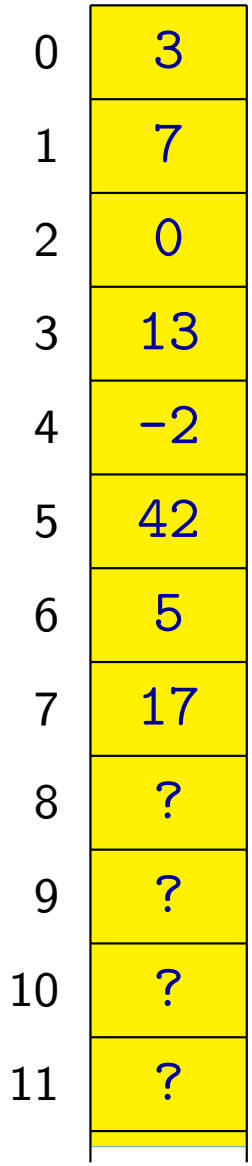


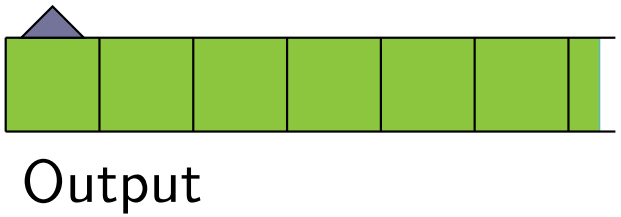


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

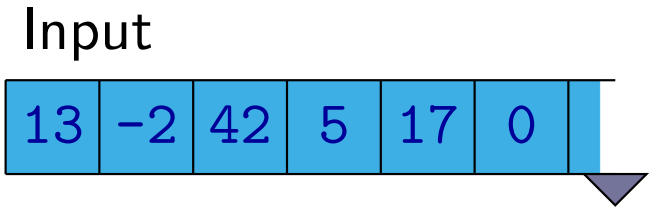


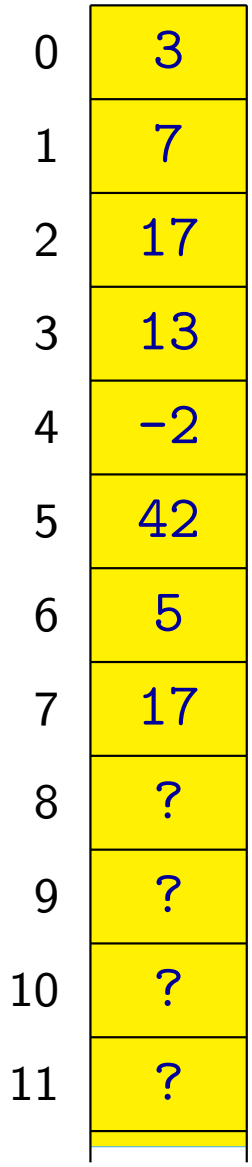


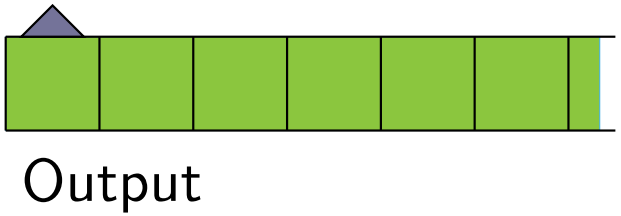


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

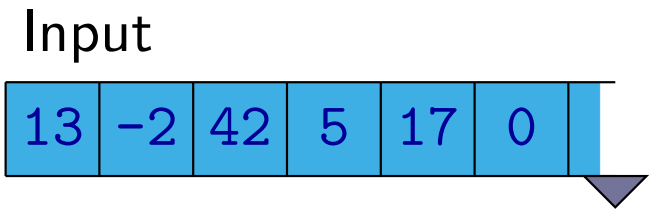


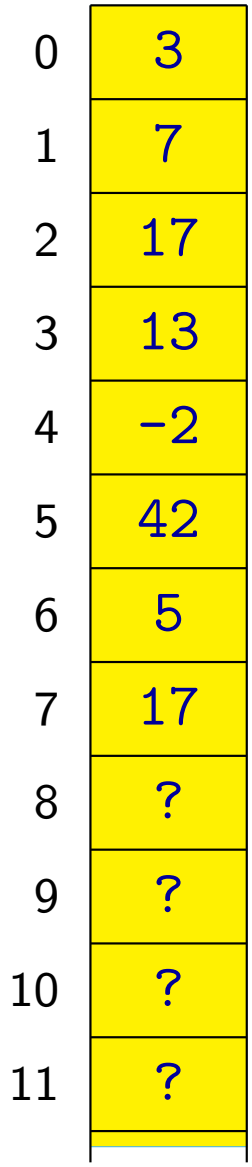


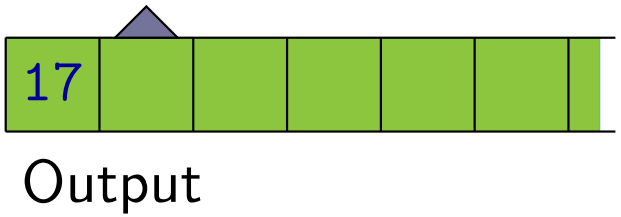


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

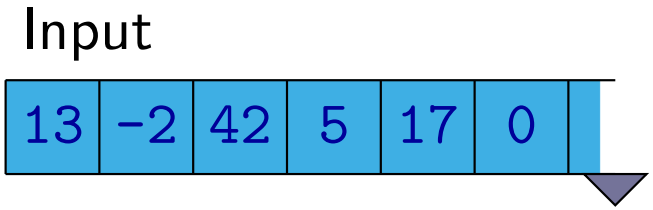


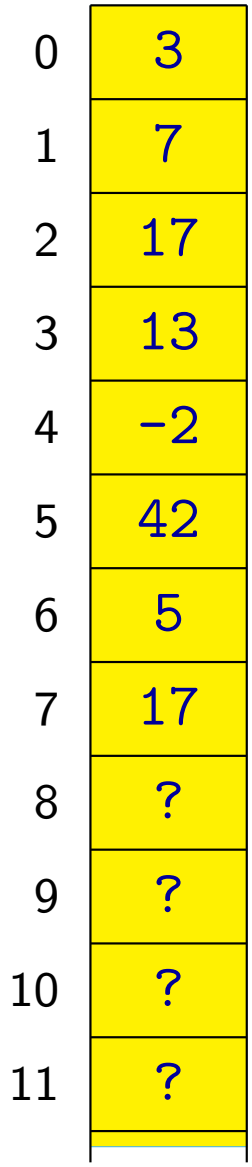


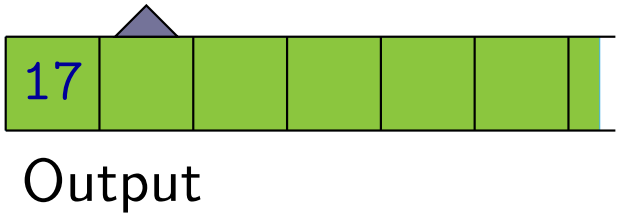


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

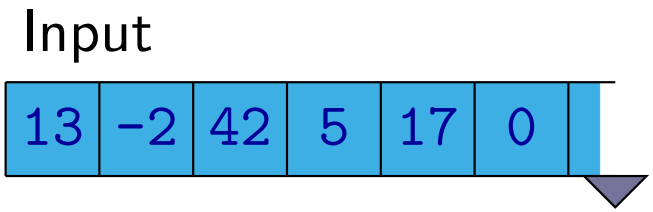


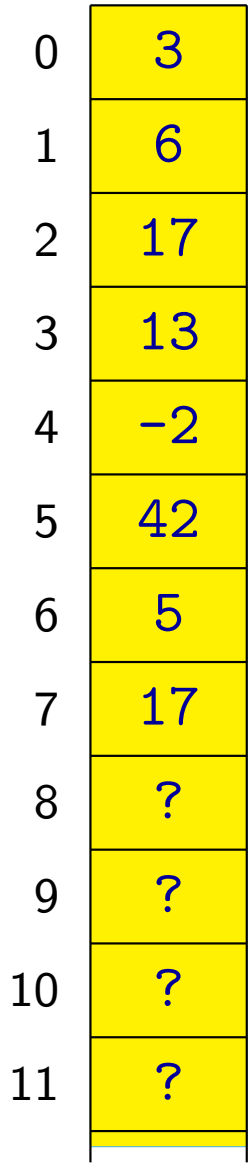


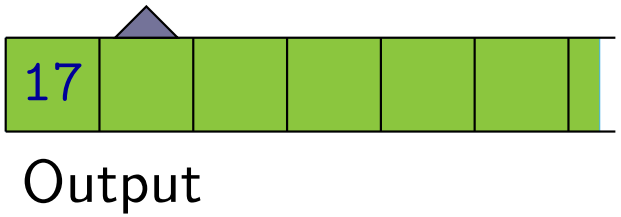


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

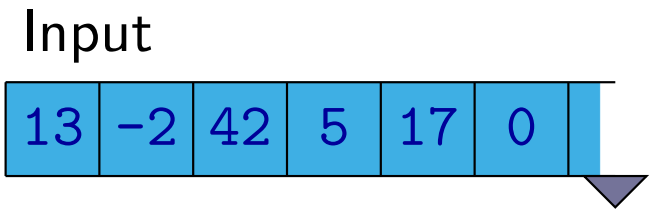


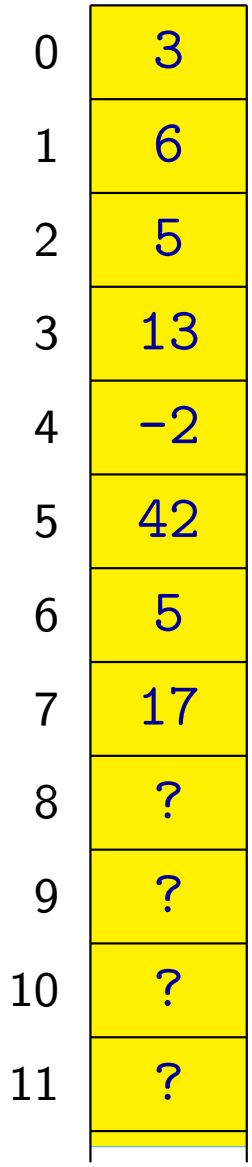


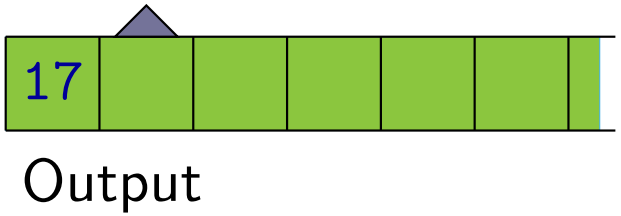


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

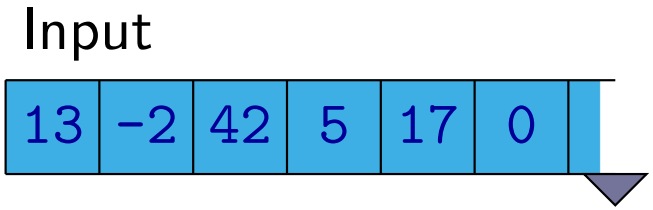


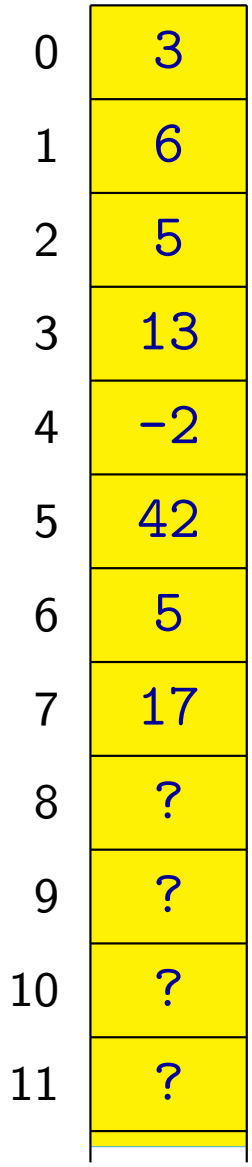


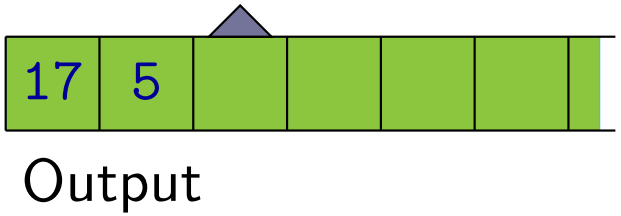


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
L_3 : **if** $(R_1 > R_0)$ **goto** L_2
halt

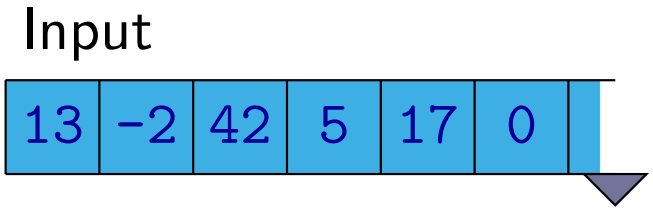


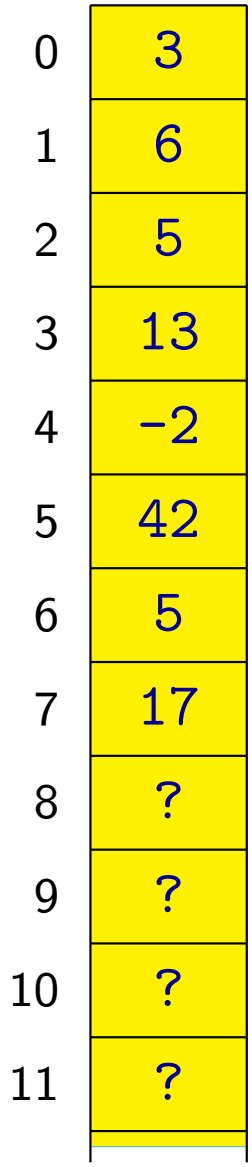


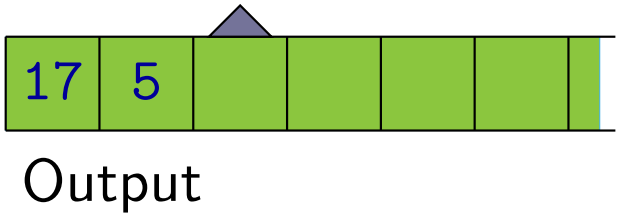


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

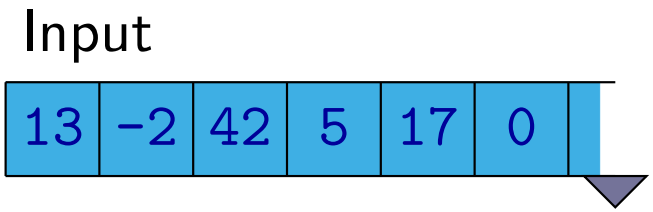


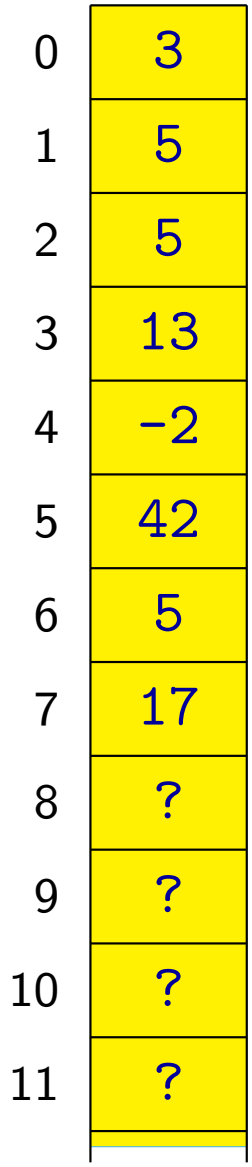


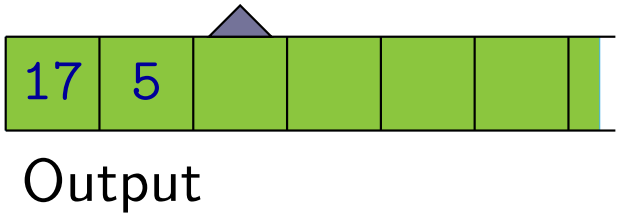


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

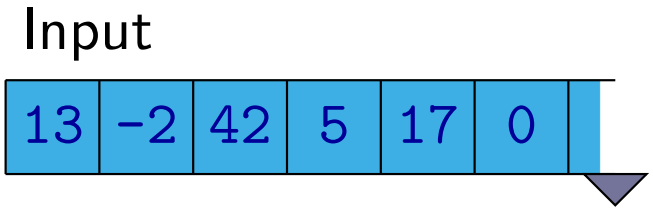


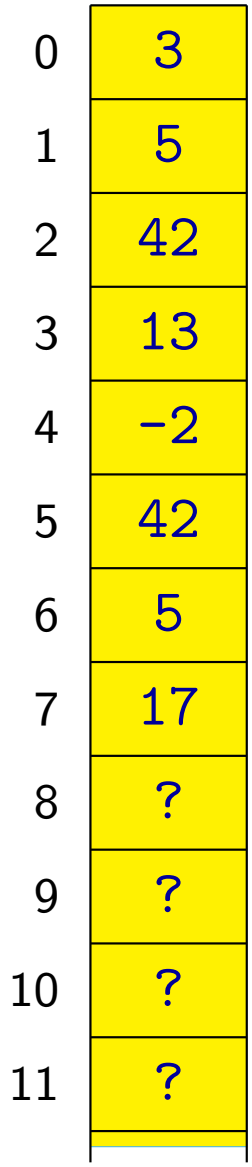


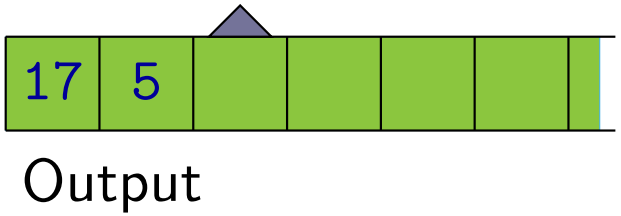


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

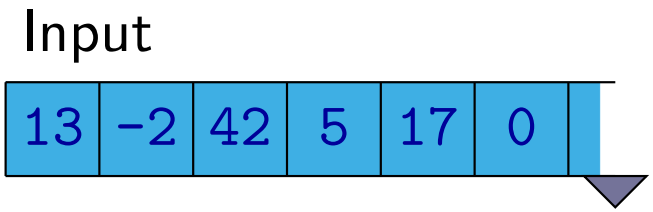


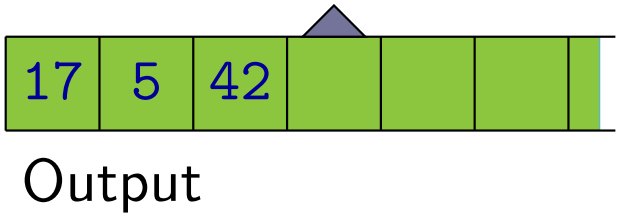


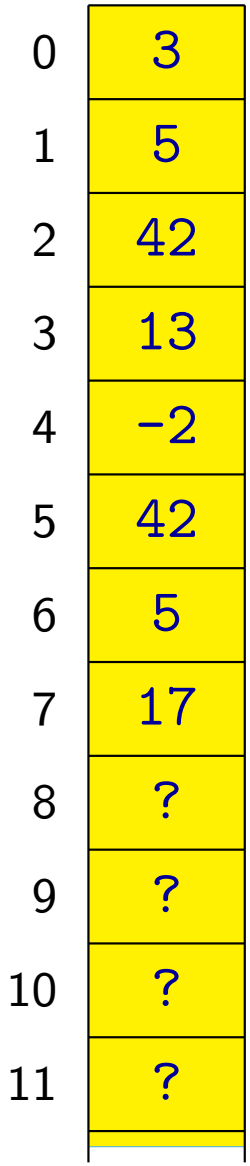


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

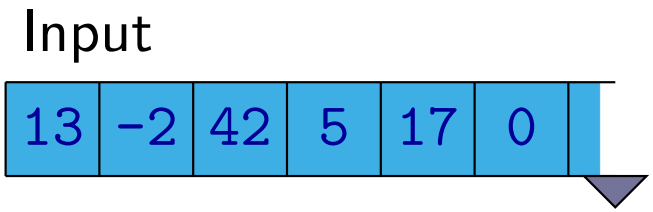


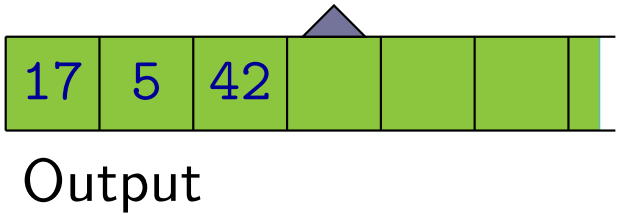


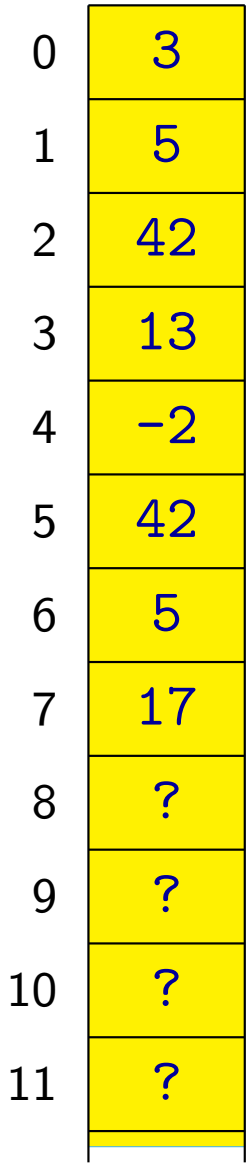


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

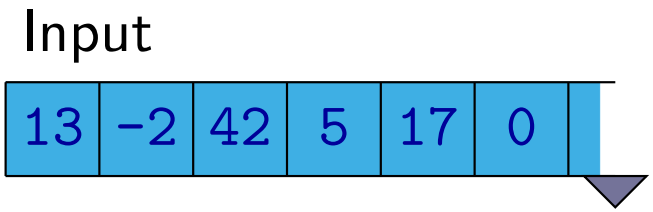


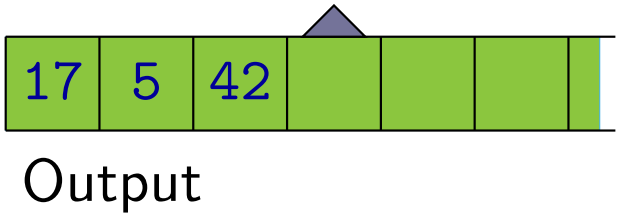


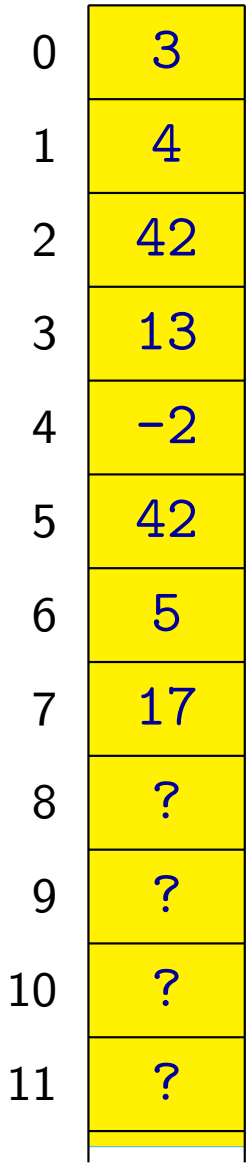


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

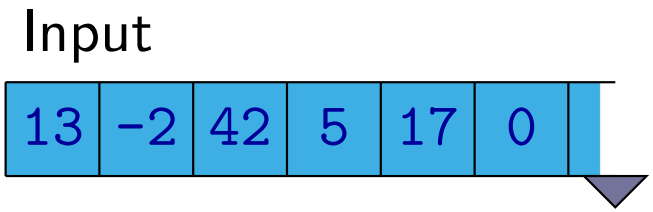


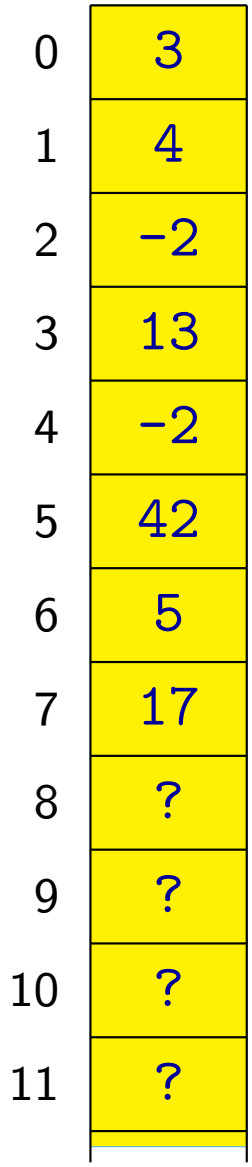


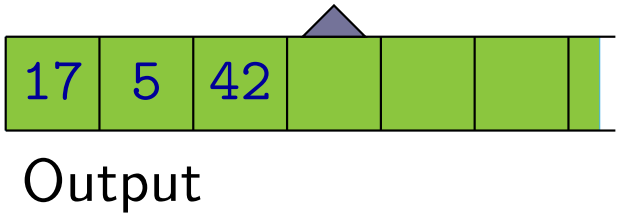


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

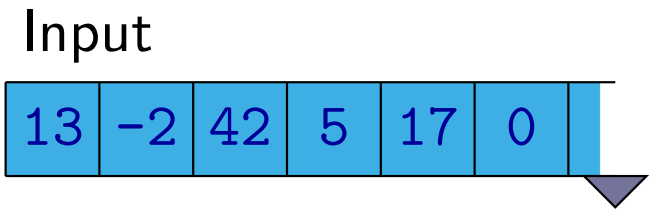


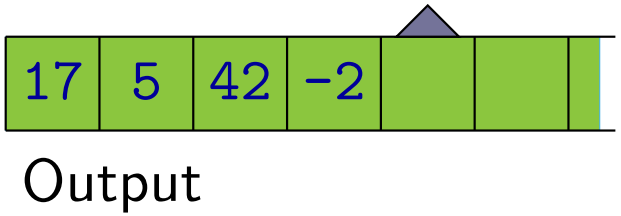


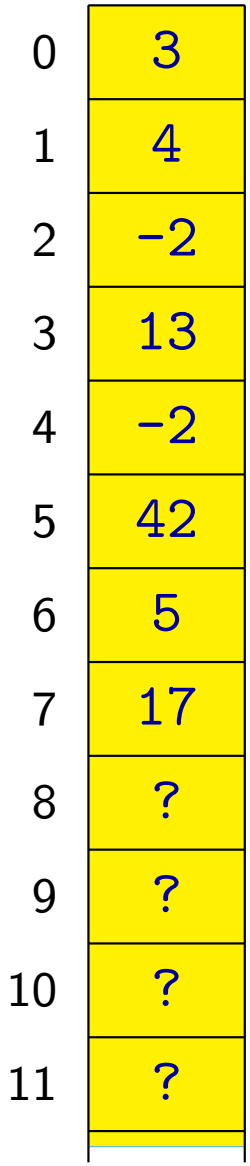


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

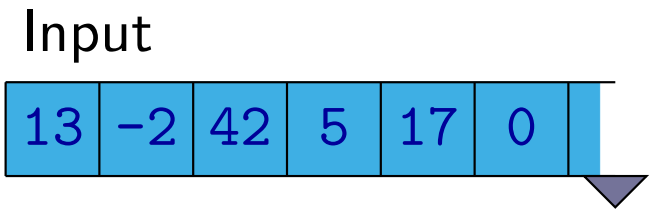


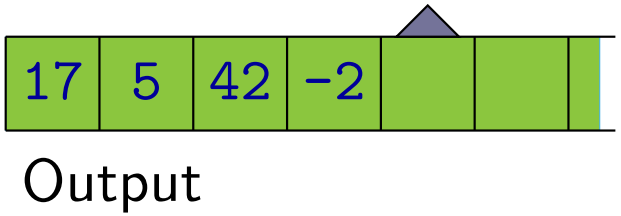


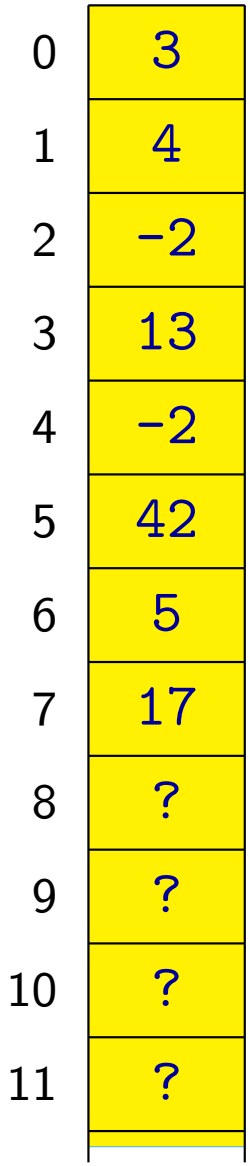


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

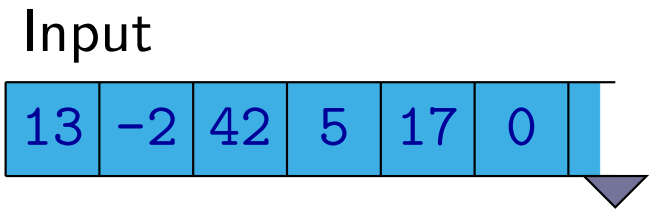


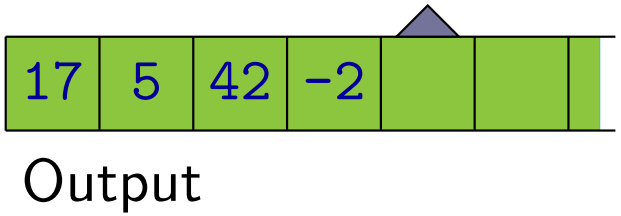


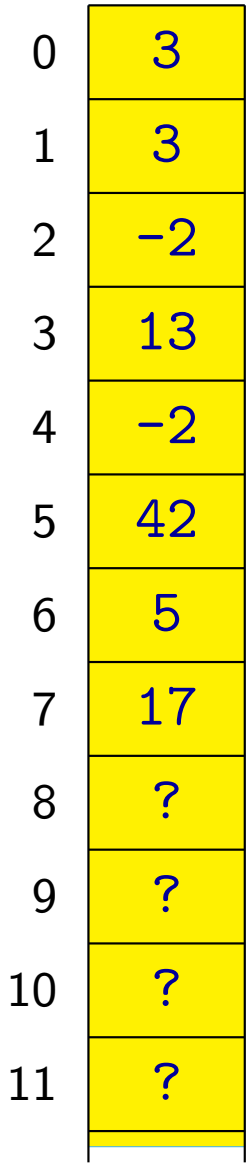


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

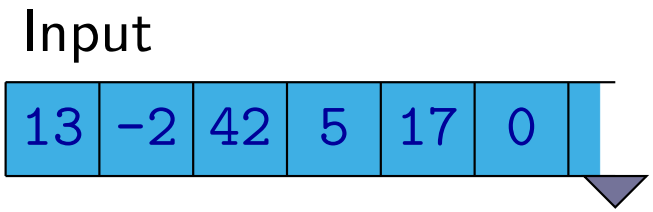


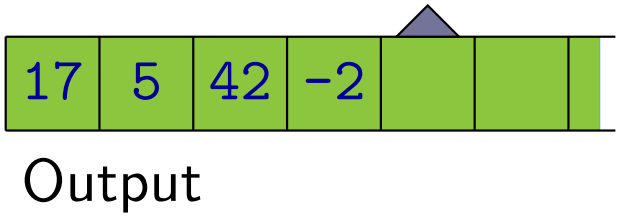


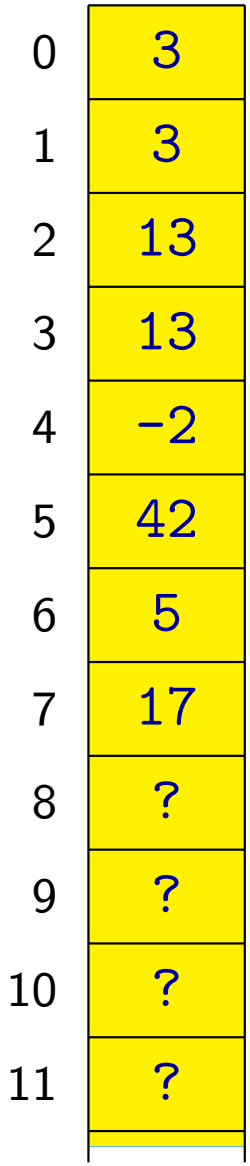


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

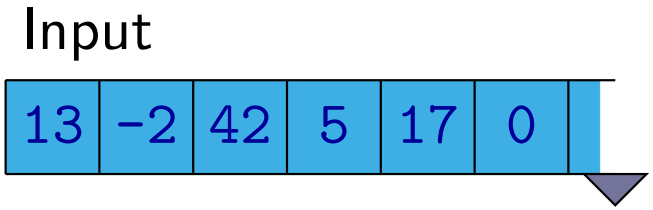


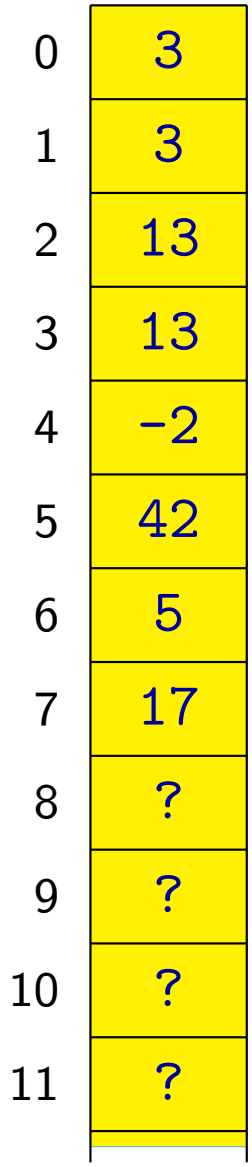


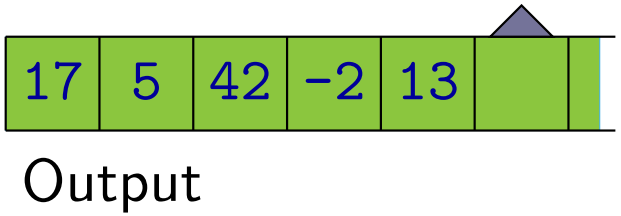


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

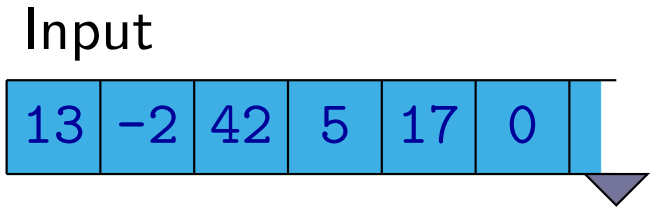


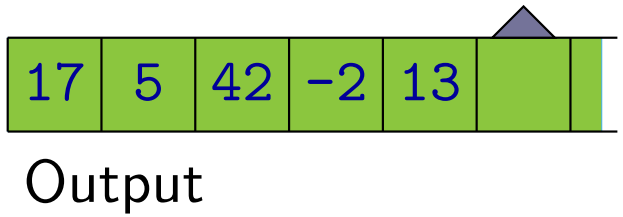


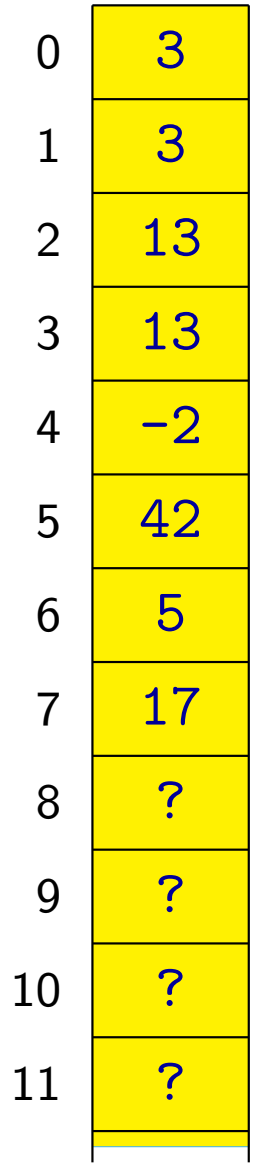


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt







Stroj RAM

Rozdíly oproti skutečnému počítači:

- Velikost paměti není omezena (adresa může být libovolné přirozené číslo).
- Velikost obsahu jednotlivých buněk není omezena (buňka může obsahovat libovolné celé číslo).
- Čte data sekvenčně ze vstupu, který je tvořen sekvencí celých čísel. Ze vstupu lze pouze číst.
- Zapisuje data sekvenčně na výstup, který je tvořen sekvencí celých čísel. Na výstup je možné pouze zapisovat.

Stroj RAM

- Operace jako přístup k buňce paměti na adrese menší než nula nebo dělení nulou vedou k chybě — výpočet se (neúspěšně) zastaví.

- Co se týká počátečního obsahu paměti, jsou dvě možnosti, jak ho definovat:
  - Všechny buňky jsou inicializovány hodnotou 0.
  - Čtení obsahu buňky, do které nebylo dosud nic zapsáno, způsobí chybu.

    Buňky na začátku obsahují speciální hodnotu (označenou zde symbolem '?'), která reprezentuje to, že buňka nebyla dosud inicializována.

- Uvažují se i varianty strojů RAM, kde buňky paměti (a vstupu a výstupu) neobsahují celá čísla (tj. prvky množiny $\mathbb{Z}$), ale mohou obsahovat jen přirozená čísla (tj. prvky množiny $\mathbb{N}$).

  Například operace odčítání ($R_i := R_j - R_k$) se pak chová tak, že pokud by výsledkem mělo být záporné číslo, je jako výsledek operace přiřazena hodnota 0.

Stroj RAM

- Různé varianty strojů RAM se mohou lišit tím, jaké konkrétní operace v aritmetických instrukcích podporují nebo naopak nepodporují.

  Například:

  - podpora bitových operací (and, or, not, xor, ...), bitový posunů, ...
  - varianta stroje RAM, která nemá operace násobení a dělení

- Mohli bychom také uvažovat variantu stroje RAM, kde místo instrukcí tvaru

  $$\textbf{if } (R_i \ rel \ R_j) \textbf{ goto } \ell \qquad \text{nebo} \qquad \textbf{if } (R_i \ rel \ c) \textbf{ goto } \ell$$

  jsou všechny podmíněné skoky jen tvaru

  $$\textbf{if } (R_i \ rel \ 0) \textbf{ goto } \ell$$

  Místo všech relací $\{=, \neq, \leq, \geq, <, >\}$ může být podporována jen nějaká podmnožina z nich, např. $\{=, >\}$.

Stroj RAM

- V některých variantách stroje RAM nemají vstup a výstup podobu sekvence čísel.

  Místo toho pracuje stroj z hlediska vstupu a výstupu s páskami obsahujícími sekvence symbolů z nějaké dané abecedy, např. $\{0, 1\}$.

  Stroj má pak například instrukce, které mu umožňují větvit výpočet podle symbolu přečteného ze vstupu.

  Vnitřní paměť ovšem i v této variantě pracuje s čísly.

- Pokud má stroj jako výsledek dávat jen odpověď Ano/Ne (tj. přijmout nebo nepřijmout daný vstup), nemusí mít výstupní pásku.

  Instrukce **halt** je pak nahrazena instrukcemi **accept** a **reject**.

Stroj RAM

- Ve standardní definici stroje RAM se většinou neuvažují instrukce skoku na adresu instrukce uloženou v buňce paměti, tj. instrukce typu

  goto R_i

  Stroj RAM bychom mohli rozšířit o tento druh instrukcí.

- Jako standardní se u stroje RAM bere to, že kód programu není uložen v pracovní paměti, ale má zvláštní samostatnou paměť, která je jen pro čtení.

  V průběhu výpočtu se tedy kód programu nemůže měnit.

Stroj RASP

- Druh stroje podobný stroji RAM, kde je ovšem program uložen v pracovní paměti (instrukce jsou kódovány čísly) a je možné ho průběhu výpočtu měnit, se označuje jako stroj **RASP** (**random-access stored program**).

  Stroj RASP tak umožňuje provádět činnost sebemodifikujících se programů.

Turingův stroj simulující činnost stroje RAM

Není těžké si rozmyslet, že činnost libovolného Turingova stroje je možné simulovat pomocí stroje RAM.

Promyslet si, že i naopak činnost každého stroje RAM je možné simulovat Turingovým strojem, je o něco komplikovanější.

Při popisu toho, jak simulovat činnost stroje RAM pomocí Turingova stroje, budeme postupovat po menších krocích:

- Ukážeme, jak činnost stroje RAM ve variantě, kterou jsme si popsali, simulovat variantou stroje RAM s poněkud jednodušími instrukcemi.
- Ukážeme, jak činnost této jednodušší varianty stroje RAM simulovat vícepáskovým Turingovým strojem.
- Už dříve jsme viděli, jak činnosti vícepáskového Turingovat stroje simulovat pomocí jednopáskového Turingovat stroje.

Jednodušší varianta stroje RAM

Tato jednodušší varianta stroje RAM bude mít kromě pracovní paměti tři **registry**:

- **registr A** — téměř všechny instrukce pracují s tímto registrem, výsledky všech operací se ukládají do tohoto registru

  Poznámka: Tento druh registru se často označuje jako **akumulátor**.

- **registr B** — tento registr slouží k uložení druhého operandu pro aritmetické instrukce (první operand je vždy v akumulátoru)

- **registr C** — tento registr slouží k uložení adresy, na kterou bude zapisovat instrukce store

Jednodušší varianta stroje RAM

Přehled instrukcí:

$A := c$	– přiřazení konstanty
$B := A$	– přiřazení do registru B
$C := A$	– přiřazení do registru C
$A := [A]$	– load (čtení z paměti)
$[C] := A$	– store (zápis do paměti)
$A := A \; op \; B$	– aritmetické instrukce, $op \in \{+, -, *, /\}$
if $(A \; rel \; 0)$ **goto** ℓ	– podmíněný skok, $rel \in \{=, \neq, \leq, \geq, <, >\}$
goto ℓ	– nepodmíněný skok
$A :=$ READ $()$	– čtení ze vstupu
WRITE (A)	– zápis na výstup
halt	– zastavení programu

Jednodušší varianta stroje RAM

Například instrukce

$$R_7 := R_3 + R_6$$

může být nahrazena posloupností instrukcí:

$$A := 7$$
$$C := A$$
$$A := 6$$
$$A := [A]$$
$$B := A$$
$$A := 3$$
$$A := [A]$$
$$A := A + B$$
$$[C] := A$$

Jednodušší varianta stroje RAM

Například instrukce

$$[R_{15}] := R_9$$

může být nahrazena posloupností instrukcí:

$$A := 15$$
$$A := [A]$$
$$C := A$$
$$A := 9$$
$$A := [A]$$
$$[C] := A$$

Jednodušší varianta stroje RAM

Například instrukce

$$\textbf{if } (R_4 \geq R_{11}) \textbf{ goto } \ell$$

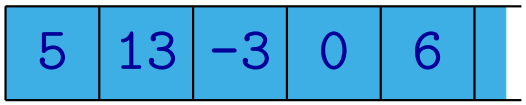
může být nahrazena posloupností instrukcí:

$$A := 11$$
$$A := [A]$$
$$B := A$$
$$A := 4$$
$$A := [A]$$
$$A := A - B$$
$$\textbf{if } (A \geq 0) \textbf{ goto } \ell$$

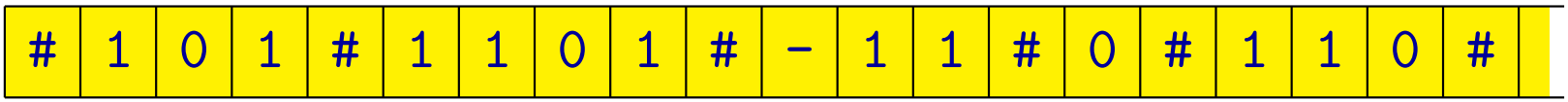
Turingův stroj simulující činnost stroje RAM

Turingův stroj pracuje se slovy nad nějakou abecedou, zatímco stroj RAM s čísly. Čísla ale můžeme zapisovat jako sekvence symbolů a naopak symboly nějaké abecedy můžeme zapisovat jako čísla.

Například následující vstup stroje RAM



může být v případě Turingova stroje reprezentován jako



Turingův stroj simulující činnost stroje RAM

Turingův stroj simulující činnost stroje RAM bude mít několik pásek:

- Pásku, na které bude uložen obsah pracovní paměti stroje RAM.
- Tři pásky, na kterých budou uloženy hodnoty registrů A, B a C.
  (Hodnoty registrů A, B a C budou na těchto páskách zapsány binárně bez vedoucích nul a zleva a zprava budou ohraničeny symboly #.)
- Pásku reprezentující vstupní pásku stroje RAM.
- Pásku reprezentující výstupní pásku stroje RAM.
- Jednu pomocnou pásku používanou při implementaci simulace jednotlivých instrukcí.

Turingův stroj simulující činnost stroje RAM

Turingův stroj si bude v řídící jednotce pamatovat, která instrukce stroje RAM se právě provádí.

Provedení většiny instrukcí není složité:

- $A := c$

  zapíše jednotlivé bity konstanty c na pásku registru A

- $B := A$ nebo $C := A$

  zkopíruje obsah pásky registru A na pásku registru B nebo C

- **goto** ℓ

  změní se jen stav řídící jednotky Turingova stroje

- **if** $(A \; rel \; 0)$ **goto** ℓ, kde $rel \in \{=, \neq, \leq, \geq, <, >\}$

  snadno se otestuje obsah registru A a podle výsledku se změní stav řídící jednotky Turingova stroje

Turingův stroj simulující činnost stroje RAM

- $A :=$ READ $()$

  zkopírování hodnoty (ohraničené znaky "#") ze vstupní pásky na pásku registru A

- WRITE (A)

  zkopírování hodnoty registru A na výstupní pásku.

- **halt**

  výpočet se zastaví

Turingův stroj simulující činnost stroje RAM

Také aritmetické instrukce jsou poměrně jednoduché, i když o něco složitější než předchozí instrukce:

- $A := A \; op \; B$, kde $op \in \{+, -, *, /\}$

  Příslušnou operaci (např. sčítání nebo odčítání) provede Turingův stroj bit po bitu, výsledek je ukládán do registru A.

Poznámka: Násobení a dělení je možné realizovat pomocí série sčítání, odčítání a bitových posunů.

Při implementaci násobení a dělení může být potřeba použít pomocnou pásku k ukládání mezivýsledků.

Turingův stroj simulující činnost stroje RAM

Asi nejsložitější je realizace pracovní paměti stroje RAM.

Jednou z možností je pamatovat si jen obsah těch buněk, se kterými stroj RAM v průběhu své činnosti někdy pracoval.

Příklad: Stroj RAM zatím pracoval jen s buňkami 2, 3 a 6:

- Buňka 2 obsahuje hodnotu 11.
- Buňka 3 obsahuje hodnotu -1.
- Buňka 6 obsahuje hodnotu 2.

Obsah pásky Turingova stroje reprezentující buňky paměti stroje RAM bude následující:

$	#	1	0	:	1	0	1	1	#	1	1	:	-	1	#	1	1	0	:	1	0	#	$	

Turingův stroj simulující činnost stroje RAM

Instrukce load, tj. $A := [A]$:

- Turingův stroj bude hledat příslušnou adresu uloženou v registru A na pásce reprezentující obsah paměti stroje RAM.
  (Pokud ji nenajdeme, přidá ji na konec, s tím, že obsahuje hodnotu 0.)
- Příslušnou hodnotu zkopíruje na pásku registru A.

Turingův stroj simulující činnost stroje RAM

Instrukce store, tj. $[C] := A$:

- Podobně jako u instrukce load se najde příslušné místo na pásce reprezentující pracovní paměť, kde se nachází obsah buňky, jejíž adresa je v registru C.
- Zbytek pásky s obsahem paměti stroje RAM se zkopíruje na pomocnou pásku.
- Na příslušné místo se zkopíruje obsah pásky registru A.
- Zbytek pásky, který byl zkopírován na pomocnou pásku, se zkopíruje zpět (za nově zapsanou hodnotu).

Algoritmy

Pseudokód

Algoritmy většinou nebudeme zapisovat jako programy pro stroj RAM, ale spíše jako programy v nějakém vyšším programovacím jazyce.

Nebudeme se vázat na nějaký konkrétní programovací jazyk.

Programy budeme zapisovat pomocí **pseudokódu**, jehož syntaxi si budeme libovolně přizpůsobovat podle potřeby (např. použití libovolné matematické notace, slovních popisů, apod.).

Příklad:

Algoritmus: Algoritmus pro nalezení největšího prvku v poli

```
FIND-MAX (A, n):
    k := 0
    for i := 1 to n − 1 do
        if A[i] > A[k] then
            k := i
    return A[k]
```

Algoritmy

Poznámka:

Z hlediska analýzy toho, jak daný algoritmus funguje, většinou není příliš podstatný rozdíl v tom, jestli algoritmus:

- čte vstupní data z nějakého vstupního zařízení (např. ze souboru na disku, z klávesnice, apod.)
- zapisuje data na nějaké výstupní zařízení (např. do souboru, na obrazovku, apod.)

nebo

- čte vstupní data z paměti (např. jsou mu předány jako parametry)
- zapisuje data na do paměti (např. je vrátí jako návratovou hodnotu)

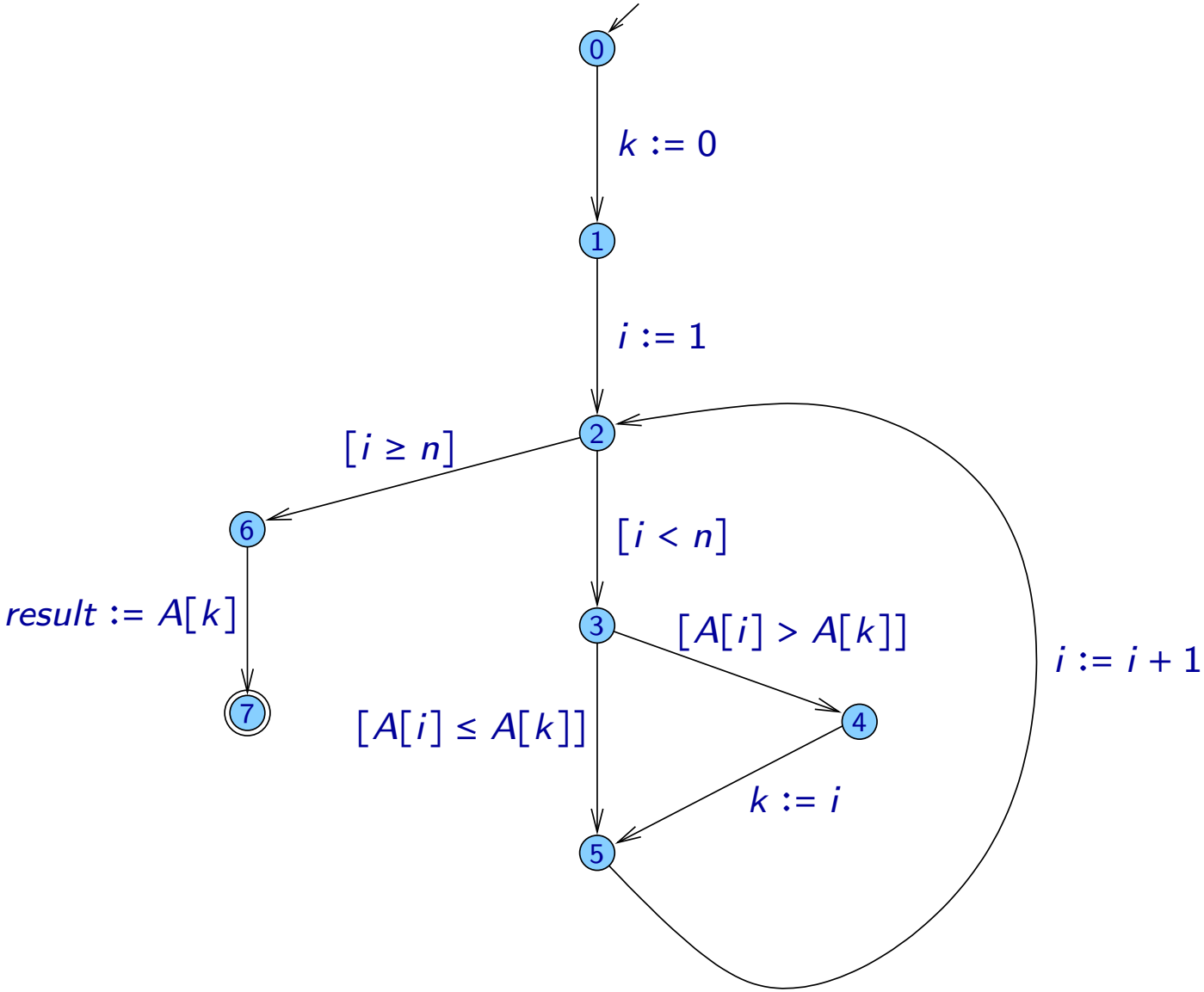
V pseudokódu tak tedy většinou budou vstupní data předávána jako argumenty dané funkce a výstup bude představován návratovou hodnotou této funkce.

Řídící tok

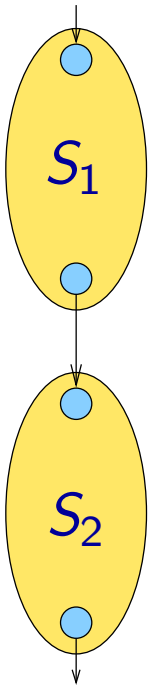
Instrukce lze zhruba rozdělit na dvě skupiny:

- instrukce přímo pracující s daty:
  - přiřazení
  - vyhodnocení hodnot výrazů v podmínkách
  - čtení vstupu, zápis na výstup
  - ...
- instrukce ovlivňující **řídící tok** — určují, které instrukce se budou provádět, v jakém pořadí, apod.:
  - větvení (if, switch, ...)
  - cykly (while, do .. while, for, ...)
  - uspořádání instrukcí do bloků
  - návraty z podpogramů (return, ...)
  - ...

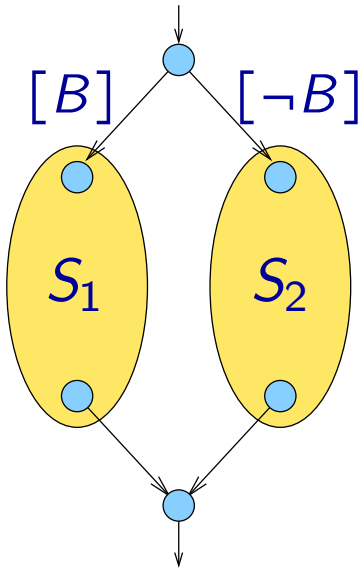
Graf řídícího toku



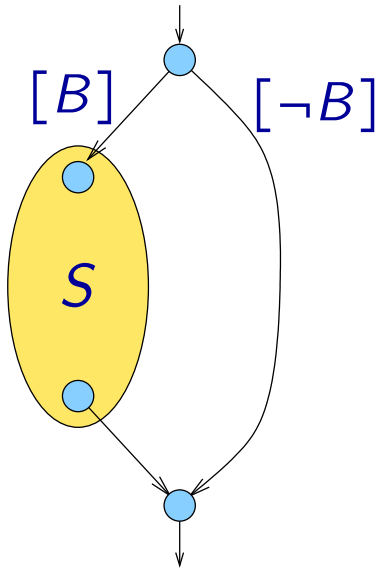
Některé základní konstrukce strukturovaného programování



$S_1; S_2$

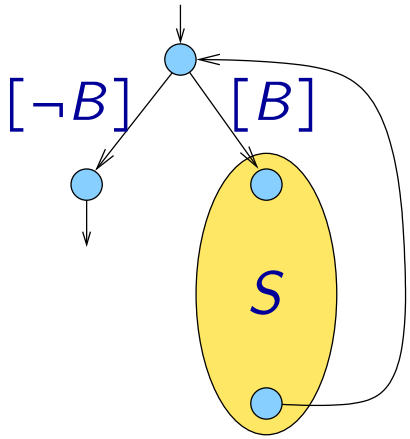


if B **then** S_1 **else** S_2

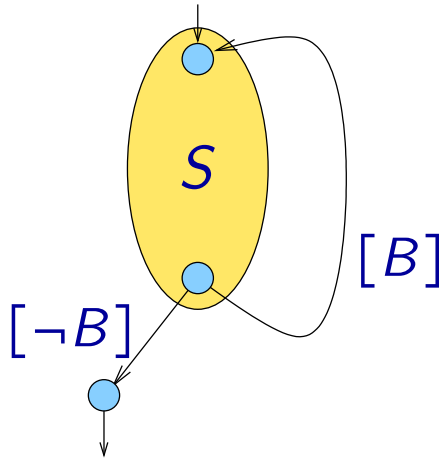


if B **then** S

Některé základní konstrukce strukturovaného programování

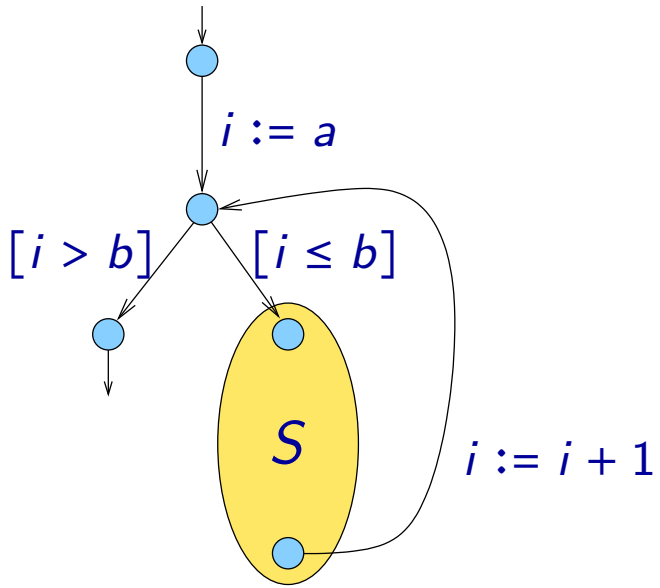


while B **do** S



do S **while** B

Některé základní konstrukce strukturovaného programování



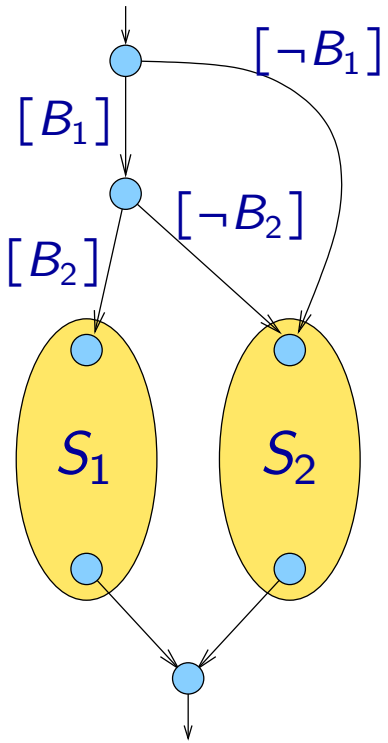
$i := a$
while $i \leq b$ **do**
  S
  $i := i + 1$

for $i := a$ **to** b **do** S

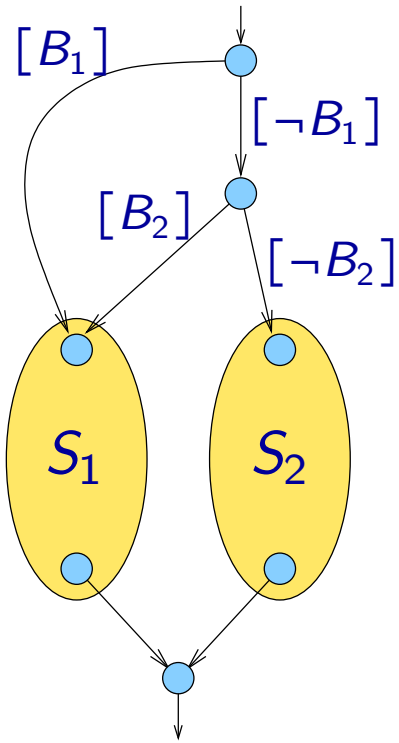
Některé základní konstrukce strukturovaného programování

Zkrácené vyhodnocování složených podmínek, např.:

while $i < n$ **and** $A[i] > x$ **do** ...



if B_1 **and** B_2 **then** S_1 **else** S_2



if B_1 **or** B_2 **then** S_1 **else** S_2

Řídící tok realizovaný pomocí goto

- **goto** ℓ — **nepodmíněný skok**
- **if** B **then goto** ℓ — **podmíněný skok**

Příklad:

0: $k := 0$
1: $i := 1$
2: **goto** *6*
3: **if** $A[i] \leq A[k]$ **then goto** *5*
4: $k := i$
5: $i := i + 1$
6: **if** $i < n$ **then goto** *3*
7: **return** $A[k]$

Řídící tok realizovaný pomocí goto

- **goto** ℓ — **nepodmíněný skok**
- **if** B **then goto** ℓ — **podmíněný skok**

Příklad:

$$
\begin{array}{rl}
\mathit{start}: & k := 0 \\
& i := 1 \\
& \textbf{goto } \mathit{L3} \\
\mathit{L1}: & \textbf{if } A[i] \leq A[k] \textbf{ then goto } \mathit{L2} \\
& k := i \\
\mathit{L2}: & i := i + 1 \\
\mathit{L3}: & \textbf{if } i < n \textbf{ then goto } \mathit{L1} \\
& \textbf{return } A[k]
\end{array}
$$

Vyhodnocení složitých výrazů

Vyhodnocení složitého výrazu, jako třeba

$$A[i + s] := (B[3 * j + 1] + x) * y + 8$$

může být na nižší úrovni nahrazeno posloupností jednodušších příkazů, jako třeba

$$\begin{aligned} t_1 &:= i + s \\ t_2 &:= 3 * j \\ t_2 &:= t_2 + 1 \\ t_3 &:= B[t_2] \\ t_3 &:= t_3 + x \\ t_3 &:= t_3 * y \\ t_3 &:= t_3 + 8 \\ A[t_1] &:= t_3 \end{aligned}$$

Výpočet algoritmu

Konfigurace — popis celkového stavu stroje v nějakém okamžiku během výpočtu

Příklad: Konfigurace tvaru

$$(q, mem)$$

kde

- q — aktuální řídící stav
- mem — představuje aktuální obsah paměti stroje — jaké hodnoty jsou momentálně přiřazeny jednotlivým proměnným.

Příklad obsahu paměti mem:

$$\langle A\colon [3,8,1,3,6],\quad n\colon 5,\quad i\colon 1,\quad k\colon 0,\quad result\colon ?\rangle$$

Výpočet algoritmu

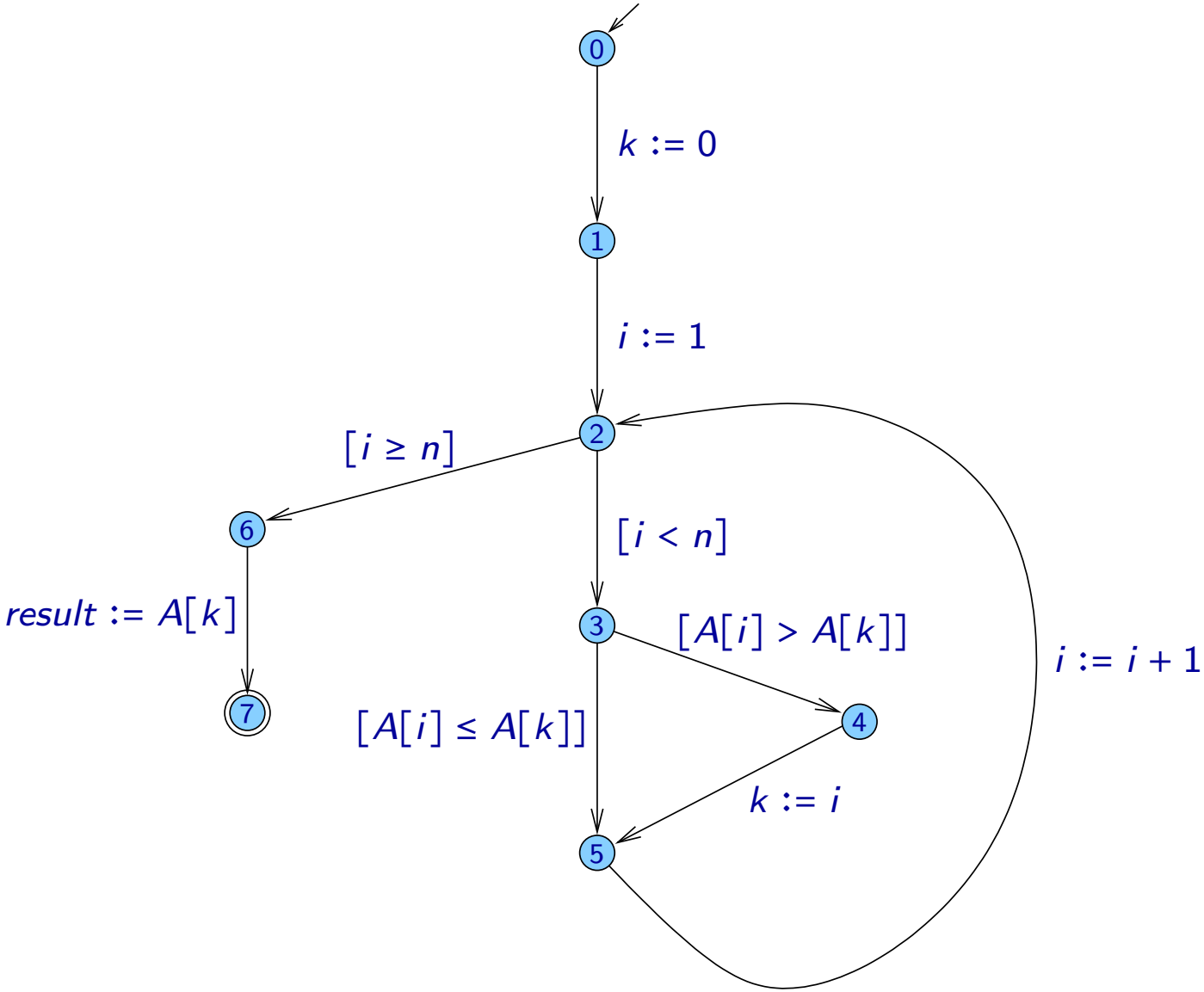
Příklad konfigurace:

$$(2,\ \langle A\colon [3,8,1,3,6],\quad n\colon 5,\quad i\colon 1,\quad k\colon 0,\quad \mathit{result}\colon ?\rangle)$$

Výpočet stroje $\mathcal{M}$ provádějícího algoritmus Alg, kde zpracovává vstup w, je posloupnost konfigurací.

- Začíná se v **počáteční konfiguraci**.
- Každým krokem stroj přechází z jedné konfigurace do další.
- Výpočet končí v **koncové konfiguraci**.

Výpočet algoritmu



Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\text{: } [3, 8, 1, 3, 6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } ?, \quad \mathit{result}\text{: } ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } ?, \quad \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } ?, \quad result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } ?, \quad result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } ?, \quad result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } ?, \quad \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 1, \quad \mathit{result}\text{: } ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon ?,\ result\colon ?\rangle)$
α_1: $(1, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon 0,\ result\colon ?\rangle)$
α_2: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_3: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_4: $(4, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_5: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 1,\ result\colon ?\rangle)$
α_6: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon ?,\ result\colon ?\rangle)$
α_1: $(1, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon 0,\ result\colon ?\rangle)$
α_2: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_3: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_4: $(4, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_5: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 1,\ result\colon ?\rangle)$
α_6: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_7: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } ?, \quad result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 2, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 2, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 2, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } ?, \quad result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 2, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 2, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 2, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 3, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } ?, \quad \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 1, \quad \mathit{result}\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 2, \quad k\text{: } 1, \quad \mathit{result}\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 2, \quad k\text{: } 1, \quad \mathit{result}\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 2, \quad k\text{: } 1, \quad \mathit{result}\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 3, \quad k\text{: } 1, \quad \mathit{result}\text{: } ?\rangle)$
α_{10}: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 3, \quad k\text{: } 1, \quad \mathit{result}\text{: } ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } ?, \quad \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 1, \quad \mathit{result}\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 2, \quad k\text{: } 1, \quad \mathit{result}\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 2, \quad k\text{: } 1, \quad \mathit{result}\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 2, \quad k\text{: } 1, \quad \mathit{result}\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 3, \quad k\text{: } 1, \quad \mathit{result}\text{: } ?\rangle)$
α_{10}: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 3, \quad k\text{: } 1, \quad \mathit{result}\text{: } ?\rangle)$
α_{11}: $(5, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 3, \quad k\text{: } 1, \quad \mathit{result}\text{: } ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } ?, \quad result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 2, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 2, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 2, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 3, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$
α_{10}: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 3, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$
α_{11}: $(5, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 3, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$
α_{12}: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 4, \quad k\text{: } 1, \quad result\text{: } ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon ?,\ result\colon ?\rangle)$
α_1: $(1, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon 0,\ result\colon ?\rangle)$
α_2: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_3: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_4: $(4, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_5: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 1,\ result\colon ?\rangle)$
α_6: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_7: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_8: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_9: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{10}: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{11}: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{12}: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$
α_{13}: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon ?,\ result\colon ?\rangle)$
α_1: $(1, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon 0,\ result\colon ?\rangle)$
α_2: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_3: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_4: $(4, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_5: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 1,\ result\colon ?\rangle)$
α_6: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_7: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_8: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_9: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{10}: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{11}: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{12}: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$
α_{13}: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$
α_{14}: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon ?, \; k\colon ?, \; result\colon ?\rangle)$
α_1: $(1, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon ?, \; k\colon 0, \; result\colon ?\rangle)$
α_2: $(2, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 1, \; k\colon 0, \; result\colon ?\rangle)$
α_3: $(3, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 1, \; k\colon 0, \; result\colon ?\rangle)$
α_4: $(4, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 1, \; k\colon 0, \; result\colon ?\rangle)$
α_5: $(5, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 1, \; k\colon 1, \; result\colon ?\rangle)$
α_6: $(2, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 2, \; k\colon 1, \; result\colon ?\rangle)$
α_7: $(3, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 2, \; k\colon 1, \; result\colon ?\rangle)$
α_8: $(5, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 2, \; k\colon 1, \; result\colon ?\rangle)$
α_9: $(2, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 3, \; k\colon 1, \; result\colon ?\rangle)$
α_{10}: $(3, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 3, \; k\colon 1, \; result\colon ?\rangle)$
α_{11}: $(5, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 3, \; k\colon 1, \; result\colon ?\rangle)$
α_{12}: $(2, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 4, \; k\colon 1, \; result\colon ?\rangle)$
α_{13}: $(3, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 4, \; k\colon 1, \; result\colon ?\rangle)$
α_{14}: $(5, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 4, \; k\colon 1, \; result\colon ?\rangle)$
α_{15}: $(2, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 5, \; k\colon 1, \; result\colon ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon ?, \; k\colon ?, \; result\colon ?\rangle)$
α_1: $(1, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon ?, \; k\colon 0, \; result\colon ?\rangle)$
α_2: $(2, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 1, \; k\colon 0, \; result\colon ?\rangle)$
α_3: $(3, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 1, \; k\colon 0, \; result\colon ?\rangle)$
α_4: $(4, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 1, \; k\colon 0, \; result\colon ?\rangle)$
α_5: $(5, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 1, \; k\colon 1, \; result\colon ?\rangle)$
α_6: $(2, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 2, \; k\colon 1, \; result\colon ?\rangle)$
α_7: $(3, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 2, \; k\colon 1, \; result\colon ?\rangle)$
α_8: $(5, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 2, \; k\colon 1, \; result\colon ?\rangle)$
α_9: $(2, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 3, \; k\colon 1, \; result\colon ?\rangle)$
α_{10}: $(3, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 3, \; k\colon 1, \; result\colon ?\rangle)$
α_{11}: $(5, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 3, \; k\colon 1, \; result\colon ?\rangle)$
α_{12}: $(2, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 4, \; k\colon 1, \; result\colon ?\rangle)$
α_{13}: $(3, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 4, \; k\colon 1, \; result\colon ?\rangle)$
α_{14}: $(5, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 4, \; k\colon 1, \; result\colon ?\rangle)$
α_{15}: $(2, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 5, \; k\colon 1, \; result\colon ?\rangle)$
α_{16}: $(6, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 5, \; k\colon 1, \; result\colon ?\rangle)$

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

α_0: $(0, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon ?,\ result\colon ?\rangle)$
α_1: $(1, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon 0,\ result\colon ?\rangle)$
α_2: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_3: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_4: $(4, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_5: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 1,\ result\colon ?\rangle)$
α_6: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_7: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_8: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_9: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{10}: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{11}: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{12}: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$
α_{13}: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$
α_{14}: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$
α_{15}: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 5,\ k\colon 1,\ result\colon ?\rangle)$
α_{16}: $(6, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 5,\ k\colon 1,\ result\colon ?\rangle)$
α_{17}: $(7, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 5,\ k\colon 1,\ result\colon 8\rangle)$

Výpočet algoritmu

Provedením instrukce I se přejde z konfigurace α do konfigurace α':

$$\alpha \xrightarrow{I} \alpha'$$

Výpočet může být:

- **Konečný**:

$$\alpha_0 \xrightarrow{I_0} \alpha_1 \xrightarrow{I_1} \alpha_2 \xrightarrow{I_2} \alpha_3 \xrightarrow{I_3} \alpha_4 \xrightarrow{I_4} \cdots \xrightarrow{I_{t-2}} \alpha_{t-1} \xrightarrow{I_{t-1}} \alpha_t$$

  kde α_t je buď koncová konfigurace nebo konfigurace, kde došlo k chybě a není možné pokračovat

- **Nekonečný**:

$$\alpha_0 \xrightarrow{I_0} \alpha_1 \xrightarrow{I_1} \alpha_2 \xrightarrow{I_2} \alpha_3 \xrightarrow{I_3} \alpha_4 \xrightarrow{I_4} \cdots$$

Výpočet algoritmu

Výpočet je možné popsat dvěma různými způsoby:

- jako posloupnost konfigurací $\alpha_0, \alpha_1, \alpha_2, \ldots$
- jako posloupnost provedených instrukcí $I_0, I_1, I_2, \ldots$

Churchova-Turingova teze

Z přechozího by mělo být jasné, že:

- Program v libovolném programovacím jazyce je možné přeložit do podoby programu pro stroj RAM.
- Činnost stroje RAM je možné simulovat Turingovým strojem.

Činost každého programu v nějakém libovolném programovacím jazyce je tedy možné vykonávat Turingovým strojem.

Churchova-Turingova teze

Každý algoritmus je možné realizovat nějakým Turingovým strojem.

Není to věta, kterou by bylo možno dokázat v matematickém smyslu – není formálně definováno, co je to algoritmus.

Tezi formulovali nezávisle na sobě v polovině 30. let 20. století Alan Turing a Alonzo Church.

Churchova-Turingova teze

Příklady matematických formalismů zachycujících pojem algoritmus:

- Turingovy stroje
- stroje RAM
- lambda kalkulus
- rekurzivní funkce
- ...

Dále můžeme uvést:

- Libovolný (obecný) programovací jazyk (jako např. C, Java, Python, Lisp, Haskell, Prolog apod.).

Všechny tyto modely jsou ekvivalentní z hlediska algoritmů, které jsou schopny realizovat.

Dokazování korektnosti algoritmů

Korektnost algoritmu

Algoritmy slouží k řešení **problémů**.

- **Problém** — specifikace toho, **co** má algoritmus dělat:
  - Popis vstupu
  - Popis výstupu
  - Vztah mezi vstupy a výstupy
- **Algoritmus** — konkrétní postup, **jak** při výpočtu postupovat

Algoritmus je korektním řešením daného problému, jestliže se pro všechny vstupy zastaví a vydá správný výsledek.

Příklad:

Problém: Problém třídění
Algoritmus: Quicksort

Korektnost algoritmu

Příklad:

Problém nalezení maximálního prvku v poli:

Vstup: Pole A indexované od nuly a číslo n udávající počet prvků v tomto poli, přičemž se předpokládá, že $n \geq 1$.

Výstup: Hodnota *result*, která je hodnotou maximálního prvku v poli A, tj. hodnota *result*, pro kterou platí:

- $A[j] \leq result$ pro všechna $j \in \mathbb{N}$, kde $0 \leq j < n$, a
- existuje $j \in \mathbb{N}$ takové, že $0 \leq j < n$ a $A[j] = result$.

Instance problému — konkrétní vstup, např.

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$

Pro tuto instanci je výstupem hodnota 11.

Korektnost algoritmu

Algoritmus: Algoritmus pro nalezení největšího prvku v poli

```
FIND-MAX (A, n):
    k := 0
    for i := 1 to n − 1 do
        if A[i] > A[k] then
            k := i
    return A[k]
```

Korektnost algoritmu

Definice

Algoritmus Alg **řeší** problém P, jestliže pro **každou** instanci w problému P jsou splněny následující dvě podmínky:

(a) Výpočet algoritmu Alg nad vstupem w se po konečném počtu kroků (korektně) zastaví.

(b) Algoritmus Alg vygeneruje pro vstup w výstup, který odpovídá podmínkám kladeným na výstup ve specifikaci problému P.

Algoritmus, který řeší problém P, je korektním řešením tohoto problému.

Korektnost algoritmu

Algoritmus *Alg* **není** korektním řešení problému *P*, jestliže existuje vstup *w* takový, že při výpočtu nad tímto vstupem nastane některá z následujících chyb:

- provedení nějaké chybné nepovolené operace (přístup k prvku pole mimo povolený rozsah indexů, dělení nulou, ...),
- vygenerovaný výstup neodpovídá podmínkám specifikovaným v zadání problému *P*,
- výpočet se nikdy nezastaví.

Testování — spustění algoritmu nad různými vstupy a zkontrolování, zda se algoritmus pro tyto vstupy chová „správně“.

Testování může prokázat přítomnost chyb, ale ne to, že se algoritmus chová korektně pro **všechny** vstupy.

Korektnost algoritmu

Typicky je množina možných instancí daného problému nekonečná (nebo přinejmenším velmi velká), takže není možné otestovat činnost algoritmu na všech instancích.

Pro zdůvodnění a ověření toho, že algoritmus je korektním řešením daného problému je třeba podat **důkaz**, který bere v úvahu všechny možné výpočty na všech možných vstupech.

Důkaz korektnosti algoritmu je obecně vhodné rozdělit na dvě části:

- Zdůvodnění toho, že algoritmus pro žádný vstup nikdy neudělá nic „špatně“:
  - během výpočtu nedojde k žádné chybné operaci
  - pokud program skončí, výstup bude „správně“
- Zdůvodnění toho, že se algoritmus pro každý vstup po konečném počtu kroků zastaví.

Invarianty

Uvažujme libovolný systém skládající se z:

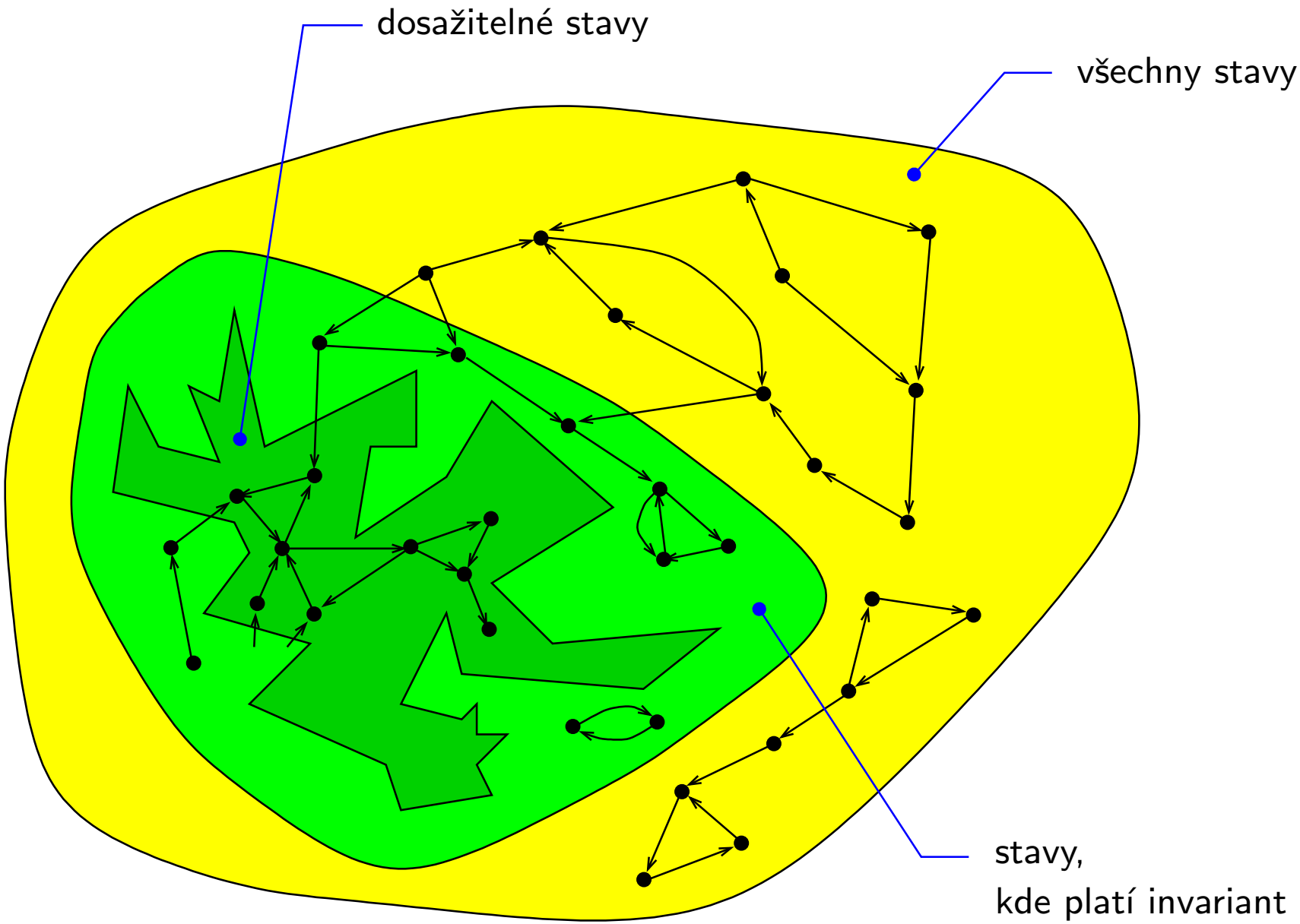
- množiny stavů (či konfigurací) — může být nekonečná
- přechodů mezi těmito stavy
- některé ze stavů jsou určené jako počáteční

Stav je **dosažitelný**, jestliže je možné se do něj dostat z některého počátečního stavu použitím nějaké posloupnosti přechodů.

Invariant je nějaká podmínka vymezující nějakou podmnožinu stavů taková, že platí ve všech dosažitelných stavech:

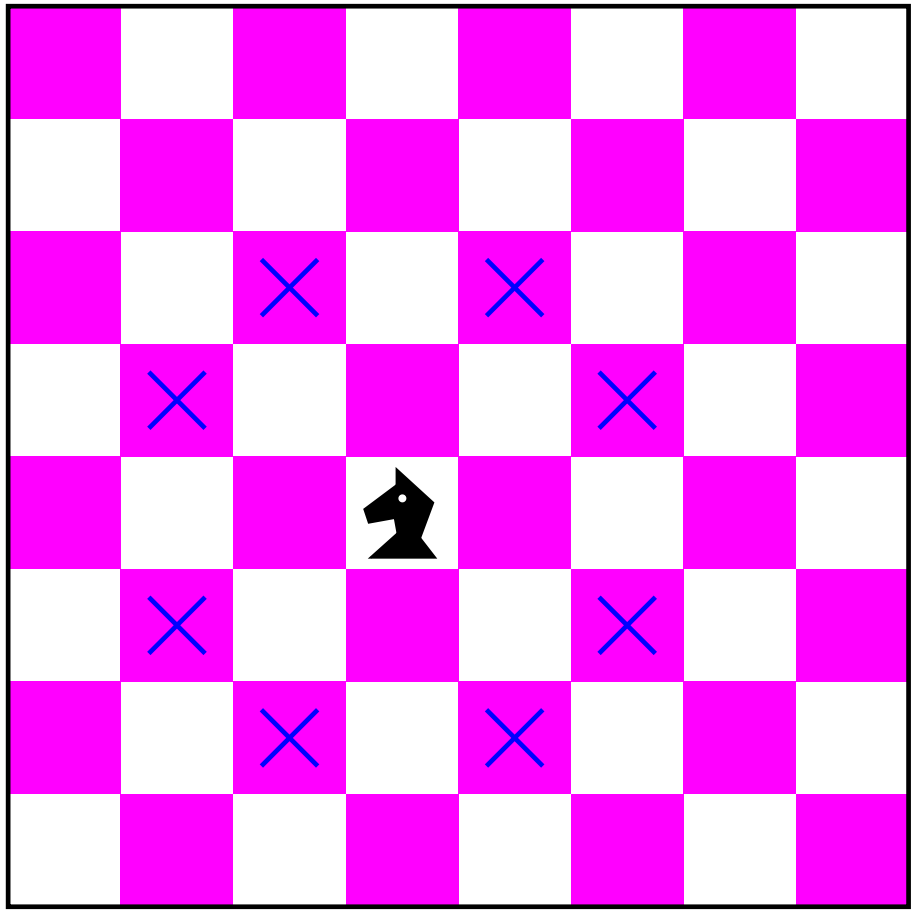
- je splněna ve všech počátečních stavech
- pokud je splněna v nějakém stavu a z tohoto stavu systém přejde jedním krokem do nějakého dalšího stavu, bude splněna i v tomto stavu

Invarianty


dosažitelné stavy
všechny stavy
stavy,
kde platí invariant

Invarianty

Příklad: Budeme skákat s figurkou jezdce po šachovnici a zároveň budeme počítat počty provedených tahů, přičemž začínáme na nějakém bílém poli v nejlevějším sloupci:



Invarianty

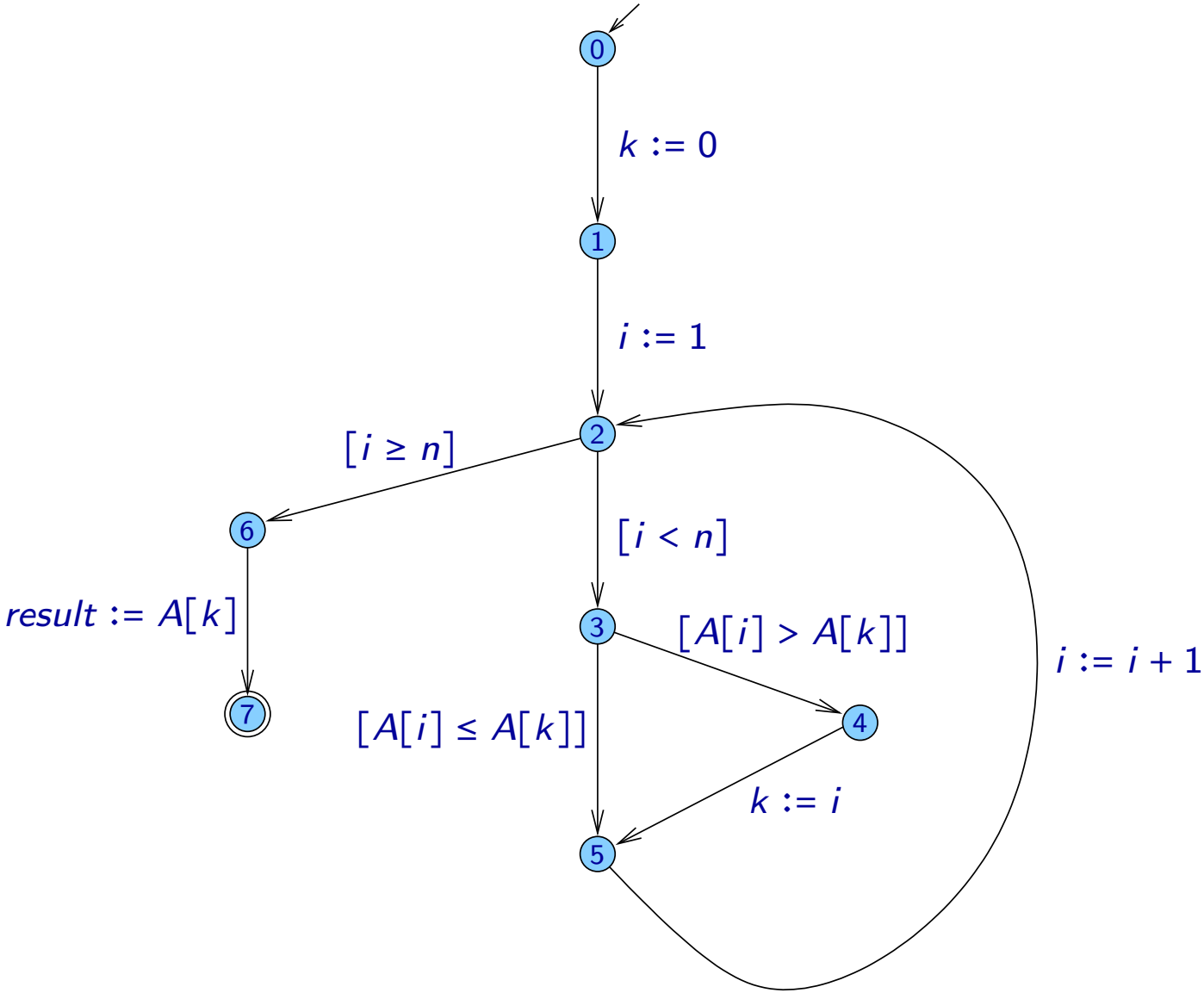
- **Stavy** — dvojice skládající se z aktuální pozice figurky jezdce na šachovnici a hodnoty čítače udávající počet zatím provedených tahů
- **Přechody** — provedení jednoho tahu jezdcem (podle pravidel šachu) a zvýšení čítače o jedna
- **Počáteční stavy** — jezdec se nachází na některém bílém poli v nejlevějším sloupci a hodnota čítače ja 0

Platí zde například následující invariant:

- jestliže je hodnota čítače sudá, jezdec se nachází na bílém poli
- jestliže je hodnota čítače lichá, jezdec se nachází na černém poli

Invarianty

Příklad: Algoritmus FIND-MAX reprezentovaný formou grafu řídícího toku



Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0,\ \langle A\text{: } [3, 8, 1, 3, 6],\ \ n\text{: } 5,\ \ i\text{: } ?,\ \ k\text{: } ?,\ \ \mathit{result}\text{: } ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0,\ \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ \mathit{result}\text{: } ?\rangle)$
α_1: $(1,\ \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } ?, \; \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } ?, \; \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } ?, \ k\text{: } ?, \ result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } ?, \ k\text{: } 0, \ result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 1, \ result\text{: } ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } ?, \; result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } 0, \; result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 1, \; result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 2, \; k\text{: } 1, \; result\text{: } ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{10}: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } ?, \ k\text{: } ?, \ result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } ?, \ k\text{: } 0, \ result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 2, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 2, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 2, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 3, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_{10}: $(3, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 3, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_{11}: $(5, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 3, \ k\text{: } 1, \ result\text{: } ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon ?,\ result\colon ?\rangle)$
α_1: $(1, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon 0,\ result\colon ?\rangle)$
α_2: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_3: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_4: $(4, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_5: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 1,\ result\colon ?\rangle)$
α_6: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_7: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_8: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_9: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{10}: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{11}: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{12}: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon ?,\ result\colon ?\rangle)$
α_1: $(1, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon 0,\ result\colon ?\rangle)$
α_2: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_3: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_4: $(4, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_5: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 1,\ result\colon ?\rangle)$
α_6: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_7: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_8: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_9: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{10}: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{11}: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{12}: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$
α_{13}: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

$$
\begin{array}{rl}
\alpha_0: & (0, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon ?,\ \mathit{result}\colon ?\rangle) \\
\alpha_1: & (1, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon 0,\ \mathit{result}\colon ?\rangle) \\
\alpha_2: & (2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ \mathit{result}\colon ?\rangle) \\
\alpha_3: & (3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ \mathit{result}\colon ?\rangle) \\
\alpha_4: & (4, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ \mathit{result}\colon ?\rangle) \\
\alpha_5: & (5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_6: & (2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_7: & (3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_8: & (5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_9: & (2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{10}: & (3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{11}: & (5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{12}: & (2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{13}: & (3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{14}: & (5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ \mathit{result}\colon ?\rangle)
\end{array}
$$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } ?, \ k\text{: } ?, \ result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } ?, \ k\text{: } 0, \ result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 2, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 2, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 2, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 3, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_{10}: $(3, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 3, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_{11}: $(5, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 3, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_{12}: $(2, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 4, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_{13}: $(3, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 4, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_{14}: $(5, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 4, \ k\text{: } 1, \ result\text{: } ?\rangle)$
α_{15}: $(2, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 5, \ k\text{: } 1, \ result\text{: } ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon ?,\ result\colon ?\rangle)$
α_1: $(1, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon 0,\ result\colon ?\rangle)$
α_2: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_3: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_4: $(4, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_5: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 1,\ result\colon ?\rangle)$
α_6: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_7: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_8: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_9: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{10}: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{11}: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{12}: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$
α_{13}: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$
α_{14}: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$
α_{15}: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 5,\ k\colon 1,\ result\colon ?\rangle)$
α_{16}: $(6, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 5,\ k\colon 1,\ result\colon ?\rangle)$

Invarianty

Výpočet pro vstup $A = [3, 8, 1, 3, 6]$ a $n = 5$ jako posloupnost konfigurací:

α_0: $(0, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon ?,\ result\colon ?\rangle)$
α_1: $(1, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon 0,\ result\colon ?\rangle)$
α_2: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_3: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_4: $(4, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_5: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 1,\ result\colon ?\rangle)$
α_6: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_7: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_8: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_9: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{10}: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{11}: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$
α_{12}: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$
α_{13}: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$
α_{14}: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ result\colon ?\rangle)$
α_{15}: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 5,\ k\colon 1,\ result\colon ?\rangle)$
α_{16}: $(6, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 5,\ k\colon 1,\ result\colon ?\rangle)$
α_{17}: $(7, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 5,\ k\colon 1,\ result\colon 8\rangle)$

Invarianty

- **Stavy** — konfigurace skládající se ze stavu řídící jednotky a obsahu paměti reprezentovaného hodnotami jednotlivých proměnných.
- **Přechody** — dány jednotlivými instrukcemi na hranách grafu, mění zároveň řídící stav i obsah paměti přiřazováním hodnot do proměnných
- **Počáteční stavy** — všechny možné počáteční konfigurace pro všechny možné vstupní instance, které jsou přípustné podle specifikace problému

Invarianty budou mít formu tvrzení vyjadřujících se o konfiguracích, tj. o stavech řídící jednotky a o hodnotách jednotlivých proměnných, např.

- Pokud je stav řídící jednotky 2, pak v dané konfiguraci platí $1 \leq i \leq n$, $0 \leq k < i$ a $A[k]$ je největší z prvků $A[0], A[1], \ldots, A[i-1]$.

Invarianty

U systémů, kde je součástí konfigurace nějaký řídící stav, může být výhodné formulovat invarianty ve formě:

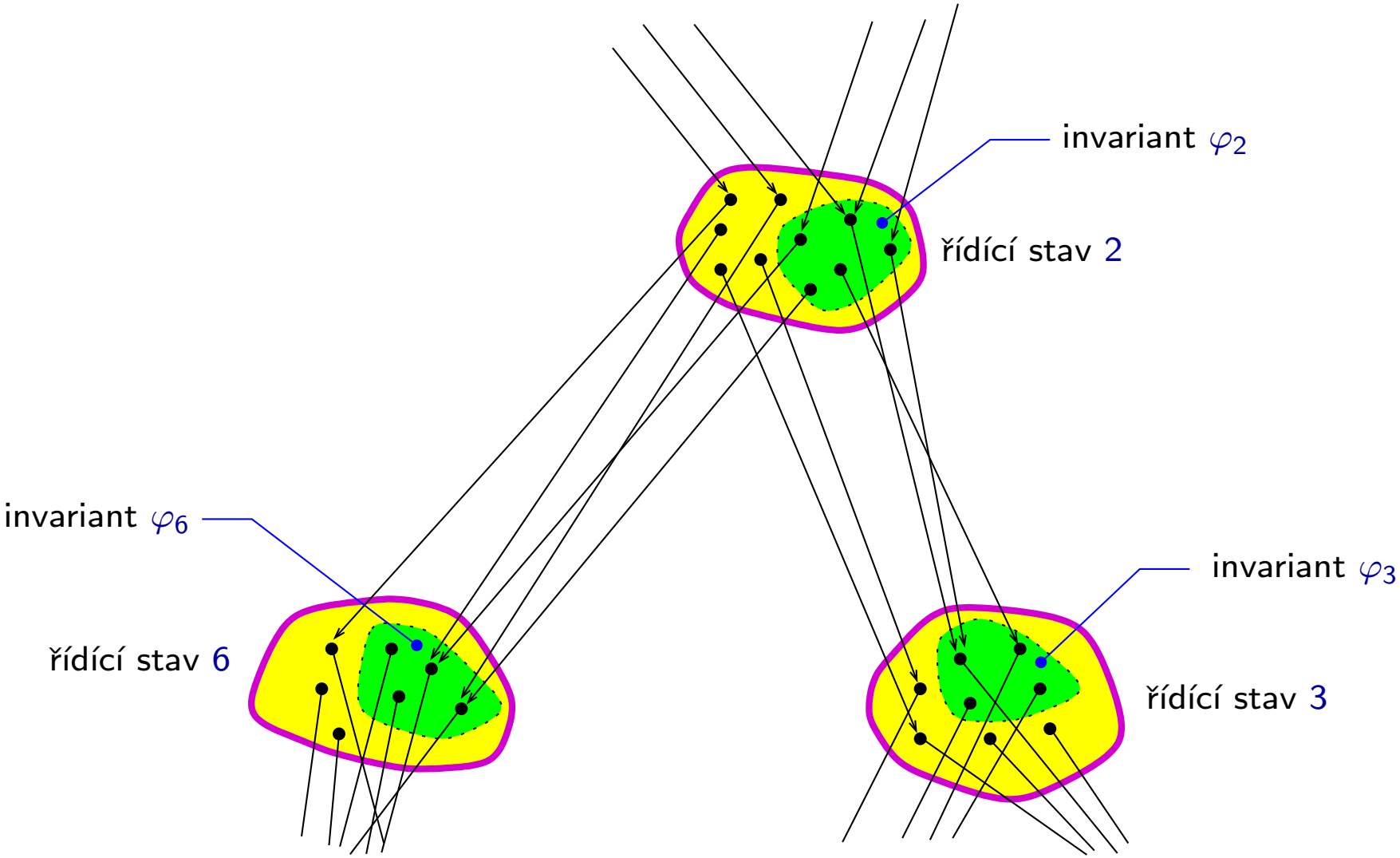
- jestliže je stav řídící jednotky 0, pak platí φ_0
- jestliže je stav řídící jednotky 1, pak platí φ_1

  $\vdots$
- jestliže je stav řídící jednotky r, pak platí φ_r

přičemž tvrzení $\varphi_0, \varphi_1, \ldots, \varphi_r$ se vyjadřují pouze o obsahu paměti, nikoli o řídících stavech.

Konfigurace můžeme rozdělit do (konečně mnoha) skupin podle stavů řídící jednotky.

Invarianty


invariant φ_2
řídící stav 2
invariant φ_6
řídící stav 6
invariant φ_3
řídící stav 3

Invarianty

Invariant — podmínka, která musí být v určitém místě kódu algoritmu vždy (tj. ve všech možných výpočtech pro všechny možné vstupy) splněna v okamžiku, kdy algoritmus tímto místem prochází.

Invarianty můžeme zapisovat formulemi predikátové logiky:

- **volné** proměnné odpovídají proměnným programu
- **valuace** je dána hodnotami proměnných programu v dané konfiguraci

Příklad: Formule

$$(1 \leq i) \wedge (i \leq n)$$

bude platit například v konfiguraci, kde proměnná i má hodnotu 5 a proměnná n má hodnotu 14.

Invarianty

Zjištěné invarianty mohou sloužit k řadě různých účelů:

- Napomohou lepšímu porozumění chování algoritmu.
- Lze pomocí nich ověřit, že nenastanou určité typy chyb — např. přístup k poli mimo povolené rozsahy indexů, dělení nulou, ...

  Ověříme, že v místech v kódu, kde by mohla daná chyba nastat, budou platit invarianty, které zaručí, že proměnné budou mít vždy takové hodnoty, aby chyba nenastala.

  Příklad: Při přístupu k prvku $A[i]$ bude vždy platit $0 \leq i < n$, kde n je délka pole.
- Invariant, který bude platit v koncových konfiguracích, zaručí, že výstup algoritmu bude odpovídat specifikaci v zadání problému.
- Při analýze výpočetní složitosti napomohou při zkoumání toho, kolikrát se provedou které instrukce nebo jak velké množství paměti je při výpočtu potřeba.

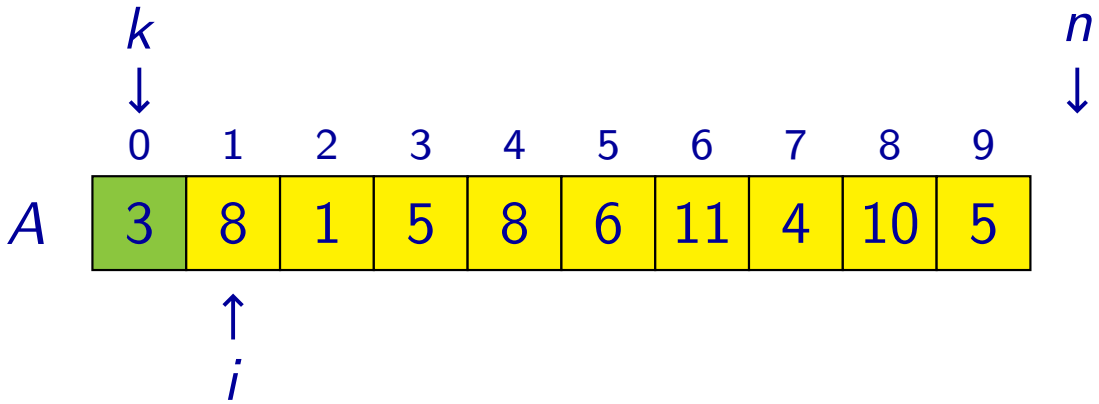
Invarianty

Stanovení invariantů není úplně mechanický proces. Vyžaduje určité pochopení chování algoritmu.

Před formulováním hypotéz o tom, jaké invarianty platí v jednotlivých řídících stavech, může být vhodné se podívat na to, jak se daný algoritmus chová na nějakých konkrétních vstupech.

Příklad: Výpočet algoritmu FIND-MAX pro vstup

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



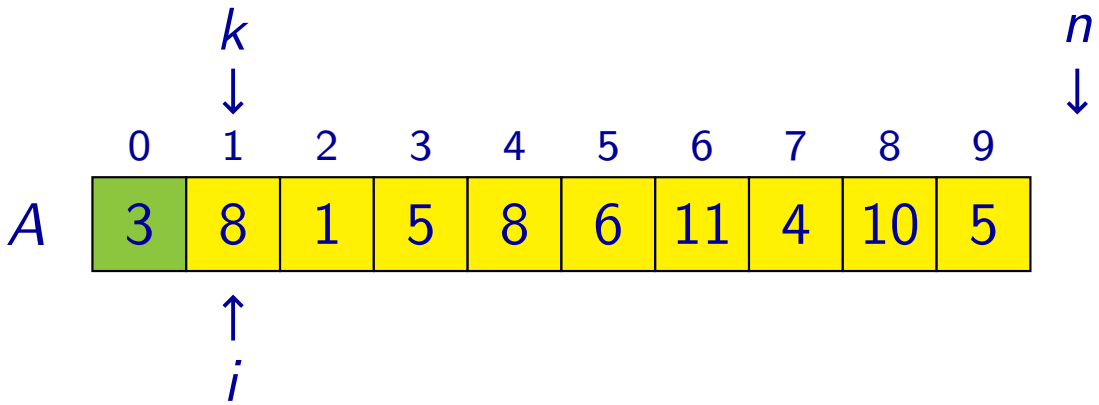
Invarianty

Stanovení invariantů není úplně mechanický proces. Vyžaduje určité pochopení chování algoritmu.

Před formulováním hypotéz o tom, jaké invarianty platí v jednotlivých řídících stavech, může být vhodné se podívat na to, jak se daný algoritmus chová na nějakých konkrétních vstupech.

Příklad: Výpočet algoritmu FIND-MAX pro vstup

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



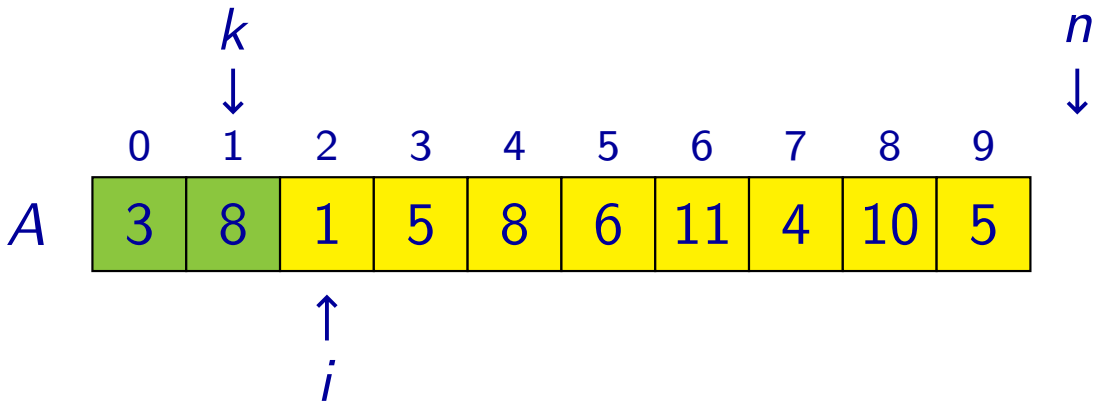
Invarianty

Stanovení invariantů není úplně mechanický proces. Vyžaduje určité pochopení chování algoritmu.

Před formulováním hypotéz o tom, jaké invarianty platí v jednotlivých řídících stavech, může být vhodné se podívat na to, jak se daný algoritmus chová na nějakých konkrétních vstupech.

Příklad: Výpočet algoritmu FIND-MAX pro vstup

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



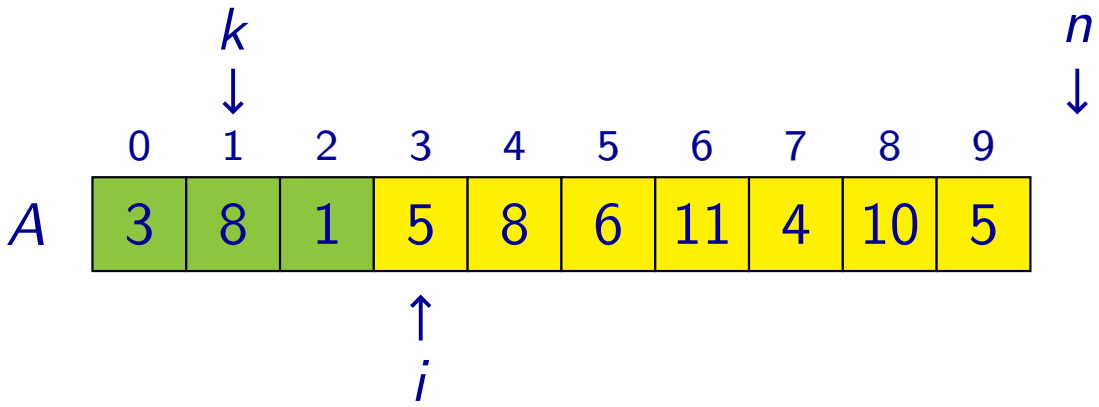
Invarianty

Stanovení invariantů není úplně mechanický proces. Vyžaduje určité pochopení chování algoritmu.

Před formulováním hypotéz o tom, jaké invarianty platí v jednotlivých řídících stavech, může být vhodné se podívat na to, jak se daný algoritmus chová na nějakých konkrétních vstupech.

Příklad: Výpočet algoritmu FIND-MAX pro vstup

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



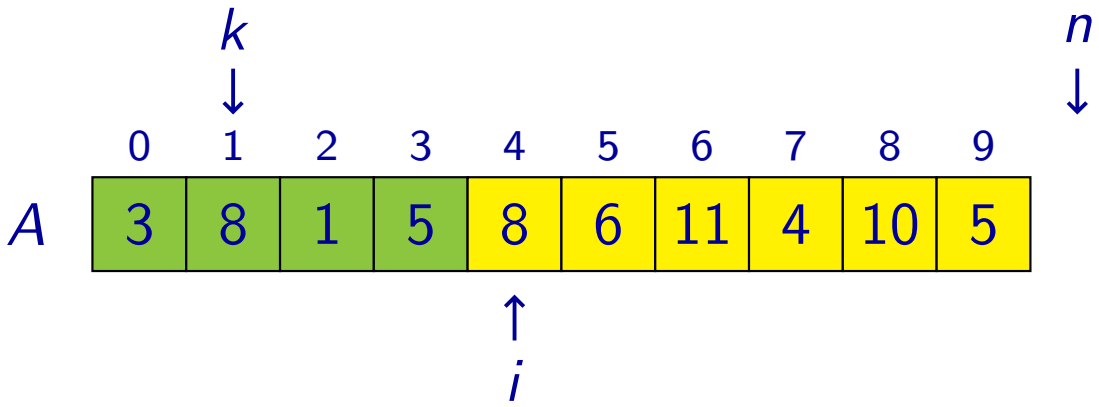
Invarianty

Stanovení invariantů není úplně mechanický proces. Vyžaduje určité pochopení chování algoritmu.

Před formulováním hypotéz o tom, jaké invarianty platí v jednotlivých řídících stavech, může být vhodné se podívat na to, jak se daný algoritmus chová na nějakých konkrétních vstupech.

Příklad: Výpočet algoritmu FIND-MAX pro vstup

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



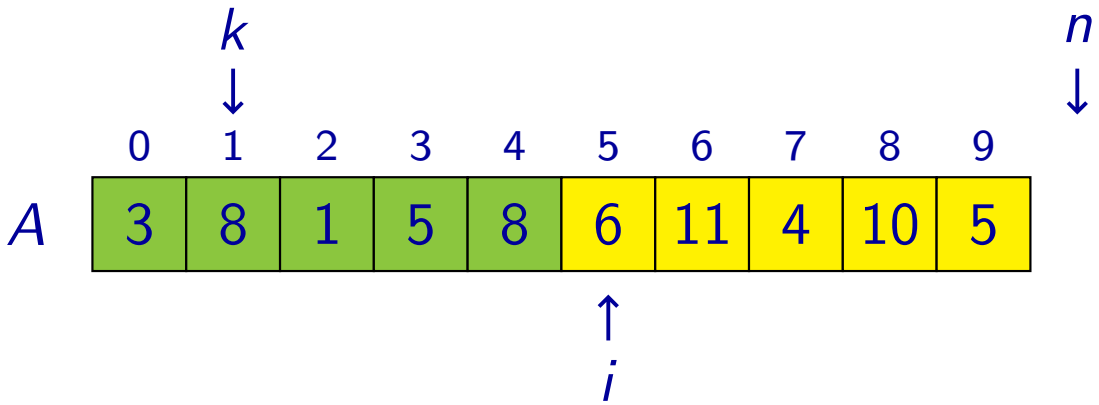
Invarianty

Stanovení invariantů není úplně mechanický proces. Vyžaduje určité pochopení chování algoritmu.

Před formulováním hypotéz o tom, jaké invarianty platí v jednotlivých řídících stavech, může být vhodné se podívat na to, jak se daný algoritmus chová na nějakých konkrétních vstupech.

Příklad: Výpočet algoritmu FIND-MAX pro vstup

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



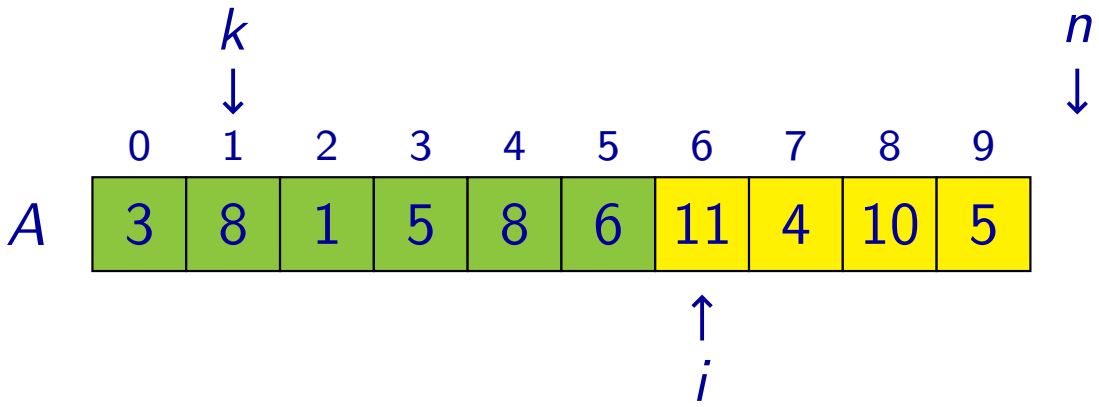
Invarianty

Stanovení invariantů není úplně mechanický proces. Vyžaduje určité pochopení chování algoritmu.

Před formulováním hypotéz o tom, jaké invarianty platí v jednotlivých řídících stavech, může být vhodné se podívat na to, jak se daný algoritmus chová na nějakých konkrétních vstupech.

Příklad: Výpočet algoritmu FIND-MAX pro vstup

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



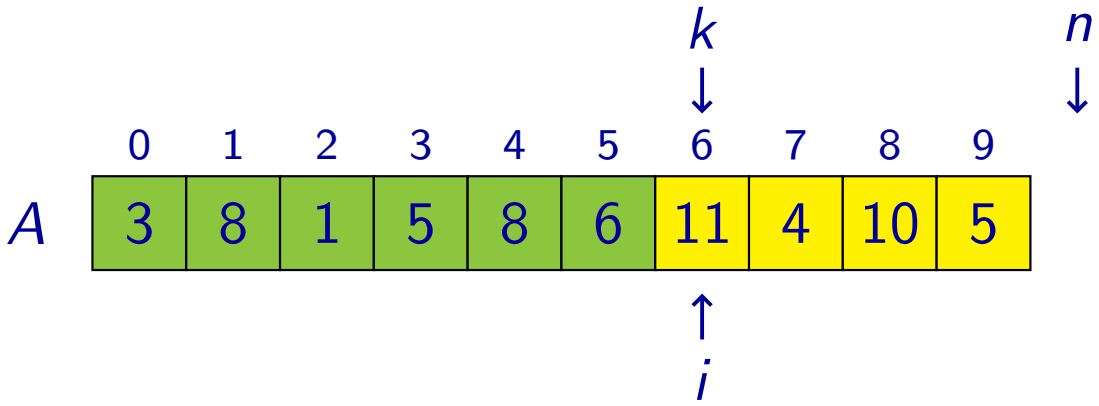
Invarianty

Stanovení invariantů není úplně mechanický proces. Vyžaduje určité pochopení chování algoritmu.

Před formulováním hypotéz o tom, jaké invarianty platí v jednotlivých řídících stavech, může být vhodné se podívat na to, jak se daný algoritmus chová na nějakých konkrétních vstupech.

Příklad: Výpočet algoritmu FIND-MAX pro vstup

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



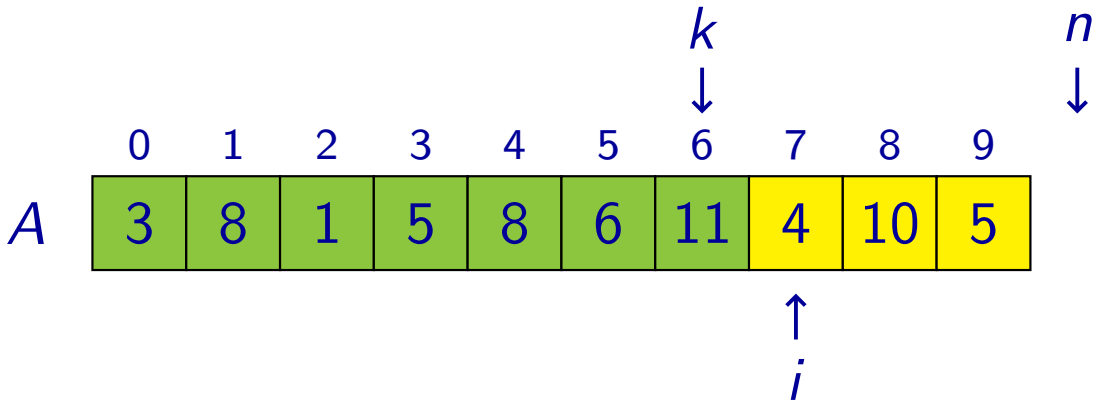
Invarianty

Stanovení invariantů není úplně mechanický proces. Vyžaduje určité pochopení chování algoritmu.

Před formulováním hypotéz o tom, jaké invarianty platí v jednotlivých řídících stavech, může být vhodné se podívat na to, jak se daný algoritmus chová na nějakých konkrétních vstupech.

Příklad: Výpočet algoritmu FIND-MAX pro vstup

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



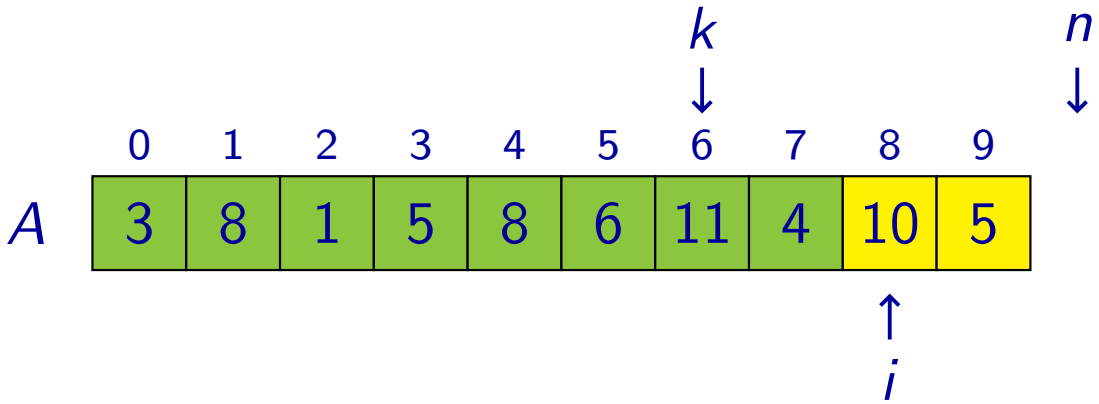
Invarianty

Stanovení invariantů není úplně mechanický proces. Vyžaduje určité pochopení chování algoritmu.

Před formulováním hypotéz o tom, jaké invarianty platí v jednotlivých řídících stavech, může být vhodné se podívat na to, jak se daný algoritmus chová na nějakých konkrétních vstupech.

Příklad: Výpočet algoritmu FIND-MAX pro vstup

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



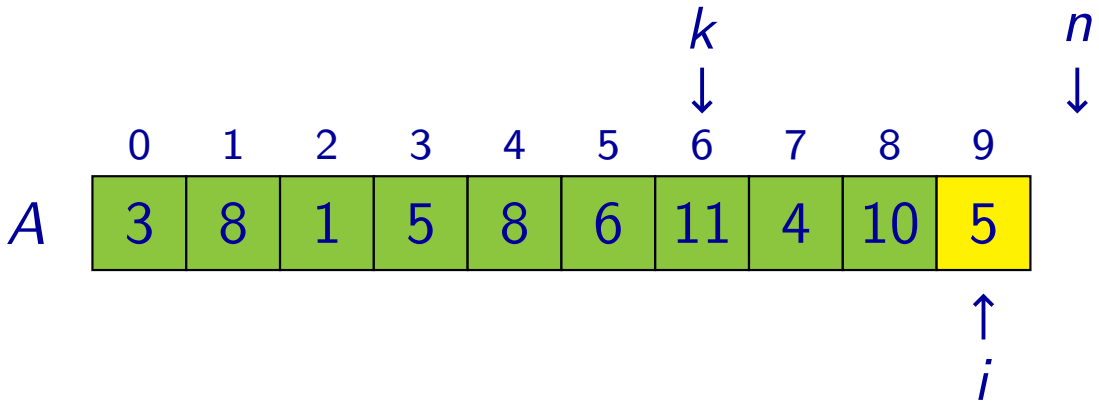
Invarianty

Stanovení invariantů není úplně mechanický proces. Vyžaduje určité pochopení chování algoritmu.

Před formulováním hypotéz o tom, jaké invarianty platí v jednotlivých řídících stavech, může být vhodné se podívat na to, jak se daný algoritmus chová na nějakých konkrétních vstupech.

Příklad: Výpočet algoritmu FIND-MAX pro vstup

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



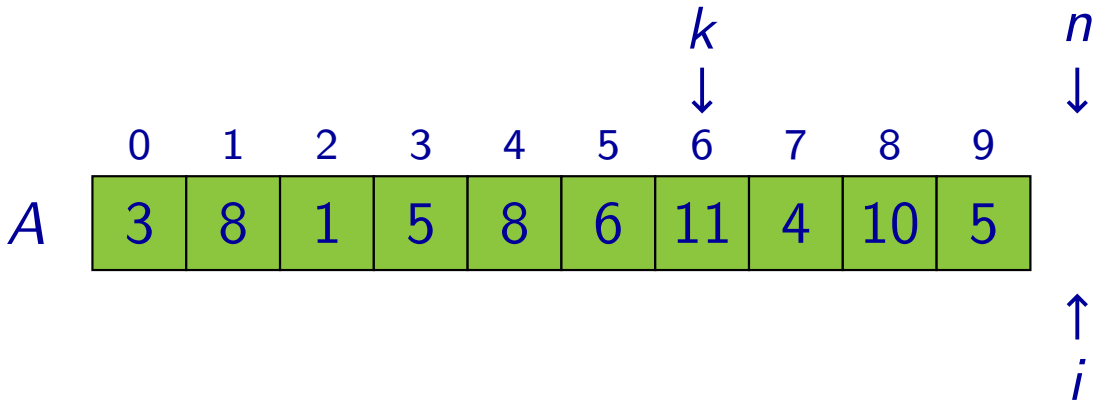
Invarianty

Stanovení invariantů není úplně mechanický proces. Vyžaduje určité pochopení chování algoritmu.

Před formulováním hypotéz o tom, jaké invarianty platí v jednotlivých řídících stavech, může být vhodné se podívat na to, jak se daný algoritmus chová na nějakých konkrétních vstupech.

Příklad: Výpočet algoritmu FIND-MAX pro vstup

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



Invarianty

Příklady invariantů:

- invariant v řídícím stavu q zapíšeme formulí φ_q

Invarianty v jednotlivých řídících stavech (zatím jen hypotézy):

- φ_0: $(n \geq 1)$
- φ_1: $(n \geq 1) \wedge (k = 0)$
- φ_2: $(n \geq 1) \wedge (1 \leq i \leq n) \wedge (0 \leq k < i)$
- φ_3: $(n \geq 1) \wedge (1 \leq i < n) \wedge (0 \leq k < i)$
- φ_4: $(n \geq 1) \wedge (1 \leq i < n) \wedge (0 \leq k < i)$
- φ_5: $(n \geq 1) \wedge (1 \leq i < n) \wedge (0 \leq k \leq i)$
- φ_6: $(n \geq 1) \wedge (i = n) \wedge (0 \leq k < n)$
- φ_7: $(n \geq 1) \wedge (i = n) \wedge (0 \leq k < n)$

Invarianty

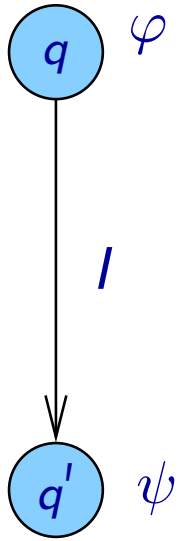
Zkontrolování toho, že invarianty opravdu platí:

- Musíme zkontrolovat, zda invarianty platí v počátečních konfiguracích — toto je většinou jednoduché.
- Pro každou instrukci algoritmu je třeba zkontrolovat, zda za předpokladu, že bude platit příslušný invariant před provedením této instrukce, bude platit i příslušný invariant po provedení této instrukce.

Předpokládejme algoritmus ve formě grafu řídícího toku:

- hrany odpovídají instrukcím
- vezměme si hranu ze stavu q do stavu q' označenou instrukcí I
- řekněme, že (zatím neověřené) invarianty pro stavy q a q' jsou vyjádřeny formulemi φ a ψ

Invarianty



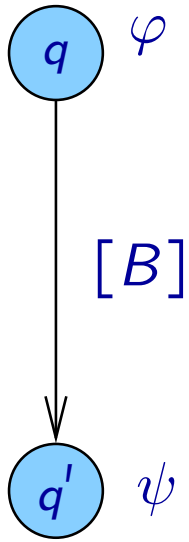
- pro tuto hranu musíme zkontrolovat, že pro všechny konfigurace $\alpha = (q, mem)$ a $\alpha' = (q', mem')$ takové, že $\alpha \xrightarrow{l} \alpha'$, platí, že pokud
  - v konfiguraci α platí φ,

  pak
  - v konfiguraci α' platí ψ

Invarianty

Zkontrolování instrukcí, které jsou testy podmínek:

- hrana označená testem podmínky $[B]$

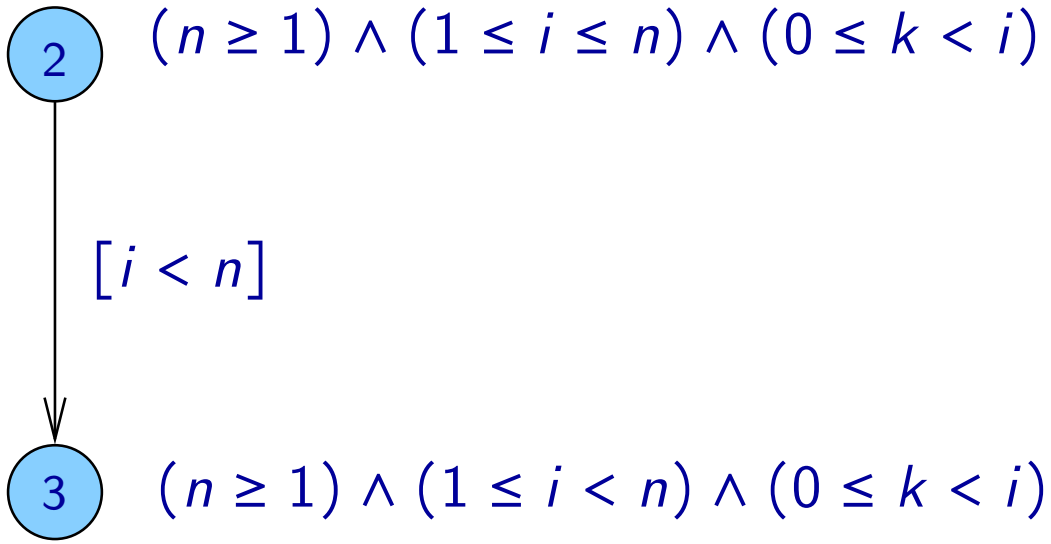


Obsah paměti se nemění, takže stačí ověřit, že platí implikace

$$(\varphi \wedge B) \Rightarrow \psi$$

Invarianty

Příklad:

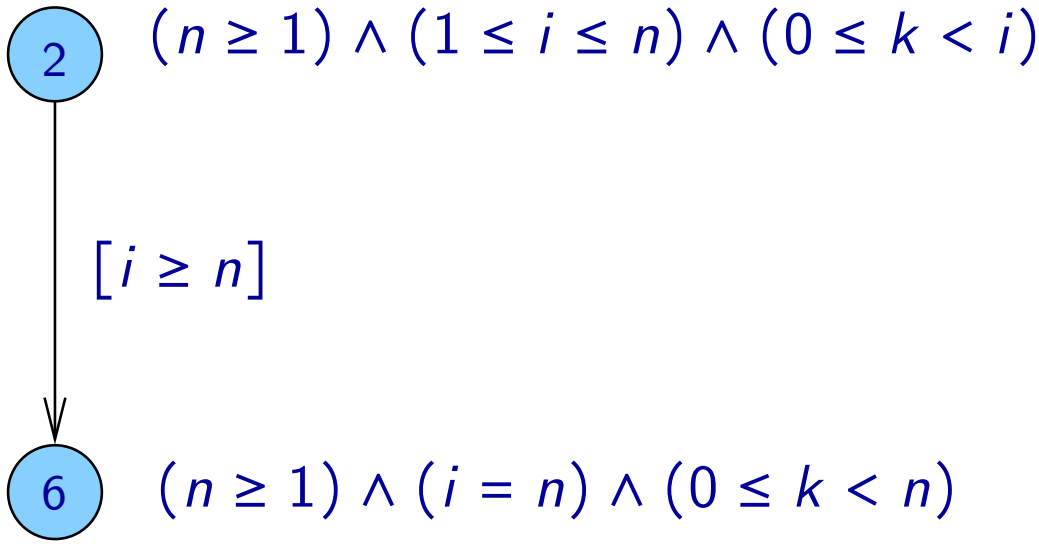


Stačí ověřit, že platí následující implikace:

- Jestliže $(n \geq 1) \wedge (1 \leq i \leq n) \wedge (0 \leq k < i) \wedge (i < n)$, pak $(n \geq 1) \wedge (1 \leq i < n) \wedge (0 \leq k < i)$.

Invarianty

Příklad:



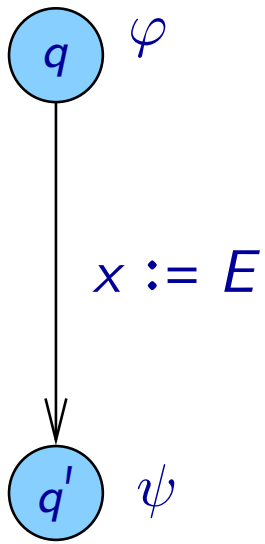
Stačí ověřit, že platí následující implikace:

- Jestliže $(n \geq 1) \wedge (1 \leq i \leq n) \wedge (0 \leq k < i) \wedge (i \geq n)$, pak $(n \geq 1) \wedge (i = n) \wedge (0 \leq k < n)$.

Invarianty

Zkontrolování instrukcí, které přiřazují hodnoty proměnným (mění obsah paměti):

- hrana označená přiřazením $x := E$



Je třeba rozlišovat mezi hodnotou proměnné x před tímto přiřazením a po tomto přiřazení.

Invarianty

Pro následující konstrukce budeme potřebavat operaci **substituce** na formulích:

$$\varphi[E/x]$$

označuje formuli, kterou dostaneme z formule φ dosazením výrazu E za všechny volné výskyty proměnné x ve formuli φ.

Příklad: Řekněme, že φ je formule $(1 \leq i) \wedge (i \leq n)$.

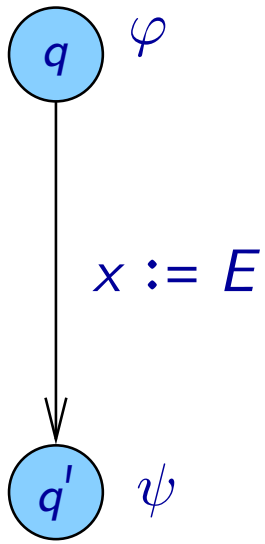
Zápis $\varphi[i'/i]$ pak označuje formuli

$$(1 \leq i') \wedge (i' \leq n)$$

a zápis $\varphi[(i+1)/i]$ formuli

$$(1 \leq i+1) \wedge (i+1 \leq n)$$

Invarianty



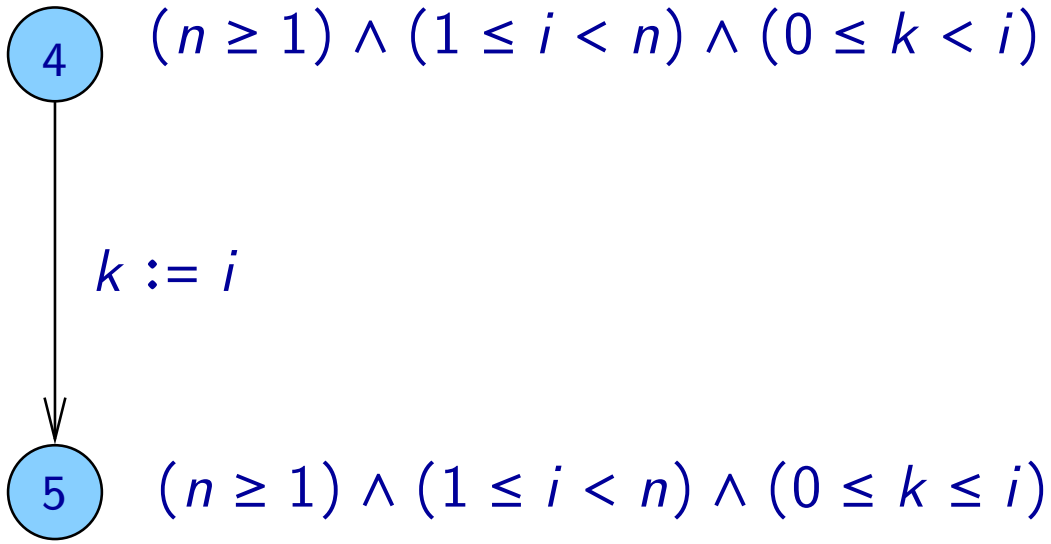
Zavedeme novou proměnnou x' reprezentující hodnotu proměnné x po provedení tohoto přiřazení.

Je třeba ověřit následující implikaci:

$$(\varphi \wedge (x' = E)) \Rightarrow \psi[x'/x]$$

Invarianty

Příklad:

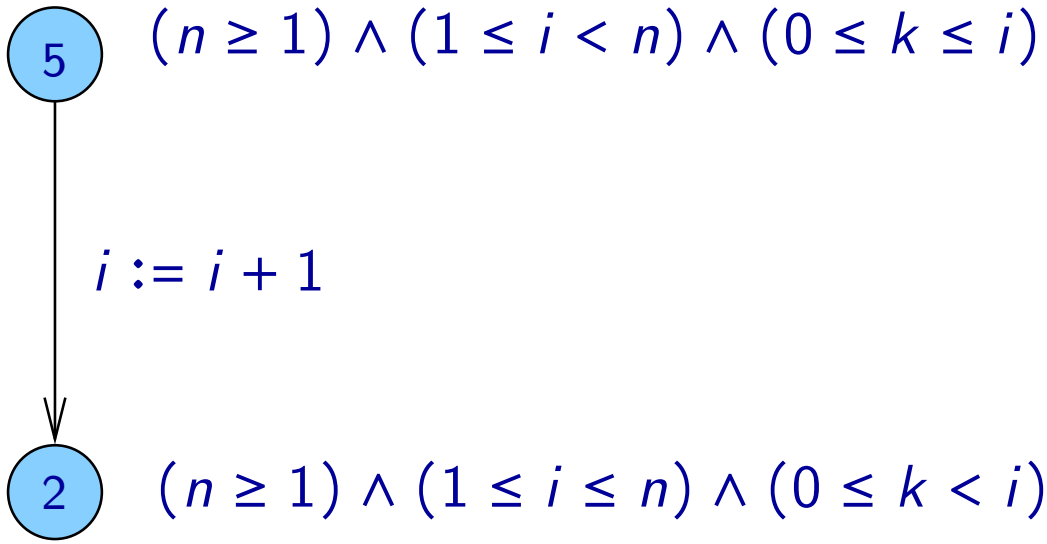


Stačí ověřit, že platí následující implikace:

- Jestliže $(n \geq 1) \wedge (1 \leq i < n) \wedge (0 \leq k < i) \wedge (k' = i)$, pak $(n \geq 1) \wedge (1 \leq i < n) \wedge (0 \leq k' \leq i)$.

Invarianty

Příklad:



Stačí ověřit, že platí následující implikace:

- Jestliže $(n \geq 1) \wedge (1 \leq i < n) \wedge (0 \leq k \leq i) \wedge (i' = i + 1)$, pak $(n \geq 1) \wedge (1 \leq i' \leq n) \wedge (0 \leq k < i')$.

Invarianty

Dokončení ověření toho, že algoritmus FIND-MAX vrací správný výsledek (za předpokladu, že skončí):

- ψ_0: φ_0
- ψ_1: $\varphi_1 \wedge (\forall j \in \mathbb{N})(0 \leq j < 1 \rightarrow A[j] \leq A[k])$
- ψ_2: $\varphi_2 \wedge (\forall j \in \mathbb{N})(0 \leq j < i \rightarrow A[j] \leq A[k])$
- ψ_3: $\varphi_3 \wedge (\forall j \in \mathbb{N})(0 \leq j < i \rightarrow A[j] \leq A[k])$
- ψ_4: $\varphi_4 \wedge (\forall j \in \mathbb{N})(0 \leq j < i \rightarrow A[j] \leq A[k]) \wedge (A[i] > A[k])$
- ψ_5: $\varphi_5 \wedge (\forall j \in \mathbb{N})(0 \leq j \leq i \rightarrow A[j] \leq A[k])$
- ψ_6: $\varphi_6 \wedge (\forall j \in \mathbb{N})(0 \leq j < n \rightarrow A[j] \leq A[k])$
- ψ_7: $\varphi_7 \wedge (\mathit{result} = A[k]) \wedge (\forall j \in \mathbb{N})(0 \leq j < n \rightarrow A[j] \leq \mathit{result}) \wedge (\exists j \in \mathbb{N})(0 \leq j < n \wedge A[j] = \mathit{result})$

Invarianty

Často není třeba specifikovat invarianty ve všech řídících stavech, ale jen v některých „důležitých“ — zejména stavy, kde se vstupuje do nebo vystupuje z cyklů:

Je pak třeba ověřit:

- Že invariant platí před vstupem do cyklu.
- Že pokud invariant platí před provedením cyklu, tak bude platit i po jeho provedení.
- Že invariant platí při opuštění cyklu.

Invarianty

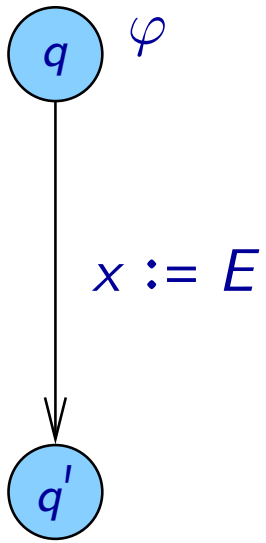
Příklad: V algoritmu FIND-MAX je takovým „důležitým“ stavem stav 2.

Ve stavu 2 platí:

- $n \geq 1$
- $1 \leq i \leq n$
- $0 \leq k < i$
- Pro všechna j taková, že $0 \leq j < i$, platí $A[j] \leq A[k]$.

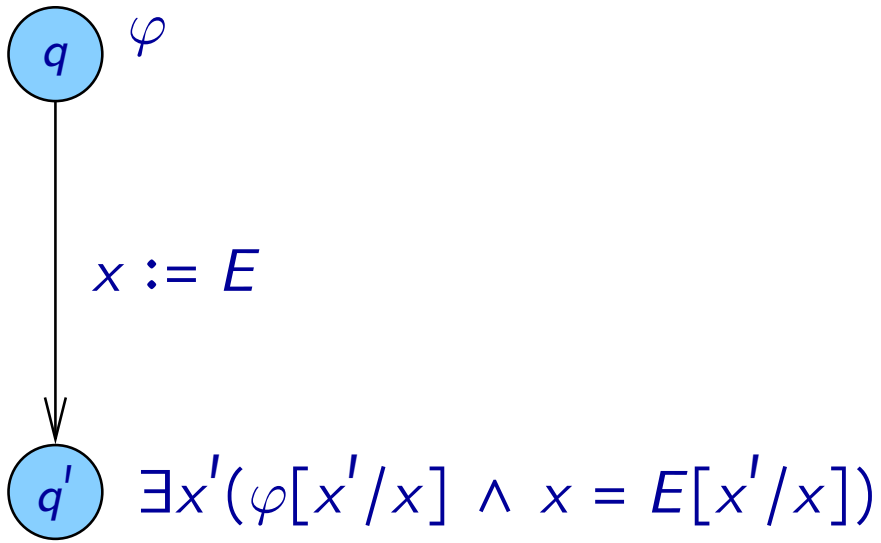
Invarianty

Příklady toho, jak určit invarianty u některých dalších stavů, pokud už u některých stavů invarianty máme:



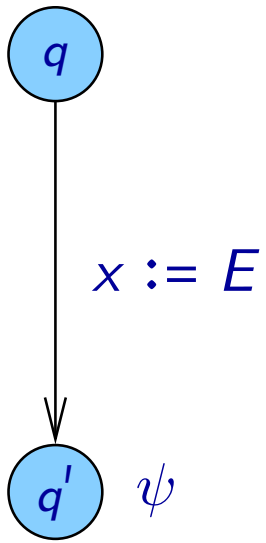
Invarianty

Příklady toho, jak určit invarianty u některých dalších stavů, pokud už u některých stavů invarianty máme:



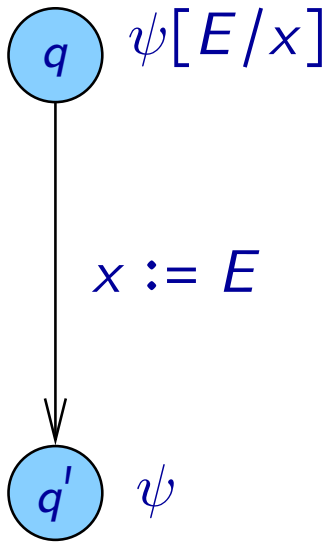
Invarianty

Příklady toho, jak určit invarianty u některých dalších stavů, pokud už u některých stavů invarianty máme:



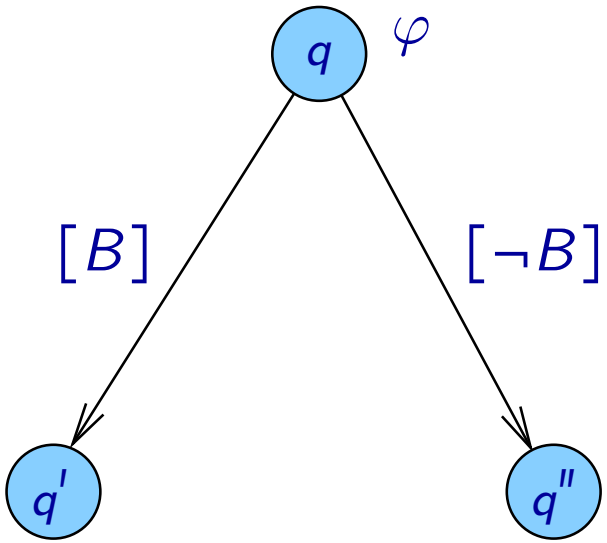
Invarianty

Příklady toho, jak určit invarianty u některých dalších stavů, pokud už u některých stavů invarianty máme:



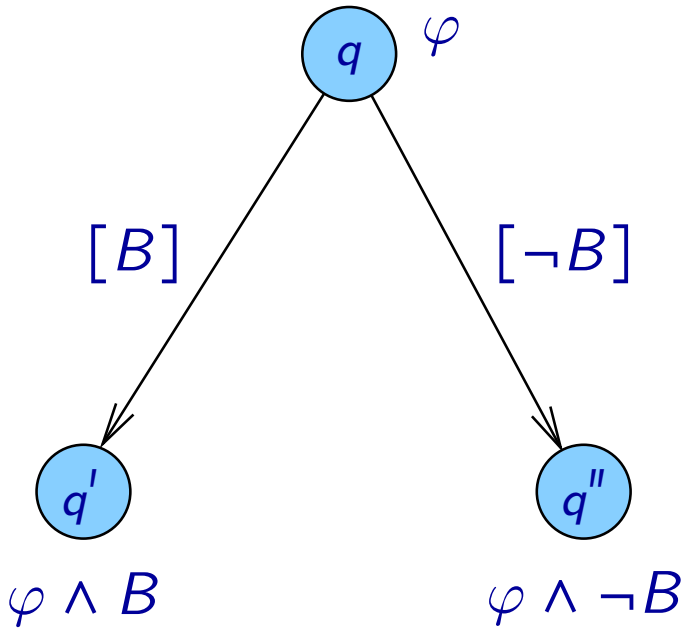
Invarianty

Příklady toho, jak určit invarianty u některých dalších stavů, pokud už u některých stavů invarianty máme:



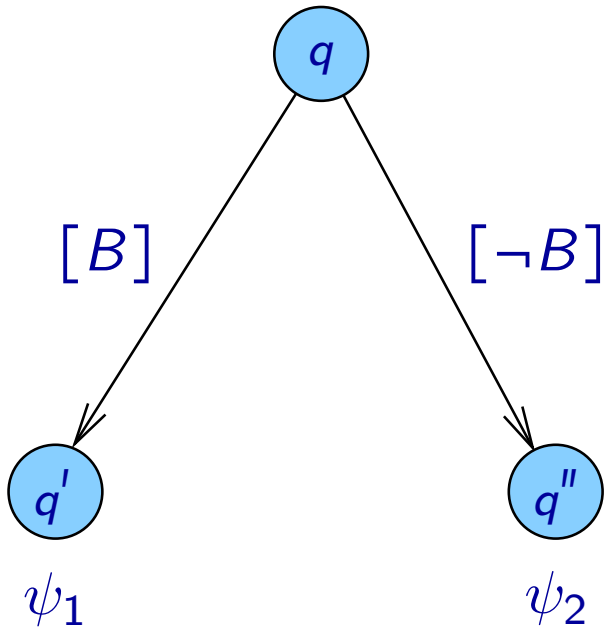
Invarianty

Příklady toho, jak určit invarianty u některých dalších stavů, pokud už u některých stavů invarianty máme:



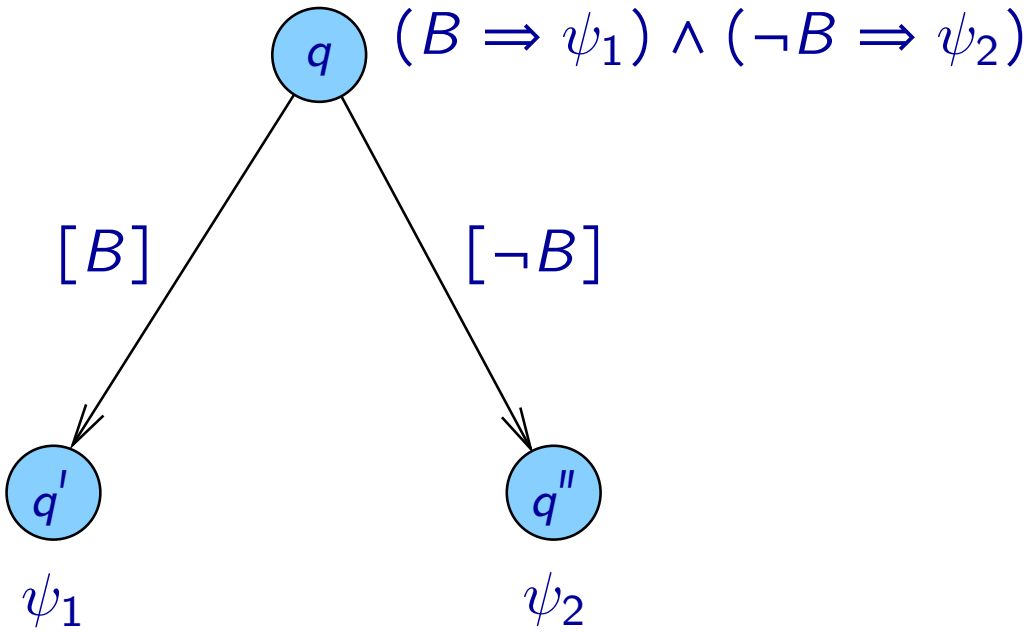
Invarianty

Příklady toho, jak určit invarianty u některých dalších stavů, pokud už u některých stavů invarianty máme:



Invarianty

Příklady toho, jak určit invarianty u některých dalších stavů, pokud už u některých stavů invarianty máme:



Příklad:

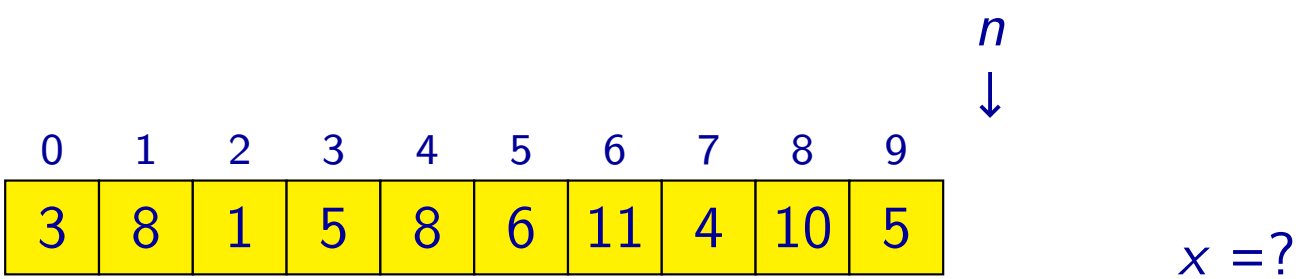
Algoritmus: Třídění přímým vkládáním

```
INSERTION-SORT (A, n):
    for j := 1 to n − 1 do
        x := A[j]
        i := j − 1
        while i ≥ 0 and A[i] > x do
            A[i + 1] := A[i]
            i := i − 1
        A[i + 1] := x
```

Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

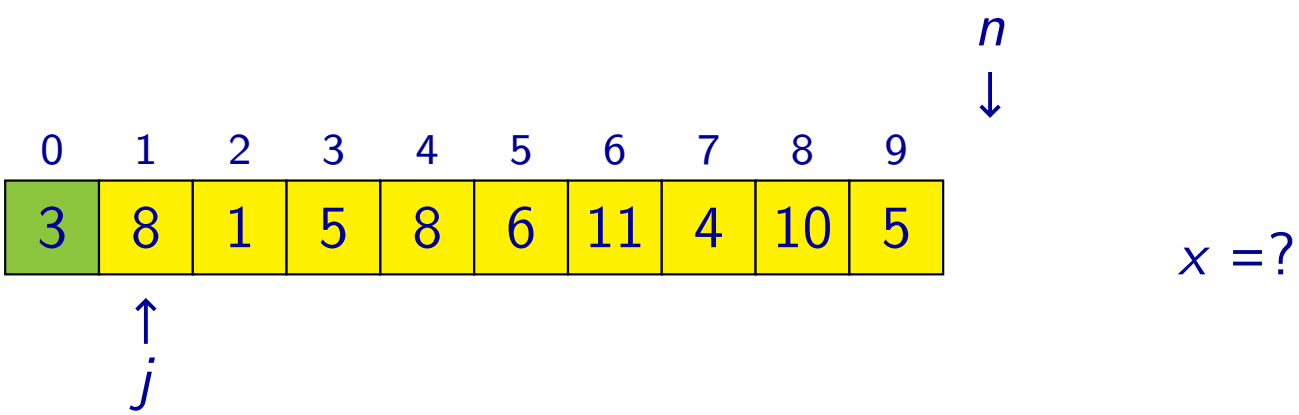
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

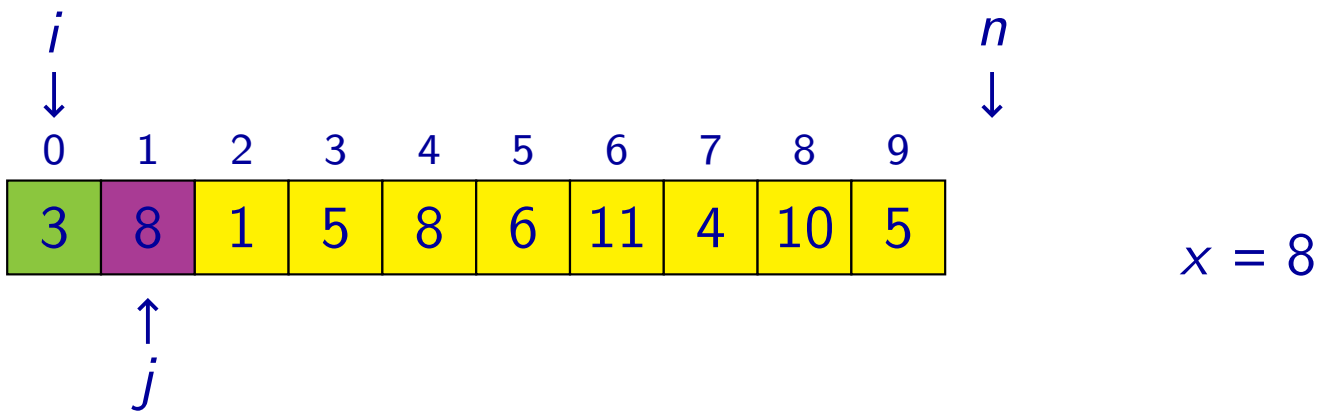
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

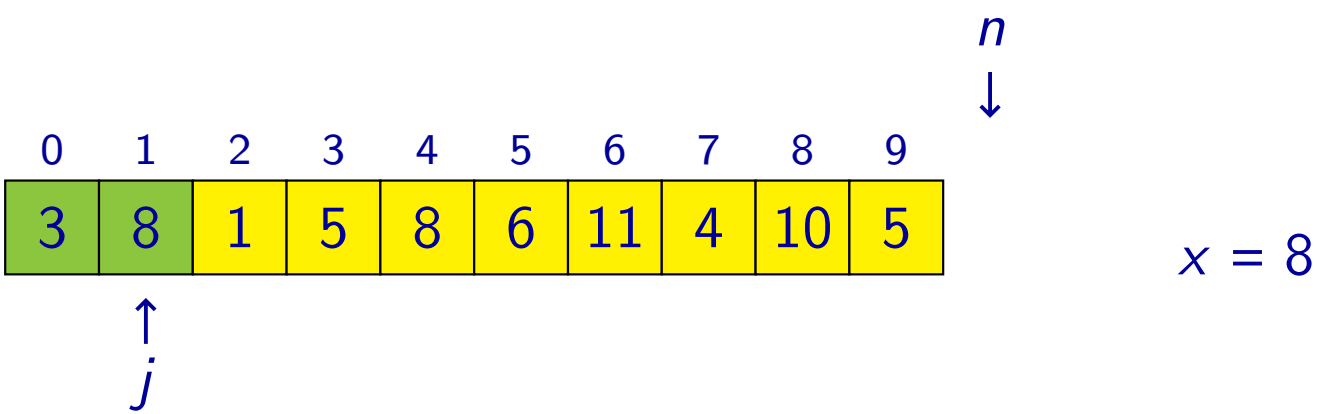
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

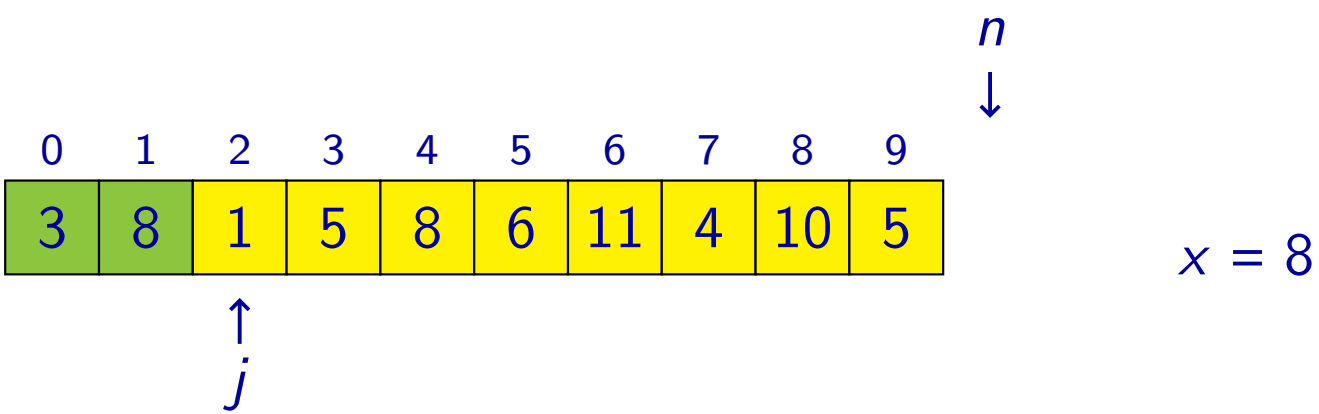
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

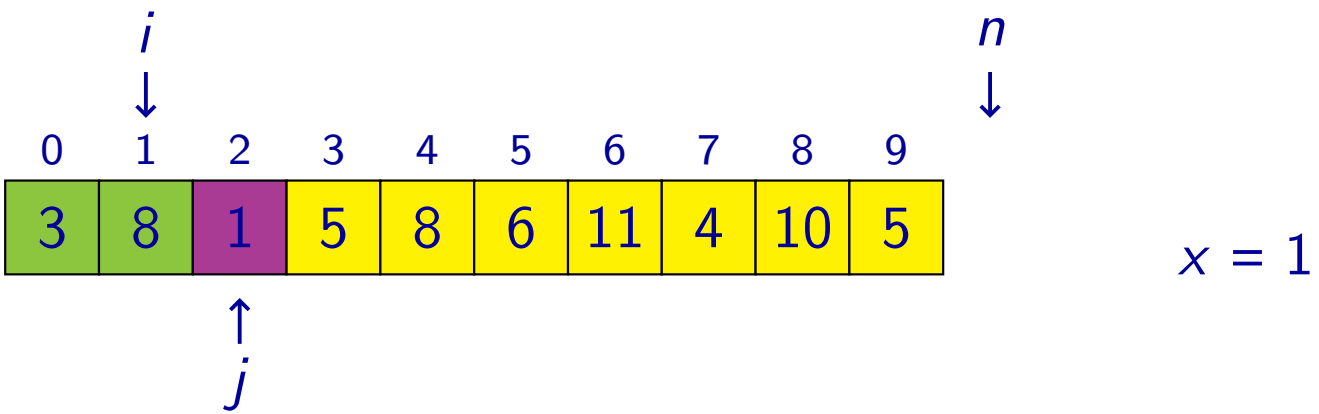
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

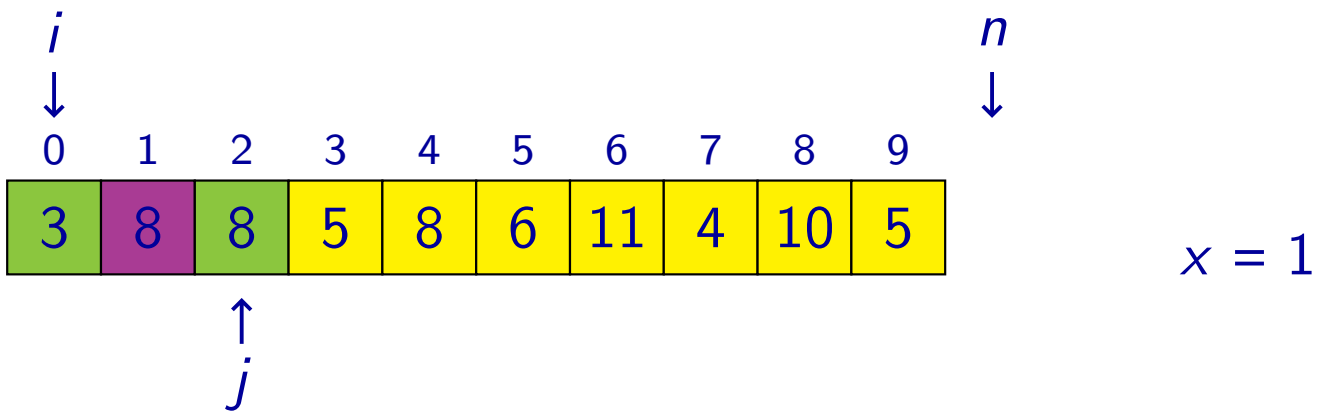
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

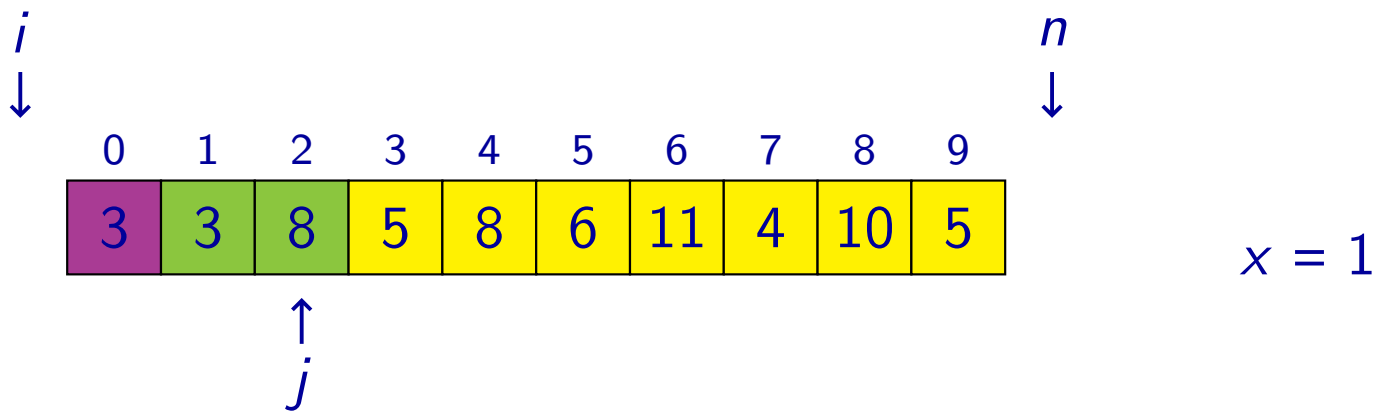
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

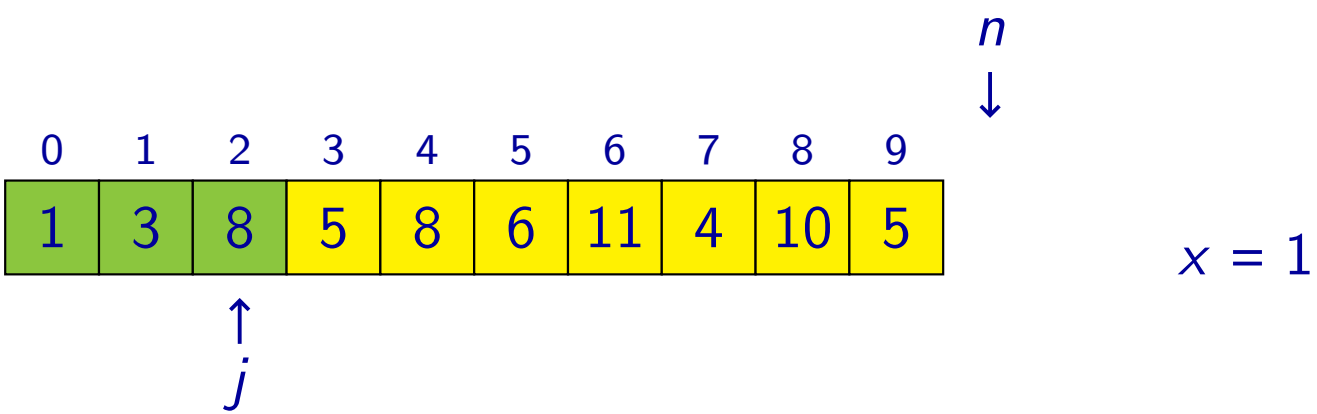
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

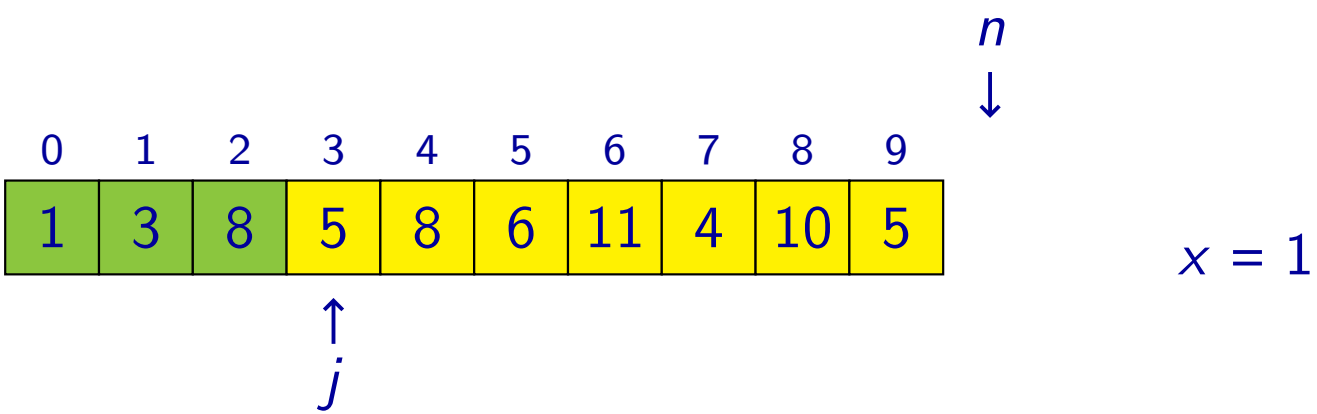
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

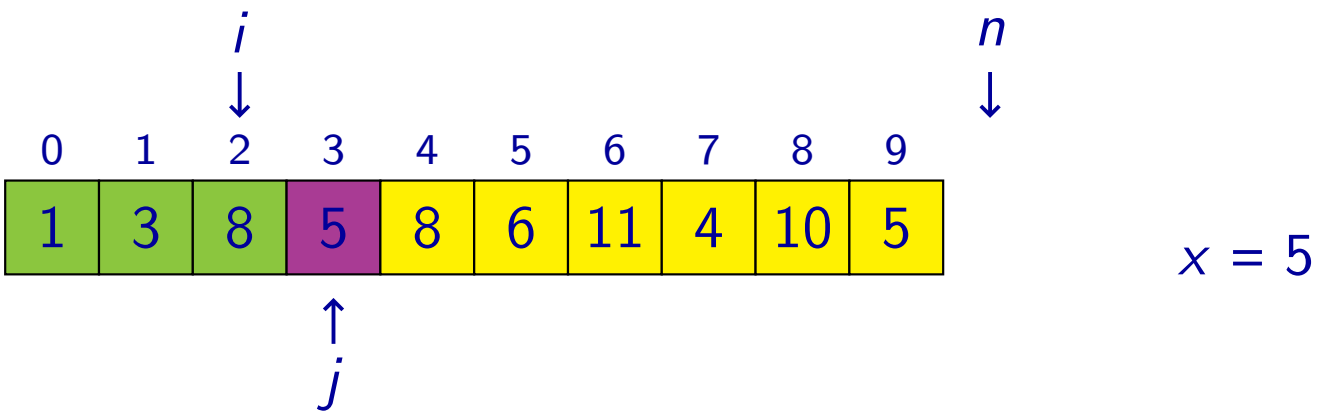
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

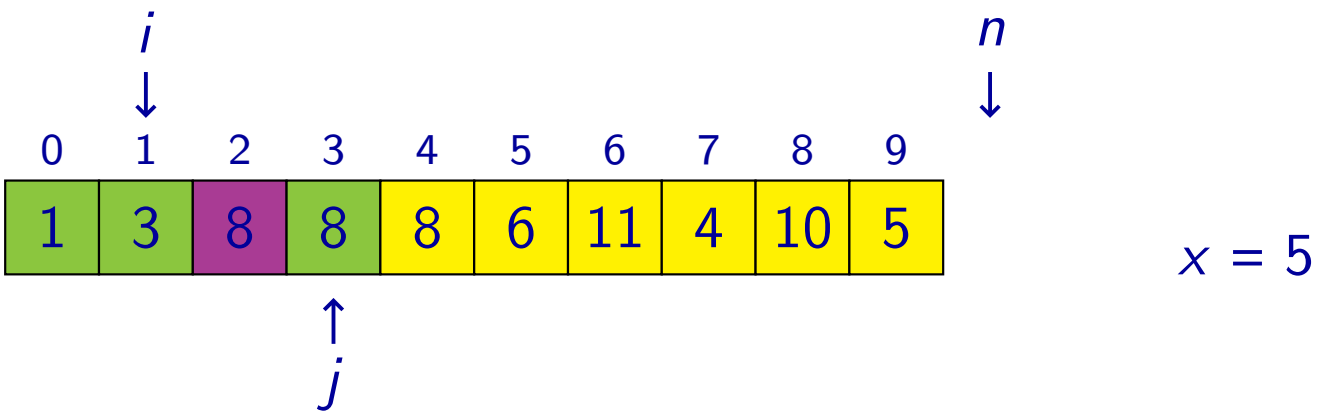
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

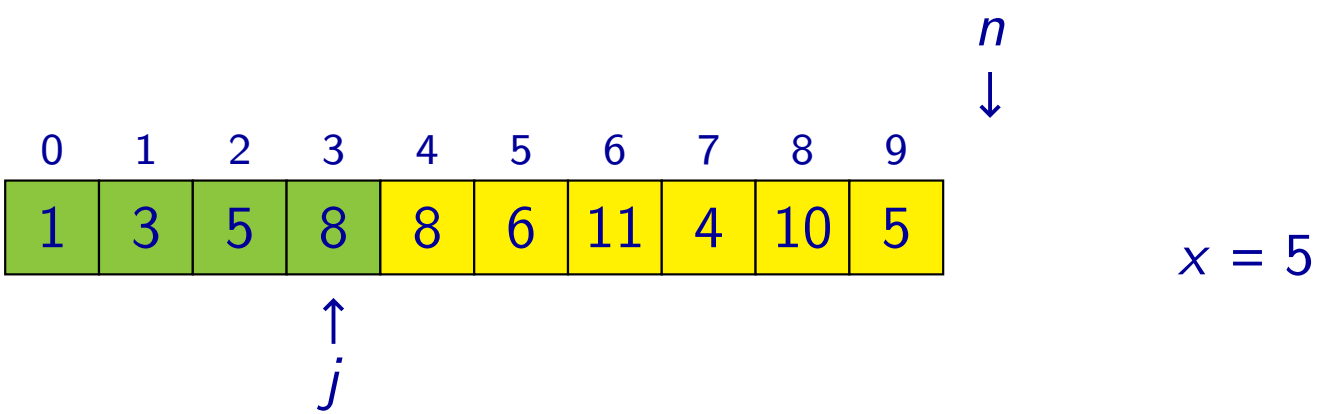
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

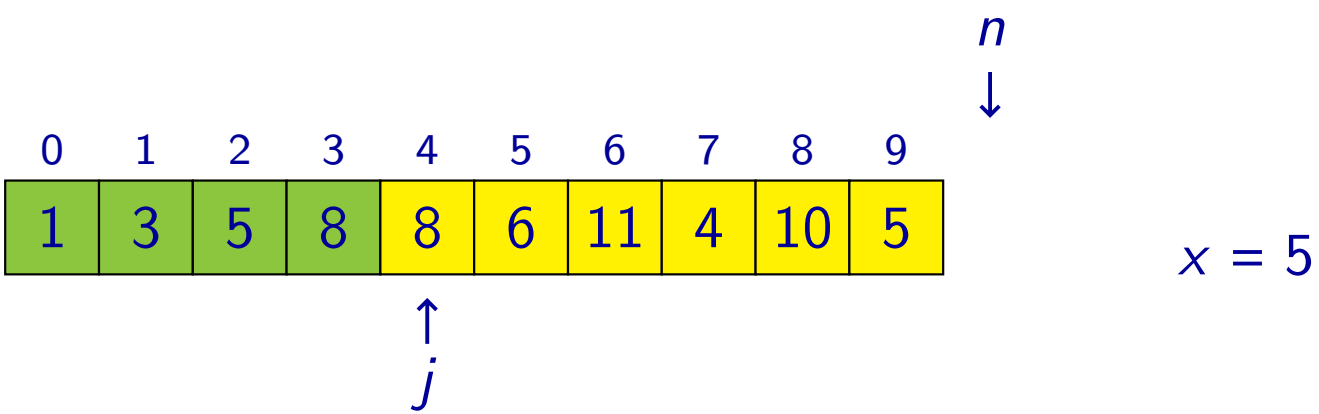
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

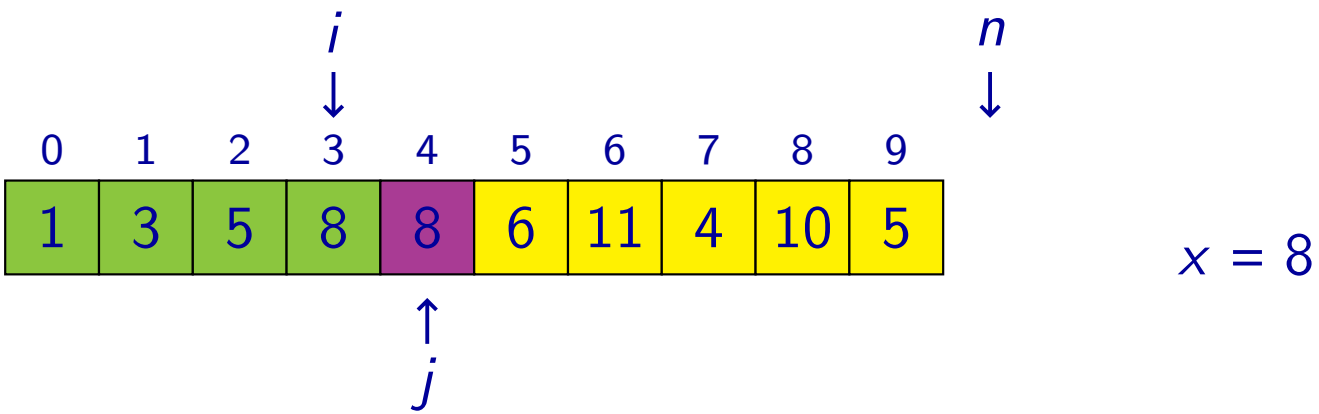
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

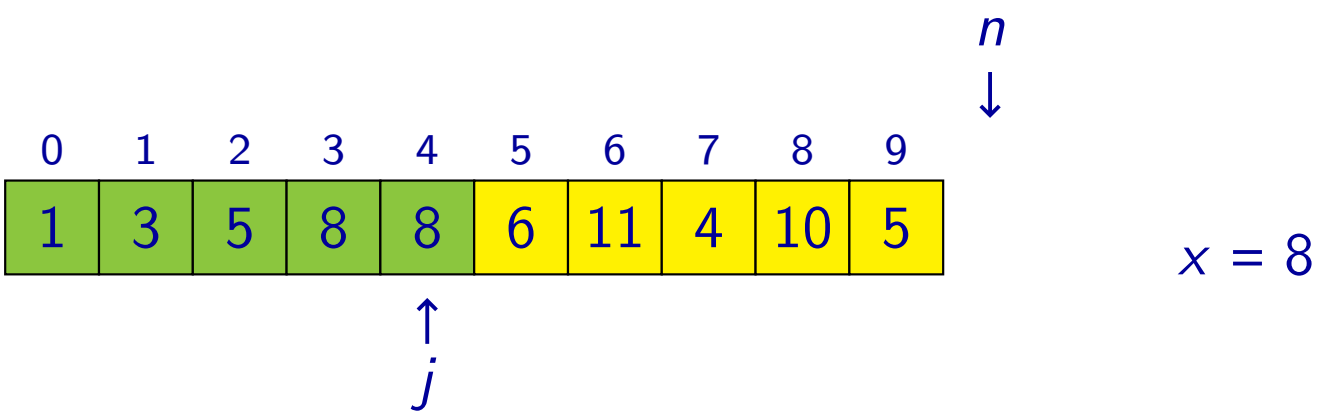
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

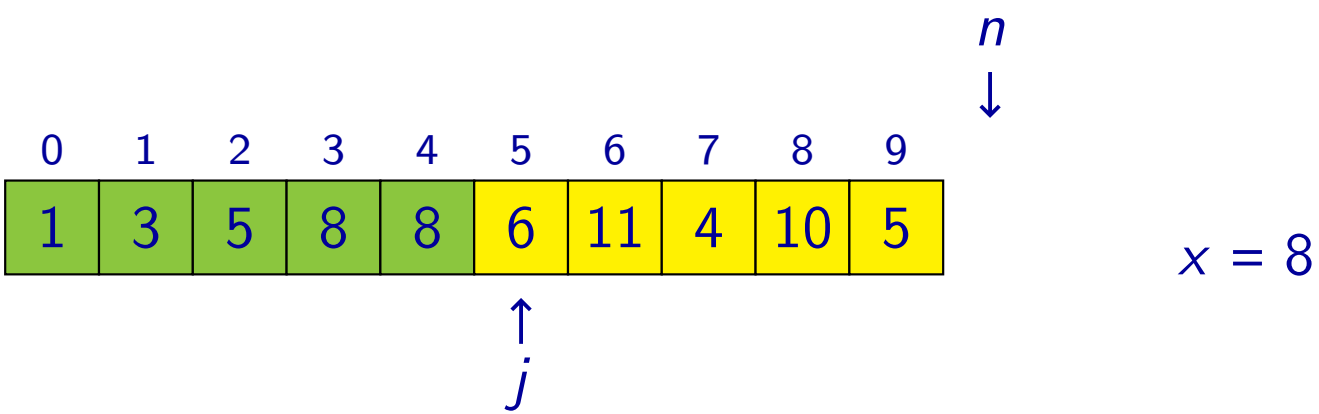
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$

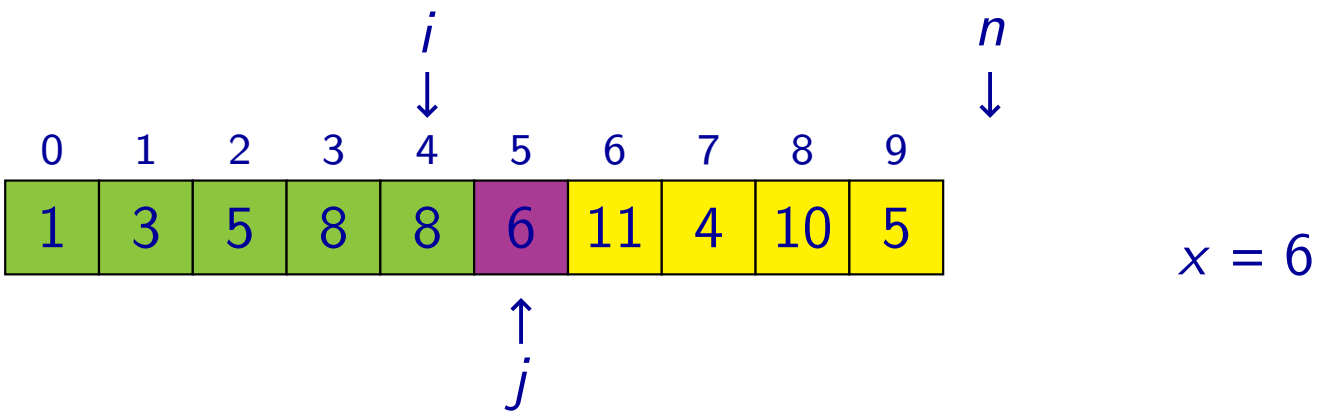


$x = 8$

Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

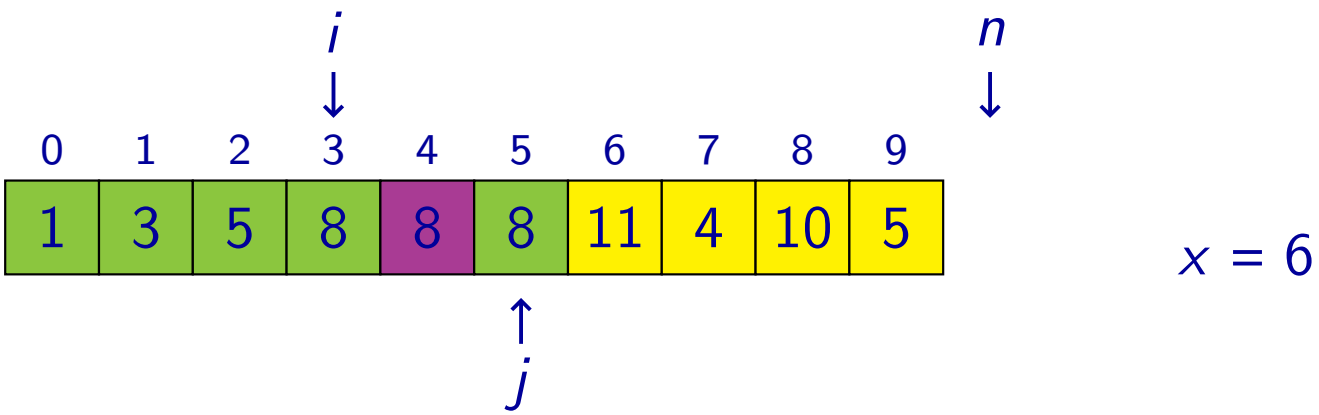
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

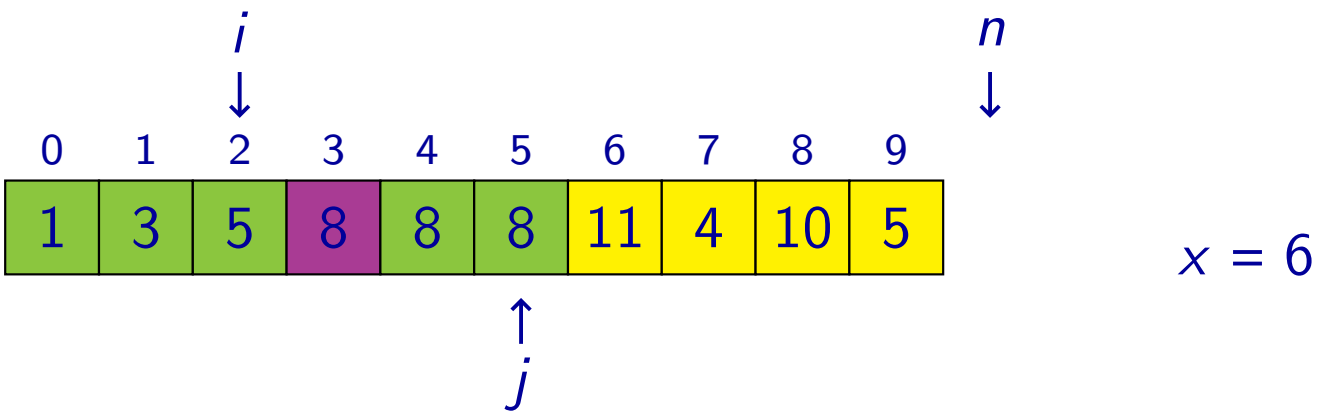
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

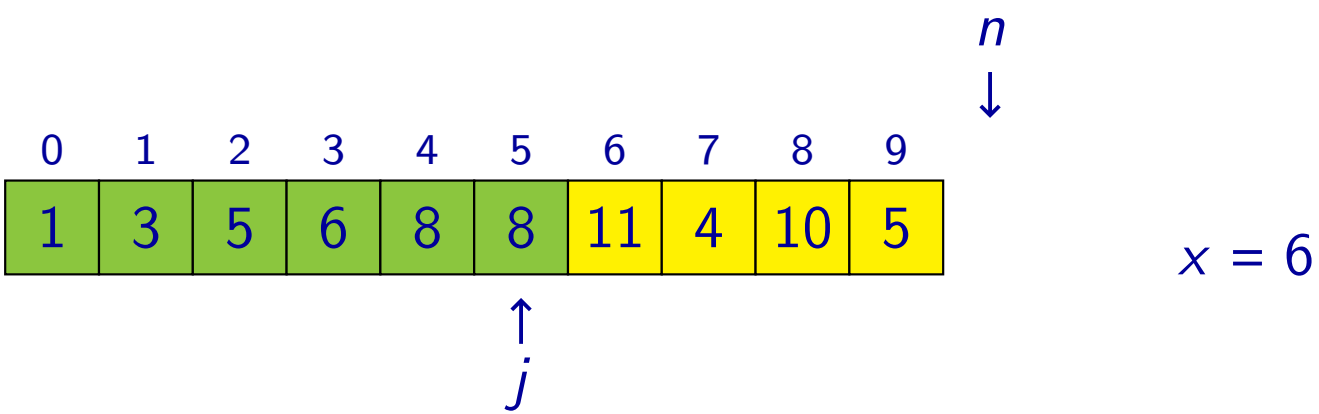
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$

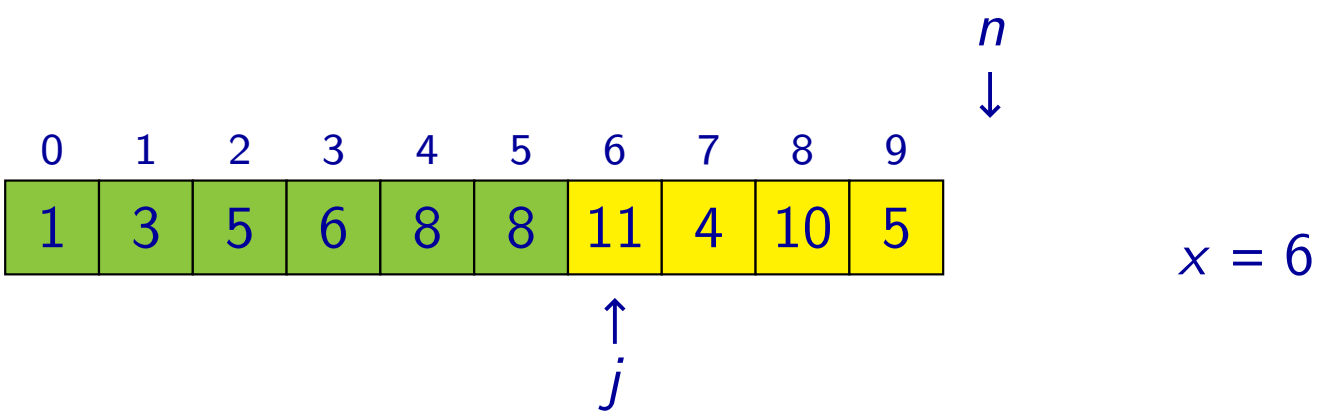


$x = 6$

Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

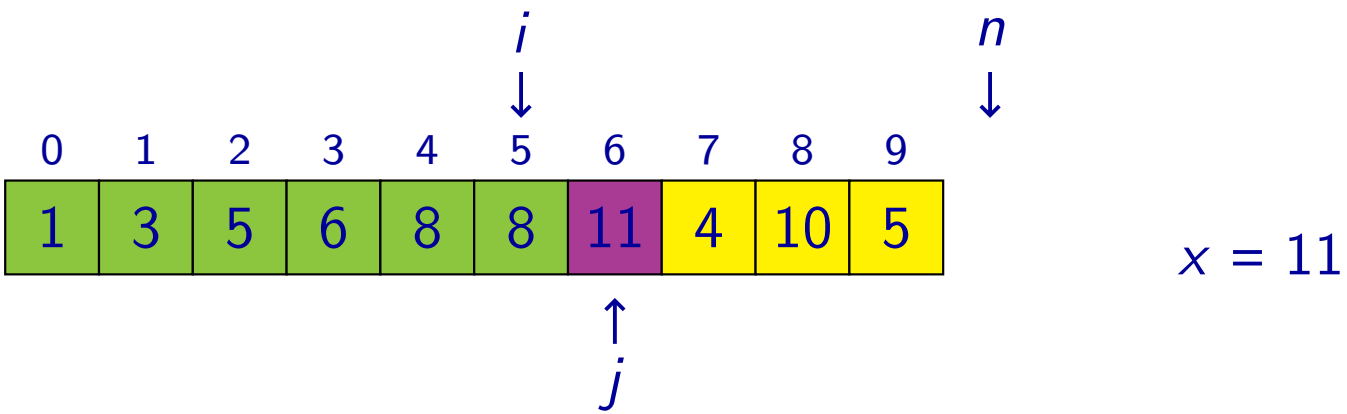
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

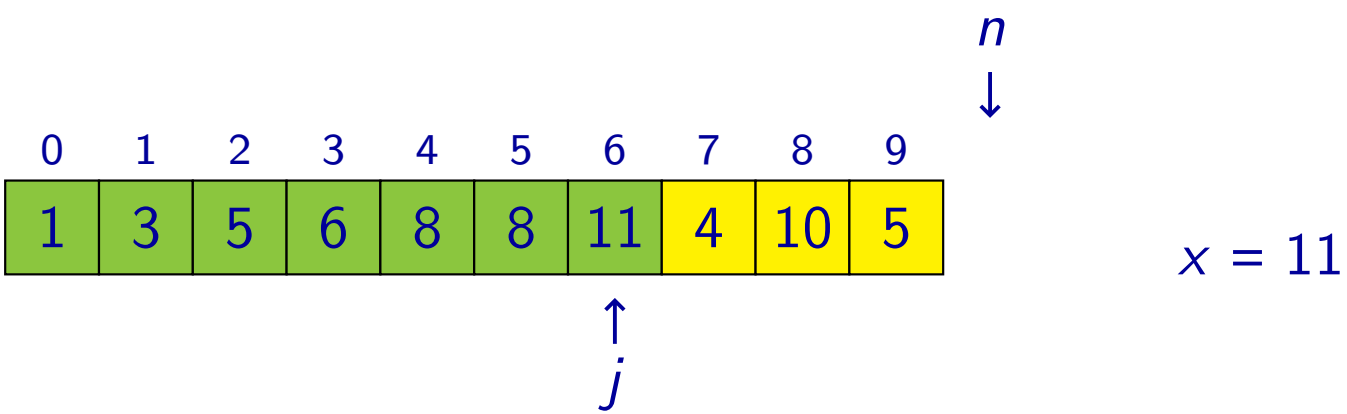
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

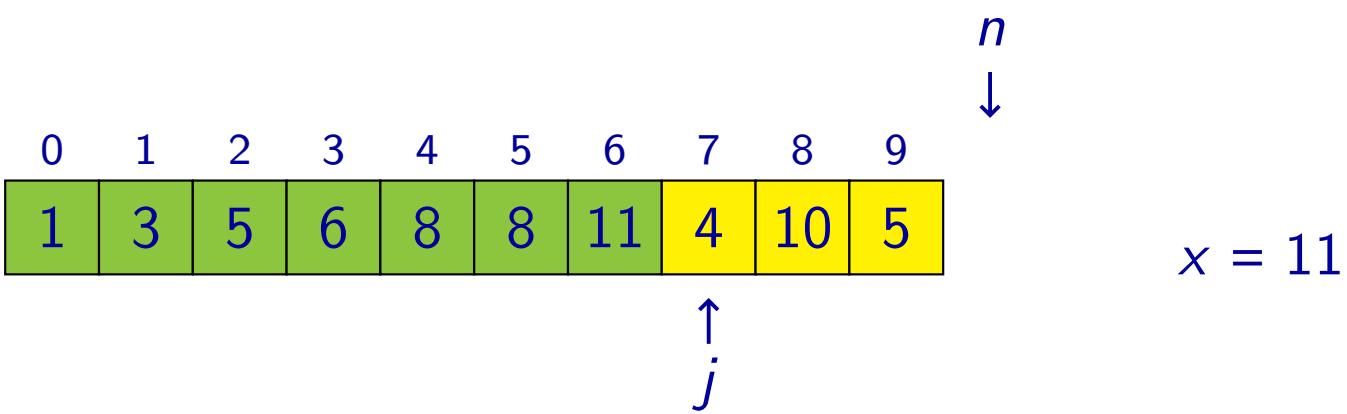
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

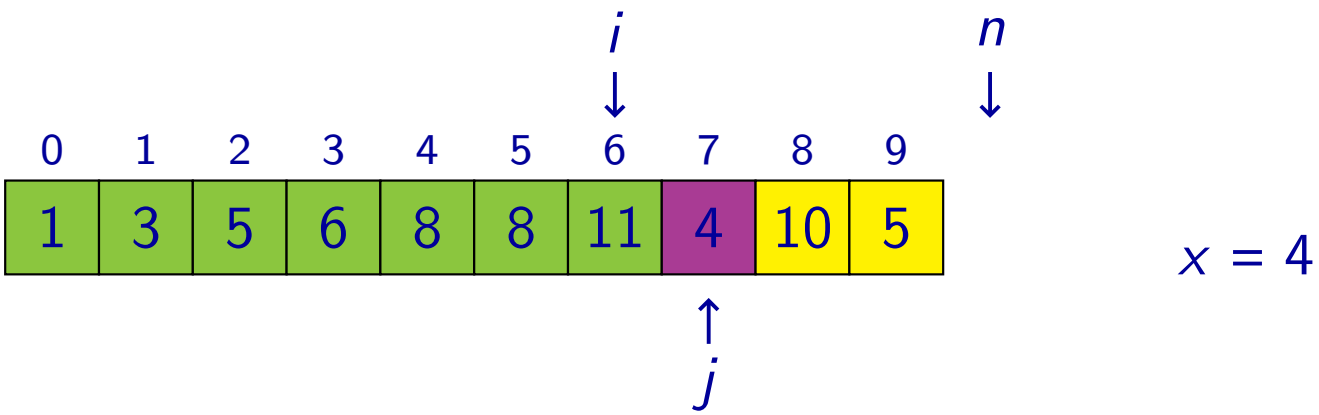
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

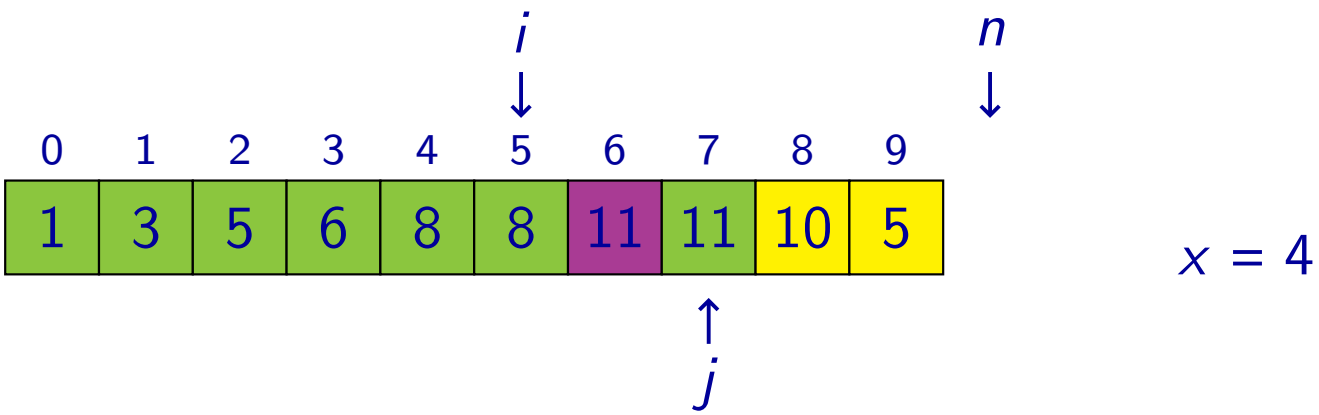
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

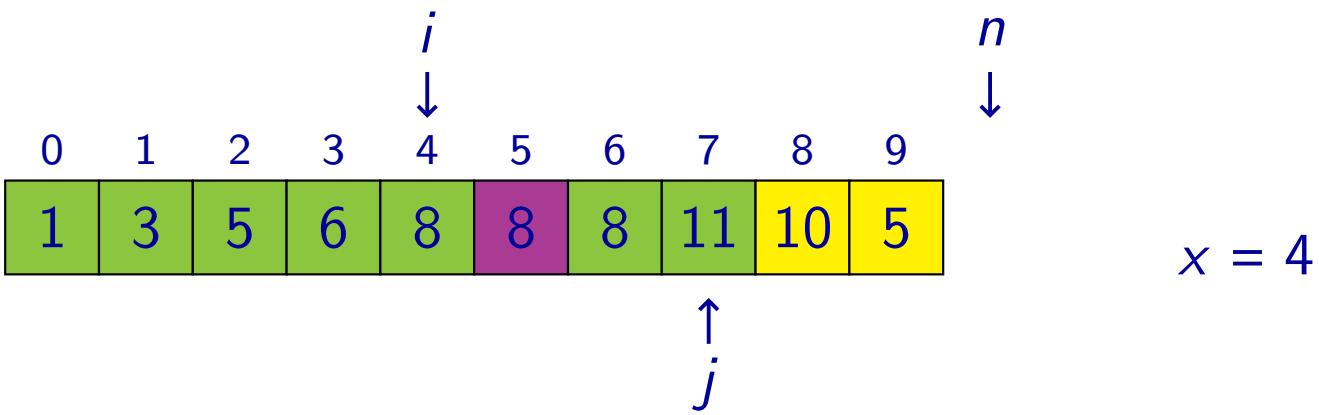
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

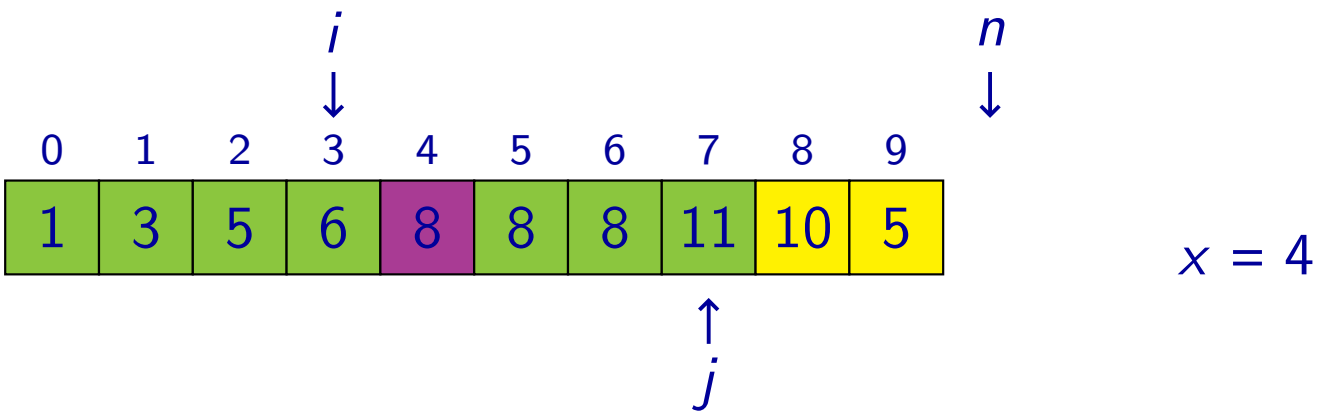
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

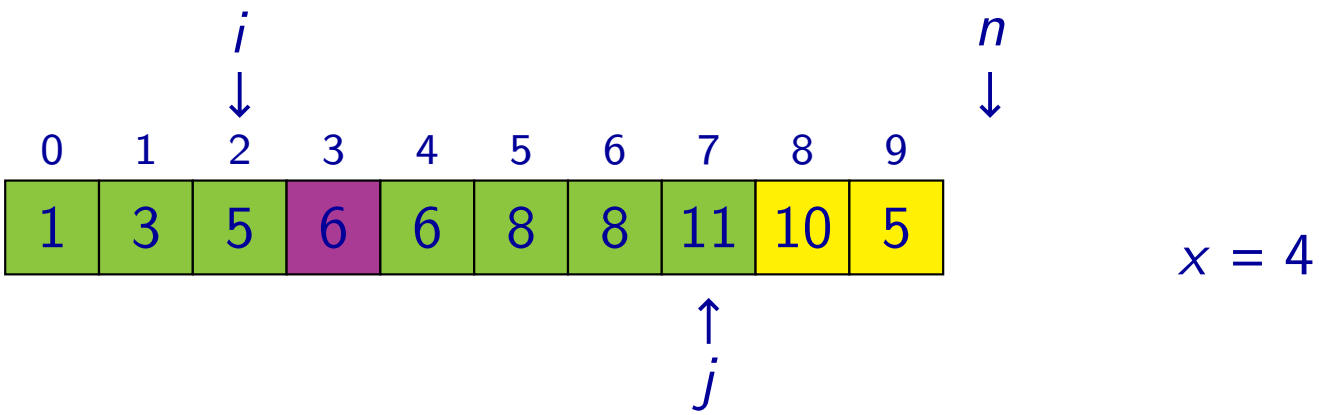
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

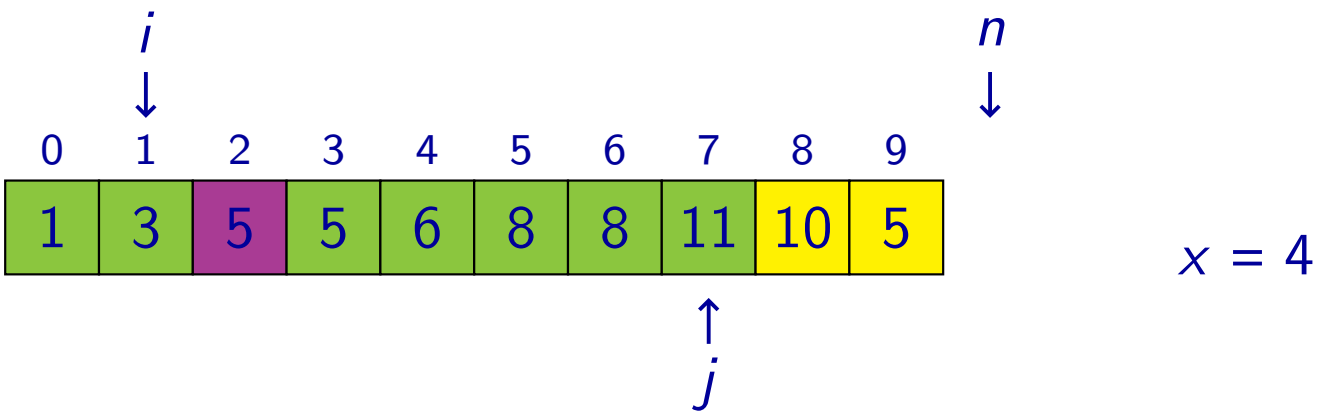
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

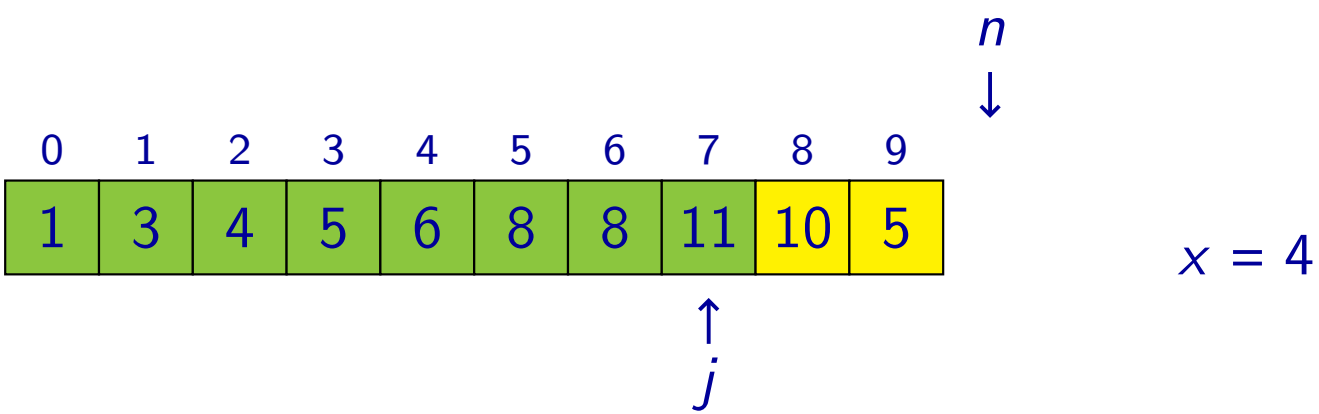
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

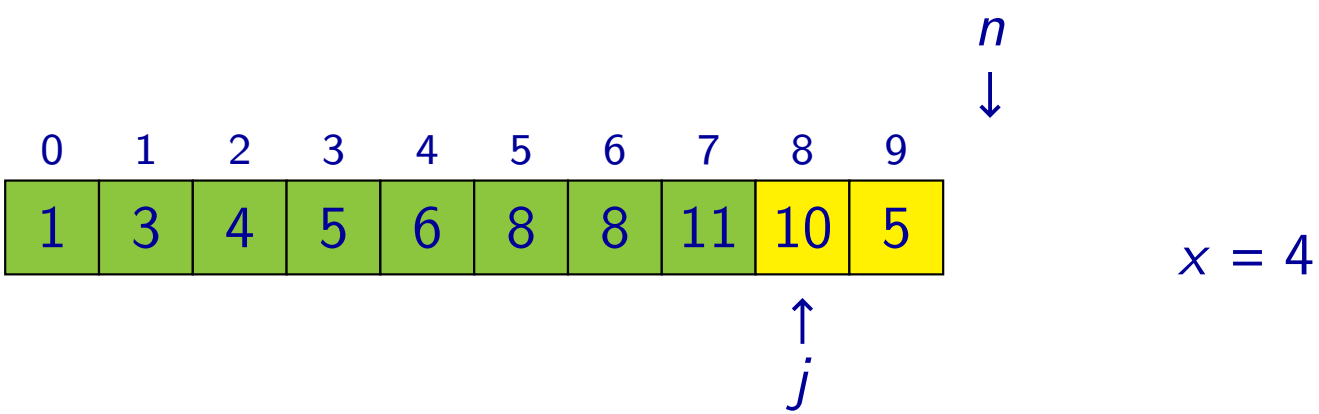
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

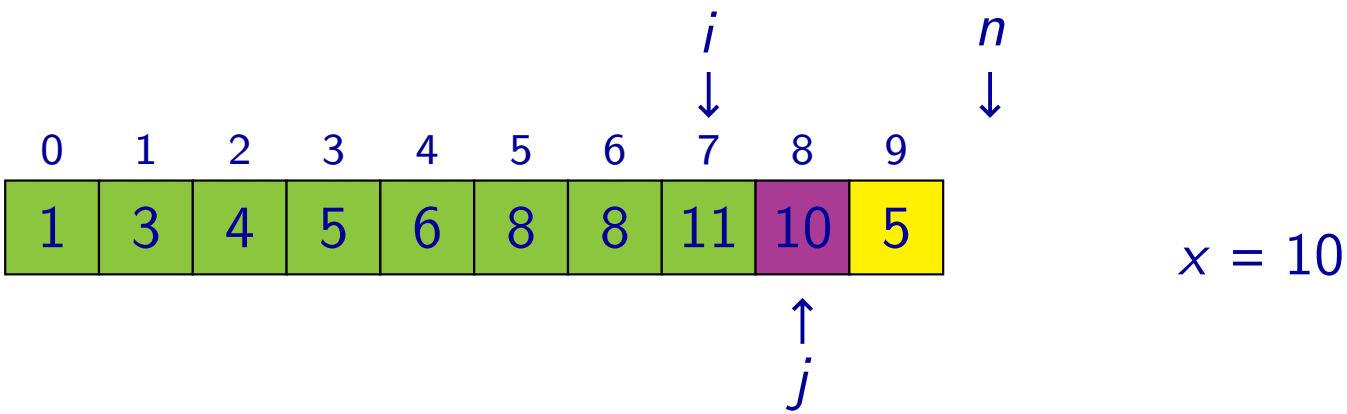
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

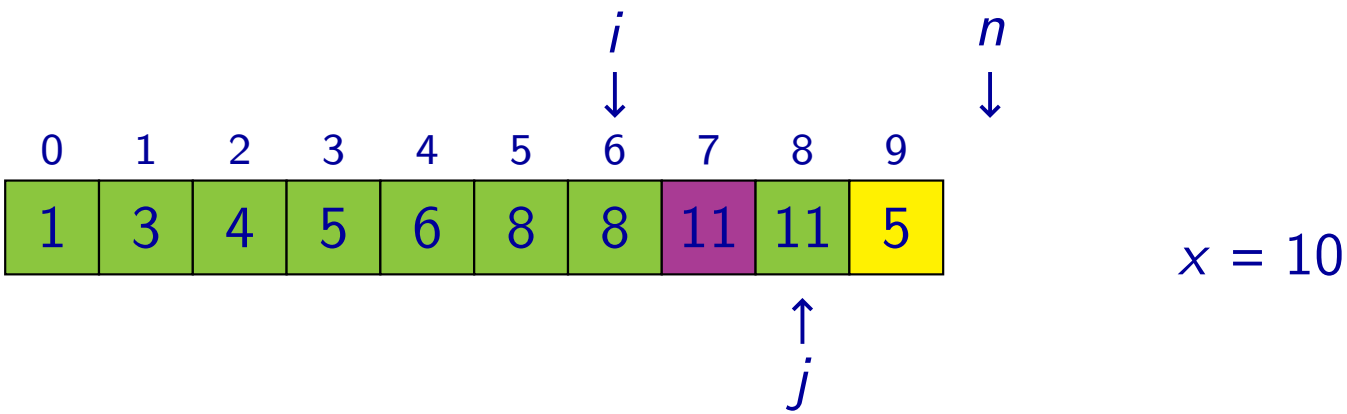
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

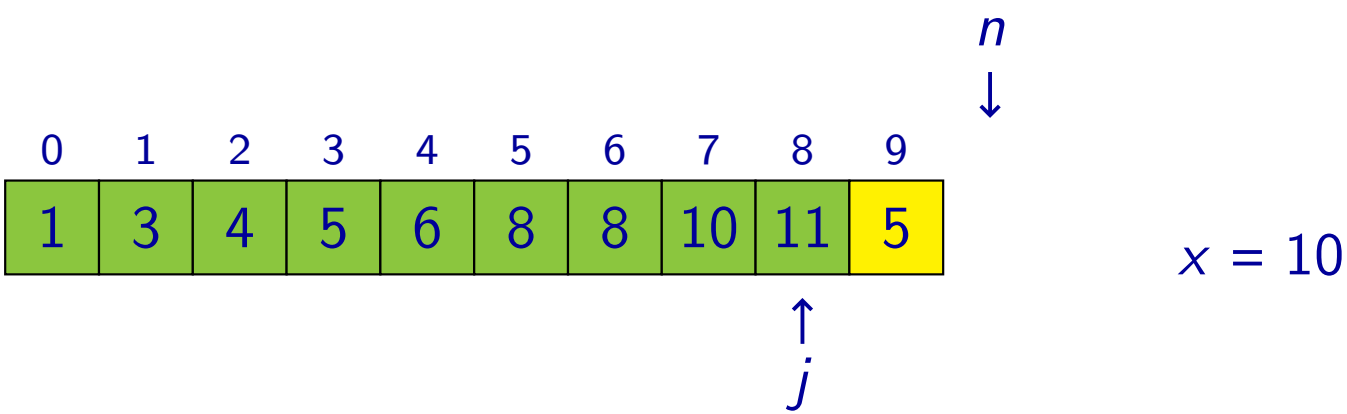
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

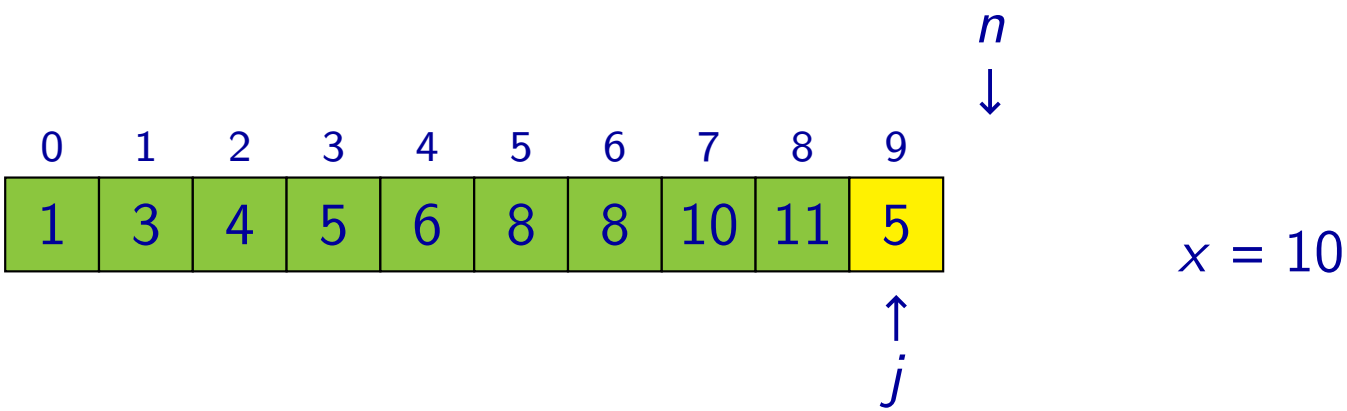
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

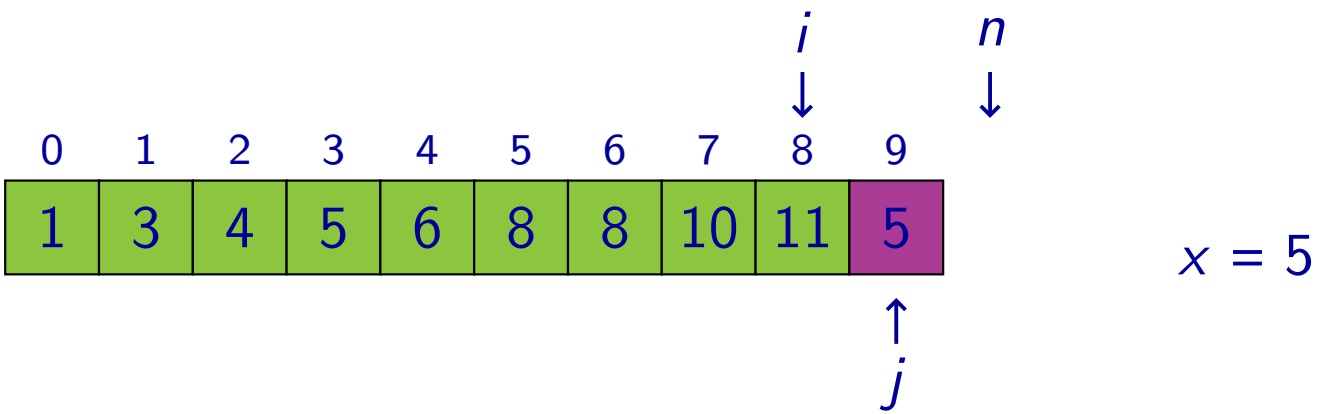
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

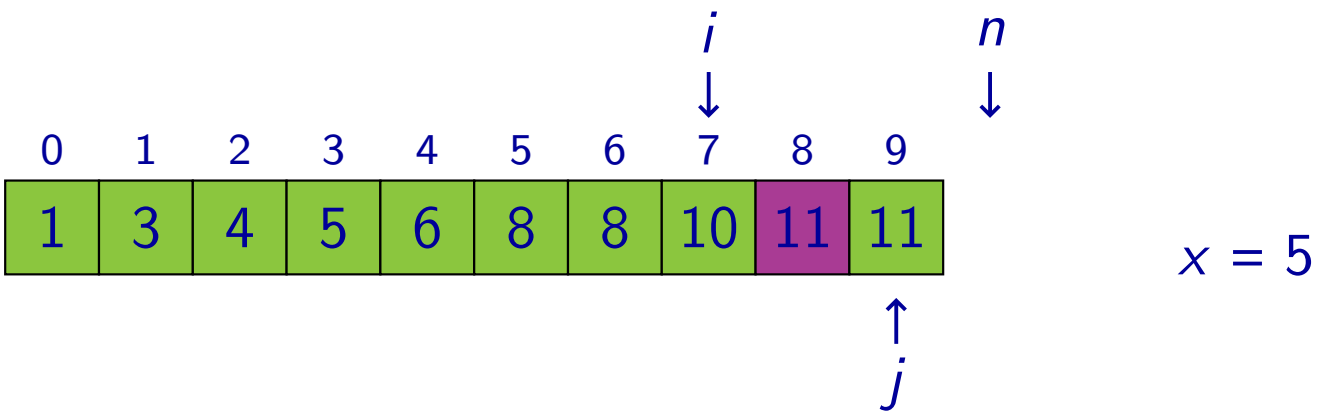
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

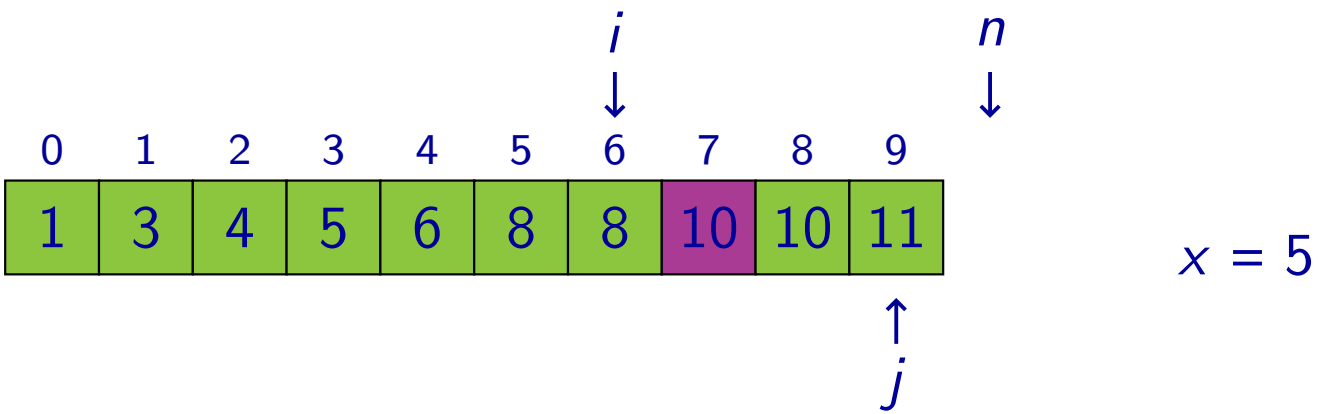
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

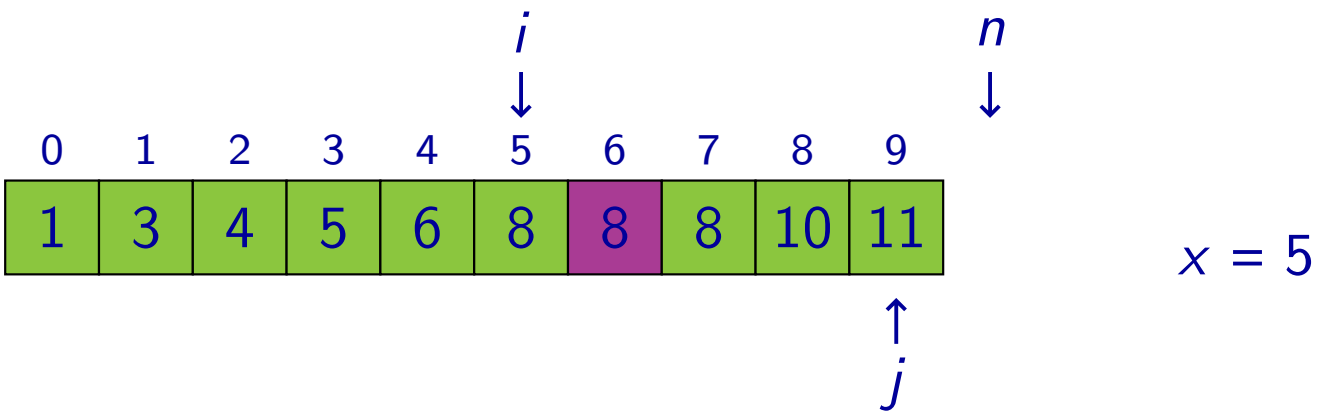
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

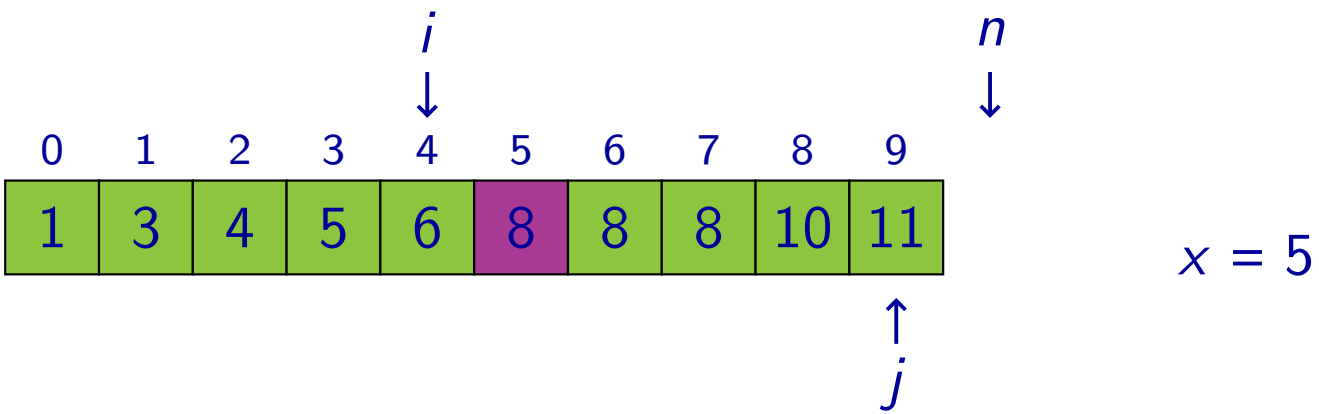
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

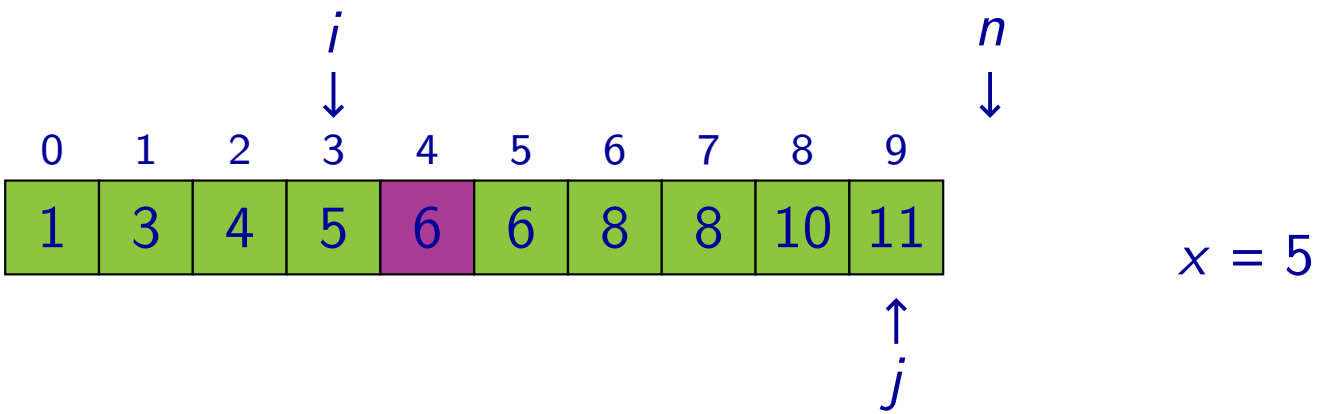
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

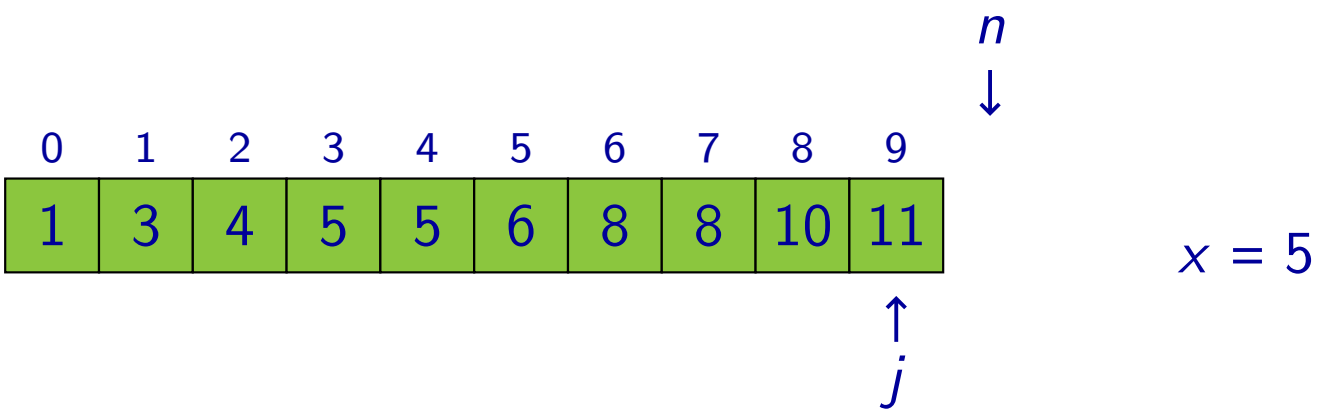
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

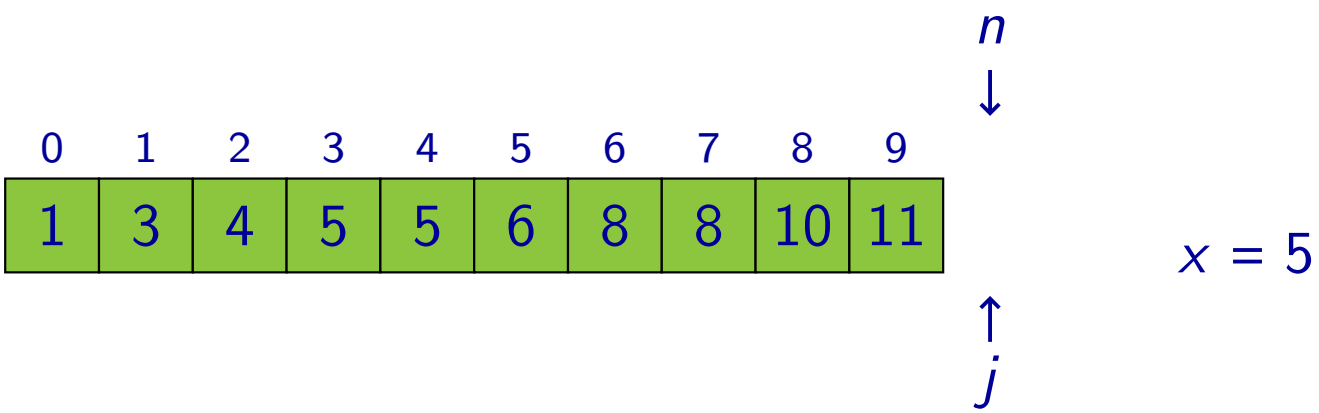
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invarianty

Předpokládejme, že vstupem je pole $A = [a_0, a_1, \ldots, a_{n-1}]$ a číslo n (kde $n \geq 1$) udávající délku tohoto pole, tj. že na začátku pro každé i, kde $0 \leq i < n$, platí $A[i] = a_i$.

- Na začátku cyklu **for** (tj. vždy před provedením testu $j < n$, resp. $j \leq n - 1$) platí následující invarianty:
  - $1 \leq j \leq n$
  - Prvky pole $A[0], A[1], \ldots, A[j-1]$ obsahují hodnoty $a_0, a_1, \ldots, a_{j-1}$ seřazené od nejmenší po největší, tj.
    $$A[0] \leq A[1] \leq \cdots \leq A[j-1]$$
  - Prvky pole $A[j], A[j+1], \ldots, A[n-1]$ obsahují hodnoty $a_j, a_{j+1}, \ldots, a_{n-1}$, tj.
    $$A[j] = a_j,\ A[j+1] = a_{j+1},\ \ldots,\ A[n-1] = a_{n-1}$$

Invarianty

- Na začátku cyklu **while** (tj. vždy před provedením testu $i \geq 0$) platí následující invarianty:
  - $1 \leq j < n$
  - $-1 \leq i < j$
  - Proměnná x obsahuje hodnotu a_j, tj. $x = a_j$.
  - Prvky pole $A[0], A[1], \ldots, A[i]$ a $A[i+2], A[i+3], \ldots, A[j]$ obsahují hodnoty $a_0, a_1, \ldots, a_{j-1}$ seřazené od nejmenší po největší, tj.
    $$A[0] \leq A[1] \leq \cdots \leq A[i] \leq A[i+2] \leq A[i+3] \leq \cdots \leq A[j]$$
  - Všechny prvky $A[i+2], A[i+3], \ldots, A[j]$ jsou ostře větší než x.
  - Prvky pole $A[j+1], A[j+2], \ldots, A[n-1]$ obsahují hodnoty $a_{j+1}, a_{j+2}, \ldots, a_{n-1}$, tj.
    $$A[j+1] = a_{j+1},\ A[j+2] = a_{j+2},\ \ldots,\ A[n-1] = a_{n-1}$$

Konečnost výpočtu

Dva možné případy, jak může vypadat nekonečný výpočet:

- nějaká konfigurace se zopakuje — následující konfigurace se opakují stále dokola
- objevují se stále nové a nové konfigurace

Konečnost výpočtu

Jeden z běžných způsobů dokazování toho, že se algoritmus zaručeně pro každý vstup po konečném počtu kroků zastaví:

- každé (dosažitelné) konfiguraci přiřadit hodnotu z nějaké vhodně zvolené množiny W
- na množině W definovat uspořádání $\leq$ takové, že ve W neexistují nekonečné (ostře) klesající posloupnosti
- ukázat, že s provedením každé instrukce se hodnota přiřazená konfiguraci zmenšuje, tj. pro $\alpha \xrightarrow{I} \alpha'$ je

$$f(\alpha) > f(\alpha')$$

($f(\alpha)$, $f(\alpha')$ jsou hodnoty z množiny W přiřazené konfiguracím α a α')

Konečnost výpočtu

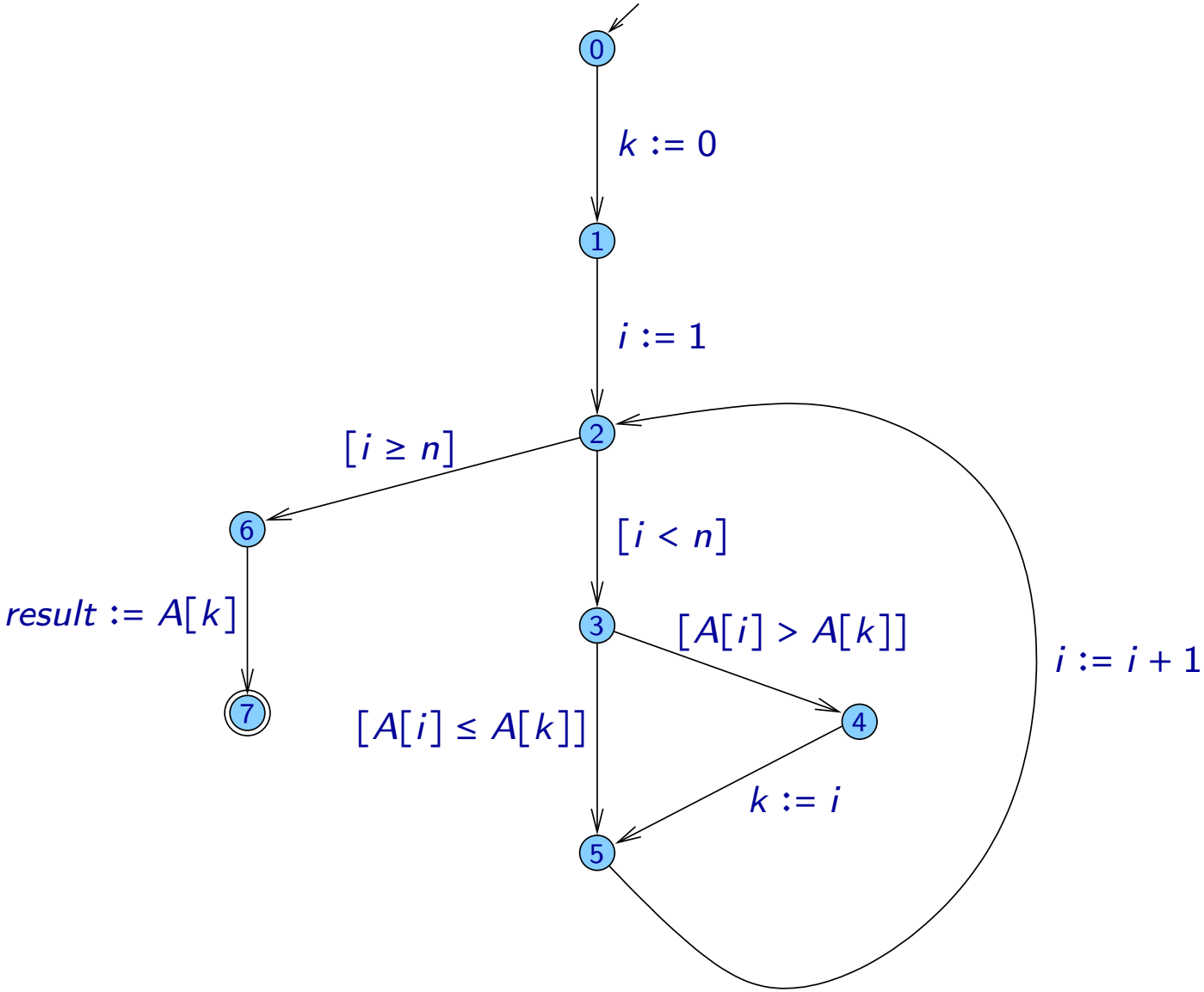
Jako množinu W je možno použít například:

- Množinu přirozených čísel $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ s uspořádáním $\leq$.
- Množinu vektorů přirozených čísel s lexikografickým uspořádáním, tj. s uspořádáním, kde vektor $(a_1, a_2, \dots, a_m)$ je menší než vektor $(b_1, b_2, \dots, b_n)$, jestliže
  - existuje i takové, že $1 \leq i \leq m$ a $i \leq n$, kde $a_i < b_i$ a pro všechna j taková, že $1 \leq j < i$, platí $a_j = b_j$, nebo
  - $m < n$ a pro všechna j taková, že $1 \leq j \leq m$, je $a_j = b_j$.

  Například $(5, 1, 3, 6, 4) < (5, 1, 4, 1)$ a $(4, 1, 1) < (4, 1, 1, 3)$.

  Poznámka: Počet prvků vektorů musí být omezen nějakou konstantou.

Konečnost výpočtu

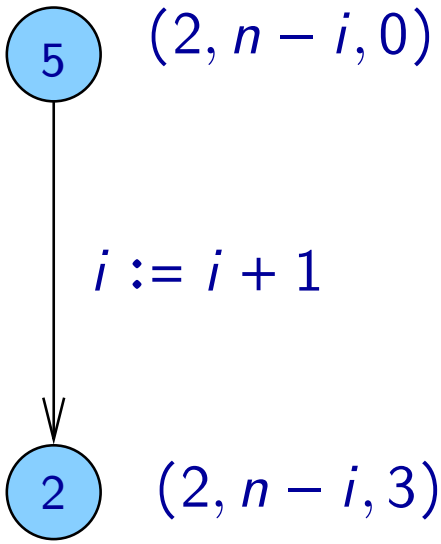


Konečnost výpočtu

Příklad: Vektory přiřazené jednotlivým konfiguracím:

- Stav 0: $f(\alpha) = (4)$
- Stav 1: $f(\alpha) = (3)$
- Stav 2: $f(\alpha) = (2, n - i, 3)$
- Stav 3: $f(\alpha) = (2, n - i, 2)$
- Stav 4: $f(\alpha) = (2, n - i, 1)$
- Stav 5: $f(\alpha) = (2, n - i, 0)$
- Stav 6: $f(\alpha) = (1)$
- Stav 7: $f(\alpha) = (0)$

Konečnost výpočtu



Je třeba brát v úvahu, že se touto instrukcí hodnota proměnné i mění.

Z konfigurace s přiřazeným vektorem $(2, n-i, 0)$ se přejde do konfigurace s přiřazeným vektorem $(2, n-i', 3)$, kde $i' = i + 1$.

Zjevně platí $n - i' < n - i$, neboť $n - (i + 1) < n - i$.

Výpočetní složitost algoritmů

Složitost algoritmu

- Počítače pracují rychle, ale ne nekonečně rychle. Provedení každé instrukce trvá nějakou (i když velmi krátkou) dobu.
- Stejný problém může řešit více různých algoritmů a doba výpočtu (daná hlavně počtem provedených instrukcí) může být pro různé algoritmy různá.
- Algoritmy bychom chtěli mezi sebou porovnávat a zvolit si ten lepší.
- Algoritmy můžeme naprogramovat a změřit čas výpočtu. Tím zjistíme jak dlouho trvá výpočet na konkrétních datech, na kterých algoritmus testujeme.
- Chtěli bychom mít i nějakou přesnější představu o tom, jak dlouho bude trvat výpočet na všech možných vstupních datech.

Složitost algoritmu

- Doba výpočtu je ovlivněna mnoha faktory, např.:
  - použitý algoritmus
  - množství vstupních dat
  - použitý hardware (důležitá může být např. taktovací frekvence procesoru)
  - použitý programovací jazyk — a jeho konkrétní implementace (překladač/interpreter)
  - ...
- Pokud potřebujeme řešit problém pro „malá“ vstupní data, doba výpočtu je většinou zanedbatelná.
- S narůstajícím množstvím vstupních dat (velikosti vstupu) může doba výpočtu růst, někdy velmi výrazně.

Složitost algoritmu

- **Časová složitost algoritmu** — jak závisí doba výpočtu na množství vstupních dat
- **Paměťová** (resp. **prostorová**) **složitost algoritmu** — jak závisí množství použité paměti na množství vstupních dat

Poznámka: Přesné definice těchto pojmů budou uvedeny za chvíli.

Poznámka:

- Existují i další typy výpočetní složitosti, kterými se nebudeme zabývat (např. komunikační složitost).

Složitost algoritmu

Vezměme si nějaký konkrétní stroj vykonávající nějaký algoritmus — např. stroj RAM, Turingův stroj, ...

Budeme předpokládat, že pro daný stroj $\mathcal{M}$ máme nějak definované pro libovolný vstup w z množiny všech vstupů In následující dvě funkce:

- $time_{\mathcal{M}} : In \to \mathbb{N}$ — vyjadřuje dobu výpočtu stroje $\mathcal{M}$ nad vstupem w
- $space_{\mathcal{M}} : In \to \mathbb{N}$ — vyjadřuje množství paměti použité strojem $\mathcal{M}$ při výpočtu nad vstupem w

Poznámka: Předpokládáme, že výpočet stroje $\mathcal{M}$ nad libovolným vstupem w se po konečném počtu kroků zastaví.

Složitost algoritmu

Příklad:

- Jednopáskový Turingův stroj $\mathcal{M}$:
  - $time_{\mathcal{M}}(w)$ — počet kroků, které vykoná $\mathcal{M}$ při výpočtu nad vstupem w
  - $space_{\mathcal{M}}(w)$ — počet políček navštívených na pásce během výpočtu nad vstupem w

- Stroj RAM:
  - $time_{\mathcal{M}}(w)$ — počet kroků, které vykoná daný stroj RAM při výpočtu nad vstupem w
  - $space_{\mathcal{M}}(w)$ — počet buněk poměti, které byly použity během výpočtu nad vstupem w (bylo do nich něco zapsáno nebo z nich bylo čteno)

Velikost vstupu

Pro různé vstupy provede program různý počet instrukcí.

Pokud chceme počet provedených instrukcí nějak analyzovat, je vhodné si zavést pojem **velikost vstupu**.

Typicky je velikost vstupu číslo, které udává, jak je daná instance „velká“ (čím větší číslo, tím větší instance).

Poznámka: Velikost vstupu si v daném konkrétním případě můžeme definovat, jak chceme a jak je to pro další analýzu výhodné.

Co přesně zvolíme jako velikost vstupu není předem dáno, ale z podstaty zadaného problému většinou nějak přirozeně vyplývá, co za velikost vstupu zvolit.

Velikost vstupu

Příklady:

- Pro problém „Třídění“, kde vstupem je sekvence čísel $a_1, a_2, \ldots, a_n$ a výstupem jsou tato čísla setříděná, můžeme vzít jako velikost vstupu hodnotu n.
- Pro problém „Prvočíselnost“, kde vstupem je přirozené číslo x, a kde se ptáme, zda x je prvočíslo, můžeme vzít jako velikost vstupu počet bitů čísla x.

  (Jinou možností by bylo vzít jako velikost vstupu přímo hodnotu x.)

Velikost vstupu

Někdy je vhodné popsat velikost vstupu pomocí více čísel.

Například u problémů, kde vstupem je graf, můžeme definovat velikost vstupu jako dvojici čísel n, m, kde:

- n – počet vrcholů grafu
- m – počet hran grafu

Poznámka: Jinou možností by bylo definovat velikost vstupu jako jediné číslo $n + m$.

Velikost vstupu

Obecně můžeme pro libovolný problém definovat velikost vstupu následovně:

- Pokud je vstupem slovo w z nějaké obecedy Σ:
  délka slova w
- Pokud je vstupem sekvence bitů (tj. slovo z abecedy $\{0, 1\}$):
  počet bitů v této sekvenci
- Pokud je vstupem přirozené číslo x:
  počet bitů nutných k zápisu čísla x

Časová složitost

Chceme analyzovat konkrétní algoritmus (jeho konkrétní implementaci).

Zajímá nás, kolik instrukcí se provede, pokud algoritmus dostane vstup velikosti $0, 1, 2, 3, 4, \ldots$.

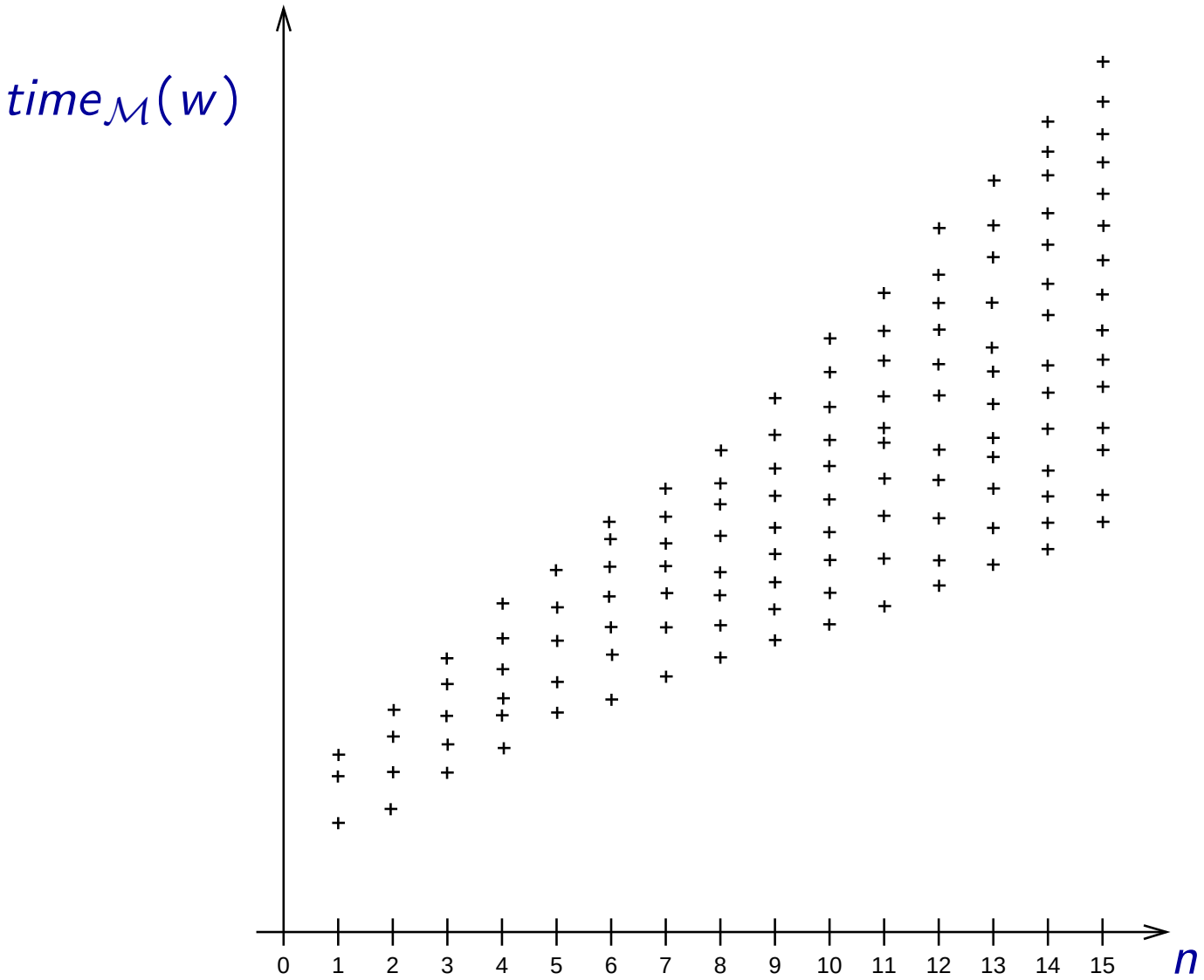
Je zřejmé, že i pro vstupy, které mají stejnou velikost, může být počet provedených instrukcí různý.

Označme si velikost vstupu $w \in In$ jako $size(w)$.

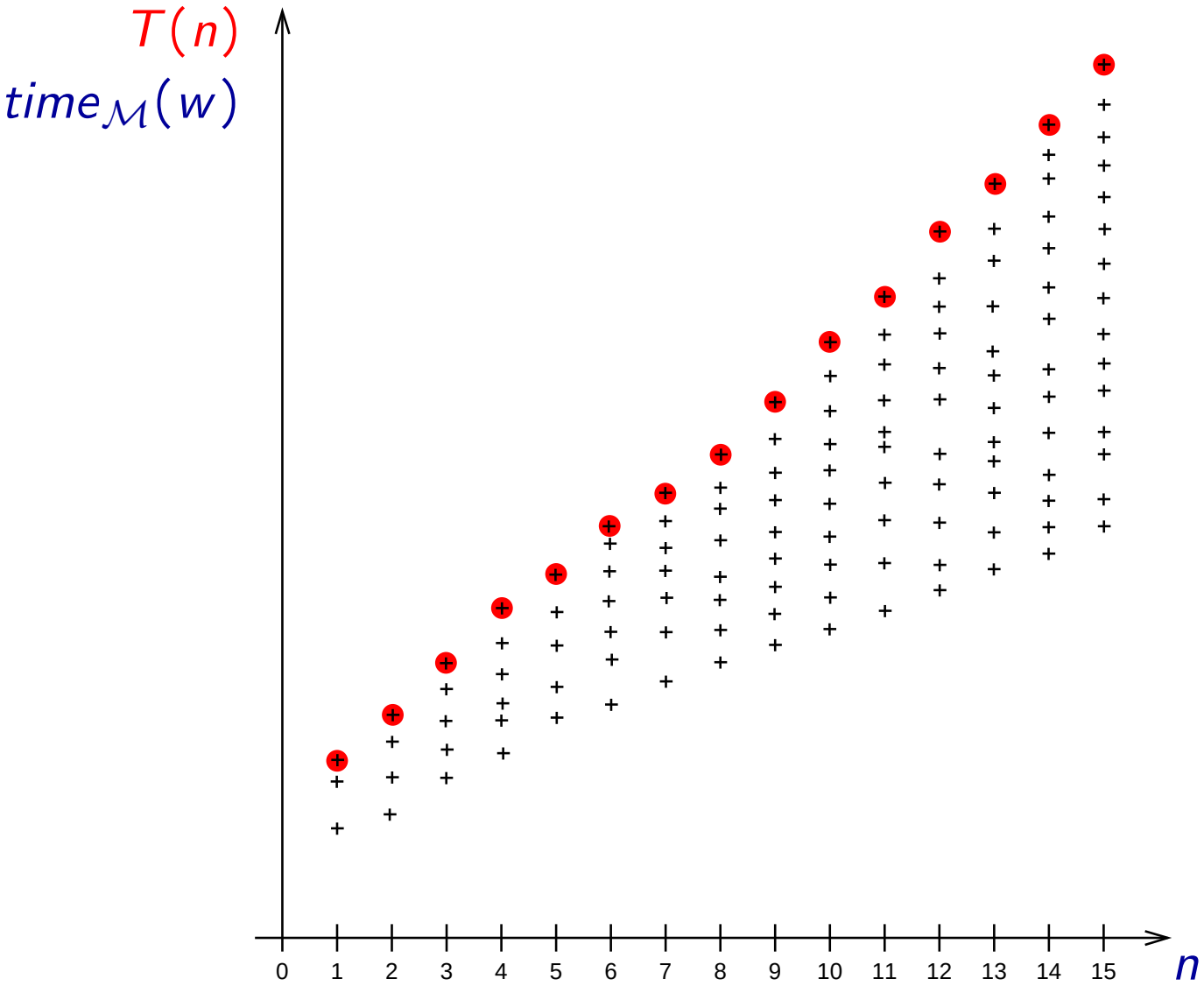
Nyní definujme následující funkci $T : \mathbb{N} \to \mathbb{N}$ takovou, že pro $n \in \mathbb{N}$ je

$$T(n) = \max\{\, time_{\mathcal{M}}(w) \mid w \in In,\ size(w) = n \,\}$$

Časová složitost v nejhorším případě



Časová složitost v nejhorším případě



Časová a prostorová složitost v nejhorším případě

Takto definované funkci $T(n)$ (tj. funkci, která pro daný algoritmus a danou definici velikosti vstupu přiřazuje každému přirozenému číslu n maximální počet instrukcí, které algoritmus provede, pokud dostane vstup velikosti n) se říká **časová složitost algorimu v nejhorším případě**.

$$T(n) = \max\{\, time_{\mathcal{M}}(w) \mid w \in In,\ size(w) = n \,\}$$

Analogicky můžeme definovat **prostorovou (paměťovou) složitost algoritmu v nejhorším případě** jako funkci $S(n)$, kde as a function $S(n)$ where:

$$S(n) = \max\{\, space_{\mathcal{M}}(w) \mid w \in In,\ size(w) = n \,\}$$

Časová složitost v průměrném případě

Kromě časové složitosti v nejhorším případě má smysl zkoumat i časovou složitost **v průměrném případě**.

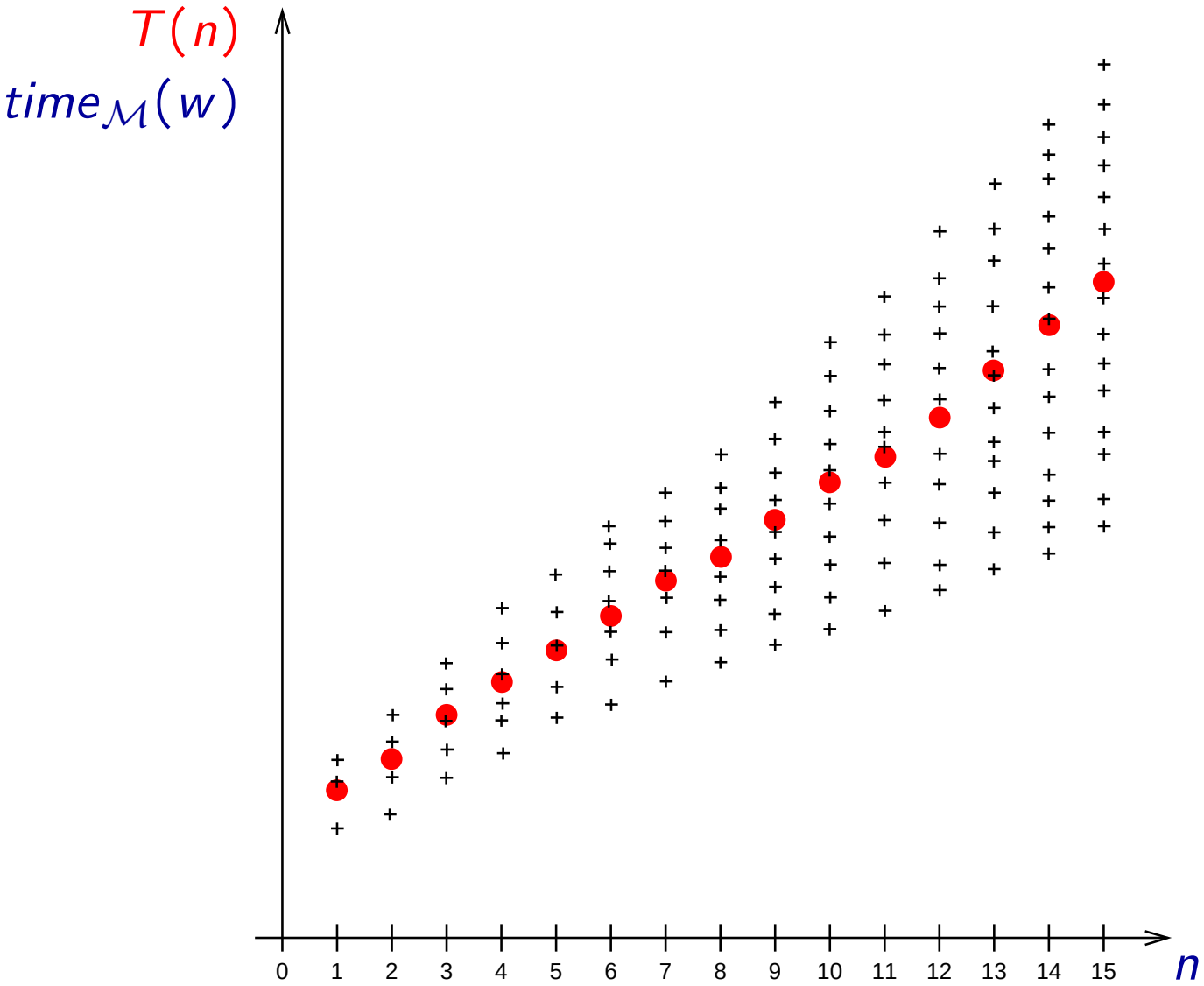
V tomto případě $T(n)$ nedefinujeme jako maximum, ale jako aritmetický průměr z hodnot

$$\{ time_{\mathcal{M}}(w) \mid w \in In, size(w) = n \}$$

- Určit časovou složitost v průměrném případě je většinou těžší než určit časovou složitost v nejhorším případě.
- Často se tyto dvě funkce příliš neliší, někdy je ale rozdíl významný.

Poznámka: Zkoumat složitost v **nejlepším případě** většinou moc smysl nemá.

Časová složitost v průměrném případě



Výpočetní složitost algoritmu

Z definice vidíme, že jak časová, tak prostorová, složitost algoritmu jsou funkce, jejíchž přesné hodnoty závistí nejen na daném algoritmu Alg, ale také na následujících věcech:

- na stroji $\mathcal{M}$, na kterém algoritmus Alg běží,
- na definici doby výpočtu $time_{\mathcal{M}}(w)$ a množství použité paměti $space_{\mathcal{M}}(w)$ algoritmu Alg na stroji $\mathcal{M}$ pro vstup $w \in In$,
- na definici velikosti vstupu (tj. definici funkce $size$).

Výpočetní složitost algoritmu

Přesné určení doby výpočtu nebo množství použité paměti může být extrémně komplikované.

Většinou se při analýze výpočetní složitosti algoritmu používá celá řada zjednodušení:

- Většinou se neanalyzuje, jak závisí doba výpočtu nebo množství použité paměti na konkrétních vstupních datech, ale pouze, jak závistí na **velikosti vstupu**, tj. na množství těchto dat.
- Funkce vyjadřující, jak roste doba výpočtu nebo množství použité paměti v závislosti na velikosti vstupu, se nepočítají přesně — počítají se **odhady** těchto funkcí.
- Odhady těchto funkcí se vyjadřují pomocí tzv. **asymptotické notace** — např. se řekne, že časová složitost algoritmu MergeSort je $O(n \log n)$, zatímco časová složitost algoritmu BubbleSort je $O(n^2)$.

Časová složitost algoritmu

Příklad analýzy časové složitosti algoritmu **bez** použití asymptotické notace:

- Takto podrobně se analýza výpočetní složitosti algoritmu téměř nikdy **nedělá** — je to příliš pracné a komplikované.
- Uvidíme tak ale, co vše je při použití asymptotické notace zanedbáno a o kolik je analýza s použitím asymptotické notace jednodušší.
- Budeme počítat s konstantami $c_0, c_1, \ldots, c_k$, které udávají dobu trvání jednotlivých instrukcí — nebudeme počítat s konkrétními čísly.

Doba výpočtu

Řekněme, že máme algoritmus reprezentován ve formě grafu řídícího toku:

- Každé instrukci (tj. každé hraně) přiřadíme hodnotu udávající, jak dlouho trvá provedení této instrukce.
- Provedení různých instrukcí může trvat různou dobu.
- Pro jednoduchost předpokládejme, že provedení té samé instrukce trvá pokaždé stejnou dobu — hodnota přiřazená dané instrukci je číslo z množiny $\mathbb{R}_+$ (množina nezáporných reálných čísel).

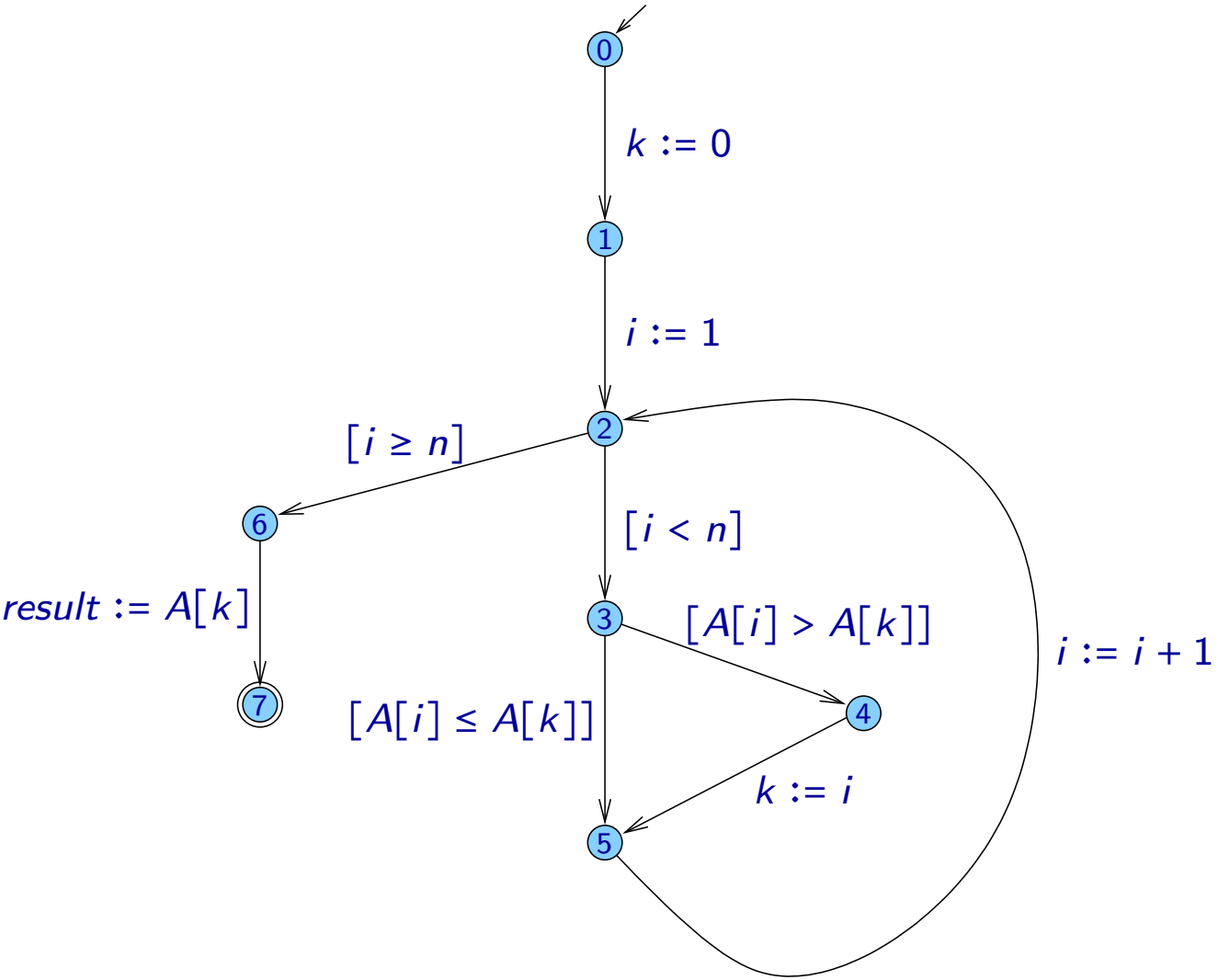
Doba výpočtu

Příklad:

Algoritmus: Nalezení největšího prvku v poli

```
FIND-MAX (A, n):
    k := 0
    for i := 1 to n − 1 do
        if A[i] > A[k] then
            k := i
    return A[k]
```

Doba výpočtu



Instr.	doba
$k := 0$	c_0
$i := 1$	c_1
$[i < n]$	c_2
$[i \geq n]$	c_3
$[A[i] \leq A[k]]$	c_4
$[A[i] > A[k]]$	c_5
$k := i$	c_6
$i := i + 1$	c_7
$result := A[k]$	c_8

Doba výpočtu

Příklad: Doby provedení jednotlivých instrukcí by mohly být třeba:

Instr.	označení	doba
$k := 0$	c_0	4
$i := 1$	c_1	4
$[i < n]$	c_2	10
$[i \geq n]$	c_3	12
$[A[i] \leq A[k]]$	c_4	14
$[A[i] > A[k]]$	c_5	12
$k := i$	c_6	5
$i := i + 1$	c_7	6
$result := A[k]$	c_8	5

Pro konkrétní vstup w, např. pro $w = ([3, 8, 4, 5, 2], 5)$, bychom mohli výpočet odsimulovat a určit konkrétní dobu výpočtu $t(w)$.

Časová složitost algoritmu

Předpokládáme vstupy tvaru (A, n), kde A je pole a n počet prvků tohoto pole (přičemž $n \geq 1$).

Jako velikost vstupu (A, n) zvolme n.

Uvažujme nyní o nějaké jednom vstupu $w = (A, n)$ velikosti n:

- Dobu výpočtu $t(w)$ nad vstupem w můžeme vyjádřit jako

$$t(w) = c_0 \cdot m_0(w) + c_1 \cdot m_1(w) + \cdots + c_8 \cdot m_8(w),$$

kde $m_0, m_1, \ldots, m_8$ jsou funkce udávající, kolikrát je daná instukce při výpočtu nad vstupem w provedena.

Časová složitost algoritmu

Instr.	doba	počet provedení	hodnota $m_i(w)$
$k := 0$	c_0	$m_0(w)$	1
$i := 1$	c_1	$m_1(w)$	1
$[i < n]$	c_2	$m_2(w)$	$n-1$
$[i \geq n]$	c_3	$m_3(w)$	1
$[A[i] \leq A[k]]$	c_4	$m_4(w)$	$n-1-\ell$
$[A[i] > A[k]]$	c_5	$m_5(w)$	ℓ
$k := i$	c_6	$m_6(w)$	ℓ
$i := i+1$	c_7	$m_7(w)$	$n-1$
$result := A[k]$	c_8	$m_8(w)$	1

ℓ — počet průchodů cyklem, kdy platí $A[i] > A[k]$ (zjevně je $0 \leq \ell < n$)

Časová složitost algoritmu

Dosazením do

$$t(w) \;=\; c_0 \cdot m_0(w) + c_1 \cdot m_1(w) + \;\cdots\; + c_8 \cdot m_8(w),$$

dostaneme

$$t(w) \;=\; d_1 \;+\; d_2 \cdot (n-1) \;+\; d_3 \cdot (n-1-\ell) \;+\; d_4 \cdot \ell,$$

kde

$$d_1 = c_0 + c_1 + c_3 + c_8 \qquad d_3 = c_4$$
$$d_2 = c_2 + c_7 \qquad d_4 = c_5 + c_6$$

Po úpravě je

$$t(w) \;=\; (d_2 + d_3) \cdot n \;+\; (d_4 - d_3) \cdot \ell \;+\; (d_1 - d_2 - d_3)$$

Poznámka: $t(w)$ není časová složitost, ale doba výpočtu pro konkrétní vstup w

Časová složitost algoritmu

Například pokud budou doby provedení jednotlivých instrukcí následující:

Instr.	označení	doba
$k := 0$	c_0	4
$i := 1$	c_1	4
$[i < n]$	c_2	10
$[i \geq n]$	c_3	12
$[A[i] \leq A[k]]$	c_4	14
$[A[i] > A[k]]$	c_5	12
$k := i$	c_6	5
$i := i + 1$	c_7	6
$result := A[k]$	c_8	5

bude $d_1 = 25$, $d_2 = 16$, $d_3 = 14$ a $d_4 = 17$.

V takovém případě je $t(w) = 30n + 3\ell - 5$.

Pro konkrétní vstup $w = ([3, 8, 4, 5, 2], 5)$ je $n = 5$ a $\ell = 1$, takže $t(w) = 30 \cdot 5 + 3 \cdot 1 - 5 = 148$.

Časová složitost algoritmu

Pro které vstupy velikosti n bude výpočet trvat nejdéle (tj. které vstupy představují nejhorší případ), může záviset na detailech implementace a přesných hodnotách konstant:

Doba výpočtu algoritmu FIND-MAX pro vstup $w = (A, n)$ velikosti n:

$$t(w) \;=\; (d_2 + d_3) \cdot n \;+\; (d_4 - d_3) \cdot \ell \;+\; (d_1 - d_2 - d_3)$$

- Pokud $d_3 \geq d_4$ — nejhorší jsou případy, kdy má ℓ co nejmenší hodnotu $\ell = 0$ — například vstupy tvaru $[0, 0, \ldots, 0]$ nebo třeba $[n, n-1, n-2, \ldots, 2, 1]$
- Pokud $d_3 \leq d_4$ — nejhorší jsou případy, kdy má ℓ co největší hodnotu $\ell = n - 1$ — například vstupy tvaru $[0, 1, \ldots, n-1]$

Časová složitost algoritmu

Časová složitost $T(n)$ algoritmu FIND-MAX v nejhorším případě je tedy dána následovně:

- Pokud $d_3 \geq d_4$:

$$T(n) = (d_2 + d_3) \cdot n + (d_1 - d_2 - d_3)$$

- Pokud $d_3 \leq d_4$:

$$\begin{aligned} T(n) &= (d_2 + d_3) \cdot n + (d_4 - d_3) \cdot (n-1) + (d_1 - d_2 - d_3) \\ &= (d_2 + d_4) \cdot n + (d_1 - d_2 - d_4) \end{aligned}$$

Příklad: Pro $d_1 = 25$, $d_2 = 16$, $d_3 = 14$ a $d_4 = 17$ bude

$$\begin{aligned} T(n) &= (16 + 17) \cdot n + (25 - 16 - 17) \\ &= 33n - 8 \end{aligned}$$

Časová složitost algoritmu

V obou případech (ať už $d_3 \geq d_4$ nebo $d_3 \leq d_4$) bude časová složitost algoritmu FIND-MAX funkce tvaru

$$T(n) = an + b$$

kde a a b jsou nějaké konstanty, jejichž přesné hodnoty závisí na délce trvání jednotlivých instrukcí.

Poznámka: Konkrétně bychom tyto konstanty mohli vyjádřit jako

$$a = d_2 + \max\{d_3, d_4\} \qquad b = d_1 - d_2 - \max\{d_3, d_4\}$$

Například

$$T(n) = 33n - 8$$

Časová složitost algoritmu

Pokud bychom se spokojili s tím, že časová složitost algoritmu FIND-MAX je nějaká funkce tvaru

$$T(n) = an + b,$$

kde by nás ale nezajímaly konkrétní hodnoty konstant a a b, celá analýza mohla být výrazně jednodušší.

- Ve skutečnosti ani většinou nechceme vědět, jak přesně funkce $T(n)$ vypadá (obecně to může být nějaká velmi komplikovaná funkce), a stačilo by nám, že víme, že hodnoty funkce $T(n)$ „zhruba“ odpovídají hodnotám nějaké funkce $S(n) = an + b$, kde a a b jsou nějaké konstanty.

Časová složitost algoritmu

U dané funkce $T(n)$ vyjadřující časovou nebo paměťovou složitost se tak většinou spokojíme s jejím přibližným vyjádřením — **odhadem**, kde

- zanedbáme méně významné členy

  (např. ve funkci $T(n) = 15n^2 + 40n - 5$ zanedbáme členy $40n$ a -5 a místo původní funkce budeme uvažovat jen o funkci $T(n) = 15n^2$),

- zanedbáme konstanty, kterými se násobí

  (např. místo funkce $T(n) = 15n^2$ budeme uvažovat o funkci $T(n) = n^2$)

- konstanty v exponentech ignorovat nebudeme — například je podstatný rozdíl mezi funkcemi $T_1(n) = n^2$ a $T_2(n) = n^3$.

- bude nás zajímat, jak se funkce $T(n)$ chová pro „velké“ hodnoty n, chování na malých hodnotách budeme ignorovat

Rychlost růstu funkcí

Program zpracovává vstup velikosti n.
Předpokládejme, že pro vstup velikosti n provede $T(n)$ operací, a že provedení jedné operace trvá $1\,\mu\mathrm{s}$ ($10^{-6}\,\mathrm{s}$).

	n							
$T(n)$	20	40	60	80	100	200	500	1000
n	$20\,\mu$s	$40\,\mu$s	$60\,\mu$s	$80\,\mu$s	0.1 ms	0.2 ms	0.5 ms	1 ms
$n \log n$	$86\,\mu$s	0.213 ms	0.354 ms	0.506 ms	0.664 ms	1.528 ms	4.48 ms	9.96 ms
n^2	0.4 ms	1.6 ms	3.6 ms	6.4 ms	10 ms	40 ms	0.25 s	1 s
n^3	8 ms	64 ms	0.216 s	0.512 s	1 s	8 s	125 s	16.7 min.
n^4	0.16 s	2.56 s	12.96 s	42 s	100 s	26.6 min.	17.36 hod.	11.57 dní
2^n	1.05 s	12.75 dní	36560 let	$38.3 \cdot 10^9$ let	$40.1 \cdot 10^{15}$ let	$50 \cdot 10^{45}$ let	$10.4 \cdot 10^{136}$ let	–
$n!$	77147 let	$2.59 \cdot 10^{34}$ let	$2.64 \cdot 10^{68}$ let	$2.27 \cdot 10^{105}$ let	$2.96 \cdot 10^{144}$ let	–	–	–

Rychlost růstu funkcí

Uvažujme 3 algoritmy se složitostmi $T_1(n) = n$, $T_2(n) = n^3$, $T_3(n) = 2^n$. Náš počítač zvládne v reálném čase (kolik jsme ochotni počkat) 10^{12} kroků.

Složitost	Velikost vstupu
$T_1(n) = n$	10^{12}
$T_2(n) = n^3$	10^4
$T_3(n) = 2^n$	40

Rychlost růstu funkcí

Uvažujme 3 algoritmy se složitostmi $T_1(n) = n$, $T_2(n) = n^3$, $T_3(n) = 2^n$. Náš počítač zvládne v reálném čase (kolik jsme ochotni počkat) 10^{12} kroků.

Složitost	Velikost vstupu
$T_1(n) = n$	10^{12}
$T_2(n) = n^3$	10^4
$T_3(n) = 2^n$	40

Nyní počítač 1000 násobně zrychlíme. Zvládne tedy 10^{15} kroků.

Složitost	Velikost vstupu	Nárust
$T_1(n) = n$	10^{15}	1000×
$T_2(n) = n^3$	10^5	10×
$T_3(n) = 2^n$	50	+10

Asymptotická notace

V následujícím se zaměříme na funkce typu $f : \mathbb{N} \to \mathbb{R}$, kde:

- Hodnota $f(n)$ nemusí být definovaná pro všechny hodnoty $n \in \mathbb{N}$, ale musí existovat nějaká konstanta n_0 taková, že hodnota $f(n)$ je definovaná pro všechna $n \in \mathbb{N}$ taková, že $n \geq n_0$.

  Příklad: Funkce $f(n) = \log_2(n)$ není definovaná pro $n = 0$, ale pro všechna $n \geq 1$ už definovaná je.

- Musí existovat taková konstanta n_0, že pro všechny hodnoty $n \in \mathbb{N}$, kde $n \geq n_0$, platí $f(n) \geq 0$.

  Příklad: Pro funkci $f(n) = n^2 - 25$ platí $f(n) \geq 0$ pro všechna $n \geq 5$.

Asymptotická notace

Vezměme si libovolnou funkci $g : \mathbb{N} \to \mathbb{R}$. Zápisy $O(g)$, $\Omega(g)$, $\Theta(g)$, $o(g)$ a $\omega(g)$ označují **množiny funkcí** typu $\mathbb{N} \to \mathbb{R}$, kde:

- $O(g)$ – množina všech funkcí, které rostou nejvýše tak rychle jako g
- $\Omega(g)$ – množina všech funkcí, které rostou alespoň tak rychle jako g
- $\Theta(g)$ – množina všech funkcí, které rostou stejně rychle jako g
- $o(g)$ – množina všech funkcí, které rostou pomaleji než funkce g
- $\omega(g)$ – množina všech funkcí, které rostou rychleji než funkce g

Poznámka: Toto nejsou definice! Ty následují na následujících slidech.

- O – velké „O“
- Ω – velké řecké písmeno „omega“
- Θ – velké řecké písmeno „theta“
- o – malé „o“
- ω – malé „omega“

Asymptotická notace – symbol O

Neformálně:

$O(g)$ – množina všech funkcí, které rostou nejvýše tak rychle jako g

Jak formálně definovat, kdy platí $f \in O(g)$?

První pokus:

- porovnat hodnoty funkcí

$$(\forall n \in \mathbb{N})(f(n) \leq g(n))$$

Problém: Neumožňuje zanedbat konstanty, např. není pravda, že $(\forall n \in \mathbb{N})(3n^2 \leq 2n^2)$.

Asymptotická notace – symbol O

Neformálně:

$O(g)$ – množina všech funkcí, které rostou nejvýše tak rychle jako g

Jak formálně definovat, kdy platí $f \in O(g)$?

Druhý pokus:

- přenásobit funkci g nějakou dostatečně velkou konstantou c

$$(\exists c > 0)(\forall n \in \mathbb{N})(f(n) \leq c \cdot g(n))$$

Problém: Nerovnost nemusí ani po přenásobení libovolně velkou konstantou platit pro malé hodnoty n.

Například funkce $g(n) = n^2$ očividně roste rychleji než funkce $f(n) = n + 5$. Ovšem bez ohledu na to, jak velkou zvolíme konstantu c, pro $n = 0$ nikdy nebude platit $n + 5 \leq c \cdot n^2$.

Asymptotická notace – symbol O

Neformálně:

$O(g)$ – množina všech funkcí, které rostou nejvýše tak rychle jako g

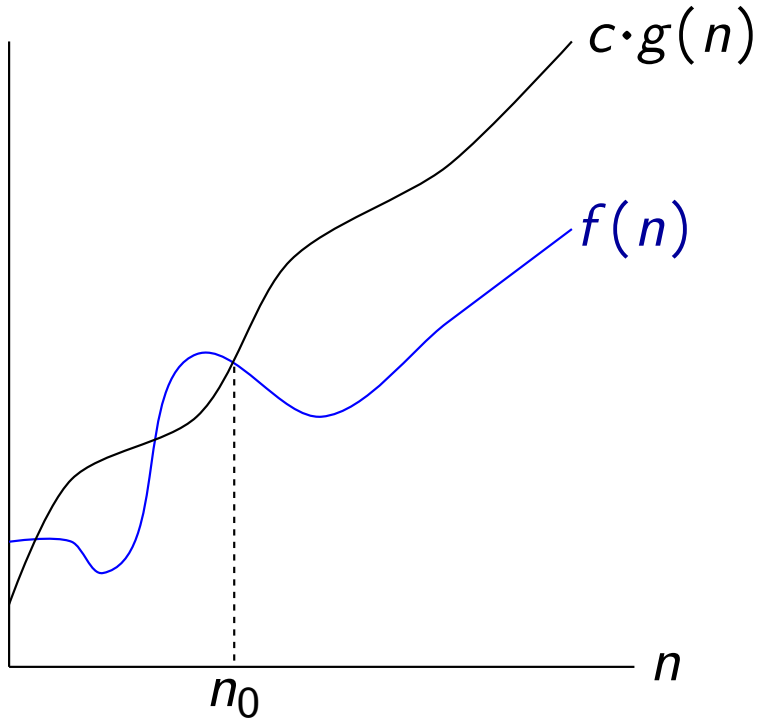
Jak formálně definovat, kdy platí $f \in O(g)$?

Třetí pokus:

- nerovnost nemusí platit pro všechna n, stačí, že bude platit pro všechny „dostatečně velké“ hodnoty n

$$(\exists c > 0)(\exists n_0 \geq 0)(\forall n \geq n_0)(f(n) \leq c \cdot g(n))$$

Asymptotická notace – symbol O



Definice

Vezměme si libovolnou funkci $g : \mathbb{N} \to \mathbb{R}$. Pro funkci $f : \mathbb{N} \to \mathbb{R}$ platí $f \in O(g)$ právě tehdy, když

$$(\exists c > 0)(\exists n_0 \geq 0)(\forall n \geq n_0)\big(f(n) \leq c \cdot g(n)\big).$$

Asymptotická notace – symbol O

Poznámky:

- c je kladné reálné číslo (tj. $c \in \mathbb{R}$ a $c > 0$)
- n_0 a n jsou přirozená čísla (tj. $n_0 \in \mathbb{N}$ a $n \in \mathbb{N}$)

Asymptotická notace – symbol O

Příklad: Vezměme si funkce $f(n) = 2n^2 + 3n + 7$ a $g(n) = n^2$.

Chceme ukázat $f \in O(g)$, tj. $f \in O(n^2)$:

- **Postup 1:**

  Zvolme například $c = 3$.

  $$c \cdot g(n) = 3n^2 = 2n^2 + \frac{1}{2}n^2 + \frac{1}{2}n^2$$

  Potřebujeme najít takové n_0, aby pro každé $n \geq n_0$ platilo současně

  $$2n^2 \geq 2n^2 \qquad \frac{1}{2}n^2 \geq 3n \qquad \frac{1}{2}n^2 \geq 7$$

  Snadno ověříme, že například $n_0 = 6$ vyhovuje těmto požadavkům.

  Pak pro každé $n \geq 6$ platí $c \cdot g(n) \geq f(n)$:

  $$cg(n) = 3n^2 = 2n^2 + \frac{1}{2}n^2 + \frac{1}{2}n^2 \geq 2n^2 + 3n + 7 = f(n)$$

Asymptotická notace – symbol O

Příklad, kde $f(n) = 2n^2 + 3n + 7$ a $g(n) = n^2$:

- **Postup 2:**

  Zvolme $c = 12$.

  $$c \cdot g(n) = 12n^2 = 2n^2 + 3n^2 + 7n^2$$

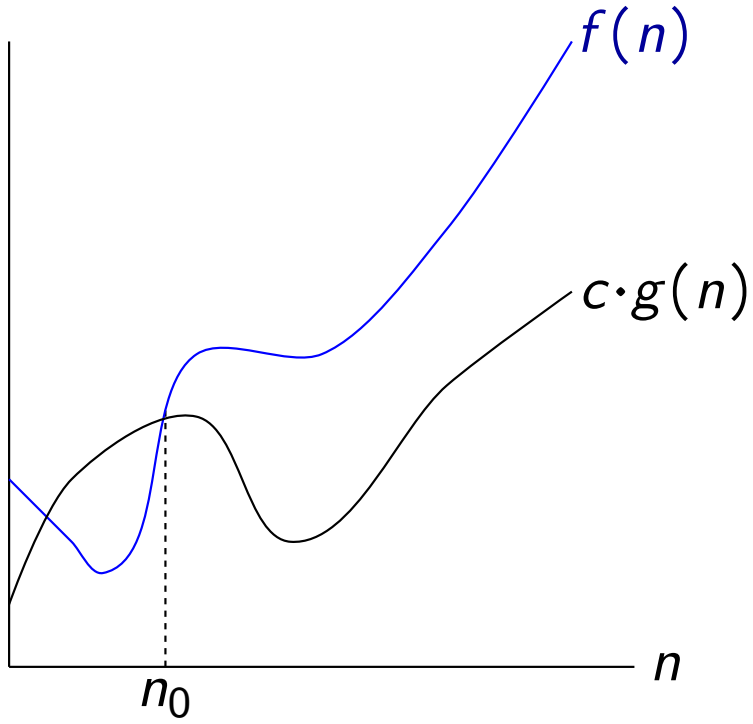
  Potřebujeme najít takové n_0, aby pro každé $n \geq n_0$ platilo současně

  $$2n^2 \geq 2n^2 \qquad 3n^2 \geq 3n \qquad 7n^2 \geq 7$$

  Uvedené vztahy zjevně platí pro $n_0 = 1$, takže pro každé $n \geq 1$ platí $f(n) \leq c \cdot g(n)$:

  $$c \cdot g(n) = 12n^2 = 2n^2 + 3n^2 + 7n^2 \geq 2n^2 + 3n + 7 = f(n)$$

Asymptotická notace – symbol Ω



Definice

Vezměme si libovolnou funkci $g : \mathbb{N} \to \mathbb{R}$. Pro funkci $f : \mathbb{N} \to \mathbb{R}$ platí $f \in \Omega(g)$ právě tehdy, když

$$(\exists c > 0)(\exists n_0 \geq 0)(\forall n \geq n_0)\big(c \cdot g(n) \leq f(n)\big).$$

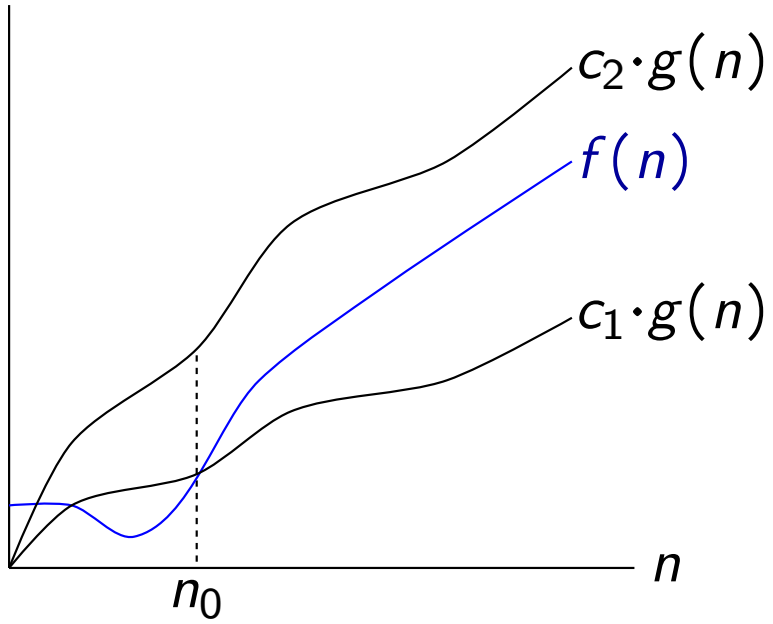
Asymptotická notace – symbol Ω

Není těžké zdůvodnit, že platí následující tvrzení:

Pro libovolné funkce f a g platí:

$$f \in O(g) \qquad \text{právě tehdy, když} \qquad g \in \Omega(f)$$

Asymptotická notace – symbol Θ



Definice

Vezměme si libovolnou funkci $g : \mathbb{N} \to \mathbb{R}$. Pro funkci $f : \mathbb{N} \to \mathbb{R}$ platí $f \in \Theta(g)$ právě tehdy, když

$$(\exists c_1 > 0)(\exists c_2 > 0)(\exists n_0 \geq 0)(\forall n \geq n_0)\big(c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\big).$$

Asymptotická notace – symbol Θ

Z definice Θ snadno vyplývá následující:

Pro libovolné funkce f a g platí:

$$f \in \Theta(g) \quad \text{právě tehdy, když} \quad f \in O(g) \text{ a } f \in \Omega(g)$$

$$f \in \Theta(g) \quad \text{právě tehdy, když} \quad f \in O(g) \text{ a } g \in O(f)$$

$$f \in \Theta(g) \quad \text{právě tehdy, když} \quad g \in \Theta(f)$$

Asymptotická notace – symboly o and ω

Definice

Vezměme si libovolnou funkci $g : \mathbb{N} \to \mathbb{R}$. Pro funkci $f : \mathbb{N} \to \mathbb{R}$ platí $f \in o(g)$ právě tehdy, když

$$\lim_{n \to +\infty} \frac{f(n)}{g(n)} = 0$$

Definice

Vezměme si libovolnou funkci $g : \mathbb{N} \to \mathbb{R}$. Pro funkci $f : \mathbb{N} \to \mathbb{R}$ platí $f \in \omega(g)$ právě tehdy, když

$$\lim_{n \to +\infty} \frac{f(n)}{g(n)} = +\infty$$

Asymptotická notace

Pro libovolné funkce f a g platí následující tvrzení:

Jestliže existuje hodnota $c \geq 0$ taková, že

$$\lim_{n \to +\infty} \frac{f(n)}{g(n)} = c$$

pak $f \in O(g)$.

Jestliže existuje hodnota $c \geq 0$ taková, že

$$\lim_{n \to +\infty} \frac{g(n)}{f(n)} = c$$

pak $f \in \Omega(g)$.

Asymptotická notace

Zjevně platí:

- Pokud $f \in o(g)$, pak $f \in O(g)$.
- Pokud $f \in \omega(g)$, pak $f \in \Omega(g)$.

Asymptotická notace

Na asymptotickou notaci se můžeme dívat jako na určitý druh porovnání **rychlosti růstu** funkcí:

$f \in O(g)$	—	rychlost růstu f "$\leq$" rychlost růstu g
$f \in \Omega(g)$	—	rychlost růstu f "$\geq$" rychlost růstu g
$f \in \Theta(g)$	—	rychlost růstu f "$=$" rychlost růstu g
$f \in o(g)$	—	rychlost růstu f "$<$" rychlost růstu g
$f \in \omega(g)$	—	rychlost růstu f "$>$" rychlost růstu g

Poznámka:

- Existují dvojice funkcí f a g takové, že

$$f \notin O(g) \quad \text{a} \quad g \notin O(f),$$

  například

$$f(n) = n^2 \qquad g(n) = \begin{cases} n & \text{pokud } n \mod 2 = 1 \\ n^3 & \text{jinak} \end{cases}$$

Asymptotická notace

- O funkci f řekneme, že je:

  lineární, pokud $f(n) \in \Theta(n)$
  kvadratická, pokud $f(n) \in \Theta(n^2)$
  kubická, pokud $f(n) \in \Theta(n^3)$
  polynomiální, pokud $f(n) \in O(n^k)$ pro nějaké $k > 0$
  exponenciální, pokud $f(n) \in O(c^{n^k})$ pro nějaké $c > 1$ a $k > 0$
  logaritmická, pokud $f(n) \in \Theta(\log n)$
  polylogaritmická, pokud $f(n) \in \Theta(\log^k n)$ pro nějaké $k > 0$

- $O(1)$ je množina všech **omezených** funkcí, tj. funkcí jejichž funkční hodnoty jsou shora omezeny nějakou konstantou.

- Exponenciální funkce se v asymptotické notaci často uvádí ve tvaru $2^{O(n^k)}$, protože potom již nemusíme uvažovat různé základy mocniny.

Asymptotická notace

Obecně platí:

- jakákoliv polylogaritmická funkce roste pomaleji než jakákoli polynomiální funkce
- jakákoli polynomiální funkce roste pomaleji než jakákoli exponenciální funkce
- při porovnávání polynomiálních funkcí n^k a n^ℓ stačí porovnat hodnoty k a ℓ
- při porovnávání polylogaritmických funkcí $\log^k n$ a $\log^\ell n$ stačí porovnat hodnoty k a ℓ
- při porovnávání exponenciálních funkcí $2^{p(n)}$ a $2^{q(n)}$ stačí porovnat polynomy $p(n)$ a $q(n)$.

Asymptotická notace

Tvrzení

Předpokládejme, že a a b jsou nějaké konstanty takové, že $a > 0$ a $b > 0$, a k a ℓ jsou nějaké libovolné konstanty, kde $k \geq 0$, $\ell \geq 0$ a $k \leq \ell$.

Uvažujme funkce

$$f(n) = a \cdot n^k \qquad\qquad g(n) = b \cdot n^\ell$$

Pro každé takové funkce f a g platí $f \in O(g)$:

Důkaz: Zvolme $c = \frac{a}{b}$.

Vzhledem k tomu, že pro $n \geq 1$ zjevně platí $n^k \leq n^\ell$ (protože $k \leq \ell$), tak pro $n \geq 1$ platí

$$c \cdot g(n) = \frac{a}{b} \cdot g(n) = \frac{a}{b} \cdot b \cdot n^\ell = a \cdot n^\ell \geq a \cdot n^k = f(n)$$

Asymptotická notace

Tvrzení

Pro libovolná $a, b > 1$ a libovolné $n > 0$ platí

$$\log_a n = \frac{\log_b n}{\log_b a}$$

Důkaz: Z $n = a^{\log_a n}$ plyne $\log_b n = \log_b(a^{\log_a n})$.
Protože $\log_b(a^{\log_a n}) = \log_a n \cdot \log_b a$, dostáváme $\log_b n = \log_a n \cdot \log_b a$,
z čehož plyne výše uvedený závěr. □

Z toho důvodu se při použití asymptotické notace základ logaritmu obvykle vynechává: například místo $\Theta(n \log_2 n)$ můžeme napsat $\Theta(n \log n)$.

Asymptotická notace

Příklady:

$n \in O(n^2)$

$1000n \in O(n)$

$2^{\log_2 n} \in \Theta(n)$

$n^3 \notin O(n^2)$

$n^2 \notin O(n)$

$n^3 + 2^n \notin O(n^2)$

$n^3 \in O(n^4)$

$0.00001n^2 - 10^{10}n \in \Theta(10^{10}n^2)$

$n^3 - n^2 \log_2^3 n + 1000n - 10^{100} \in \Theta(n^3)$

$n^3 + 1000n - 10^{100} \in O(n^3)$

$n^3 + n^2 \notin \Theta(n^2)$

$n! \notin O(2^n)$

Asymptotická notace

Pro libovolné tři funkce f, g a h platí:

- jestliže $f \in O(g)$ a $g \in O(h)$, pak $f \in O(h)$
- jestliže $f \in \Omega(g)$ a $g \in \Omega(h)$, pak $f \in \Omega(h)$
- jestliže $f \in \Theta(g)$ a $g \in \Theta(h)$, pak $f \in \Theta(h)$

Asymptotická notace

- Pro libovolnou funkci f a libovolnou konstantu $c > 0$ platí:
  - $c \cdot f \in \Theta(f)$
- Pro libovolné dvě funkce f, g platí:
  - $\max(f, g) \in \Theta(f + g)$
  - pokud $f \in O(g)$, pak $f + g \in \Theta(g)$
- Pro libovolné čtyři funkce f_1, f_2, g_1, g_2 platí:
  - pokud $f_1 \in O(f_2)$ a $g_1 \in O(g_2)$, pak $f_1 + g_1 \in O(f_2 + g_2)$ a $f_1 \cdot g_1 \in O(f_2 \cdot g_2)$
  - pokud $f_1 \in \Theta(f_2)$ a $g_1 \in \Theta(g_2)$, pak $f_1 + g_1 \in \Theta(f_2 + g_2)$ a $f_1 \cdot g_1 \in \Theta(f_2 \cdot g_2)$

Asymptotická notace

Jak bylo uvedeno, výrazy $O(g)$, $\Omega(g)$, $\Theta(g)$, $o(g)$ a $\omega(g)$ označují určité množiny funkcí.

V odborných textech se však někdy používají tyto výrazy i v poněkud odlišném významu:

- zápis $O(g)$, $\Omega(g)$, $\Theta(g)$, $o(g)$ nebo $\omega(g)$ nereprezentuje danou množinu funkcí, ale **nějakou** funkci z dané množiny.

Tato konvence se používá zejména v zápisu rovnic nebo nerovnic.

Příklad: $3n^3 + 5n^2 - 11n + 2 = 3n^3 + O(n^2)$

Při použití této konvence je tedy možné například psát $f = O(g)$ místo $f \in O(g)$.

Složitost algoritmů

Řekněme, že bychom chtěli analyzovat časovou složitost $T(n)$ nějakého algoritmu, který se skládá z instrukcí $I_1, I_2, \ldots, I_k$:

- Předpokládejme, že doby provedení jednotlivých instrukcí jsou $c_1, c_2, \ldots, c_k$, tj. doba provedení instrukce I_i je dána konstantou c_i.
- Předpokládejme, že In je množina všech možných vstupů pro daný algoritmus.

  Zaveďme si pro každou instrukci I_i odpovídající funkci

  $$m_i : In \to \mathbb{N}$$

  udávající, kolikrát se provede instrukce I_i při výpočtu nad daným vstupem, tj. hodnota $m_i(w)$ udává, kolikrát se provede instrukce I_i při výpočtu nad vstupem w.

Složitost algoritmů

- Celková doba výpočtu nad vstupem w:
$$t(w) = c_1 \cdot m_1(w) + c_2 \cdot m_2(w) + \cdots + c_k \cdot m_k(w).$$
- Připomeňme, že $T(n) = \max\{\, t(w) \mid size(w) = n \,\}$.
- Pro každou z funkcí $m_1, m_2, \ldots, m_k$ můžeme nadefinovat odpovídající funkci $f_i : \mathbb{N} \to \mathbb{R}$, kde
$$f_i(n) = \max\{\, m_i(w) \mid size(w) = n \,\}$$
tj. $f_i(n)$ je maximum z počtu provedení instrukce I_i pro všechny vstupy velikosti n.
- Zjevně platí $T \in O(f_1 + f_2 + \cdots + f_k)$.
- Připomeňme si, že pokud $f_j \in O(f_i)$, pak $c_i \cdot f_i + c_j \cdot f_j \in O(f_i)$.
- Pokud tedy pro některou funkci f_i platí, že pro všechny f_j, kde $j \neq i$, je $f_j \in O(f_i)$, pak
$$T \in O(f_i).$$

Složitost algoritmů

- Zjevně také platí, že pro libovolnou z funkcí $f_1, f_2, \ldots, f_k$ je $T \in \Omega(f_i)$.
- Při analýze celkové časové složitosti $T(n)$ se tedy většinou můžeme omezit pouze na analýzu počtu provedení nejčastěji prováděné instrukce I_i, tj. zkoumání toho, jak rychle roste funkce $f_i(n)$, protože platí

$$T \in \Theta(f_i).$$

- Pro ostatní instrukce I_j stačí ověřit, že

$$f_j \in O(f_i),$$

tj. není pro ně nutné přesně zjišťovat, jak rychle rostou, ale jen to, že rostou nanejvýš tak rychle jako f_i.

Příklad:

Algoritmus: Nalezení největšího prvku v poli

```
FIND-MAX (A, n):
    k := 0
    for i := 1 to n − 1 do
        if A[i] > A[k] then
            k := i
    return A[k]
```

Složitost algoritmů

Při analýze složitosti algoritmu FIND-MAX jsme zjistili, že časová složitost daného algoritmu v nejhorším případě je

$$f(n) = an + b\,.$$

Kdybychom to nechtěli takto podrobně zjišťovat a spokojili se s hrubším odhadem, mohli jsme určit, že časová složitost tohoto algoritmu je $\Theta(n)$, protože:

- Algoritmus obsahuje jediný cyklus, který se pro vstup velikosti n provede vždy právě $(n-1)$ krát, tj. počet průchodů cyklem je v $\Theta(n)$.
- V rámci jednoho průchodu cyklem se provede několik instrukcí, jejichž počet je shora i zdola omezen nějakými konstantami nezávislými na velikosti vstupu. Doba provedení jedné iterace cyklu je tedy v $\Theta(1)$.
- Ostatní instrukce se provedou jednou. Čas, který se stráví jejich prováděním, je v $\Theta(1)$.

Složitost algoritmů

Pokusme se analyzovat časovou složitost následujícího algoritmu:

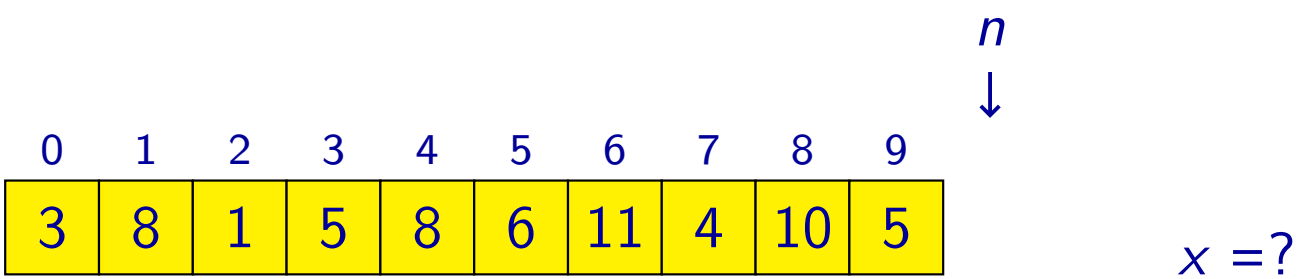
Algoritmus: Třídění přímým vkládáním

```
INSERTION-SORT (A, n):
    for j := 1 to n − 1 do
        x := A[j]
        i := j − 1
        while i ≥ 0 and A[i] > x do
            A[i + 1] := A[i]
            i := i − 1
        A[i + 1] := x
```

Tj. chceme najít funkci $T(n)$ takovou, že časová složitost algoritmu INSERTION-SORT v nejhorším případě je v $\Theta(T(n))$.

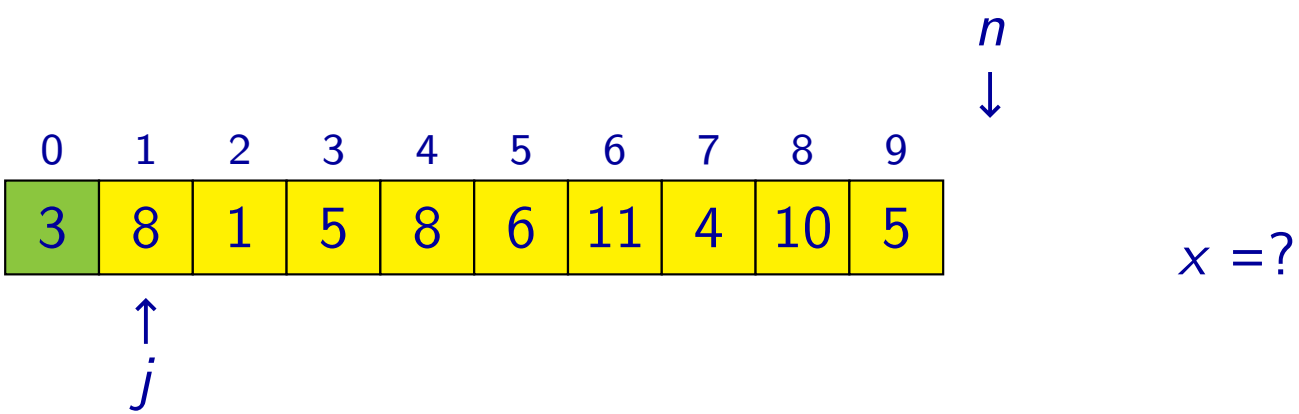
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



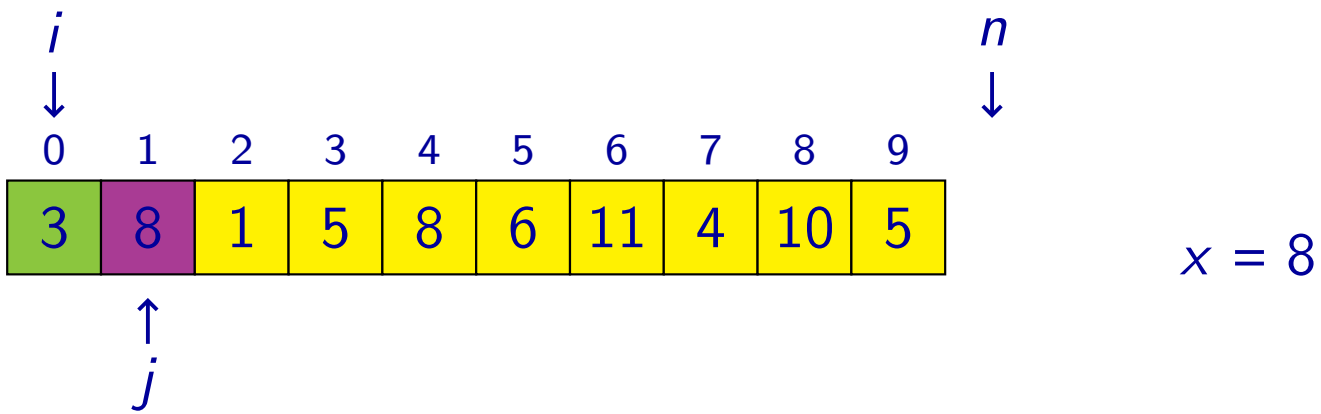
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



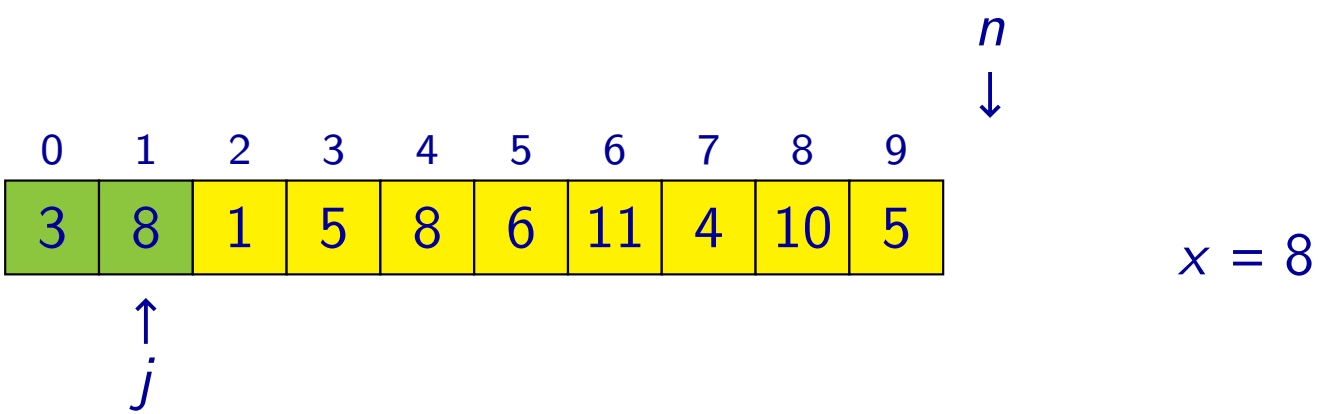
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



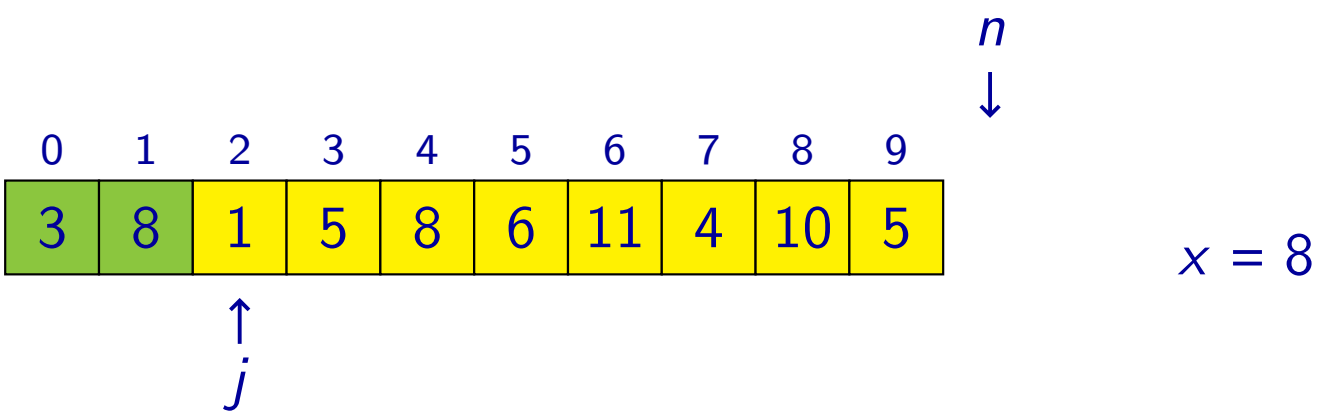
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



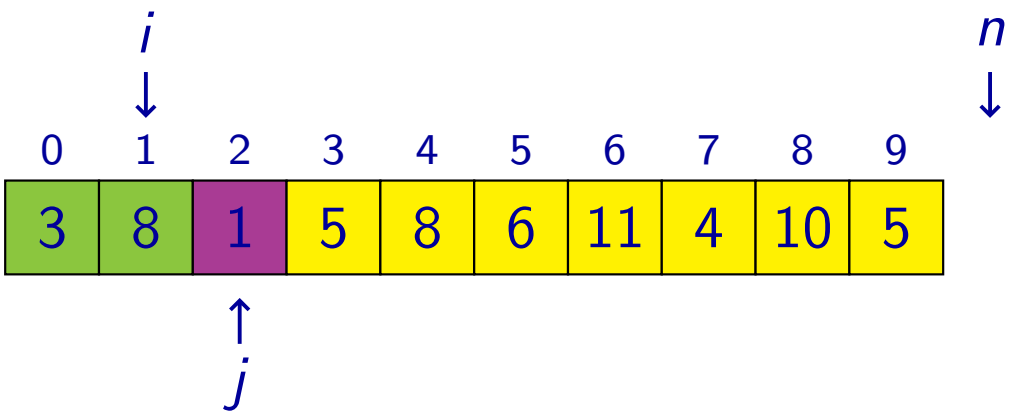
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

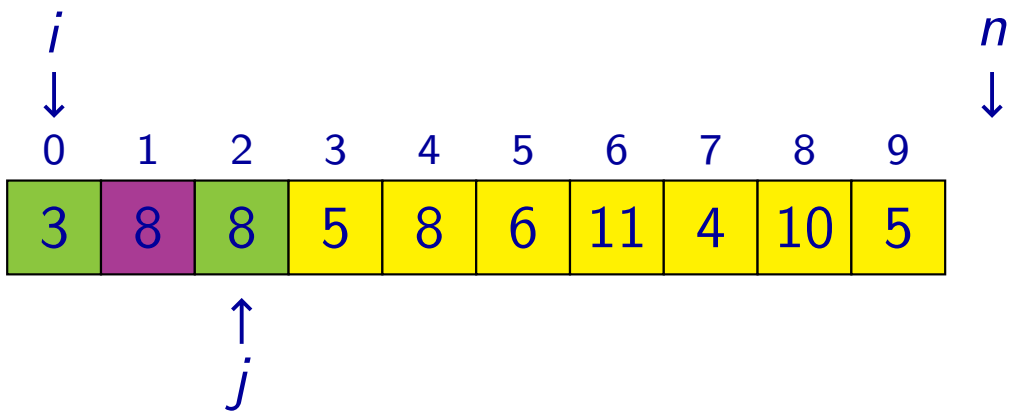
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



$x = 1$

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

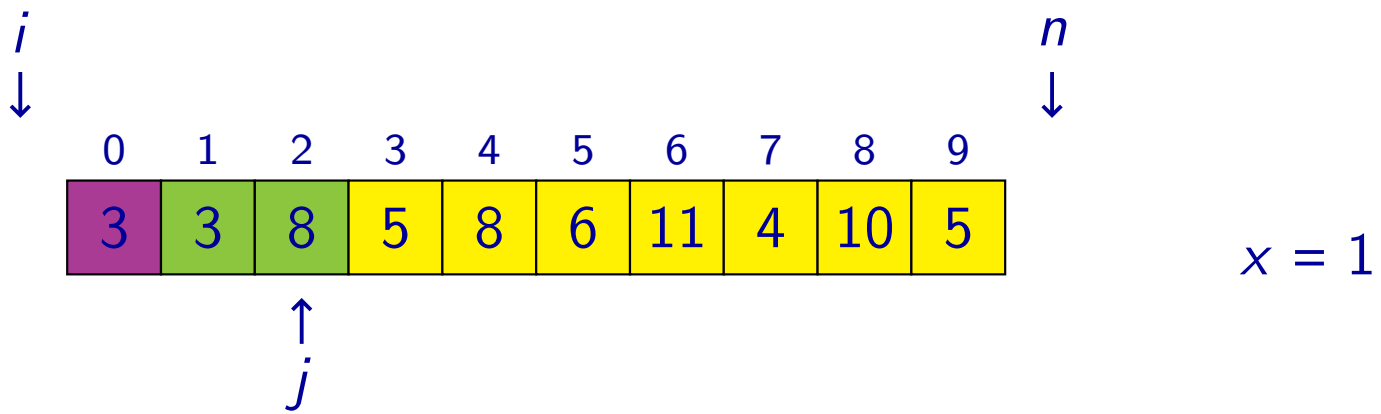
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



$x = 1$

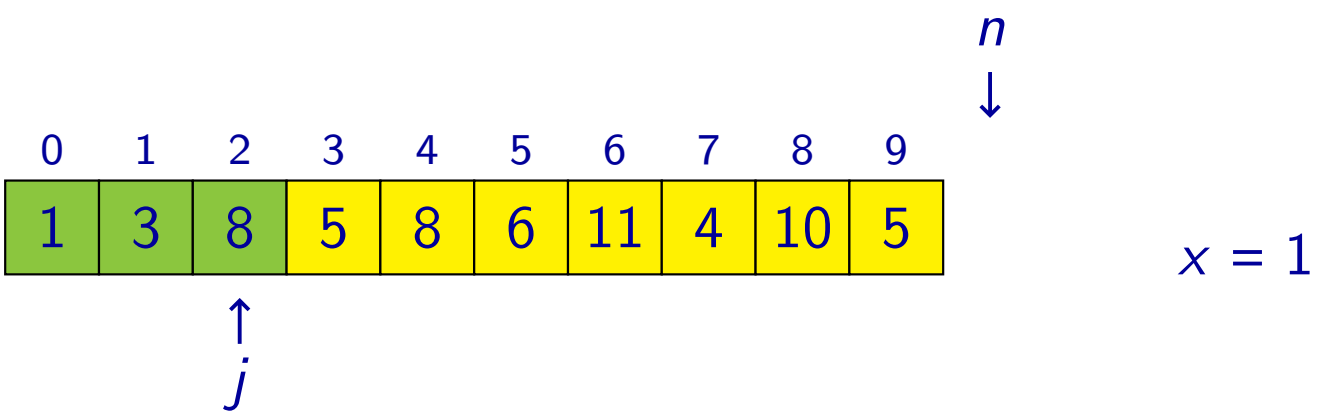
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



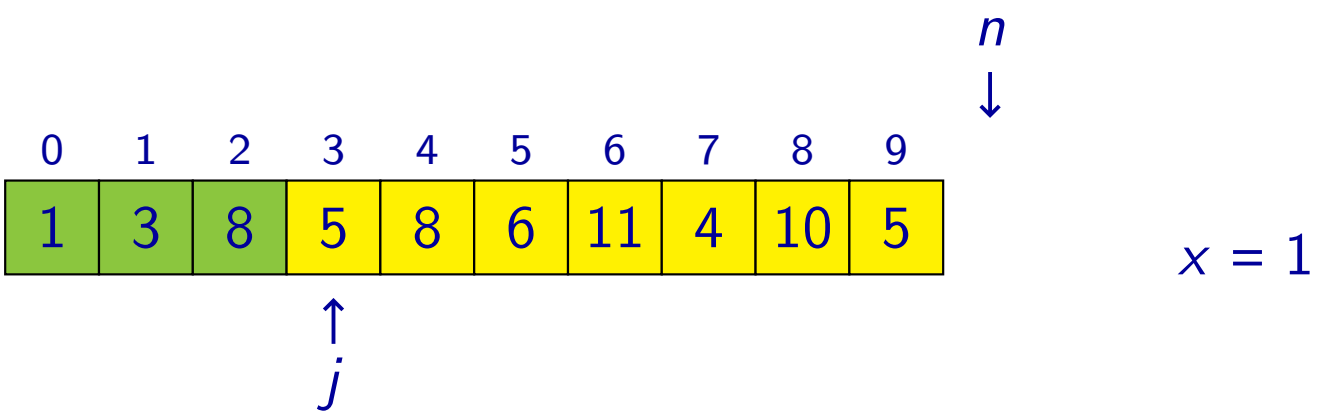
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



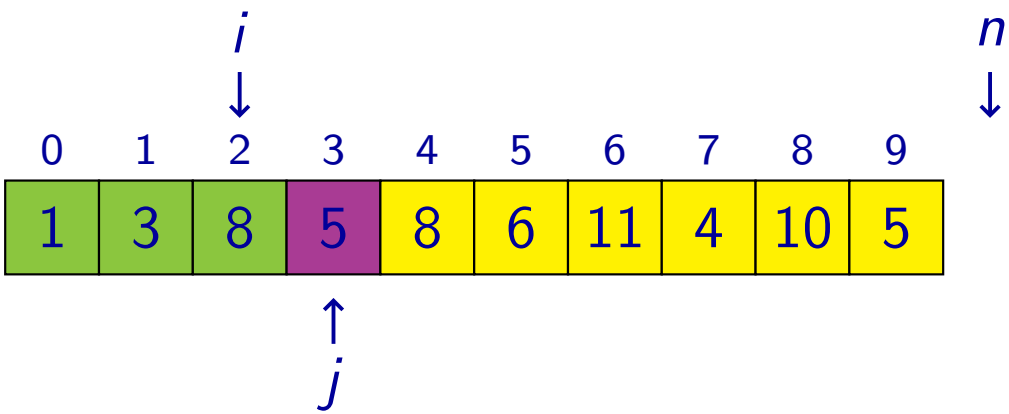
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

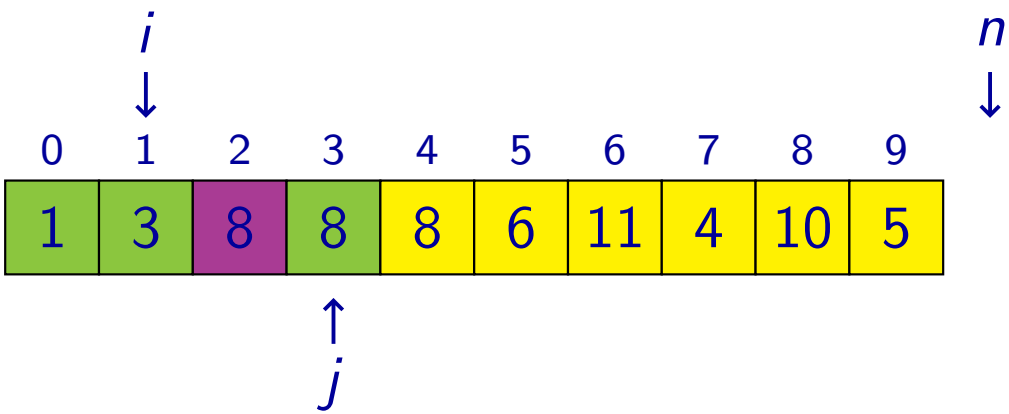
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



$x = 5$

Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

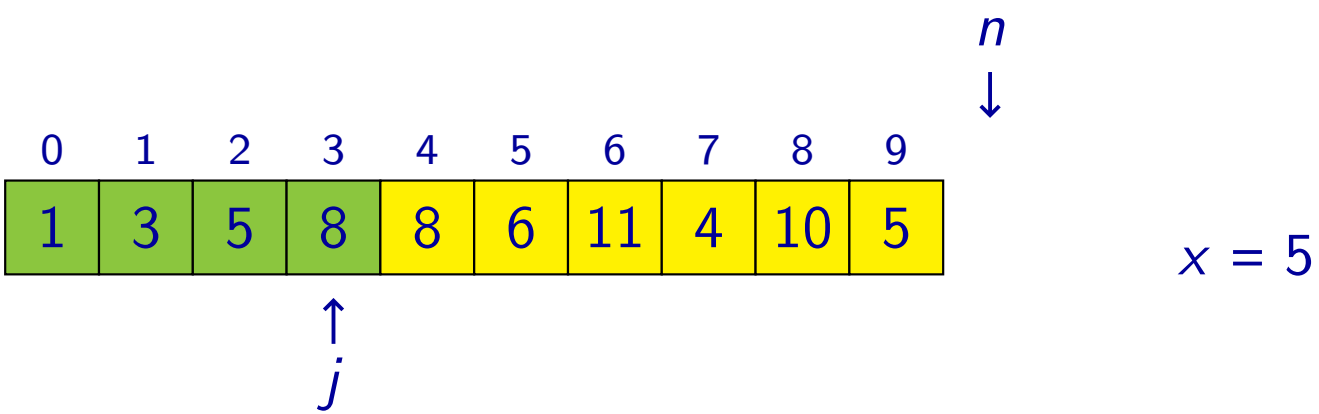
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,], \; n = 10.$$



$x = 5$

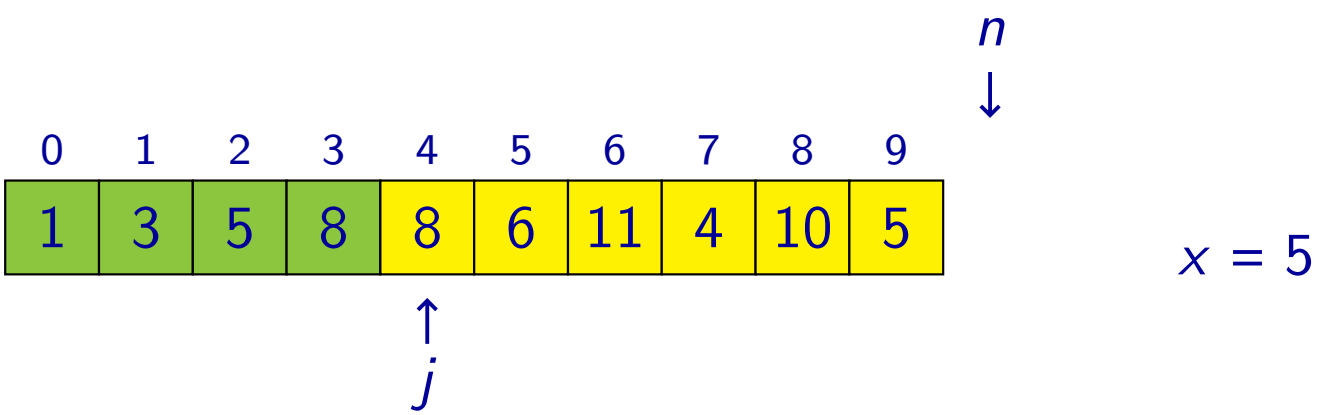
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,], \; n = 10.$$



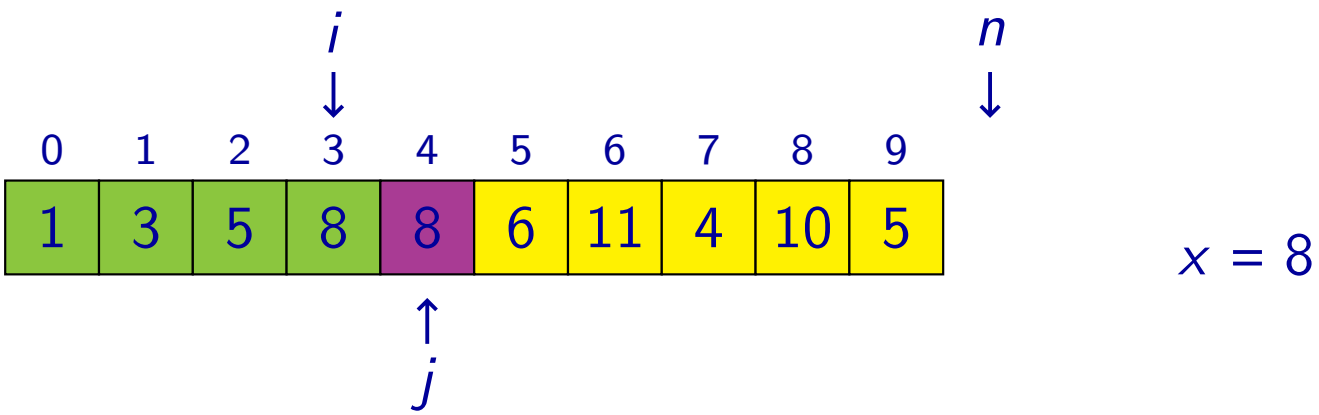
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



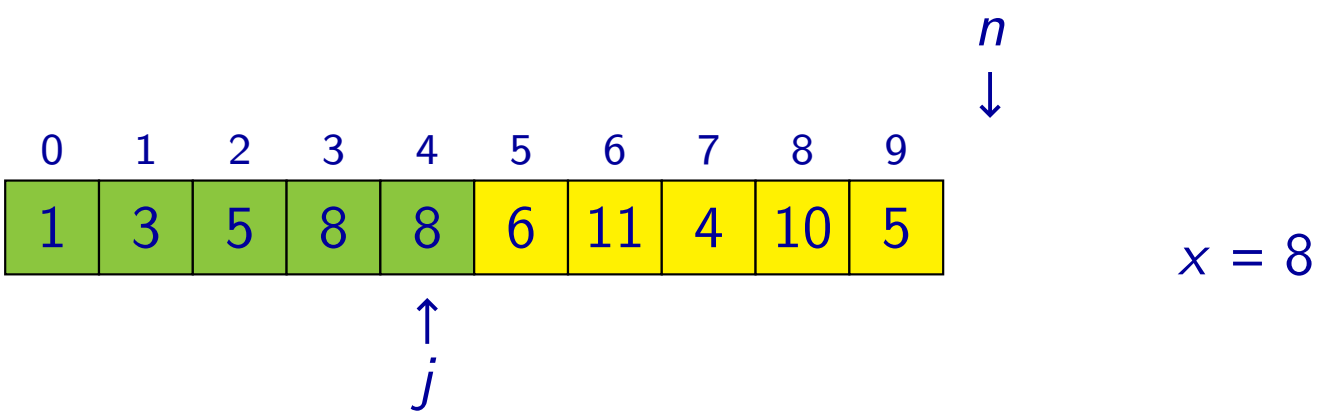
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



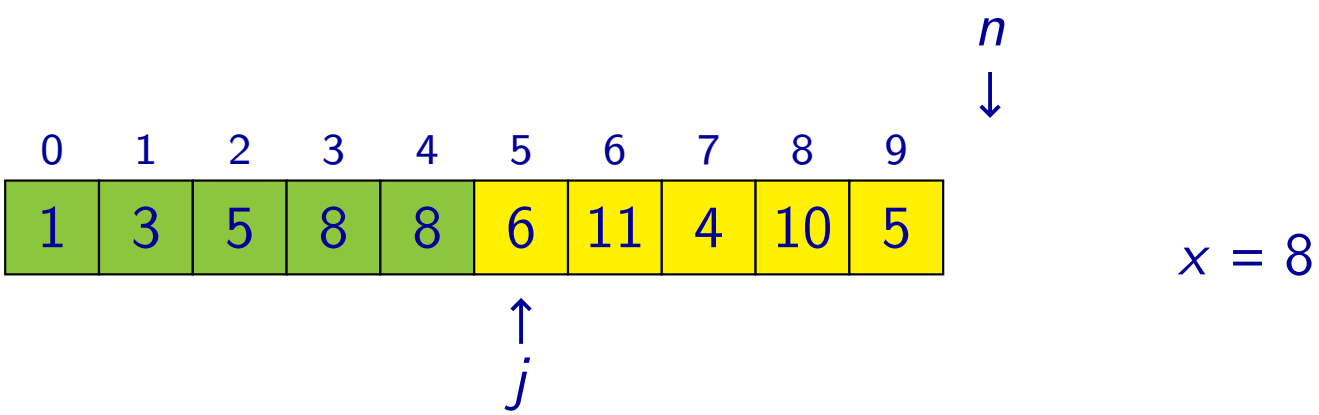
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



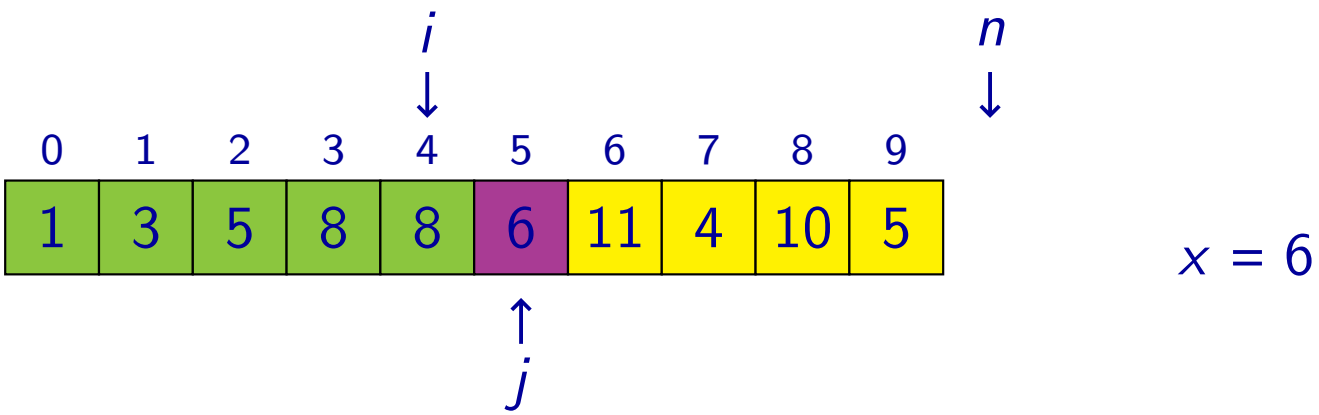
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



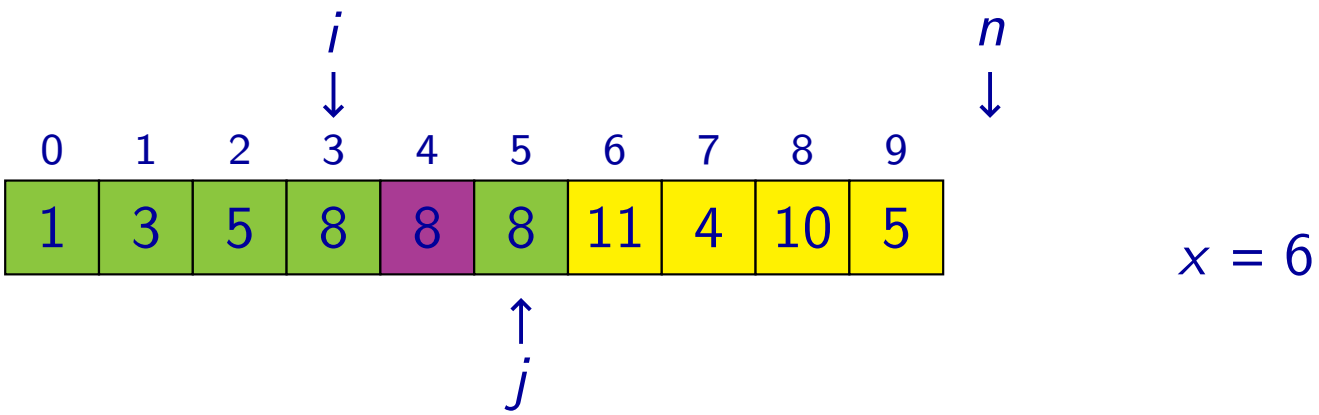
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



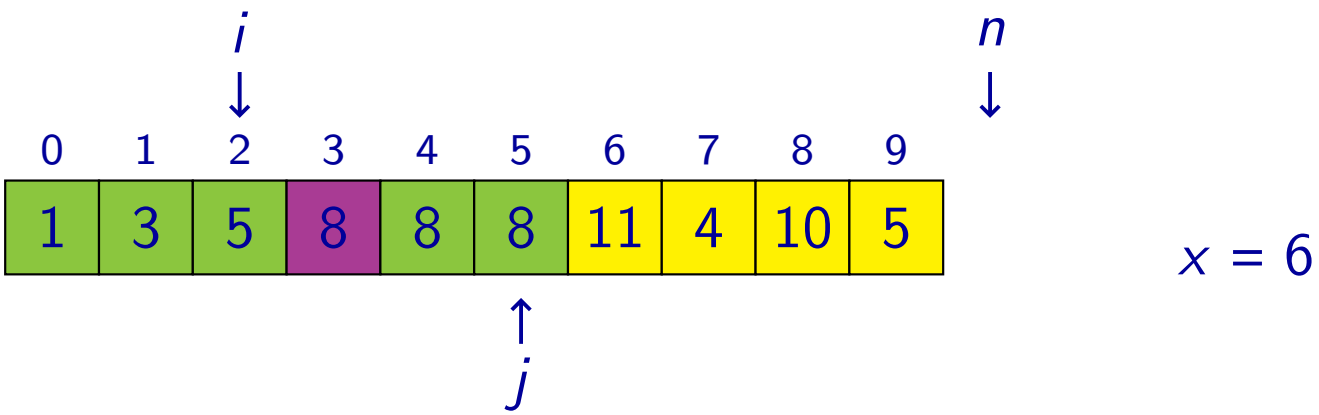
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



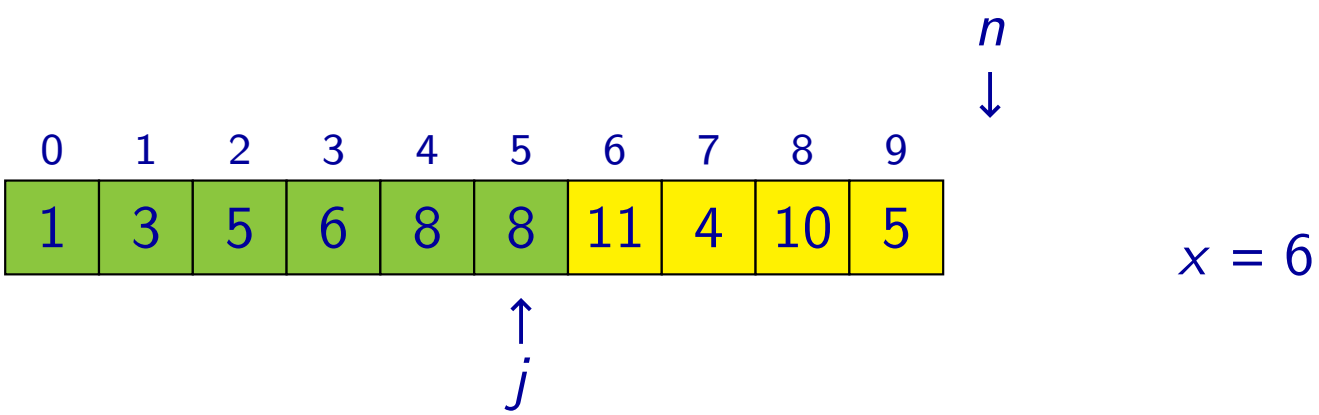
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



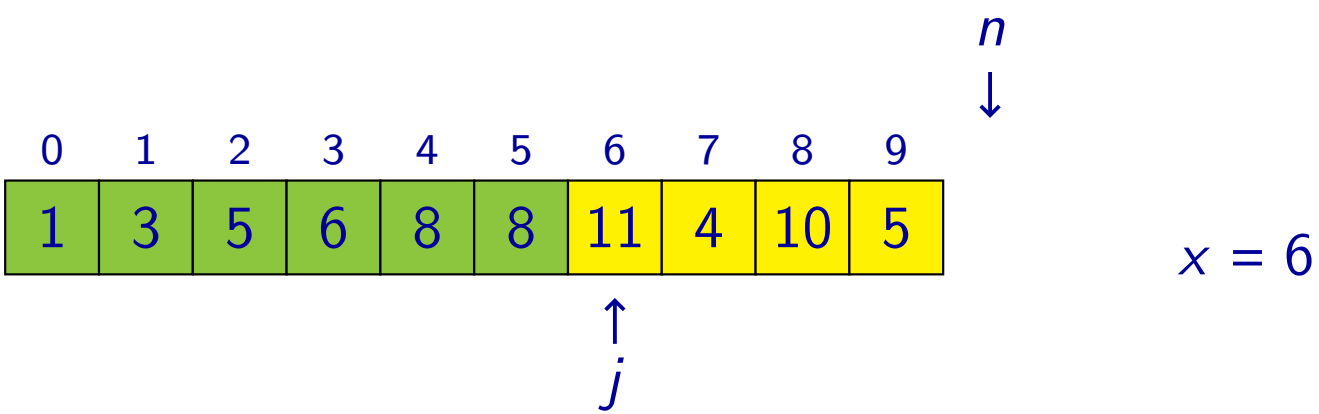
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



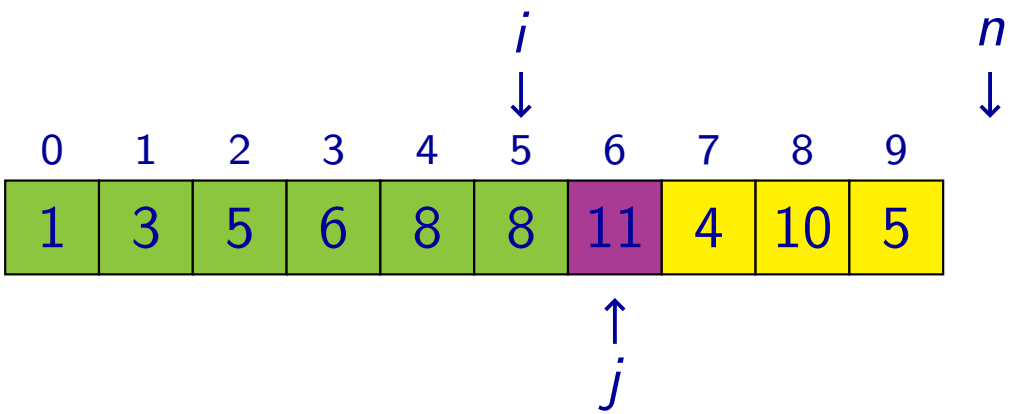
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

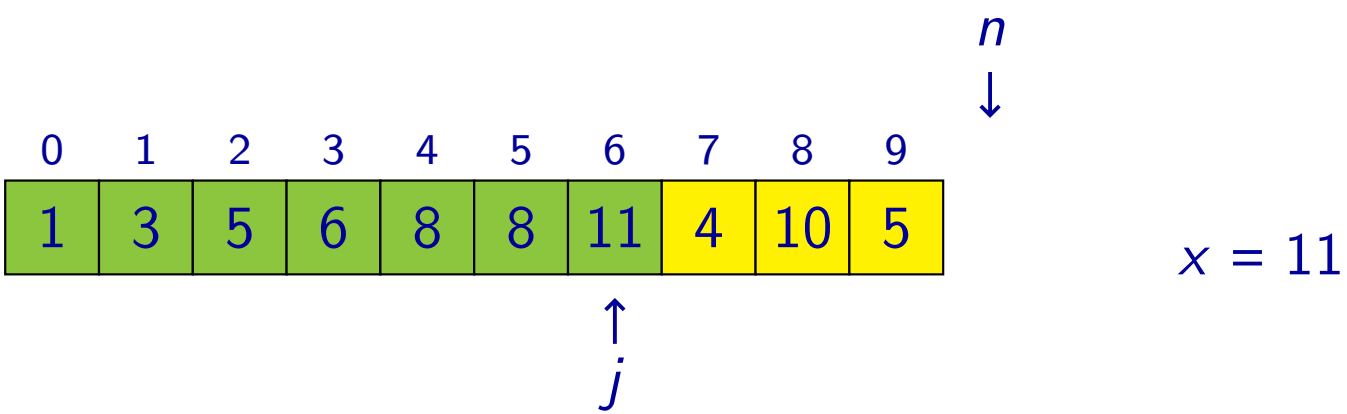
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



$x = 11$

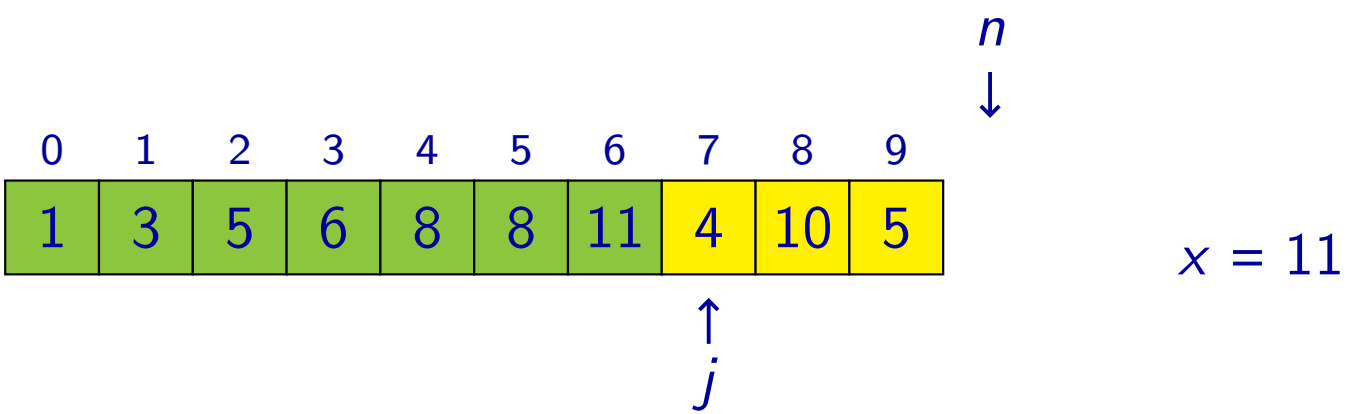
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



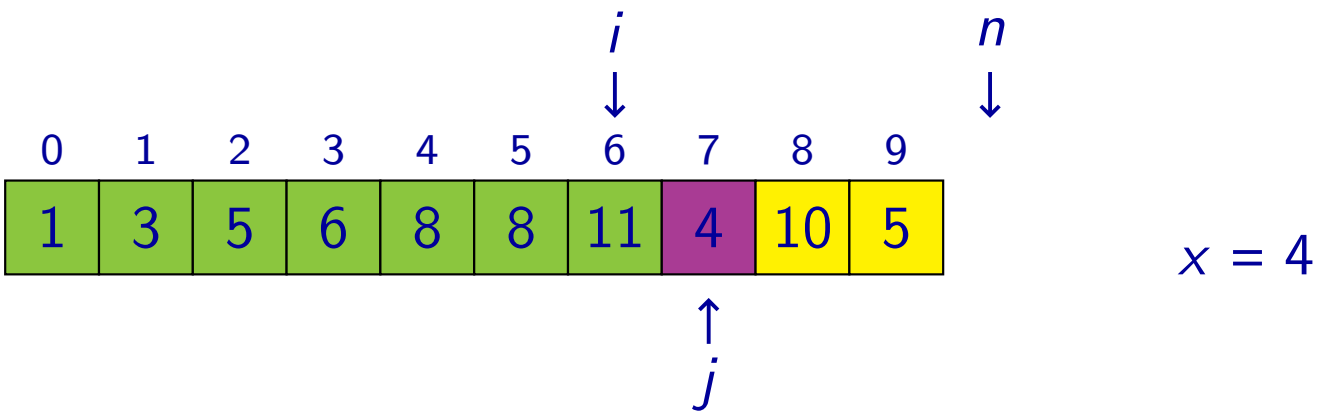
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



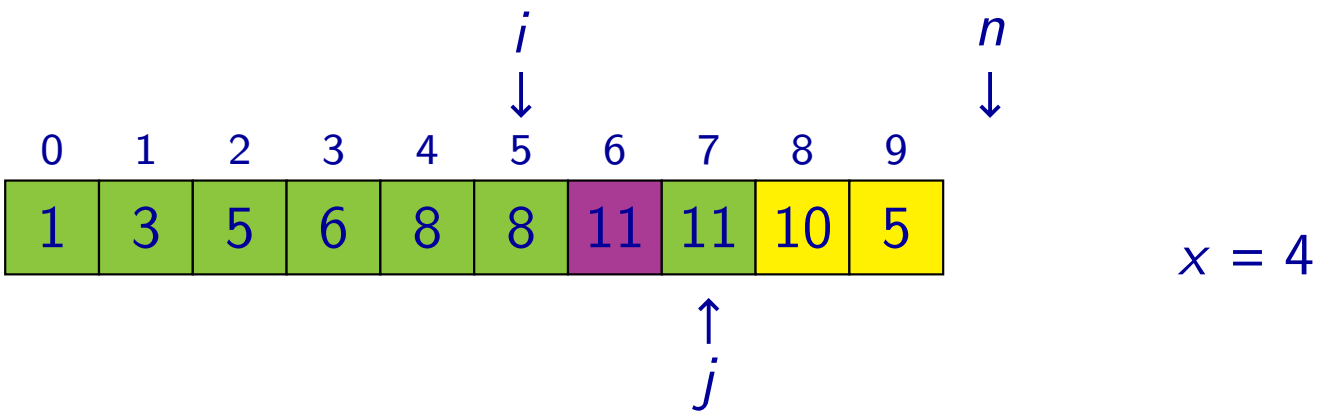
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



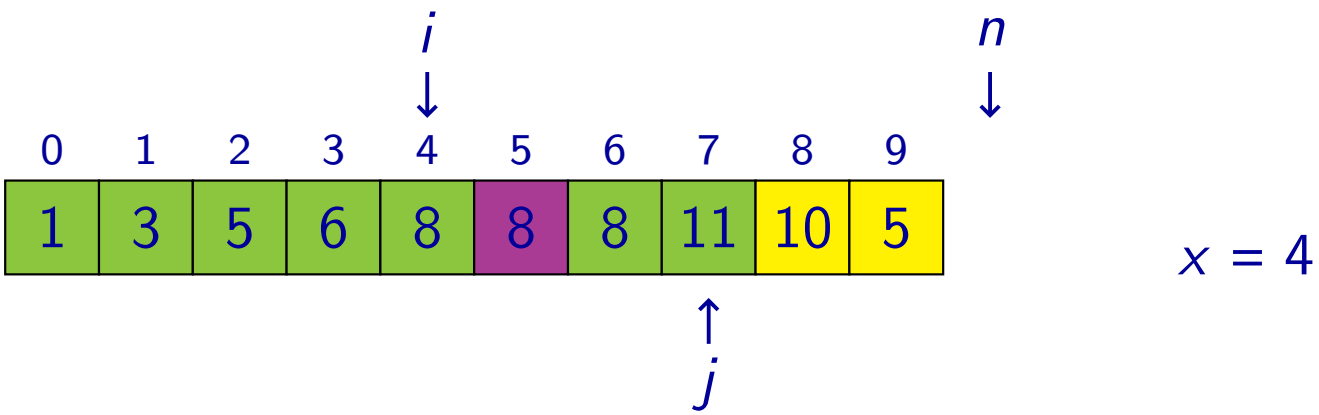
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



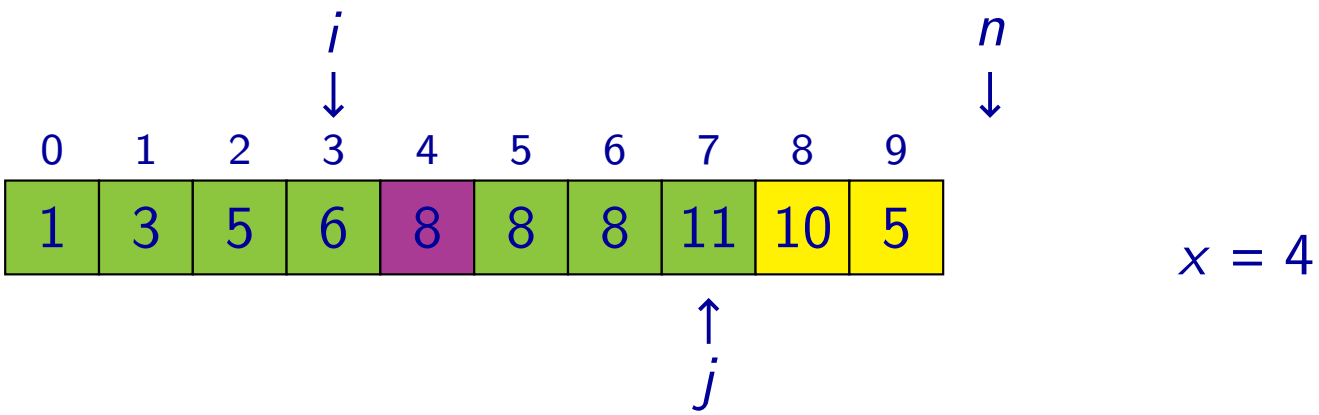
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



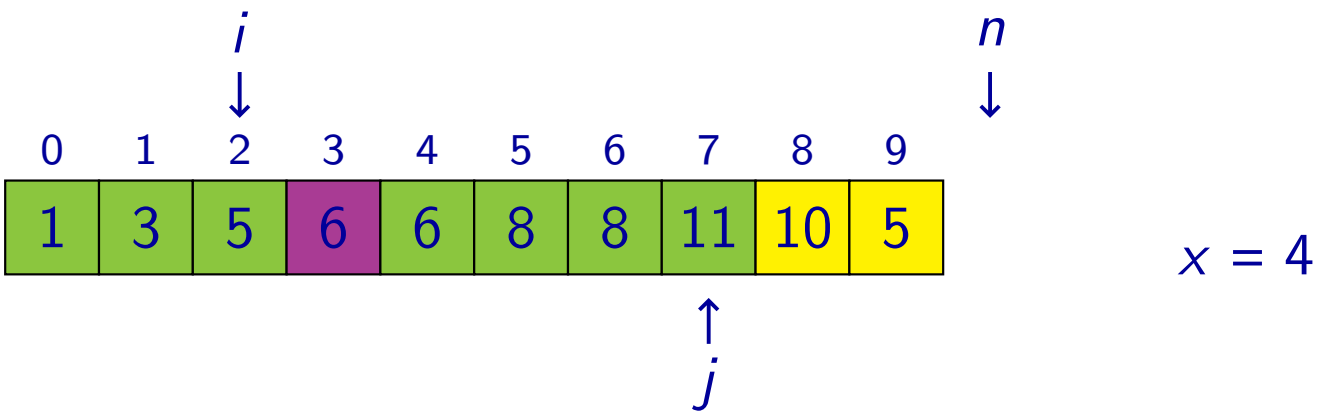
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



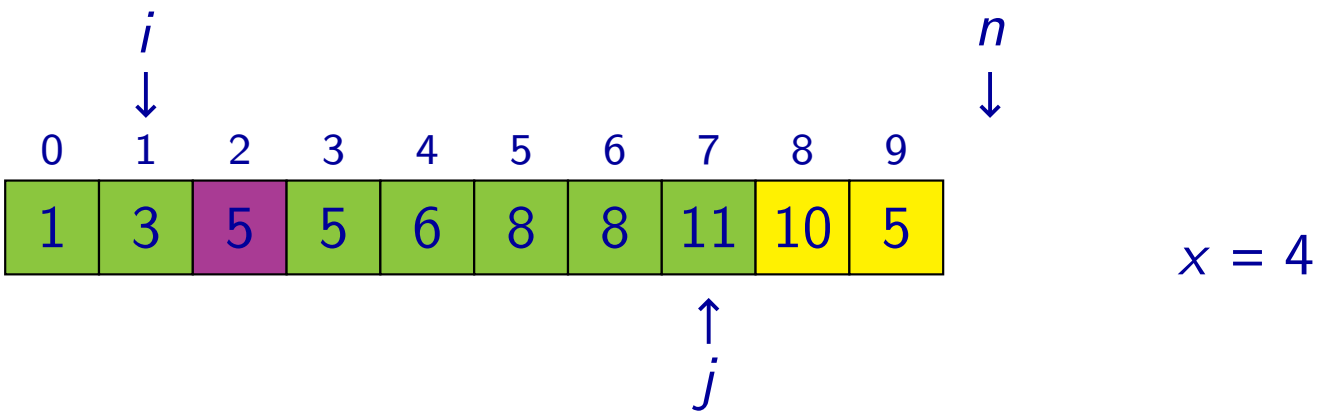
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



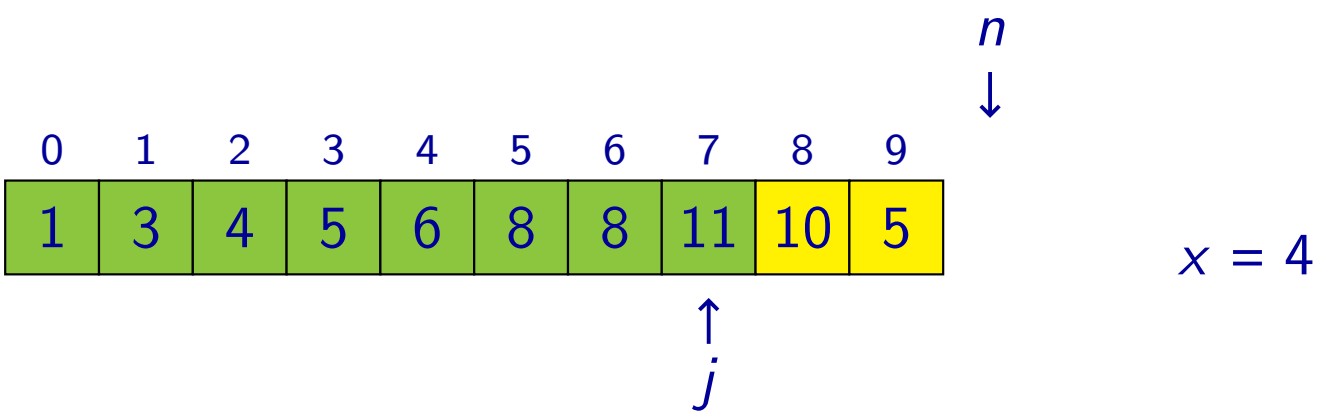
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



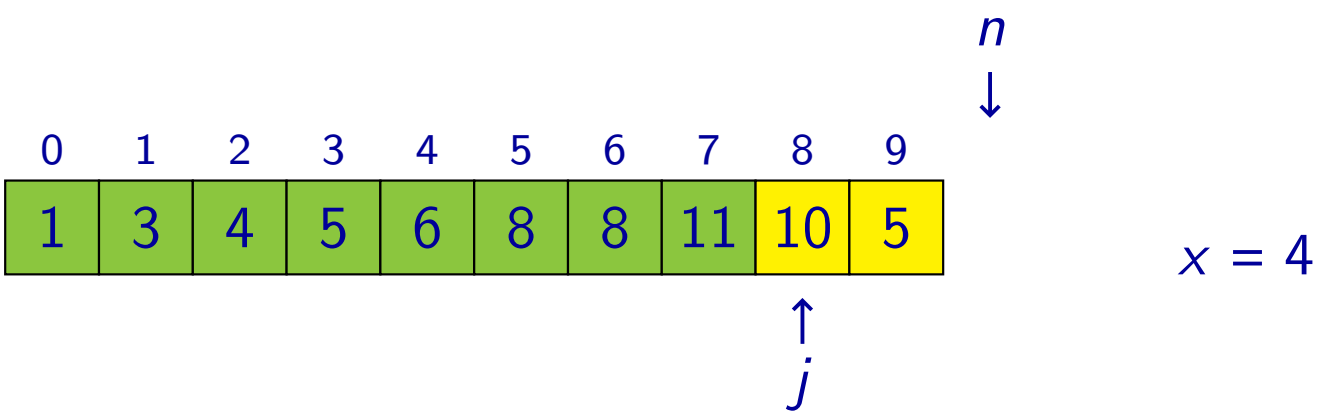
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



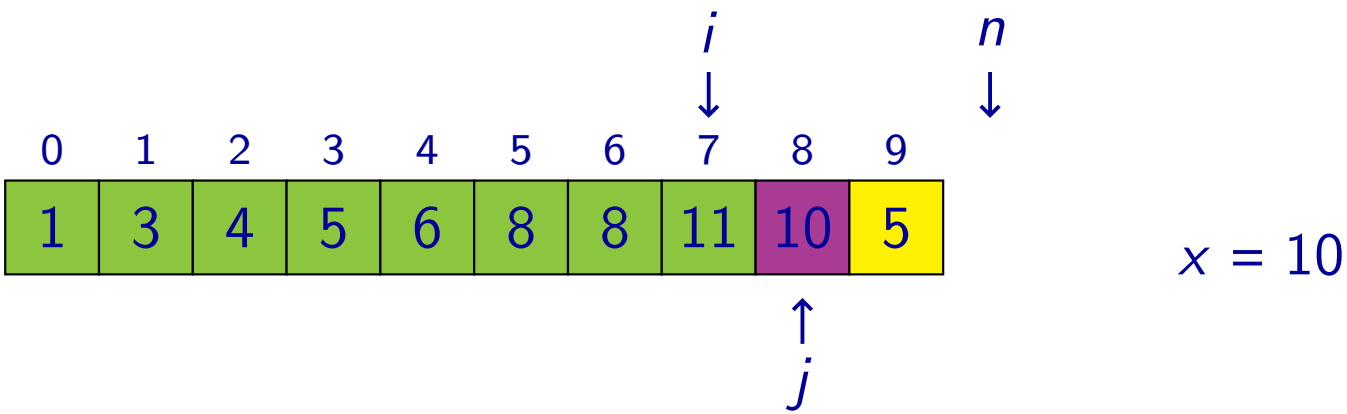
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



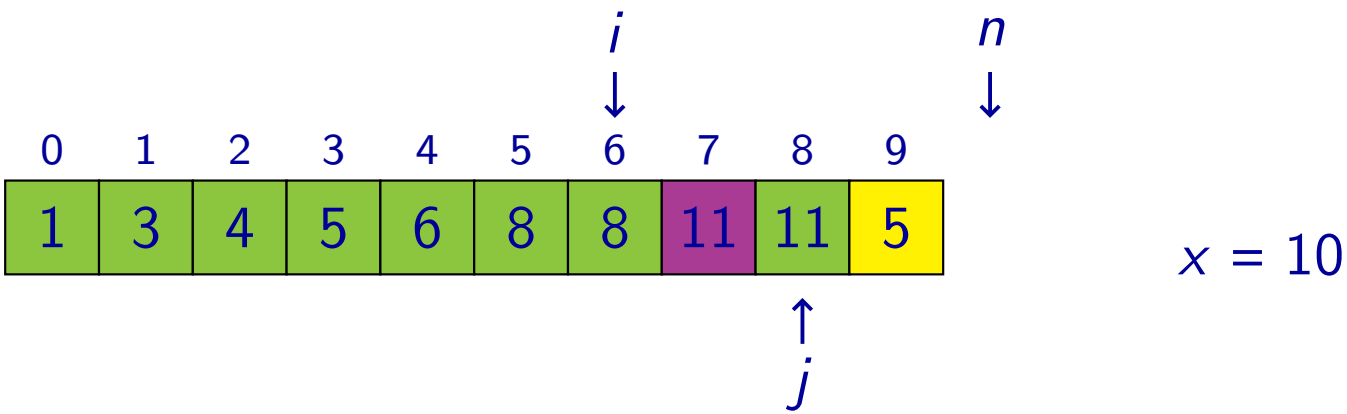
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



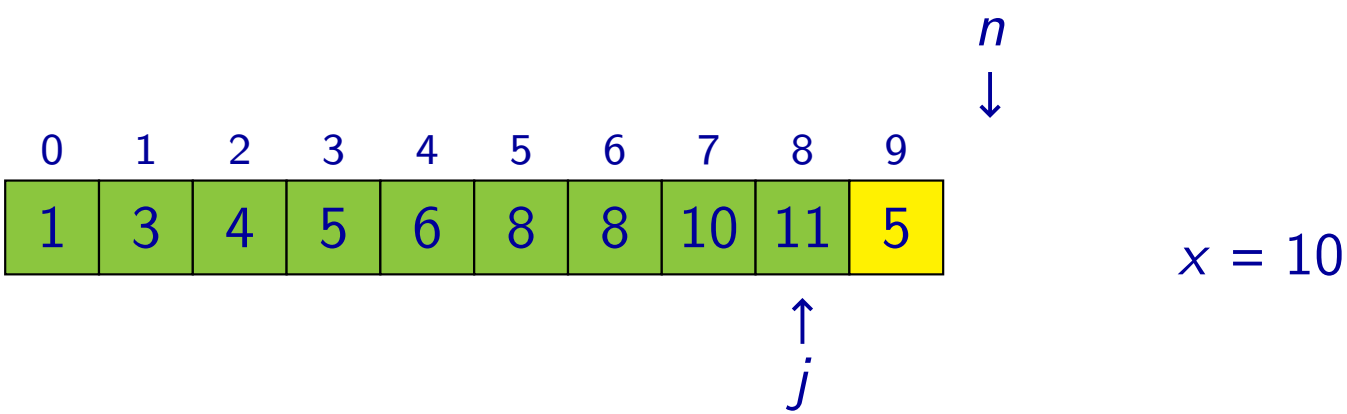
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



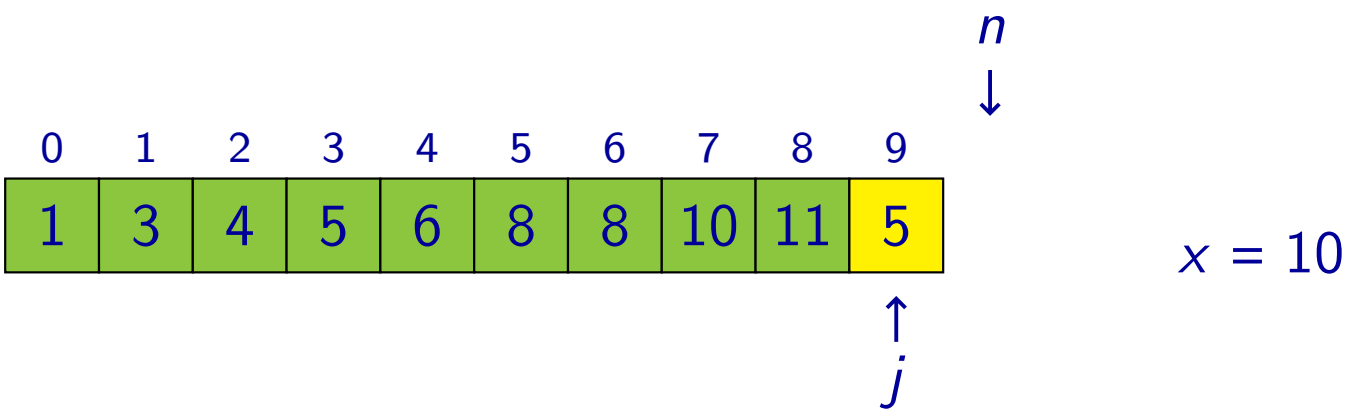
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



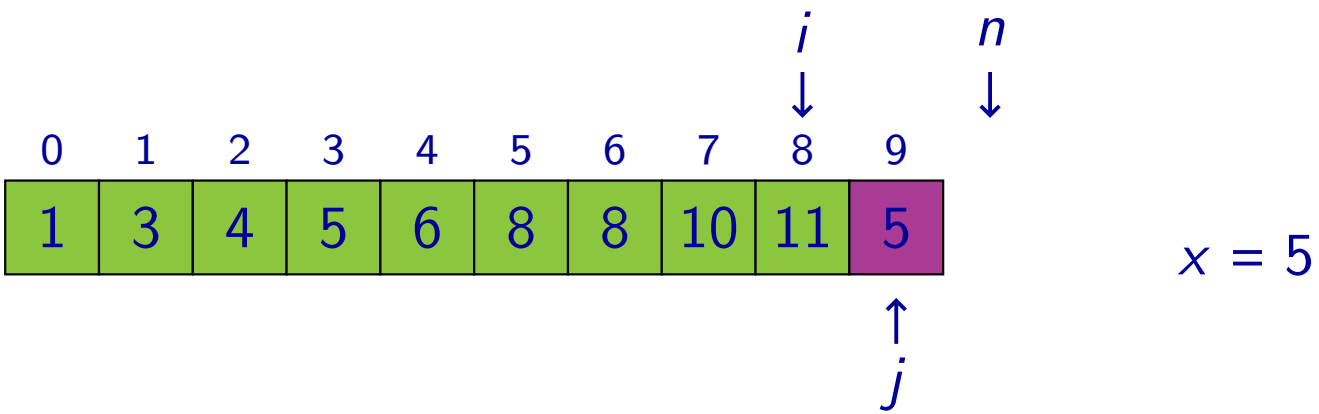
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



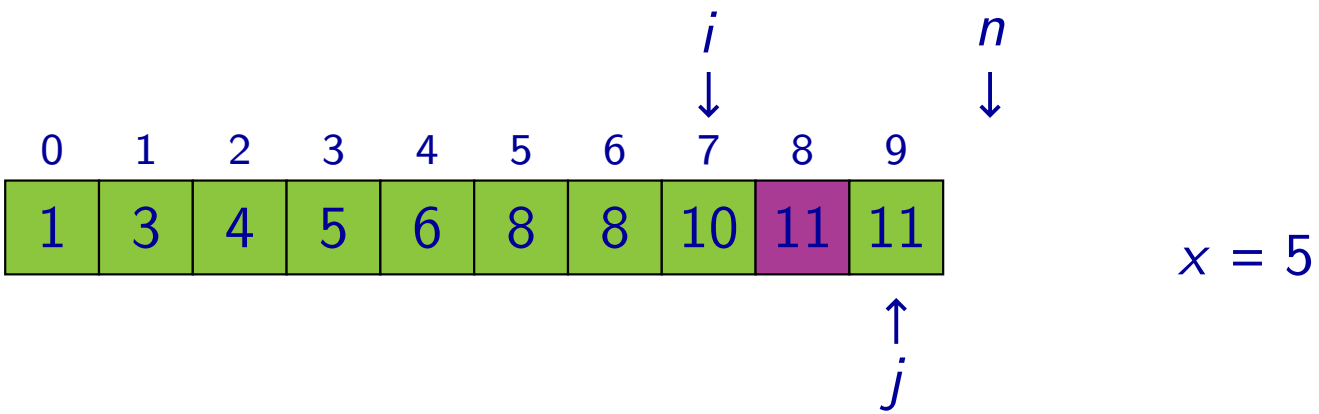
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



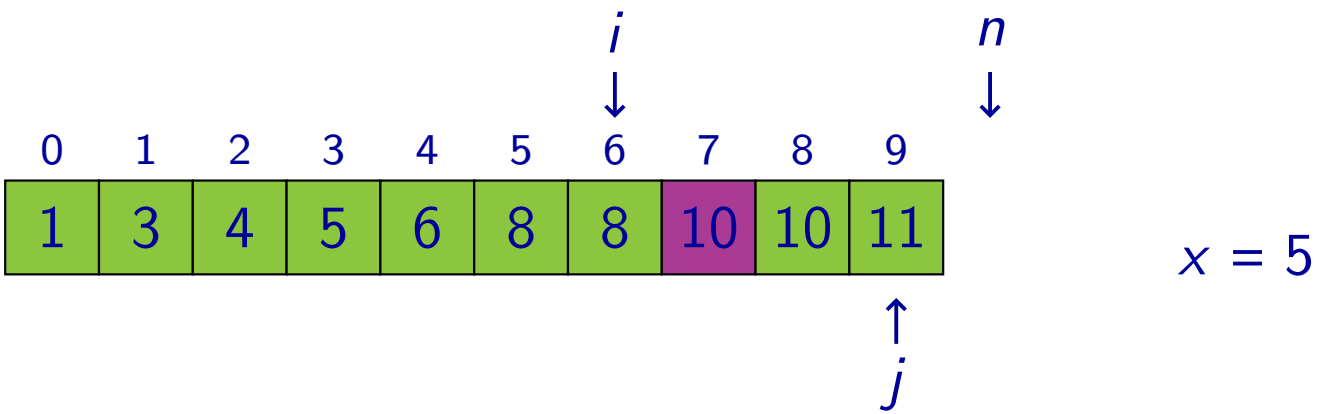
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



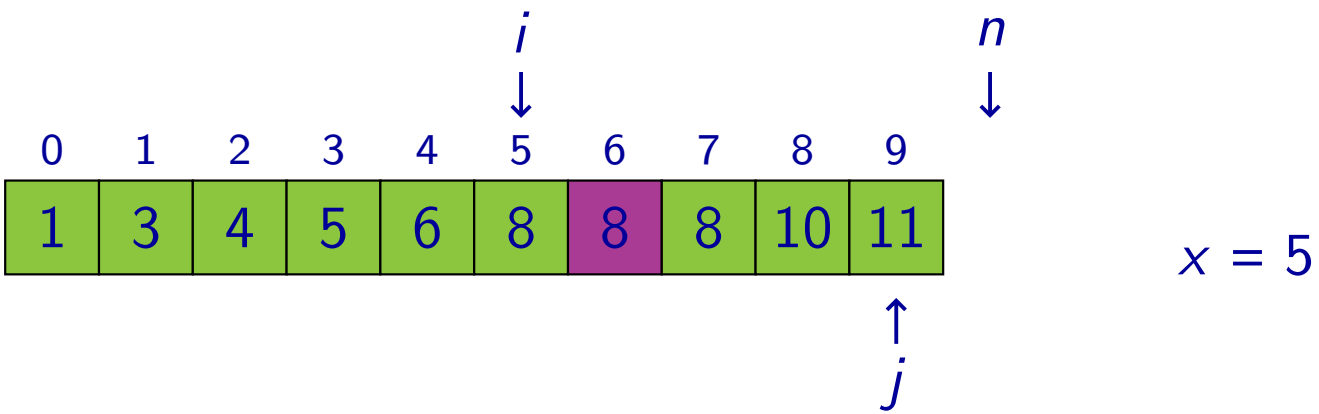
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



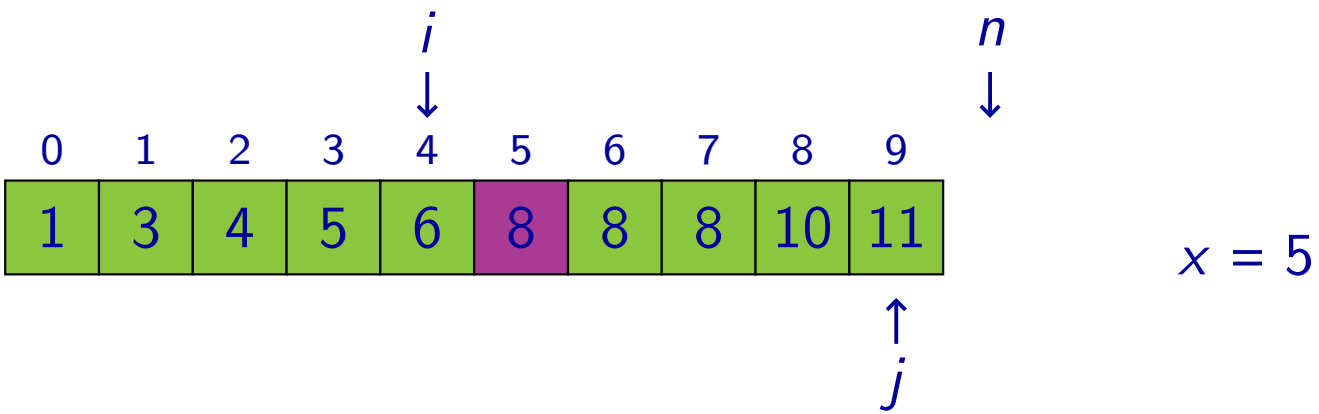
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



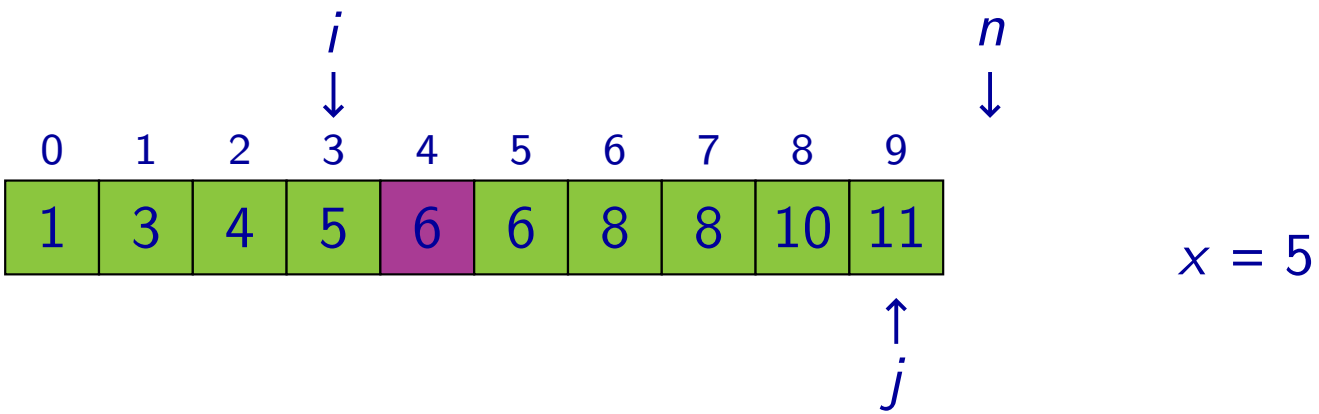
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



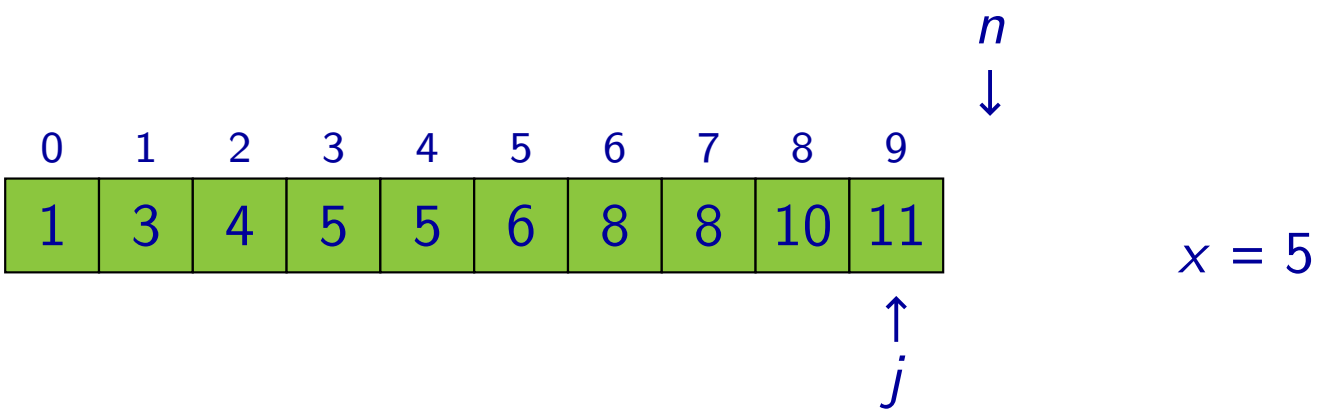
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



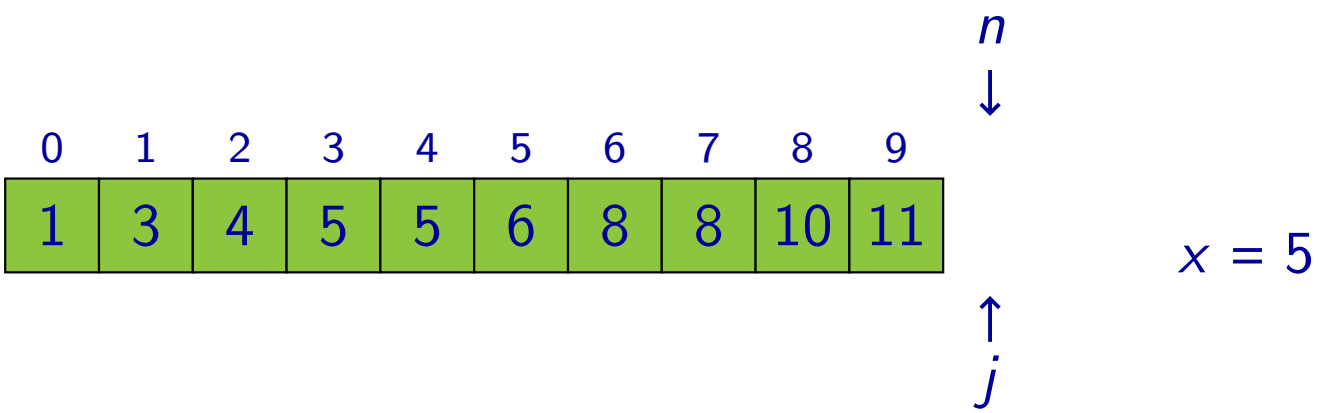
Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Příklad: Výpočet algoritmu INSERTION-SORT pro vstup

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Algoritmus: Třídění přímým vkládáním

INSERTION-SORT (A, n):

```
for j := 1 to n − 1 do
    x := A[j]
    i := j − 1
    while i ≥ 0 and A[i] > x do
        A[i + 1] := A[i]
        i := i − 1
    A[i + 1] := x
```

Složitost algoritmů

Uvažujme vstupy velikosti n:

- Vnější cyklus **for** se provede $n-1$ krát.
  (Proměnná j nabývá hodnot $1, 2, \ldots, n-1$.)
- Vnitřní cyklus **while** se pro danou hodnotu j provede maximálně j krát.
  (Proměnná i nabývá hodnot $j-1, j-2, \ldots, 1, 0$.)
- Existují vstupy, pro které platí, že pro každou hodnotu j od 1 do $n-1$ se vnitřní cyklus **while** provede právě j krát.
- V nejhorším případě se tedy cyklus **while** provede celkem m krát, kde
$$m = 1 + 2 + \cdots + (n-1) = (1 + (n-1)) \cdot \frac{n-1}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$$
- Celková časová složitost algoritmu Insertion-Sort v nejhorším případě je tedy $\Theta(n^2)$.

Složitost algoritmů

V předchozím případě jsme přesně spočítali celkový počet průchodů cyklem **while**.

Obecně to není vždy možné spočítat takto přesně nebo to může být hodně komplikované. Pokud nás zajímá jen asymptotický odhad, tak to často ani není nutné.

Složitost algoritmů

Pokud bychom například neuměli spočítat součet aritmetické posloupnosti, mohli bychom provést analýzu následovně:

- Vnější cyklus **for** se neprovede více než n krát, vnitřní cyklus **while** se při každé iteraci vnějšího cyklu provede maximálně n krát. Celkově se tedy vnitřní cyklus provede maximálně n^2 krát.

  Platí tedy $T \in O(n^2)$.

- Pro některé vstupy se při posledních $\lfloor n/2 \rfloor$ průchodech cyklem **for** provede cyklus **while** alespoň $\lceil n/2 \rceil$ krát.

  Pro některé vstupy se tedy cyklus **while** provede alespoň $\lfloor n/2 \rfloor \cdot \lceil n/2 \rceil$ krát.

  $\lfloor n/2 \rfloor \cdot \lceil n/2 \rceil \geq (n/2 - 1) \cdot (n/2) = \frac{1}{4}n^2 - \frac{1}{2}n$

  Platí tedy $T \in \Omega(n^2)$.

Složitost algoritmů

- Zatím jsme uvažovali, že provedení dané instrukce trvá vždy stejně dlouho bez ohledu na to, s jakými hodnotami pracuje.
- Při použití asymptotických odhadů tedy doba trvání jednotlivých instrukcí nehrála roli a důležité bylo pouze to, kolikrát se daná instrukce při běhu algoritmu provede.
- Například při použití strojů RAM jako výpočetního modelu to odpovídá počítání počtu provedených instrukcí, tj. doba trvání provedení jedné instrukce je 1.

  Tato se označuje jako použití tzv. **jednotkové míry**.
- Odhady časové složitosti v jednotkové míře odpovídají době běhu na skutečných počítačích za předpokladu, že operace, které provádí stroj RAM, může skutečný počítač provést v konstantním čase.

  To platí, pokud čísla, se kterými algoritmus pracuje, jsou malá (vejdou se např. do 32 nebo 64 bitů).

Složitost algoritmů

- Pokud by stroj RAM pracoval s „velkými“ čísly (např. 1000 bitovými), bude odhad časové složitosti v jednotkové míře nerealistický v tom smyslu, že výpočet na skutečném počítači bude trvat mnohem déle.
- Proto se při analýze časové složitosti algoritmů, u kterých se předpokládá práce s velkými čísly, používá tzv. **logaritmická míra**, kdy je doba provedení jedné instrukce úměrná počtu **bitových operací**, které je třeba pro provedení dané instrukce provést.
- Doba trvání instrukce je tedy závislá na aktuálních hodnotách jejích operandů.
- Například doba provádění instrukcí sčítání a odčítání je rovna součtu počtů bitů jejich operandů.
- Doba provádění instrukcí násobení a dělení je rovna součinu počtů bitů jejich operandů.

Složitost algoritmů

Poznámka: Zápisem $blen(x)$ označme počet bitů v binárním zápise přirozeného čísla x.

Platí

$$blen(x) = \max(1, \lceil \log_2(x+1) \rceil)$$

Prostorová (paměťová) složitost algoritmů

- Zatím jsme se zajímali o čas, který potřebujeme k výpočtu
- Někdy bývá kritickou velikost paměti potřebné k provedení výpočtu.

V případě strojů RAM opět můžeme i z hlediska množství použité paměti rozlišovat mezi použitím jednotkové a logaritmické míry:

Množstvím paměti stroje RAM $\mathcal{M}$ použitým pro vstup w rozumíme buď počet buněk paměti nebo počet bitů paměti, které stroj $\mathcal{M}$ během svého výpočtu nad vstupem w použije.

Definice

Prostorová složitost stroje RAM $\mathcal{M}$ (v nejhorším případě) je funkce $S : \mathbb{N} \to \mathbb{N}$, kde $S(n)$ udává maximální množství paměti použité strojem $\mathcal{M}$ pro vstupy délky n.

Prostorová (paměťová) složitost algoritmů

- Pro konkrétní problém můžeme mít dva algoritmy takové, že jeden má menší prostorovou složitost a druhý zase časovou složitost.
- Je-li časová složitost algoritmu v $O(f(n))$ je i prostorová v $O(f(n))$ (počet buněk navštívených RAMem nemůže být řádově větší než počet kroků, protože v každém kroku použije nejvýše tři buňky paměti — nejvýše dvě pro čtení a nejvýše jednu pro zápis).

Příklady analýzy složitosti algoritmů

(a příklady technik používaných při návrhu efektivních algoritmů)

Složitost algoritmů

Orientační typické hodnoty velikosti vstupu n, pro které algoritmus s danou časovou složitostí ještě většinou zvládne na „běžném PC“ spočítat výsledek ve zlomku sekundy nebo maximálně v řádu sekund.

(Závisí to samozřejmě výrazně na konkrétních detailech. Navíc se zde předpokládá, že v asymptotické notaci nejsou skryty nějaké velké konstanty.)

$O(n)$	$O(n \log n)$	$O(n^2)$	$O(n^3)$
$1\,000\,000 - 100\,000\,000$	$100\,000 - 1\,000\,000$	$1000 - 10\,000$	$100 - 1000$

$2^{O(n)}$	$O(n!)$
$20 - 30$	$10 - 15$

Složitost algoritmů

Při používání asymptotických odhadů časové složitosti algoritmů bychom si měli být vědomi některých úskalí:

- Asyptotické odhady se týkají pouze toho, jak roste čas s rostoucí velikostí vstupu.
- Neříkají nic o konkrétní době výpočtu. V asymptotické notaci mohou být skryty velké konstanty.
- Algoritmus, který má lepší asymptotickou časovou složitost než nějaký jiný algoritmus, může být ve skutečnosti rychlejší až pro nějaké hodně velké vstupy.
- Většinou analyzujeme složitost v nejhorším případě. Pro některé algoritmy může být doba výpočtu v nejhorším případě mnohem větší než doba výpočtu na „typických“ instancích.

Složitost algoritmů

- Můžeme si to ilustrovat na algoritmech pro třídění.

Algoritmus	Nejhorší případ	Průměrný případ
Bubblesort	$\Theta(n^2)$	$\Theta(n^2)$
Heapsort	$\Theta(n \log n)$	$\Theta(n \log n)$
Quicksort	$\Theta(n^2)$	$\Theta(n \log n)$

- Quicksort má horší asymptotickou složitost v nejhorším případě než Heapsort, stejnou asymptotickou složitost v průměrném případě a přesto je v praxi nejrychlejší.

Složitost algoritmů

Polynom — funkce tvaru

$$a_k n^k + a_{k-1} n^{k-1} + \cdots + a_2 n^2 + a_1 n + a_0$$

kde $a_0, a_1, \ldots, a_k$ jsou konstanty.

Příklady polynomů:

$$4n^3 - 2n^2 + 8n + 13 \qquad 2n + 1 \qquad n^{100}$$

Funkce f je **polynomiální**, jestliže je shora omezena nějakým polynomem, tj. jestliže existuje nějaká konstanta k taková, že $f \in O(n^k)$.

Polynomiální jsou například funkce, které patří do následujících tříd:

$$O(n) \qquad O(n \log n) \qquad O(n^2) \qquad O(n^5) \qquad O(\sqrt{n}) \qquad O(n^{100})$$

Složitost algoritmů

Funkce jako 2^n nebo $n!$ polynomiální nejsou — pro libovolně velkou konstantu k platí

$$2^n \in \Omega(n^k) \qquad\qquad n! \in \Omega(n^k)$$

Polynomiální algoritmus — algoritmus, jehož časová složitost je polynomiální (tj. shora omezená nějakým polynomem)

Zhruba se dá říct, že:

- polynomiální algoritmy jsou efektivní algoritmy, které se dají prakticky použít i pro relativně velké vstupy
- algoritmy, které polynomiální nejsou, se dají použít jen pro poměrně malé vstupy

Složitost algoritmů

Rozdělení na polynomiální a nepolynomiální algoritmy je velmi hrubé — nelze kategoricky tvrdit, že polynomiální algoritmy jsou vždy prakticky použitelné a nepolynomiální naopak nikdy nejsou:

- algoritmus se složitostí $\Theta(n^{100})$ pravděpodobně příliš prakticky použitelný nebude,
- některé algoritmy, které nejsou polynomiální, mohou fungovat efektivně pro velkou část vstupů, a složitost větší než polynomiální mají jen kvůli některým problematickým vstupům, na kterých může výpočet trvat velmi dlouhou dobu.

Poznámka: Polynomiální algoritmy, kde by konstanta v exponentu bylo nějaké velké číslo (např. algoritmy se složitostí $\Theta(n^{100})$), se při řešení běžných algoritmických problémů prakticky nevyskytují.

Složitost algoritmů

Pro většinu běžných algoritmických problémů nastává jedna ze tří možností:

- Je znám polynomiální algoritmus se složitostí $O(n^k)$, kde k je nějaké velmi malé číslo (např. 5 a častěji třeba 3 a méně).

- Není znám žádný polynomiální algoritmus a nejlepší známé algoritmy mají složitosti jako třeba $2^{\Theta(n)}$, $\Theta(n!)$ nebo nějaké ještě větší.

  V některých případech může být znám i důkaz, že pro daný problém žádný polynomiální algoritmus neexistuje (tj. nedá se vytvořit).

- Není znám žádný algoritmus, který řeší daný problém (a případně je i dokázáno, že žádný takový algoritmus neexistuje).

Složitost algoritmů

Typický příklad polynomiálního algoritmu — násobení matic s časovou složitostí $\Theta(n^3)$ a paměťovou složitostí $\Theta(n^2)$:

Algoritmus: Násobení matic

```
MATRIX-MULT(A, B, C, n):
    for i := 1 to n do
        for j := 1 to n do
            x := 0
            for k := 1 to n do
                x := x + A[i][k] * B[k][j]
            C[i][j] := x
```

Složitost algoritmů

- Při hrubé analýze složitosti často stačí spočítat počet do sebe vnořených smyček — a tento počet pak udává stupeň polynomu

  Příklad: Tři vnořené cykly při násobení matic — časová složitost algoritmu je $O(n^3)$.

- Pokud neprobíhají všechny smyčky např. od 0 do n, ale počet průchodů vnitřními smyčkami se při různých iteracích vnější smyčky mění, podrobnější analýza může být komplikovanější.

  Většinou to pak vede na počítání součtů různých typů číselných řad (např. aritmetické, geometrické, apod.).

  Často dá taková podrobnější analýza podobný výsledek jako hrubá analýza, mnohdy však může být složitost zjištěná touto podrobnější analýzou podstatně nižší než by vyplývalo z hrubého odhadu.

Aritmetická posloupnost

Aritmetická posloupnost — číselná řada $a_0, a_1, \ldots, a_{n-1}$, kde

$$a_i = a_0 + i \cdot d,$$

kde d je nějaká konstanta nezávistlá na i.

V aritmetické posloupnosti tedy pro všechna i platí $a_{i+1} = a_i + d$.

Příklad: Aritmetická posloupnost, kde $a_0 = 1$, $d = 1$ a $n = 100$:

$$1, 2, 3, 4, 5, 6, \ldots, 96, 97, 98, 99, 100$$

Součet aritmetické posloupnosti:

$$\sum_{i=0}^{n-1} a_i = a_0 + a_1 + \cdots + a_{n-1} = \frac{1}{2} n (a_0 + a_{n-1})$$

Aritmetická posloupnost

Příklad:

$$1 + 2 + \cdots + n \;=\; \frac{1}{2}n(n+1) \;=\; \frac{1}{2}n^2 + \frac{1}{2}n \;=\; \Theta(n^2)$$

Konkrétně například pro $n = 100$ je

$$1 + 2 + \cdots + 100 \;=\; 50 \cdot 101 \;=\; 5050.$$

Aritmetická posloupnost

Důkaz: Označme

$$s = \sum_{i=0}^{n-1} a_i = a_0 + a_1 + \cdots + a_{n-1}$$

$$\begin{aligned}
2s &= s + s \\
&= (a_0 + a_1 + \cdots + a_{n-1}) + (a_0 + a_1 + \cdots + a_{n-1}) \\
&= (a_0 + a_1 + \cdots + a_{n-1}) + (a_{n-1} + a_{n-2} + \cdots + a_0) \\
&= (a_0 + a_{n-1}) + (a_1 + a_{n-2}) + \cdots + (a_{n-1} + a_0) \\
&= ((a_0 + 0{\cdot}d) + (a_0 + (n-1){\cdot}d)) + ((a_0 + 1{\cdot}d) + (a_0 + (n-2){\cdot}d)) + \\
&\quad \cdots + ((a_0 + (n-1){\cdot}d) + (a_0 + 0{\cdot}d)) \\
&= n \cdot (a_0 + a_0 + (n-1){\cdot}d) \\
&= n \cdot (a_0 + a_{n-1})
\end{aligned}$$

Aritmetická posloupnost

Příklad: $s = 1 + 2 + 3 + \cdots + 99 + 100$

$$\begin{aligned} 2s &= s + s \\ &= (1 + 2 + \cdots + 100) + (1 + 2 + \cdots + 100) \\ &= (1 + 2 + \cdots + 100) + (100 + 99 + \cdots + 1) \\ &= (1 + 100) + (2 + 99) + (3 + 98) + \cdots + (99 + 2) + (100 + 1) \\ &= 100 \cdot (1 + 100) = 10100 \end{aligned}$$

Takže

$$s = \frac{1}{2} \cdot 10100 = 5050$$

Geometrická posloupnost

Geometrická posloupnost — číselná řada $a_0, a_1, \ldots, a_n$, kde

$$a_i = a_0 \cdot q^i,$$

kde q je nějaká konstanta nezávistlá na i.

V geometrické posloupnosti tedy pro všechna i platí $a_{i+1} = a_i \cdot q$.

Příklad: Geometrická posloupnost, kde $a_0 = 1$, $q = 2$ a $n = 14$:

1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384

Součet geometrické posloupnosti (kde $q \neq 1$):

$$\sum_{i=0}^{n} a_i = a_0 + a_1 + \cdots + a_n = a_0 \frac{q^{n+1} - 1}{q - 1}$$

Geometrická posloupnost

Příklad:

$$1 + q + q^2 + \cdots + q^n = \frac{q^{n+1} - 1}{q - 1}$$

Speciálně pro $q = 2$:

$$1 + 2^1 + 2^2 + 2^3 + \cdots + 2^n = \frac{2^{n+1} - 1}{2 - 1} = 2 \cdot 2^n - 1 = \Theta(2^n)$$

Geometrická posloupnost

Důkaz: Označme

$$s = \sum_{i=0}^{n} a_i = a_0 + a_1 + \cdots + a_n$$

$$\begin{aligned}
s &= a_0 \cdot q^0 + a_0 \cdot q^1 + \cdots + a_0 \cdot q^n \\
s \cdot q &= (a_0 \cdot q^0 + a_0 \cdot q^1 + \cdots + a_0 \cdot q^n) \cdot q \\
&= a_0 \cdot q^1 + a_0 \cdot q^2 + \cdots + a_0 \cdot q^{n+1} \\
s \cdot q - s &= a_0 \cdot q^{n+1} - a_0 \cdot q^0 \\
s \cdot (q-1) &= a_0 \cdot (q^{n+1} - 1) \\
s &= a_0 \cdot \frac{q^{n+1} - 1}{q - 1}
\end{aligned}$$

Složitost algoritmů

Exponenciální funkce: funkce tvaru c^n, kde c je konstanta — např. funkce 2^n

Logaritmus — inverzní funkce k exponenciální funkci: pro dané n je

$$\log_c n$$

taková hodnota x, že $c^x = n$.

Složitost algoritmů

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768
16	65536
17	131072
18	262144
19	524288
20	1048576

n	$\lceil \log_2 n \rceil$
0	—
1	0
2	1
3	2
4	2
5	3
6	3
7	3
8	3
9	4
10	4
11	4
12	4
13	4
14	4
15	4
16	4
17	5
18	5
19	5
20	5

n	$\log_2 n$
1	0
2	1
4	2
8	3
16	4
32	5
64	6
128	7
256	8
512	9
1024	10
2048	11
4096	12
8192	13
16384	14
32768	15
65536	16
131072	17
262144	18
524288	19
1048576	20

Složitost algoritmů

Příklady toho, kde se při analýze algoritmů objevují exponenciální funkce a logaritmy:

- Nějaká hodnota se opakovaně zmenšuje na polovinu nebo naopak zdvojnásobuje.

  Například u **binárního vyhledávání** (metodou půlení intervalu) se s každou iterací cyklu zmenšuje velikost intervalu na polovinu.

  Předpokládejme, že pole má velikost n.

  Jaká je minimální velikost pole n, při které se provede alespoň k iterací?

  Odpověď: 2^k

  Platí tedy $k = \log_2(n)$. Časová složitost algoritmu je pak $\Theta(\log n)$.

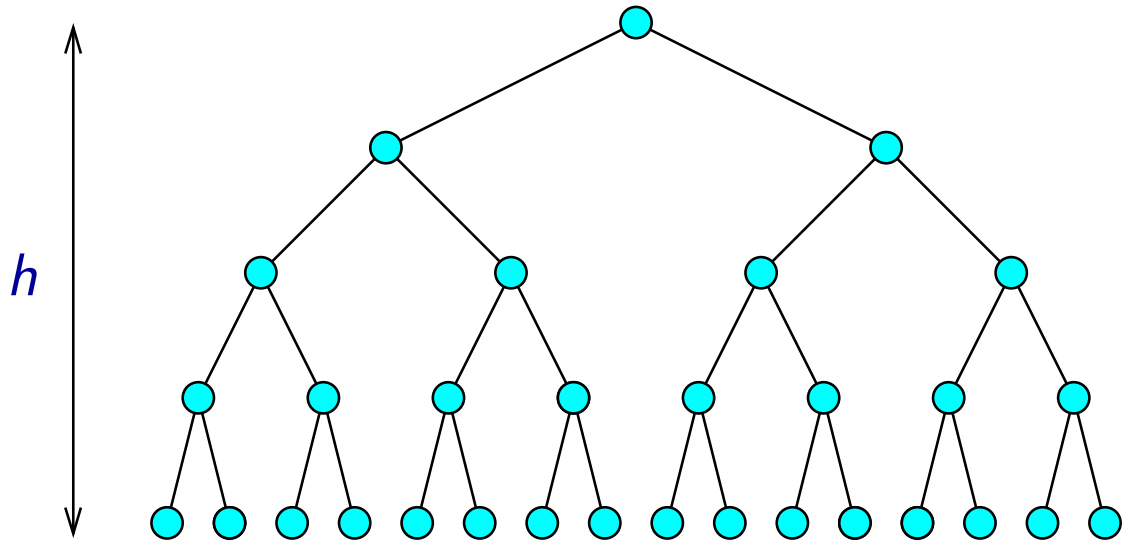
Složitost algoritmů

- Pomocí n bitů je možno reprezentovat čísla od 0 do $2^n - 1$.
- Minimální počet bitů potřebných pro uložení přirozeného čísla x reprezentovaného binárně je
$$\lceil \log_2(x+1) \rceil.$$
- Dokonale vyvážený binární strom o výšce h má $2^{h+1} - 1$ vrcholů, z čehož 2^h jsou listy.
- Dokonale vyvážený binární strom o n vrcholech má výšku zhruba $\log_2 n$.

  Ilustrační příklad: Kdybychom nakreslili vyvážený strom o $n = 1\,000\,000$ vrcholech tak, aby sousední vrcholy byly vzdáleny o 1 cm a výška každé vrstvy byla také 1 cm, měl by tento strom na šířku 10 km a na výšku zhruba 20 cm.

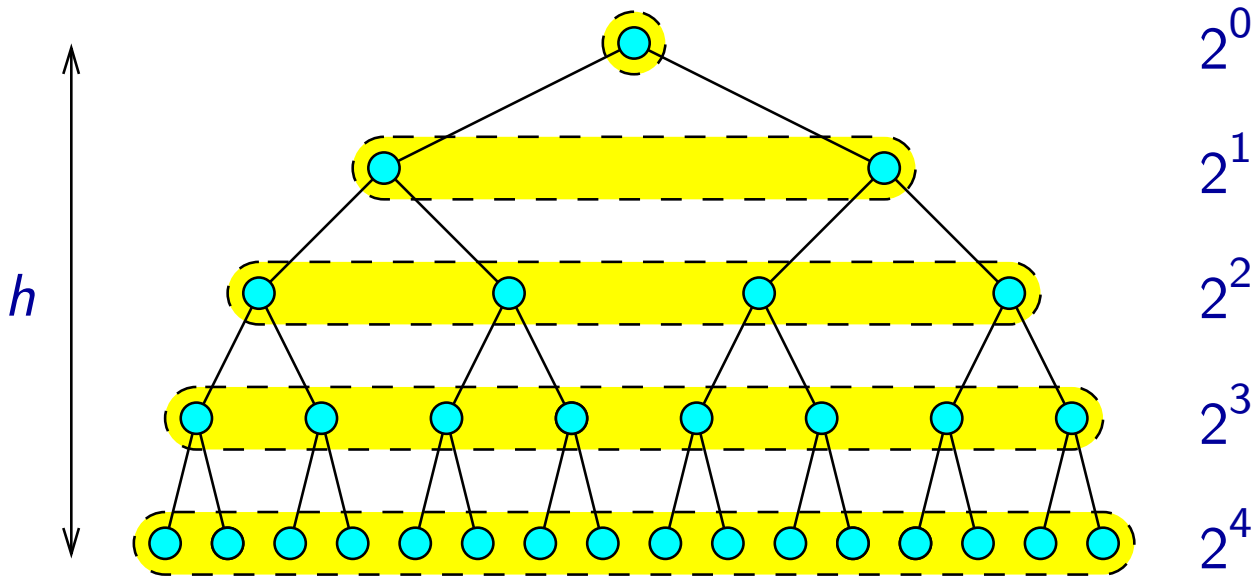
Složitost algoritmů

Dokonale vyvážený binární strom výšky h:



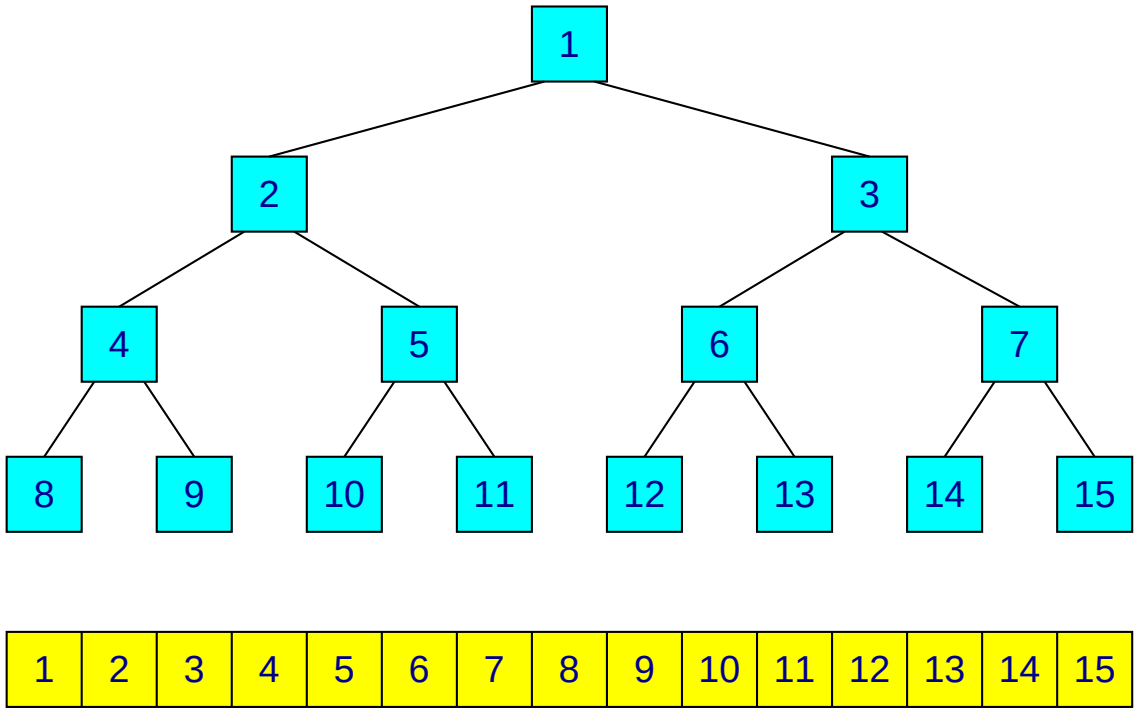
Složitost algoritmů

Dokonale vyvážený binární strom výšky h:



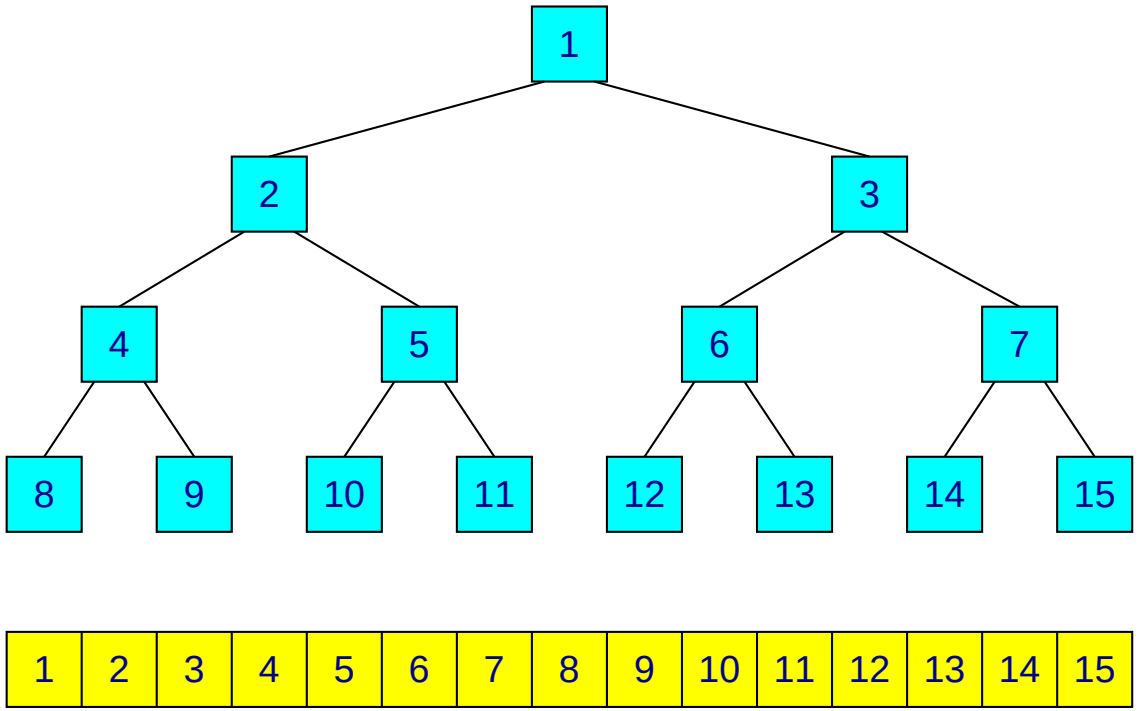
Složitost algoritmů

Efektivní uložení úplného binárního stromu v poli:



Složitost algoritmů

Efektivní uložení úplného binárního stromu v poli:



Potomci vrcholu s indexem i mají indexy $2i$ a $2i + 1$.
Rodič vrcholu s indexem i má index $\lfloor i/2 \rfloor$.

Složitost algoritmů

Halda (**heap**) — úplný binární strom uložený v poli A výše uvedeným způsobem, kde navíc pro každé $i = 1, 2, \ldots, n$ platí:

- pokud $2i \leq n$, pak $A[i] \leq A[2i]$
- pokud $2i + 1 \leq n$, pak $A[i] \leq A[2i + 1]$

Příklady využití haldy:

- třídící algoritmus **HeapSort**
- efektivní implementace **prioritní fronty** — umožňuje provádět většinu operací na této frontě s časovou složitostí v $O(\log n)$, kde n je počet prvků ve frontě

Algoritmus: Vytvoření haldy z nesetříděného pole

```
CREATE-HEAP (A, n):
    i := ⌊n/2⌋
    while i ≥ 1 do
        j := i
        x := A[j]
        while 2 * j ≤ n do
            k := 2 * j
            if k + 1 ≤ n and A[k + 1] < A[k] then
                k := k + 1
            if x ≤ A[k] then break
            A[j] := A[k]
            j := k
        A[j] := x
        i := i − 1
```

Složitost algoritmů

Časová složitost algoritmu CREATE-HEAP:

- Rychlou a hrubou analýzou lehce zjistíme, že tato složitost je v $O(n \log n)$ a v $\Omega(n)$:
  - Vnější cyklus se provede vždy $\lfloor n/2 \rfloor$ krát — počet průchodů je tedy v $\Theta(n)$.
  - Počet průchodů vnitřním cyklem v rámci jedné iterace vnějšího cyklu je očividně v $O(\log n)$.
- Daleko méně zřejmé je, že celkový počet průchodů vnitřním cyklem (tj. dohromady přes všechny iterace vnějšího cyklu) je ve skutečnosti v $O(n)$.

Celkově tedy dostáváme:

Časová složitost algoritmu CREATE-HEAP je v $\Theta(n)$.

Složitost algoritmů

Zdůvodnění toho, proč je počet průchodů vnitřním cyklem v $O(n)$:

Předpokládejme pro jednoduchost, že všechny větve stromu jsou stejně dlouhé a mají délku h — platí tedy $n = 2^{h+1} - 1$.

Označme C_i, kde $0 \leq i < h$, celkový počet průchodů vnitřním cyklem, kdy je na začátku cyklu hodnota j v i-té vrstvě stromu (vrstvy jsou číslovány odshora $0, 1, 2, \ldots$).

Zjevně je celkový počet průchodů s dán vztahem

$$s = C_{h-1} + C_{h-2} + \cdots + C_0 = \sum_{i=0}^{h-1} C_i$$

Hodnotu C_i spočítáme jako celkový počet vrcholů ve vstvách $0, 1, \ldots, i$:

$$C_i = 2^0 + 2^1 + \cdots + 2^i = \sum_{k=0}^{i} 2^k = \frac{2^{i+1} - 1}{2 - 1} = 2^{i+1} - 1$$

Celkový součet pak spočítáme následovně:

$$s = \sum_{i=0}^{h-1} C_i = \sum_{i=0}^{h-1} (2^{i+1} - 1) = 2 \cdot \left(\sum_{i=0}^{h-1} 2^i \right) - \left(\sum_{i=0}^{h-1} 1 \right)$$

$$= 2 \cdot \frac{2^h - 1}{2 - 1} - h = 2^{h+1} - 2 - h = n - 1 - h = O(n)$$

Rekurzivní algoritmy

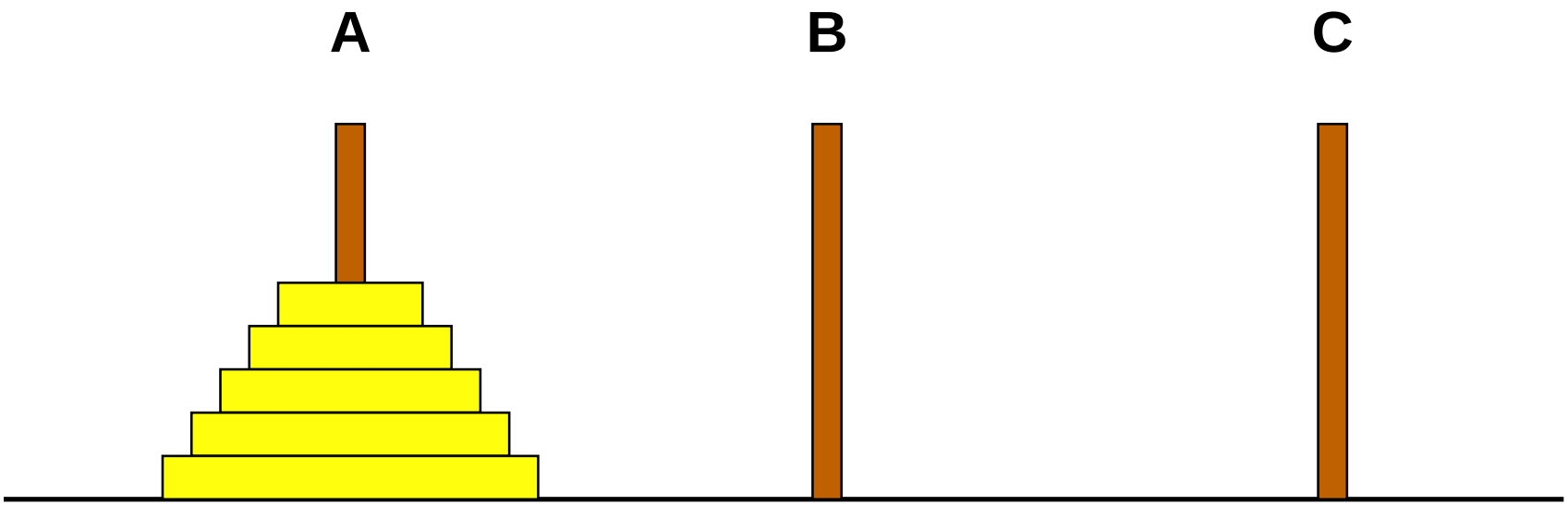
Rekurzivní algoritmus je algoritmus, který převede řešení původního problému na řešení několika podobných problémů pro menší instance.

Obecné schéma rekurzivních algoritmů:

- Pokud se jedná o elementární případ, vyřeš ho přímo a vrať výsledek.
- V opačném případě vytvoř instance podproblémů.
- Zavolej sám sebe pro každou z těchto instancí.
- Z výsledků pro jednotlivé podproblémy slož řešení původního problému a vrať ho jako výsledek.

Poznámka: Instance podproblémů musí vždy být v nějakém smyslu menší než instance původního problému. Často (ne však vždy) se zmenšuje velikost instance.

Hanojské věže



Úkol: Přemístit disky z A na B, přičemž:

- V jednom okamžiku je možné přesouvat jen jeden disk.
- Není dovaleno položit větší disk na menší.

Hanojské věže

$n = 1$:

$A \to B$

Hanojské věže

$n = 1$:

$A \to B$

$n = 2$:

$A \to C$

$A \to B$

$C \to B$

Hanojské věže

$n = 1$:

$A \to B$

$n = 2$:

$A \to C$
$A \to B$
$C \to B$

$n = 3$:

$A \to B$
$A \to C$
$B \to C$
$A \to B$
$C \to A$
$C \to B$
$A \to B$

Hanojské věže

$n = 1$:

$A \to B$

$n = 2$:

$A \to C$
$A \to B$
$C \to B$

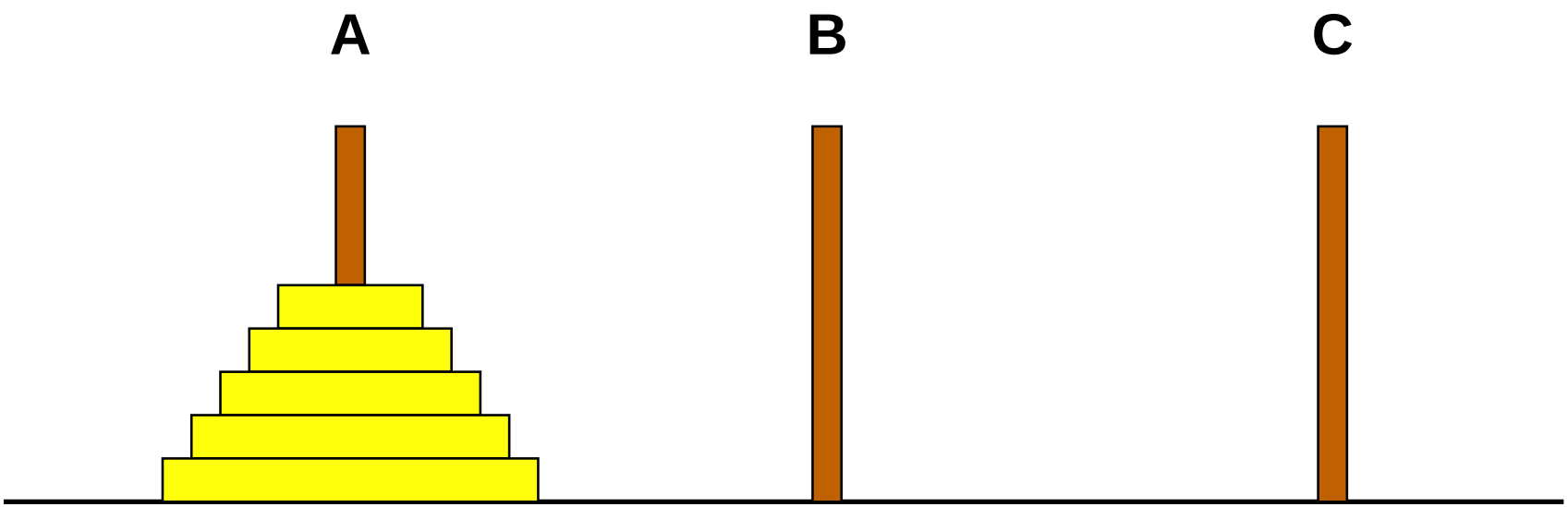
$n = 3$:

$A \to B$
$A \to C$
$B \to C$
$A \to B$
$C \to A$
$C \to B$
$A \to B$

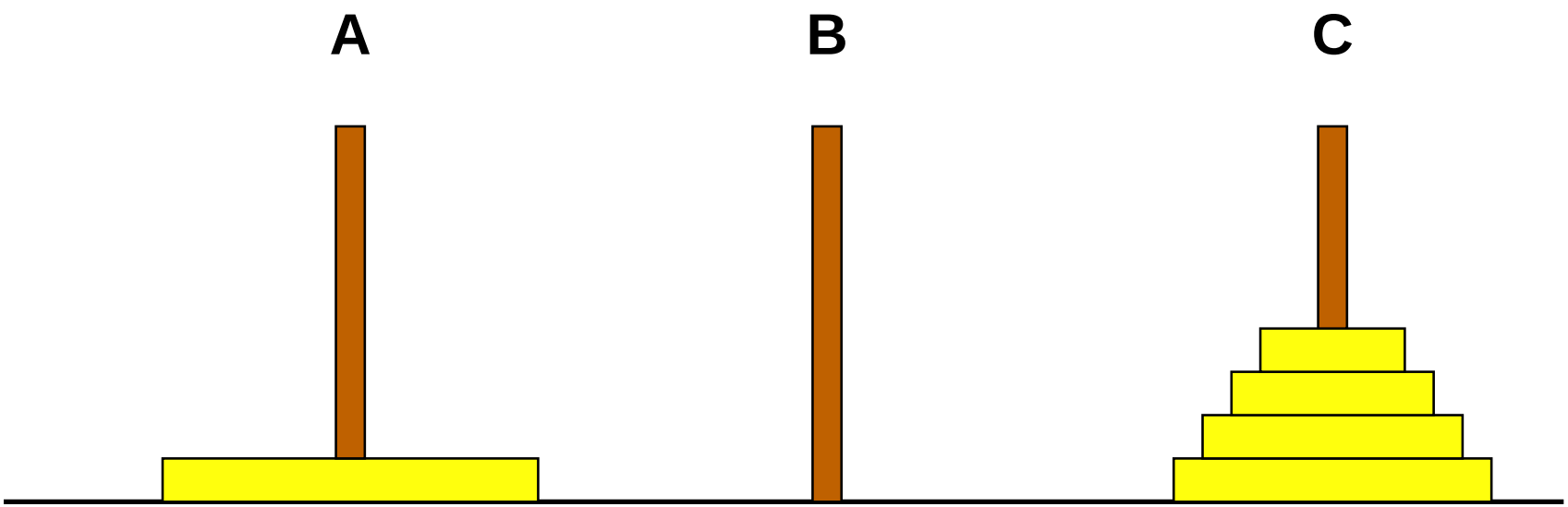
$n = 4$:

$A \to C$
$A \to B$
$C \to B$
$A \to C$
$B \to A$
$B \to C$
$A \to C$
$A \to B$
$C \to B$
$C \to A$
$B \to A$
$C \to B$
$A \to C$
$A \to B$
$C \to B$

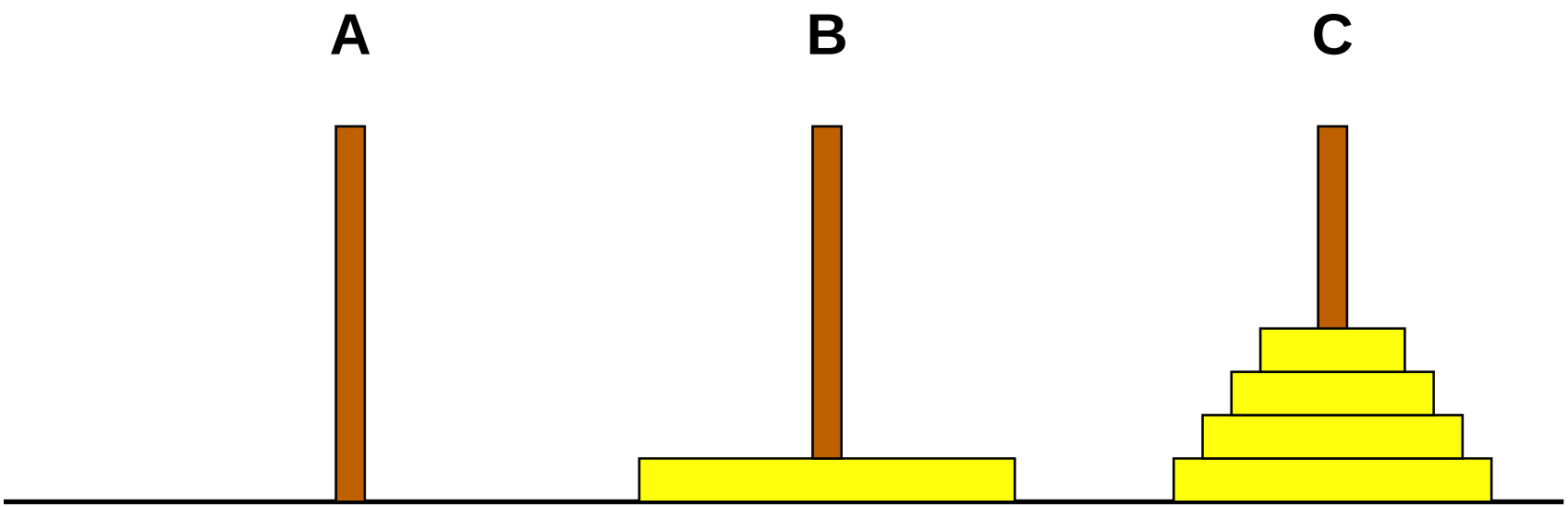
Hanojské věže



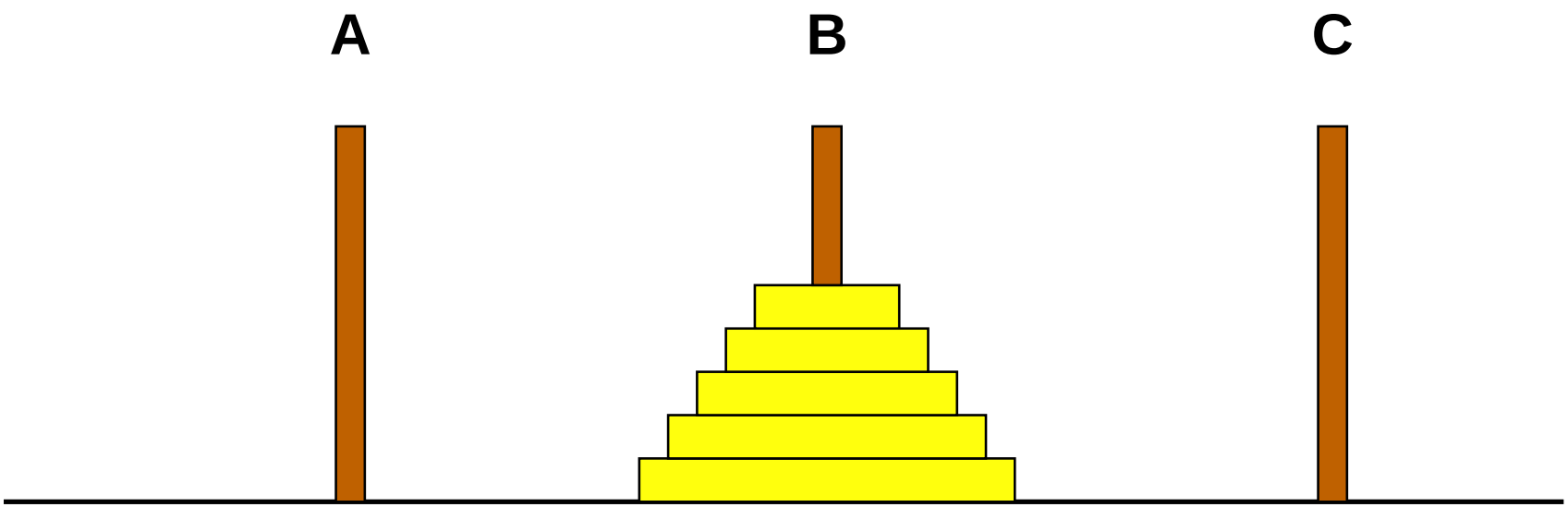
Hanojské věže



Hanojské věže



Hanojské věže



Hanojské věže

Algoritmus: Hanojské věže

```
HANOI(n, src, dst, tmp):
    if n = 0 then return
    HANOI(n − 1, src, tmp, dst)
    print(src, "→", dst)
    HANOI(n − 1, tmp, dst, src)

MAIN(n):
    HANOI(n, "A", "B", "C")
```

Hanojské věže

Označme $P(n)$ počet tahů, které provede algoritmus pro n disků.

Tvrzení

$P(n) = 2^n - 1$.

Důkaz:

- Pro $n = 0$: $P(n) = 0 = 2^0 - 1$
- Pro $n > 0$: Předpokládáme, že $P(n-1) = 2^{n-1} - 1$.

$$\begin{aligned} P(n) &= 2P(n-1) + 1 = \\ &= 2(2^{n-1} - 1) + 1 = \\ &= 2 \cdot 2^{n-1} - 2 + 1 = \\ &= 2^n - 1 \end{aligned}$$

Hanojské věže

Tvrzení

Pro přesun n disků je třeba minimálně $2^n - 1$ tahů.

Důkaz:

Indukcí.

Uvedený algoritmus nalezne tedy optimální řešení.

Poznámka

Otázka: Jak dlouho by trvalo přesunutí 64 disků, pokud by přesunutí jednoho disku trvalo 1 s ?

Hanojské věže

Tvrzení

Pro přesun n disků je třeba minimálně $2^n - 1$ tahů.

Důkaz:

Indukcí.

Uvedený algoritmus nalezne tedy optimální řešení.

Poznámka

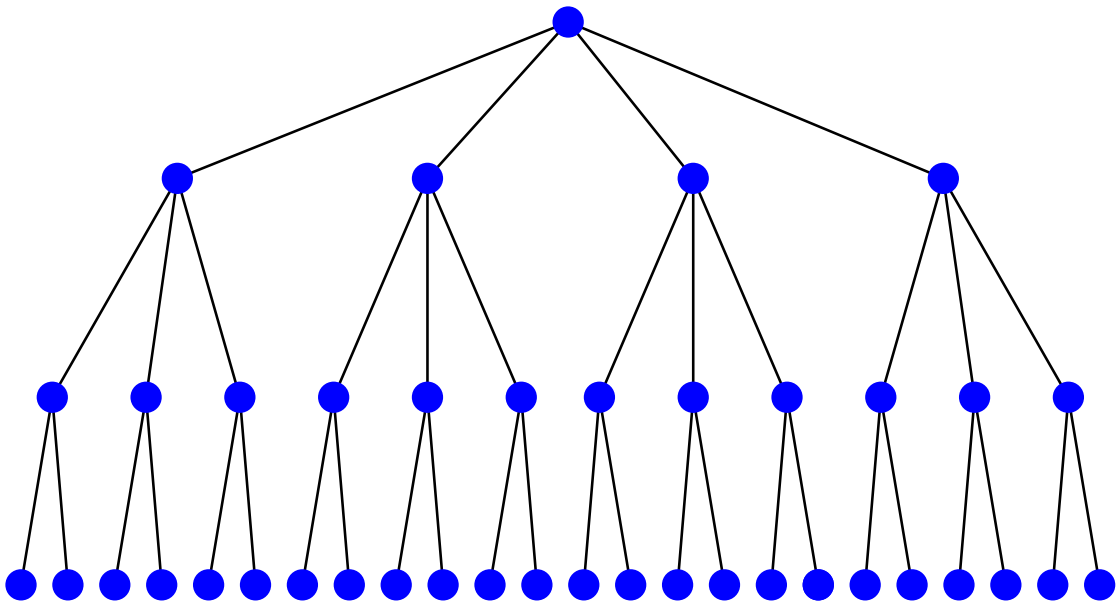
Otázka: Jak dlouho by trvalo přesunutí 64 disků, pokud by přesunutí jednoho disku trvalo 1 s ?

Odpověď: 18446744073709551615 s, tj. asi 585 miliard let.

Rekurzivní algoritmy

Výpočet rekurzivního algoritmu je možné znázornit jako strom:

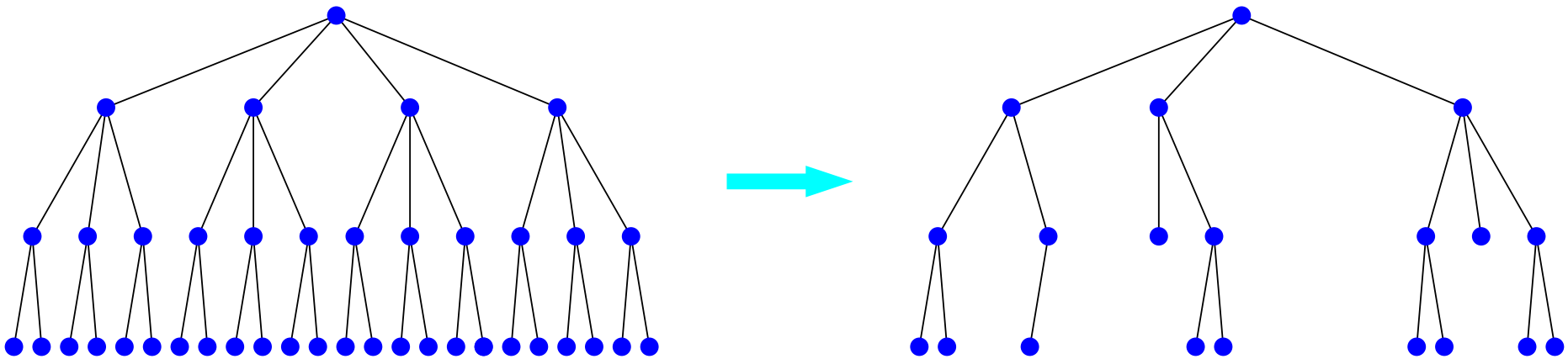
- vrcholy stromu odpovídají jednotlivým podproblémům
- kořen je původní problém
- potomci vrcholu odpovídají podproblémům daného problému



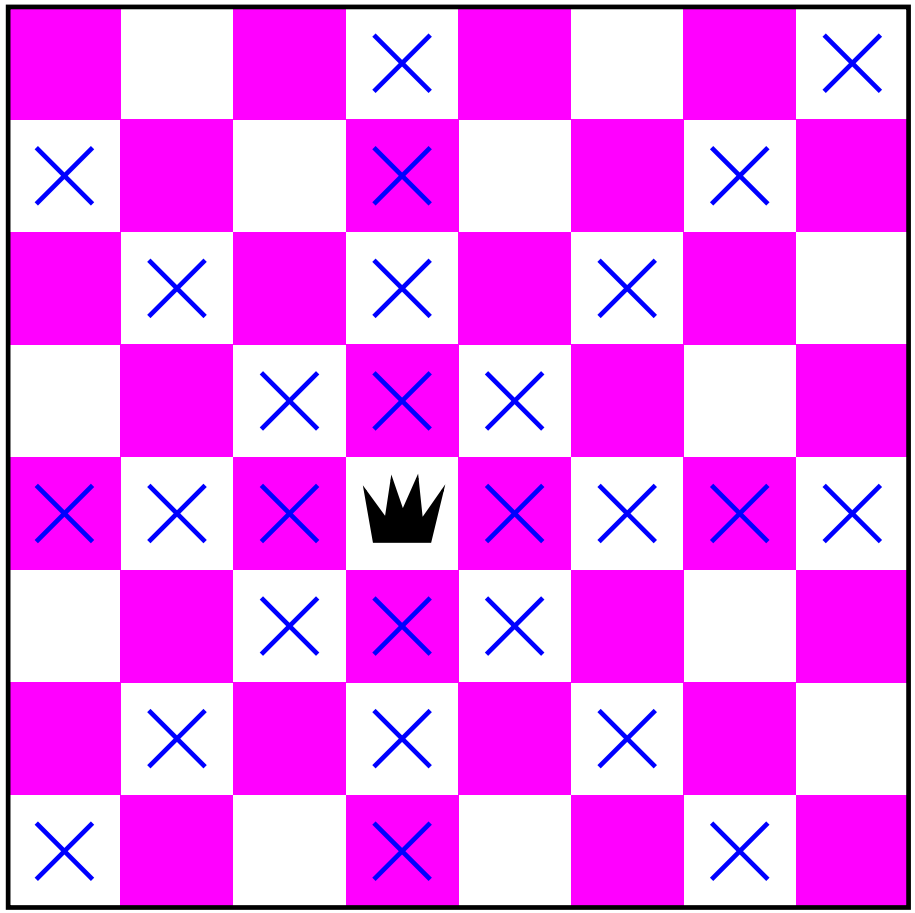
Pruning

Přístup, kdy do rekurzivním algoritmu doplníme nějaké vhodné testy, které způsobí, že se rekurzivní procedura nebude volat pro některé podproblémy, u kterých je jisté, že jejich vyřešení nepovede k řešení celého problému, se nazývá **pruning**.

Čím více větví stromu tímto způsobem odstraníme, tím lépe.



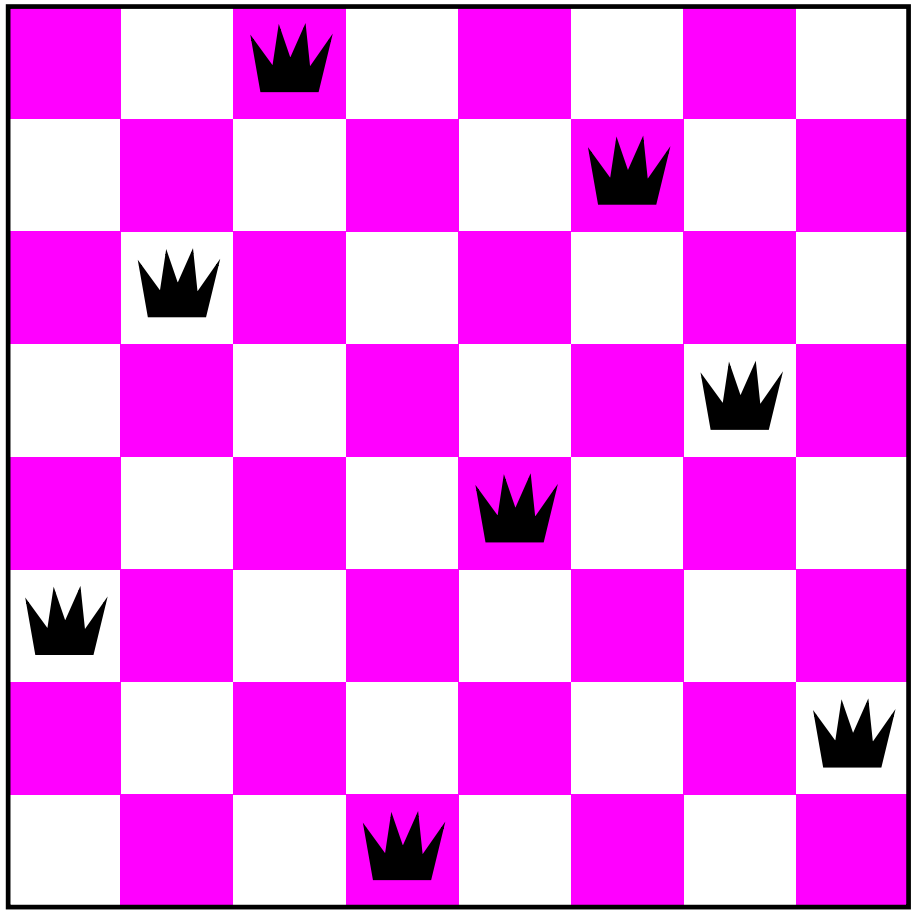
Problém osmi dam



Problém: Najít všechny možnosti, jak rozmístit n dam na šachovnici velikosti $n \times n$ tak, aby se žádné dvě dámy navzájem neohrožovaly.

Problém osmi dam

Jedno z možných řešení.



Problém osmi dam

Vstupem je číslo n.

Algoritmus bude používat následující pole:

- Y — s indexy $0 \,..\, n-1$, hodnota $Y[i]$ udává pro dámu ve sloupci i číslo řádku, na kterém se nachází
- A — s indexy $0 \,..\, n-1$, booleovská hodnota $A[j]$ udává, jestli je řádek j obsazený
- B — s indexy $0 \,..\, 2n-2$, booleovská hodnota $B[k]$ udává, jestli je diagonála k ve směru $\nearrow$ obsazená
- C — s indexy $-(n-1) \,..\, +(n-1)$, booleovská hodnota $C[k]$ udává, jestli je diagonála k ve směru $\searrow$ obsazená

Prohledávání se spustí zavoláním $\textsc{Search}\,(0)$.

Problém osmi dam

Algoritmus: Rozmístění n dam na šachovnici

```
SEACH (k):
    if k = n then
        PRINT-SOLUTION ()
        return
    for i := 0 to n − 1 do
        i_2 := k + i; i_3 := k − i
        if ¬A[i] and ¬B[i_2] and ¬C[i_3] then
            Y[k] := i
            A[i] := TRUE; B[i_2] := TRUE; C[i_3] := TRUE
            SEARCH (k + 1)
            A[i] := FALSE; B[i_2] := FALSE; C[i_3] := FALSE
```

Problém osmi dam

	0	1	2	3
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5
3	3	4	5	6

x+y

	0	1	2	3
0	0	1	2	3
1	−1	0	1	2
2	−2	−1	0	1
3	−3	−2	−1	0

x−y

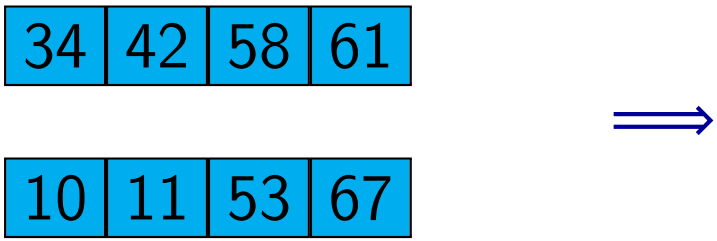
Rekurzivní algoritmy

- Uvedený algoritmus pro rozmístění n dam na šachovnici je příkladem **prohledávání s návratem** (**backtracking**).
- I při použití pruningu má tento typ algoritmů většinou exponenciální složitost.
- Obecně rekurzivní algoritmy, které převádí řešení problému velikosti n na dva nebo více problémů velikosti $n-1$, mívají většinou exponenciální složitost.
- Pokud rekurzivní algoritmus převádí řešení problému velikosti n na řešení problémů velikosti $n/2$, složitost může být (a často bývá) polynomiální.

  Tento postup může někdy vést k řešením, která mohou být efektivnější než nějaké přímočaré řešení.

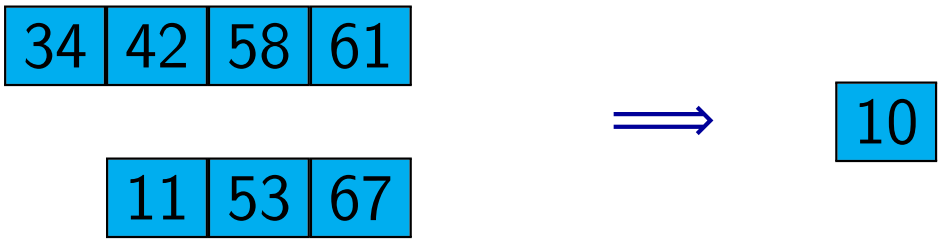
Příklad: Algoritmus MERGE-SORT.

Hlavní myšlenka algoritmu: Dvě setříděné posloupnosti snadno spojíme do jediné setříděné posloupnosti.
Pokud mají obě posloupnosti dohromady n prvků, vyžaduje tato operace n kroků.



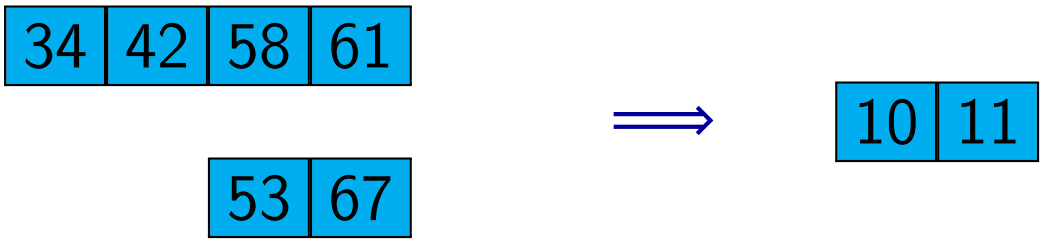
Příklad: Algoritmus MERGE-SORT.

Hlavní myšlenka algoritmu: Dvě setříděné posloupnosti snadno spojíme do jediné setříděné posloupnosti.
Pokud mají obě posloupnosti dohromady n prvků, vyžaduje tato operace n kroků.



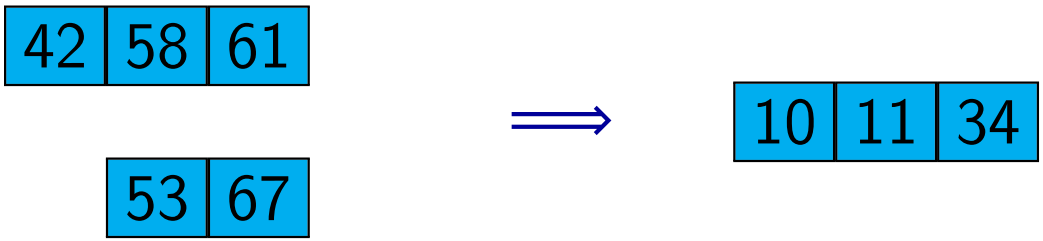
Příklad: Algoritmus MERGE-SORT.

Hlavní myšlenka algoritmu: Dvě setříděné posloupnosti snadno spojíme do jediné setříděné posloupnosti.
Pokud mají obě posloupnosti dohromady n prvků, vyžaduje tato operace n kroků.



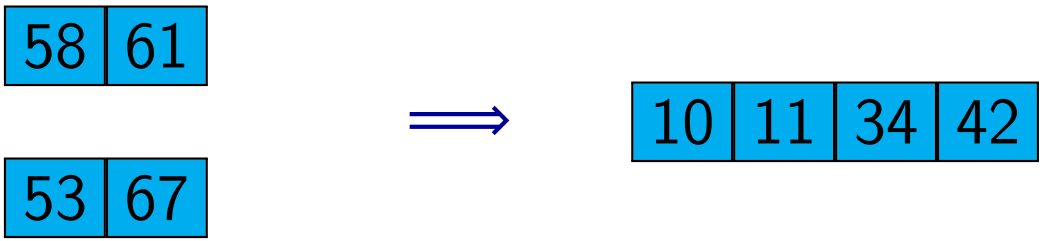
Příklad: Algoritmus MERGE-SORT.

Hlavní myšlenka algoritmu: Dvě setříděné posloupnosti snadno spojíme do jediné setříděné posloupnosti.
Pokud mají obě posloupnosti dohromady n prvků, vyžaduje tato operace n kroků.



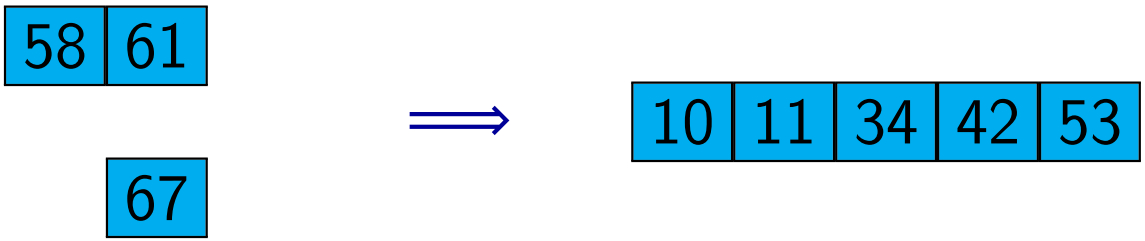
Příklad: Algoritmus MERGE-SORT.

Hlavní myšlenka algoritmu: Dvě setříděné posloupnosti snadno spojíme do jediné setříděné posloupnosti.
Pokud mají obě posloupnosti dohromady n prvků, vyžaduje tato operace n kroků.



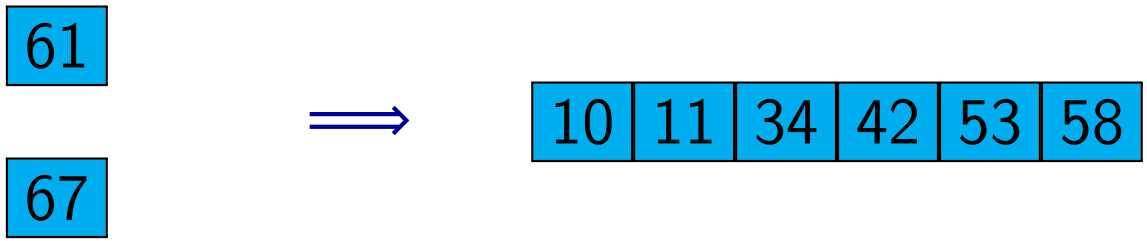
Příklad: Algoritmus MERGE-SORT.

Hlavní myšlenka algoritmu: Dvě setříděné posloupnosti snadno spojíme do jediné setříděné posloupnosti.
Pokud mají obě posloupnosti dohromady n prvků, vyžaduje tato operace n kroků.



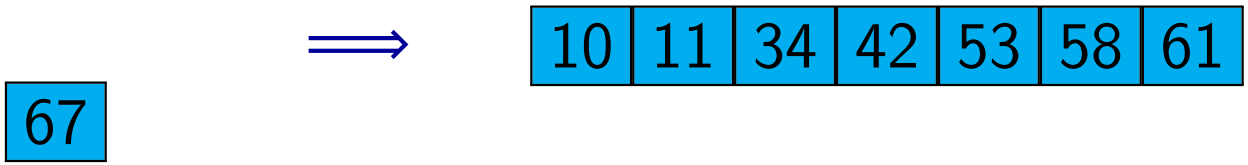
Příklad: Algoritmus MERGE-SORT.

Hlavní myšlenka algoritmu: Dvě setříděné posloupnosti snadno spojíme do jediné setříděné posloupnosti.
Pokud mají obě posloupnosti dohromady n prvků, vyžaduje tato operace n kroků.



Příklad: Algoritmus MERGE-SORT.

Hlavní myšlenka algoritmu: Dvě setříděné posloupnosti snadno spojíme do jediné setříděné posloupnosti.
Pokud mají obě posloupnosti dohromady n prvků, vyžaduje tato operace n kroků.



Příklad: Algoritmus MERGE-SORT.

Hlavní myšlenka algoritmu: Dvě setříděné posloupnosti snadno spojíme do jediné setříděné posloupnosti.
Pokud mají obě posloupnosti dohromady n prvků, vyžaduje tato operace n kroků.

Složitost algoritmů

Algoritmus: Merge sort

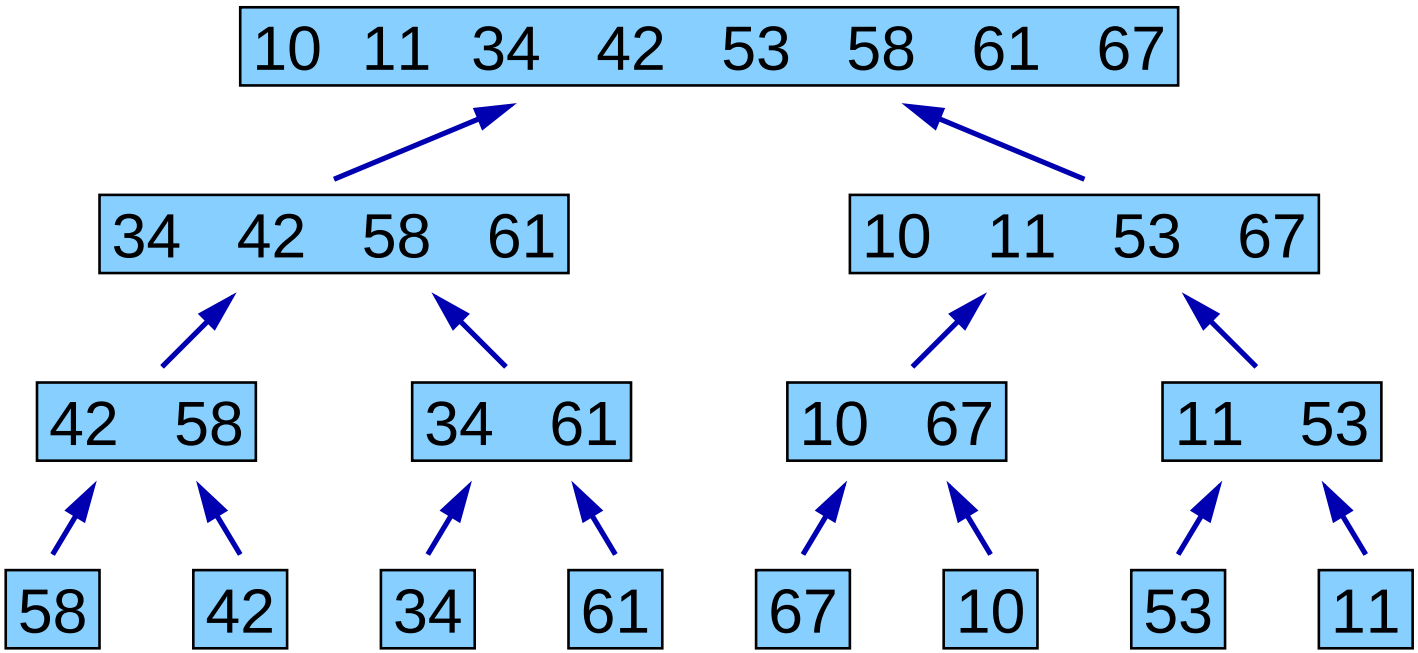
```
MERGE-SORT(A, p, r):
    if r − p > 1 then
        q := ⌊(p + r) / 2⌋
        MERGE-SORT(A, p, q)
        MERGE-SORT(A, q, r)
        MERGE(A, p, q, r)
```

Pro setřídění pole A, které obsahuje prvky $A[0], A[1], \cdots, A[n-1]$, zavoláme MERGE-SORT$(A, 0, n)$.

Poznámka: Procedura MERGE(A, p, q, r) spojí setříděné posloupnosti uložené v $A[p \,..\, q-1]$ a $A[q \,..\, r-1]$ do jedné posloupnosti uložené v $A[p \,..\, r-1]$.

Složitost algoritmů

Vstup: 58, 42, 34, 61, 67, 10, 53, 11



Strom rekurzivních volání má $\Theta(\log n)$ úrovní. Na každé úrovni se provede $\Theta(n)$ operací. Časová složitost algoritmu MERGE-SORT je $\Theta(n \log n)$.

The master theorem

Master theorem

Předpokládejme, že $a \geq 1$ a $b > 1$ jsou konstanty, že $f(n)$ je funkce a že funkce $T(n)$ je definována rekurentním předpisem

$$T(n) = a \cdot T(n/b) + f(n)$$

(kde n/b může být buď $\lfloor n/b \rfloor$ nebo $\lceil n/b \rceil$). Pak platí:

a. Pokud $f(n) \in O(n^{\log_b a - \epsilon})$ pro nějakou konstantu $\epsilon > 0$, pak $T(n) = \Theta(n^{\log_b a})$.

b. Pokud $f(n) \in \Theta(n^{\log_b a})$, pak $T(n) = \Theta(n^{\log_b a} \log n)$.

c. Pokud $f(n) \in \Omega(n^{\log_b a + \epsilon})$ pro nějakou konstantu $\epsilon > 0$ a pokud $a \cdot f(n/b) \leq c \cdot f(n)$ pro nějakou konstantu $c < 1$ a všechna dostatečně velká n, pak $T(n) = \Theta(f(n))$.

The master theorem

Master theorem je možné použít pro analýzu složitosti libovolného rekurzivního algoritmu, kde:

- Řešení jednoho podproblému velikosti n, kde $n > 1$, se převede na řešení a podproblémů, z nichž každy má velikost n/b.
- Doba, která stráví řešením jednoho podproblému velikosti n, nepočítaje v to dobu, která se stráví v rekurzivních voláních, je určená funkcí $f(n)$.

Příklad: Algoritmus MERGE-SORT: $a = 2$, $b = 2$, $f(n) \in \Theta(n)$

(v rámci jednoho volání — dva podproblémy, každý velikosti $n/2$, spojení dvou setříděných sekvencí v čase $\Theta(n)$)

Platí $f(n) \in \Theta(n^{\log_b a}) = \Theta(n)$, takže

$$T(n) \in \Theta(n^{\log_b a} \log n) = \Theta(n \log n).$$

The master theorem

Příklad: Násobení čtvercových matic A a B velikosti $n \times n$ rekurzivním způsobem:

- Pro $n = 1$ se výsledek spočítá přímo.
- Pro $n > 1$ se každá z matic A a B rozloží na čtyři podmatice velikosti $(n/2) \times (n/2)$.
- Výsledek se poskládá pomocí sčítání a násobení těchto osmi menších matic. Pro násobení těchto menších matic se funkce zavolá rekurzivně.
- Přímočarý způsob vyžaduje 8 násobení matic velikosti $(n/2) \times (n/2)$.

  Máme tedy $a = 8$, $b = 2$, $f(n) \in \Theta(n^2)$.

  Platí $f(n) \in O(n^{\log_b a - \epsilon})$, protože $n^2 \in O(n^{\log_2 8 - \epsilon}) = O(n^{3-\epsilon})$ platí např. pro $\epsilon = 1$.

  Takže $T(n) \in \Theta(n^{\log_b a}) = \Theta(n^{\log_2 8}) = \Theta(n^3)$.

Tento postup tedy není lepší než standardní jednoduchý algoritmus pro násobení matic.

The master theorem

Existuje však chytrý způsob, jak výše uvedené provést komplikovanějším způsobem tak, že v rámci jednoho rekurzivního volání postačí rekurzivně volat funkci 7 krát
(za cenu většího počtu sčítání a odčítání).

Jedná se o tzv. **Strassenův algoritmus**.

Zde je $a = 7$, $b = 2$ a $f(n) \in \Theta(n^2)$.

Opět platí $f(n) \in O(n^{\log_b a - \epsilon})$, protože $n^2 \in O(n^{\log_2 7 - \epsilon})$ platí např. pro $\epsilon = 0.5$.

($\log_2 7$ je přibližně 2.80735)

Takže $T(n) \in \Theta(n^{\log_b a}) = \Theta(n^{\log_2 7})$ a tedy $T(n) \in O(n^{2.81})$.

The master theorem

Důkaz master theoremu:

Pro jednoduchost se omezíme jen na případy, kdy $f(n) = n^c$ pro nějakou konstantu $c > 0$.

Rovněž pro jednoduchost předpokládejme, že n je mocninou čísla b, ať nemusíme řešit zaokrouhlování.

Představme si strom rekurzivních volání pro instanci velikosti n:

- Výška stromu je $\log_b n$.
- Počty vrcholů na jednotlivých úrovních jsou $a^0, a^1, \ldots, a^{\log_b n}$
- Čas, který se stráví v jednom vrcholu na úrovni i je

$$f\left(\frac{n}{b^i}\right) = \left(\frac{n}{b^i}\right)^c$$

The master theorem

Platí tedy

$$T(n) = \sum_{i=0}^{\log_b n} a^i \cdot f\left(\frac{n}{b^i}\right) = \sum_{i=0}^{\log_b n} a^i \cdot \left(\frac{n}{b^i}\right)^c = n^c \cdot \sum_{i=0}^{\log_b n} \left(\frac{a}{b^c}\right)^i$$

Označme $q = a/b^c$. Je třeba rozlišit tři případy:

- $q > 1$ — tj. když platí $a > b^c$, neboli $c < \log_b a$
- $q = 1$ — tj. když platí $a = b^c$, neboli $c = \log_b a$
- $q < 1$ — tj. když platí $a < b^c$, neboli $c > \log_b a$

The master theorem

Případ $q > 1$ — tj. když platí $a > b^c$, neboli $c < \log_b a$:

$$T(n) = n^c \cdot \sum_{i=0}^{\log_b n} \left(\frac{a}{b^c}\right)^i = n^c \cdot \frac{q^{\log_b n+1} - 1}{q-1} \in \Theta(n^c \cdot q^{\log_b n})$$

Platí

$$n^c \cdot q^{\log_b n} = n^c \cdot \left(\frac{a}{b^c}\right)^{\log_b n} = n^c \cdot n^{\log_b\left(\frac{a}{b^c}\right)}$$

$$= n^c \cdot n^{\log_b a - \log_b(b^c)} = n^{c+\log_b a - c} = n^{\log_b a}$$

Platí tedy $T(n) \in \Theta(n^{\log_b a})$.

Poznámka: Počet listů stromů (tj. podproblémů velikosti 1) je $a^{\log_b n} = n^{\log_b a}$.

Většina času se tedy tráví řešením těchto elementárních případů.

The master theorem

Případ $q = 1$ — tj. když platí $a = b^c$, neboli $c = \log_b a$:

$$T(n) = n^c \cdot \sum_{i=0}^{\log_b n} \left(\frac{a}{b^c}\right)^i = n^c \cdot \sum_{i=0}^{\log_b n} 1 = n^c \cdot (\log_b n + 1) \in \Theta(n^{\log_b a} \log n)$$

Poznámka: V každé vrstvě stromu se stráví zhruba stejný čas $\Theta(n^{\log_b a})$.
Vrstev je celkem $\Theta(\log n)$.

The master theorem

Případ $q < 1$ — tj. když platí $a < b^c$, neboli $c > \log_b a$:

$$T(n) = n^c \cdot \sum_{i=0}^{\log_b n} \left(\frac{a}{b^c}\right)^i < n^c \cdot \sum_{i=0}^{\infty} \left(\frac{a}{b^c}\right)^i = n^c \cdot \frac{1}{1-q} \in O(n^c)$$

protože pro q, kde $0 < q < 1$, platí

$$\sum_{i=0}^{\infty} q^i = \lim_{z\to\infty} \sum_{i=0}^{z} q^i = \lim_{z\to\infty} \frac{q^{z+1}-1}{q-1} = \frac{1}{1-q}$$

Zjevně platí $T(n) \in \Omega(n^c)$ (protože už v samotném kořeni se stráví čas $\Theta(n^c)$), takže celkově platí $T(n) \in \Theta(n^c)$.

Poznámka: Většina času se v tomto případě stráví v kořeni stromu.

Dynamické programování

Definice

Slovo $v = a_1 a_2 \dots a_n$ nad abecedou Σ (kde všechna $a_i \in \Sigma$) je **podsekvencí** slova $w \in \Sigma^*$, jestliže existují slova $u_0, u_1, \dots, u_n \in \Sigma^*$ taková, že

$$w = u_0\, a_1\, u_1\, a_2\, u_2 \cdots u_{n-1}\, a_n\, u_n$$

Příklad: Slovo `bcba` je podsekvencí slova `abcbdcab`.

Nejdelší společná podsekvence (**longest common subsequence**) slov u a v je nejdelší slovo w, které je podsekvencí slova u a zároveň podsekvencí slova v.

Příklad: Nejdelší společnou podsekvencí slov `abcbdab` a `bdcaba` je slovo `bcba`.

Poznámka: Nejdelší společná podsekvence vždy existuje, ale ne vždy je dána jednoznačně — např. pro `aaabb` a `bbbaa` je to `aa` i `bb`.

Problém: Nejdelší společná podsekvence

Vstup: Slova u a v nad abecedou Σ.

Výstup: Nejdelší slovo w, které je podsekvencí slova u a zároveň podsekvencí slova v.

Předpokládejme, že:

- slova u a v jsou uložena v polích A a B indexovaných od jedné
- hodnoty m a n udávají délku slov u a v

Tj. pokud $u = a_1 a_2 \cdots a_m$ a $v = b_1 b_2 \cdots b_n$, tak:

- prvky $A[1], A[2], \ldots, A[m]$ obsahují symboly $a_1, a_2, \ldots, a_m$
- prvky $B[1], B[2], \ldots, B[n]$ obsahují symboly $b_1, b_2, \ldots, b_n$

Dynamické programování

Zaměřme se nejprve na problém, zjistit pro daná slova u a v, jaká je délka jejich nejdelší společné podsekvence.

Můžeme řešit rekurzivně podproblémy následujícího typu:

- LCS-LEN (i, j) — pro dané i a j, kde $0 \leq i \leq m$ a $0 \leq j \leq n$, vrátí délku nejdelší společné podsekvence prefixu slova u délky i a prefixu slova v délky j.

  Tj. LCS-LEN (i, j) vrátí délku nejdelší společné podsekvence slov uložených v

  $$A[1], A[2], \ldots, A[i] \qquad \text{a} \qquad B[1], B[2], \ldots, B[j]$$

Délku nejdelší společné podsekvence slov u a v pak můžeme zjistit pomocí LCS-LEN (m, n).

Dynamické programování

Rekurzivní řešení:

$$\textsc{Lcs-len}\,(i,j) = \begin{cases} 0 & \text{pokud } i = 0 \text{ nebo } j = 0 \\ \textsc{Lcs-len}\,(i-1,j-1)+1 & \text{pokud } A[i] = B[j] \\ \max(\textsc{Lcs-len}\,(i-1,j), & \\ \qquad \textsc{Lcs-len}\,(i,j-1)) & \text{pokud } A[i] \neq B[j] \end{cases}$$

Dynamické programování

Rekurzivní řešení:

$$\textsc{Lcs-len}\,(i,j) = \begin{cases} 0 & \text{pokud } i = 0 \text{ nebo } j = 0 \\ \textsc{Lcs-len}\,(i-1,j-1)+1 & \text{pokud } A[i] = B[j] \\ \max(\textsc{Lcs-len}\,(i-1,j), & \\ \quad \textsc{Lcs-len}\,(i,j-1)) & \text{pokud } A[i] \neq B[j] \end{cases}$$

Toto řešení má očividně exponenciální časovou složitost.

Ve skutečnosti potřebujeme řešit jen $(m+1)\cdot(n+1)$ **různých** podproblémů (protože $i \in \{0,1,\ldots,m\}$ a $j \in \{0,1,\ldots,n\}$).

Výsledky řešení jednotlivých podproblémů si můžeme ukládat do tabulky a nemusíme je řešit opakovaně.

Dynamické programování

Algoritmus: Nalezení nejdelší společné podsekvence — vyplnění tabulky

```
LCS-COMP (A, m, B, n):
    for i := 0 to m do C[i][0] := 0
    for j := 1 to n do C[0][j] := 0
    for i := 1 to m do
        for j := 1 to n do
            if A[i] = B[j] then
                C[i][j] := C[i − 1][j − 1] + 1; D[i][j] := "↖"
            else
                if C[i − 1][j] ≤ C[i][j − 1] then
                    C[i][j] := C[i − 1][j]; D[i][j] := "↑"
                else
                    C[i][j] := C[i][j − 1]; D[i][j] := "←"
```

Složitost algoritmu je $O(m \cdot n)$.

Dynamické programování

i	j	0	1 b	2 d	3 c	4 a	5 b	6 a
0		0	0	0	0	0	0	0
1	a	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	b	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	c	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	b	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	d	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	a	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	b	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 3	↑ 4

Dynamické programování

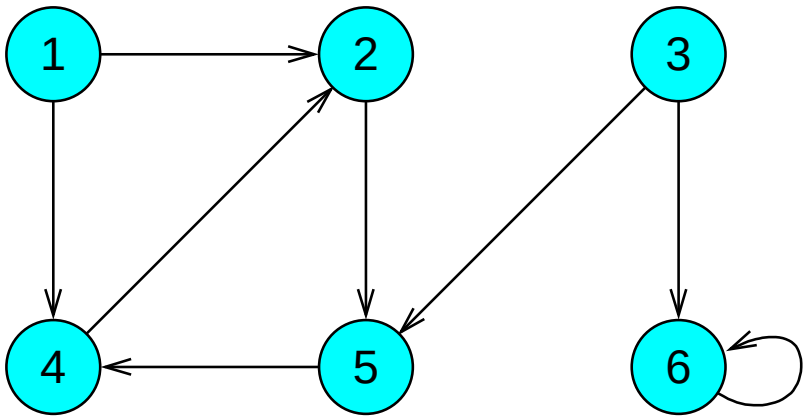
Dynamické programování:

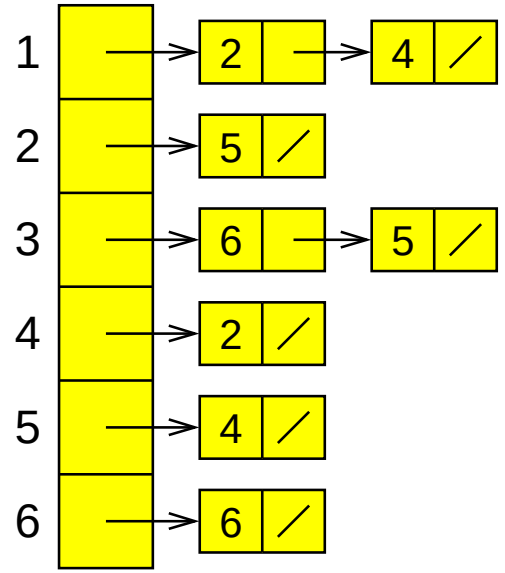
- Pokud máme rekurzivní řešení, kde se opakovaně mnohokrát řeší stejné instance podproblémů, je vhodné ukládat řešení podproblémů do nějaké datové struktury.
- Jedna možnost je ponechat rekurzivní řešení a značit si, které podproblémy již byly vyřešeny.
- Jiná možnost je systematické řešení všech podproblémů ve vhodném pořadí (od nejmenších po největší).

  Při řešení daného podproblému jsou rekurzivní volání nahrazena přečtením již dříve uložených řešení.

Reprezentace grafů

Reprezentace grafu:

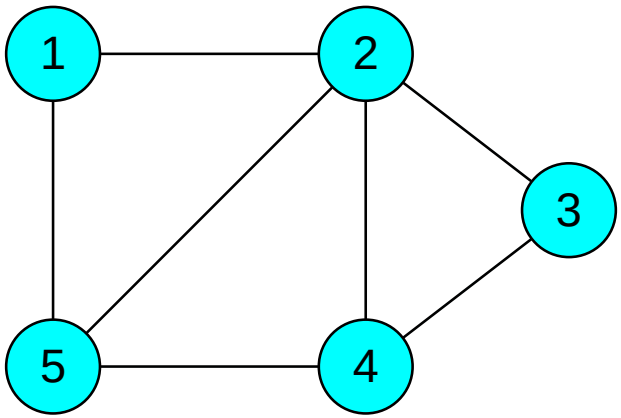


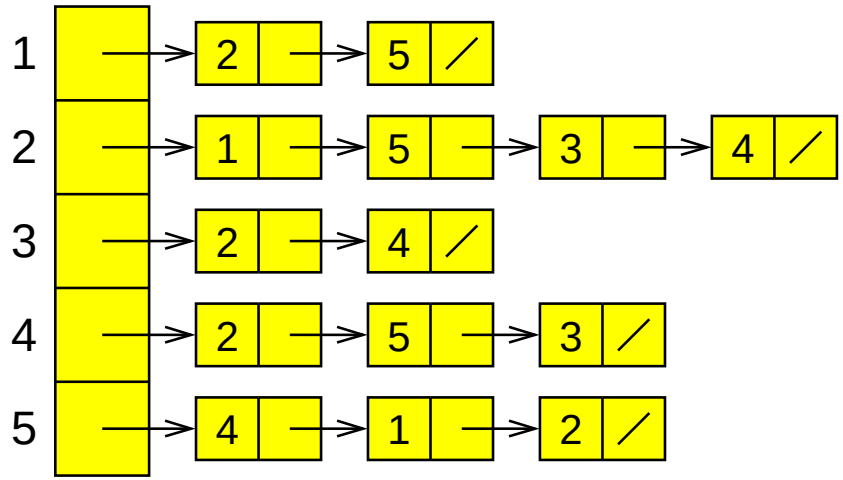


	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Reprezentace grafů

Reprezentace grafu:

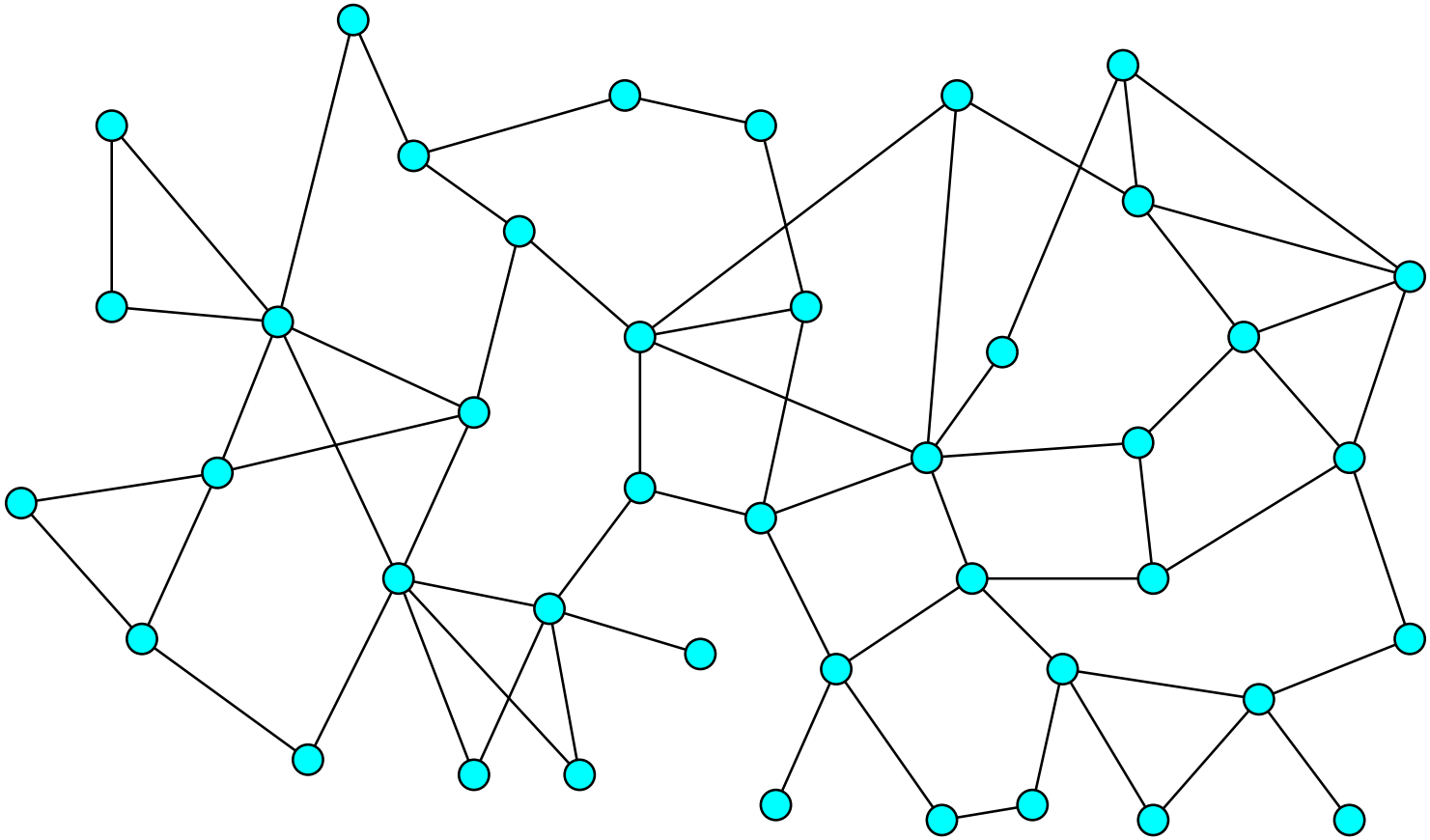




	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

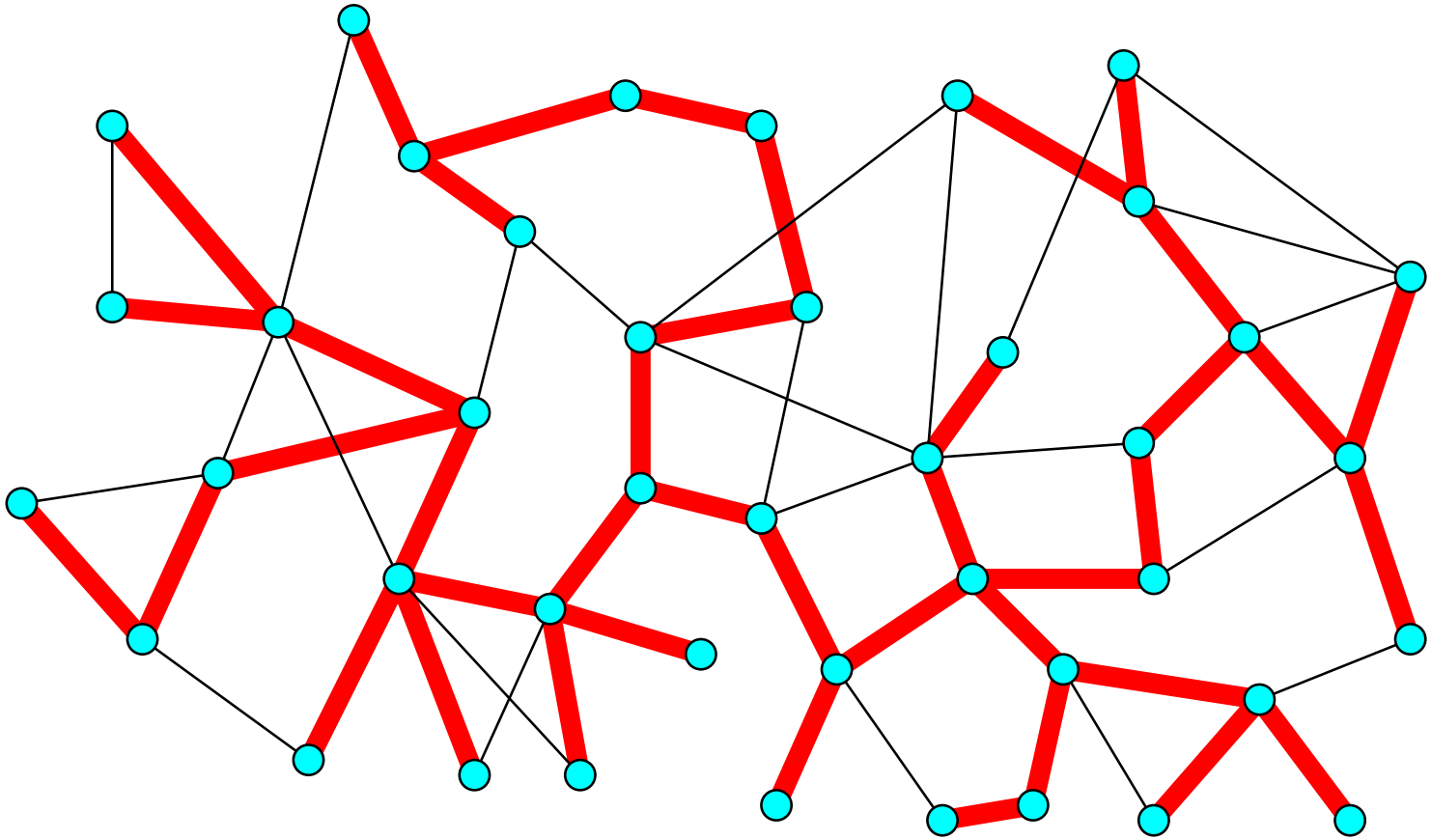
Minimální kostra grafu

Kostra grafu — souvislý podgraf grafu, který obsahuje všechny vrcholy a neobsahuje žádné cykly



Minimální kostra grafu

Kostra grafu — souvislý podgraf grafu, který obsahuje všechny vrcholy a neobsahuje žádné cykly



Minimální kostra grafu

Uvažujme neorientovaný graf $G = (V, E)$, kde navíc máme dány váhy hran, tj. funkci $w : E \to \mathbb{R}_+$, přiřazující každé hraně její váhu.

Pokud T je podmnožina hran (tj. $T \subseteq E$), můžeme funkci w rozšířit na tuto podmnožinu:

$$w(T) = \sum_{e \in T} w(e)$$

Kostra grafu G je dána takovou množinou hran T (kde $T \subseteq E$), která splňuje to, že graf (V, T) je souvislý a neobsahuje žádný cyklus.

Minimální kostra T je taková kostra grafu G, kde pro libovolnou jinou kostru grafu T' platí

$$w(T) \leq w(T')$$

Minimální kostra grafu

Problém: Minimální kostra grafu

Vstup: Souvislý neorientovaný graf $G = (V, E)$ s ohodnocením hran $w : E \to \mathbb{R}_+$.

Výstup: Některá minimální kostra grafu G.

Algoritmus, který by systematicky zkoušel všechny možné kostry, má očividně exponenciální složitost.

Efektivní algoritmy pro tento problém jsou založeny na tzv. **greedy** (**hltavém, hladovém**) řešení:

- na základě nějakého lokálního kritéria vybrat z mnoha možností jen jednu a nezkoušet všechny možnosti

  — z mnoha hran, které je možné do kostry přidat, vybrat vždy hranu s co nejmenší váhou, která nevytvoří cyklus, a tu přidat

Minimální kostra grafu

Kruskalův algoritmus:

- Setřídit hrany podle váhy od nejmenší po největší.
- $T := \emptyset$
- Probírat hrany v daném setříděném pořadí.

  Pro každou hranu e otestovat, zda jejím přidáním do T nevznikne cyklus, pokud ne, nastavit

  $$T := T \cup \{e\}$$
- Vrátit T jako výsledek.

Výpočetní složitost závisí na tom, jak je konkrétně implementováno testování toho, že nevznikne cyklus:

- Přímočaré řešení má složitost $O(n \cdot m)$.
- Existuje efektivní řešení se složitostí $O(m \log n)$.

Minimální kostra grafu

U **greedy algoritmů** je obecně většinou nesložitější částí návrhu algoritmu **důkaz korektnosti** — zdůvodnění toho lokálně optimální volby skutečně vždy vedou ke globálně optimálnímu řešení.

U Kruskalova algoritmu může být důkaz založen na tom, že se udržuje následující invariant:

Aktuální množina T je podmnožinou hran nějaké minimální kostry T_0.

Minimální kostra grafu

- Řekněme, že T je podmnožinou minimální kostry T_0, a algoritmus v následujícím kroku přidá hranu e takovou, že $T \cup \{e\}$ není podmnožinou hran žádné minimální kostry.
- Přidáním hrany e do T_0 vznikne cyklus.

  Tento cyklus musí obsahovat nějakou hranu e' takovou, že $e' \notin T$ (jinak by přidáním e do T vznikl cyklus).

  Navíc musí platit $w(e) \leq w(e')$ (jinak by algoritmus nevybral hranu e).
- Množina $T_0' = (T_0 - \{e'\}) \cup \{e\}$ je rovněž kostra.

  Navíc zjevně platí $w(T_0') \leq w(T_0)$, takže musí platit $w(T_0') = w(T_0)$ (jinak by kostra T_0 nebyla minimální).
- Kostra T_0' je tedy minimální a platí $T \cup \{e\} \subseteq T_0'$, což je ve sporu s předpokladem, že $T \cup \{e\}$ není podmnožinou hran žádné minimální kostry.

Prohledávání do šířky

Nalezení nejkratší cesty v grafu, kde hrany nejsou ohodnoceny:

- Algoritmus pro **prohledávání grafu do šířky**
- Vstupem je graf G (s množinou vrcholů V) a počáteční vrchol s.
- Algoritmus pro všechny vrcholy najde nejkratší cestu z vrcholu s.
- Pro graf, který má n vrcholů a m hran je doba výpočtu tohoto algoritmu $\Theta(n + m)$.

Prohledávání do šířky

Algoritmus: Prohledávání do šířky

```
BFS(G, s):
    BFS-INIT(G, s)
    ENQUEUE(Q, s)
    while Q ≠ ∅ do
        u := DEQUEUE(Q)
        for each v ∈ edges[u] do
            if color[v] = WHITE then
                color[v] := GRAY
                d[v] := d[u] + 1
                pred[v] := u
                ENQUEUE(Q, v)
        color[u] := BLACK
```

Prohledávání do šířky

Algoritmus: Prohledávání do šířky — inicializace

BFS-INIT (G, s):

```
for each u ∈ V − {s} do
    color[u] := WHITE
    d[u] := ∞
    pred[u] := NIL
color[s] := GRAY
d[s] := 0
pred[s] := NIL
Q := ∅
```

Nerozhodnutelné problémy

Algoritmicky řešitelné problémy

Předpokládejme, že máme dán nějaký problém P.

Jestliže existuje nějaký algoritmus, který řeší problém P, pak říkáme, že problém P je **algoritmicky řešitelný**.

Jestliže P je rozhodovací problém a jestliže existuje nějaký algoritmus, který problém P řeší, pak říkáme, že problém P je **(algoritmicky) rozhodnutelný**.

Když chceme ukázat, že problém P je algoritmicky řešitelný, stačí ukázat nějaký algoritmus, který ho řeší (a případně ukázat, že daný algoritmus problém P skutečně řeší).

Algoritmicky neřešitelné problémy

Problém, který není algoritmicky řešitelný, je **algoritmicky neřešitelný**.

Rozhodovací problém, který není rozhodnutelný, je **nerozhodnutelný**.

Kupodivu existuje řada algoritmických problémů (přesně definovaných), o kterých je dokázáno, že nejsou algoritmicky řešitelné.

Halting Problem

Vezměme si nějaký libovolný obecný programovací jazyk $\mathcal{L}$.

Navíc předpokládejme, že programy v jazyce $\mathcal{L}$ běží na nějakém idealizovaném stroji, kde mají k dispozici (potenciálně) neomezené množství paměti — tj. kde alokace paměti nikdy neselže kvůli nedostatku paměti.

Příklad: Následující problém zvaný **Problém zastavení (Halting problem)** je nerozhodnutelný:

Halting problem

Vstup: Zdrojový kód programu P v jazyce $\mathcal{L}$, vstupní data x.

Otázka: Zastaví se program P po nějakém konečném počtu kroků, pokud dostane jako vstup data x?

Halting Problem

Předpokládejme, že by existoval nějaký program, který by rozhodoval Halting problem.

Mohli bychom tedy vytvořit podprogram H, deklarovaný jako

Bool H(String kod, String vstup)

kde $H(P, x)$ vrátí:

- true pokud se program P zastaví pro vstup x,
- false pokud se program P nezastaví pro vstup x.

Poznámka: Řekněme, že podprogram $H(P, x)$ by vracel false v případě, že P není syntakticky správný kód programu.

Halting Problem

S použitím podprogramu H bychom vytvořili program D, který bude provádět následující kroky:

- Načte svůj vstup do proměnné x typu String.
- Zavolá podprogram $H(x, x)$.
- Pokud podprogram H vrátil true, skočí do nekonečné smyčky

```
loop: goto loop
```

V případě, že H vrátil false, program D se ukončí.

Co udělá program D, pokud mu předložíme jako vstup jeho vlastní kód?

Halting Problem

Pokud D dostane jako vstup svůj vlastní kód, tak se buď zastaví nebo nezastaví.

- Pokud se D zastaví, tak $H(D, D)$ vrátí true a D skočí do nekonečné smyčky. Spor!
- Pokud se D nezastaví, tak $H(D, D)$ vrátí false a D se zastaví. Spor!

V obou případech dospějeme ke sporu a další možnost není. Nemůže tedy platit předpoklad, že H řeší Halting problem.

Částečně rozhodnutelné problémy

Problém je **částečně rozhodnutelný**, jestliže existuje algoritmus, který:

- Pokud dostane jako vstup instanci, pro kterou je odpověď ANO, tak se po konečném počtu kroků zastaví a vypíše `"ANO"`.
- Pokud dostane jako vstup instanci, pro kterou je odpověď NE, tak se buď zastaví a vypíše `"NE"` nebo se nikdy nezastaví.

Je očividné, že například HP (Halting problem) je částečně rozhodnutelný.

Některé problémy však nejsou ani částečně rozhodnutelné.

Postova věta

Doplňkový problém k danému rozhodovacímu problému P je problém, kde vstupy jsou stejné jako u problému P a otázka je negací otázky z problému P.

Postova věta

Jestliže problém P i jeho doplňkový problém jsou částečně rozhodnutelné, pak je problém P rozhodnutelný.

Převody mezi problémy

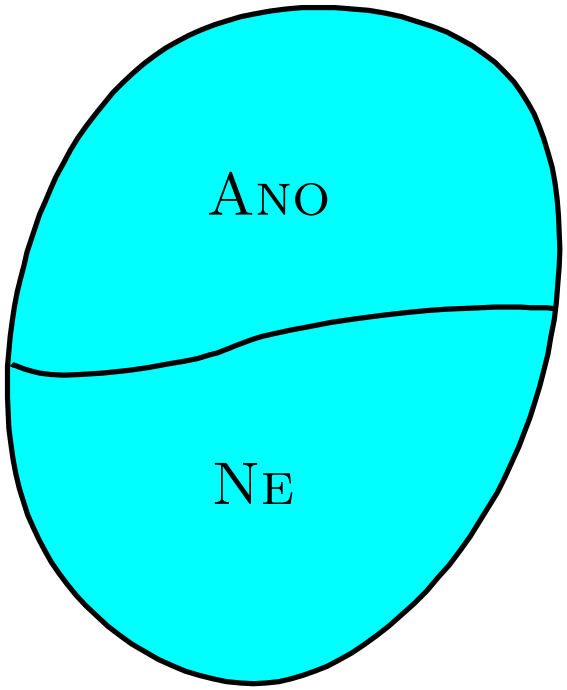
Pokud máme o nějakém (rozhodovacím) problému dokázáno, že je nerozhodnutelný, můžeme ukázat nerozhodnutelnost dalších problémů pomocí redukcí (převodů) mezi problémy.

Problém P_1 je **převeditelný** na problém P_2, jestliže existuje algoritmus Alg takový, že:

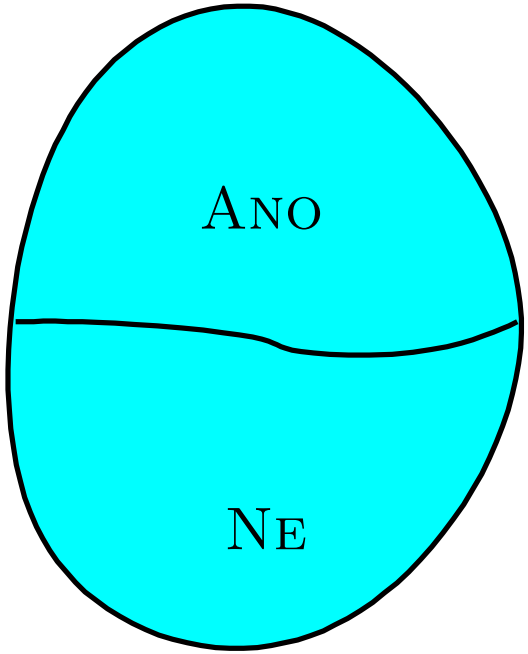
- Jako vstup může dostat libovolnou instanci problému P_1.
- K instanci problému P_1, kterou dostane jako vstup (označme ji w), vyprodukuje jako svůj výstup instanci problému P_2 (označme ji $Alg(w)$).
- Platí, že pro vstup w je v problému P_1 odpověď ANO právě tehdy, když pro vstup $Alg(w)$ je v problému P_2 odpověď ANO.

Převody mezi problémy

vstupy problému P_1



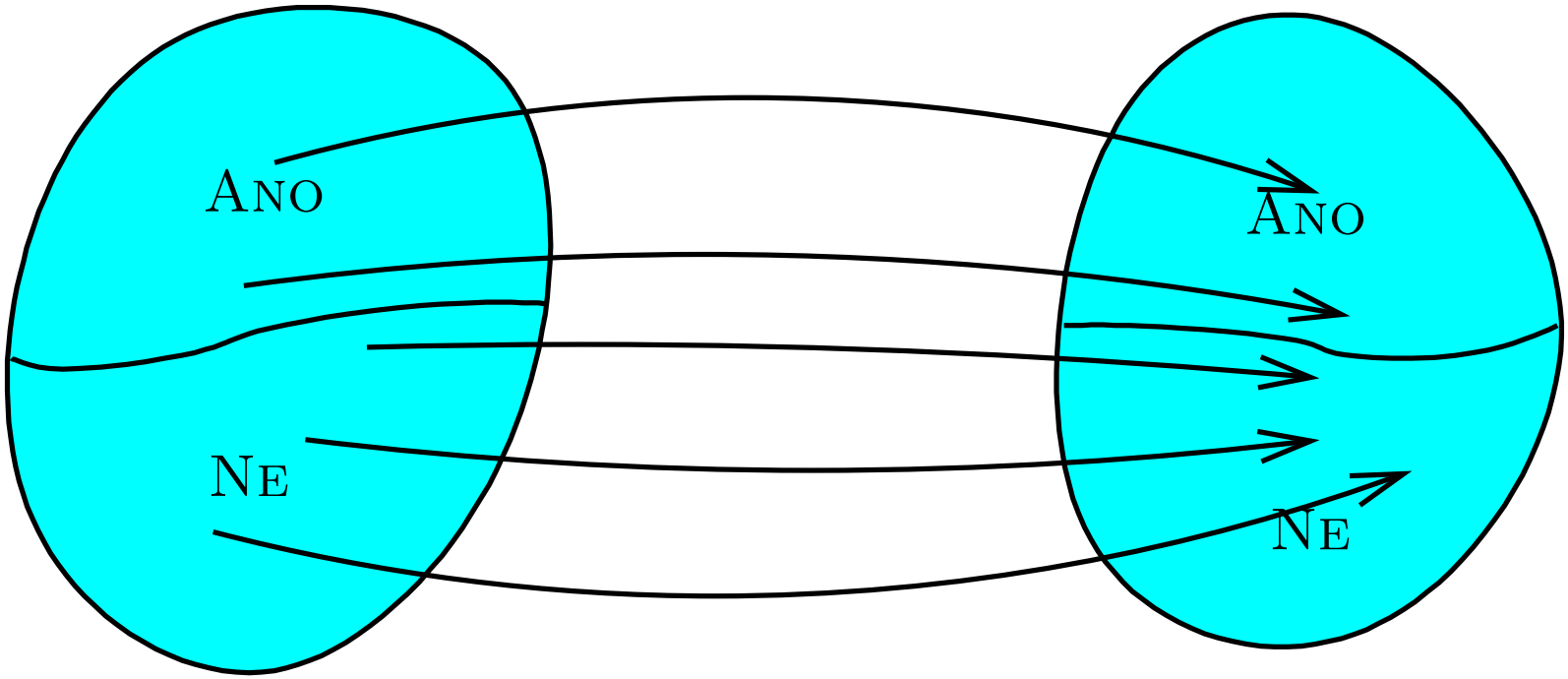
vstupy problému P_2



Převody mezi problémy

vstupy problému P_1

vstupy problému P_2



Alg

Převody mezi problémy

Řekněme, že existuje redukce Alg problému P_1 na problém P_2.

Pokud by problém P_2 byl rozhodnutelný, pak i problém P_1 je rozhodnutelný.

Řešení problému P_1 pro vstup x:

- Zavoláme Alg se vstupem x, vrátí nám hodnotu $Alg(x)$.
- Zavoláme algoritmus řešící problém P_2 se vstupem $Alg(x)$.
- Hodnotu, kterou nám vrátí vypíšeme jako výsledek.

Je zřejmé, že pokud P_1 je nerozhodnutelný, tak P_2 nemůže být rozhodnutelný.

Další nerozhodnutelné problémy

Redukcí z Halting problému se dá ukázat nerozhodnutelnost celé řady problémů, které se týkají ověřování chování programů:

- Vydá daný program pro nějaký vstup odpověď ANO?
- Zastaví se daný program pro libovolný vstup?
- Dávají dva dané programy pro stejné vstupy stejný výstup?
- ...

Halting problem

Pro účely důkazů se Halting problem nejčastěji používá v následující podobě:

Halting problem

Vstup: Popis Turingova stroje $\mathcal{M}$ a slovo w.

Otázka: Zastaví se stroj $\mathcal{M}$ po nějakém konečném počtu kroků, pokud dostane jako svůj vstup slovo w?

Další nerozhodnutelné problémy

S následujícím příkladem nerozhodnutelného problému už jsme se setkali:

Problém

Vstup: Bezkontextové gramatiky $\mathcal{G}_1$ a $\mathcal{G}_2$.

Otázka: Je $\mathcal{L}(\mathcal{G}_1) = \mathcal{L}(\mathcal{G}_2)$?

případně

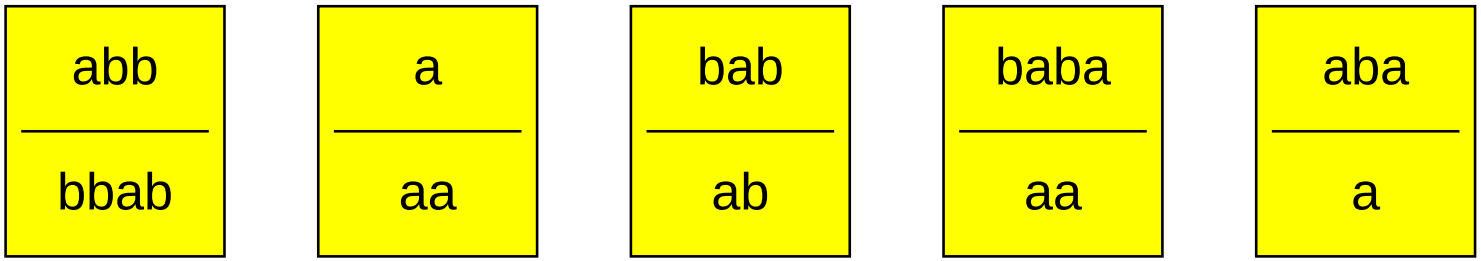
Problém

Vstup: Bezkontextová gramatika $\mathcal{G}$ generující jazyk nad abecedou Σ.

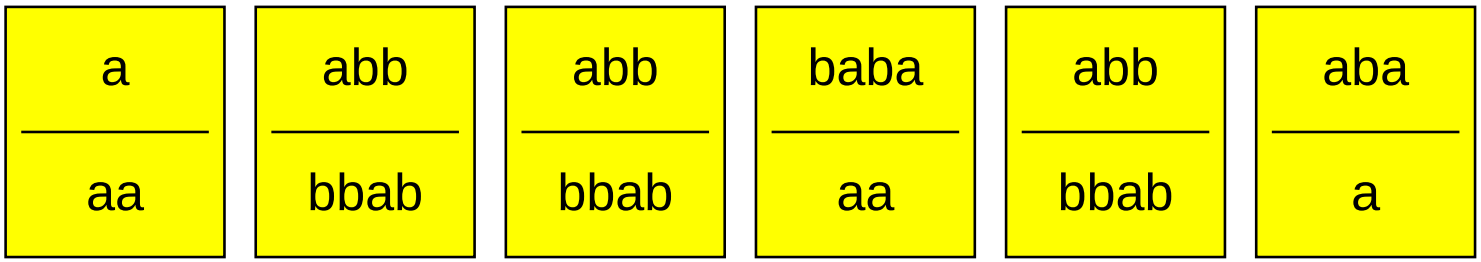
Otázka: Je $\mathcal{L}(\mathcal{G}) = \Sigma^*$?

Další nerozhodnutelné problémy

Vstupem je množina typů kartiček, jako třeba:



Otázka je, zda je možné z těchto typů kartiček vytvořit neprázdnou konečnou posloupnost, kde zřetězením slov nahoře i dole vznikne totéž slovo. Každý typ kartičky je možné používat opakovaně.



Nahoře i dole vznikne slovo `aabbabbbabaabbaba`.

Další nerozhodnutelné problémy

Redukcí z předchozího problému se dá snadno ukázat nerozhodnutelnost některých dalších problémů z oblasti bezkontextových gramatik:

Problém

Vstup: Bezkontextové gramatiky $\mathcal{G}_1$ a $\mathcal{G}_2$.

Otázka: Je $\mathcal{L}(\mathcal{G}_1) \cap \mathcal{L}(\mathcal{G}_2) = \varnothing$?

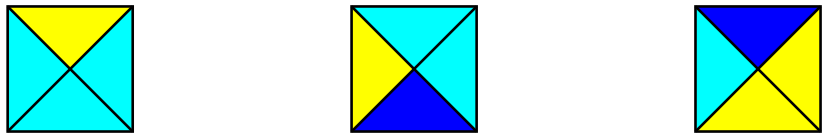
Problém

Vstup: Bezkontextová gramatika $\mathcal{G}$.

Otázka: Je $\mathcal{G}$ nejednoznačná?

Další nerozhodnutelné problémy

Vstupem je množina typů kachliček, jako třeba:

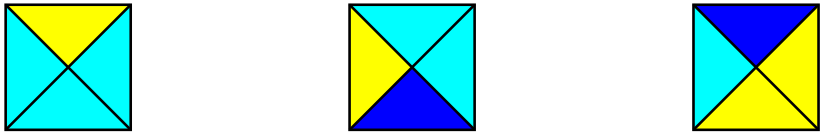


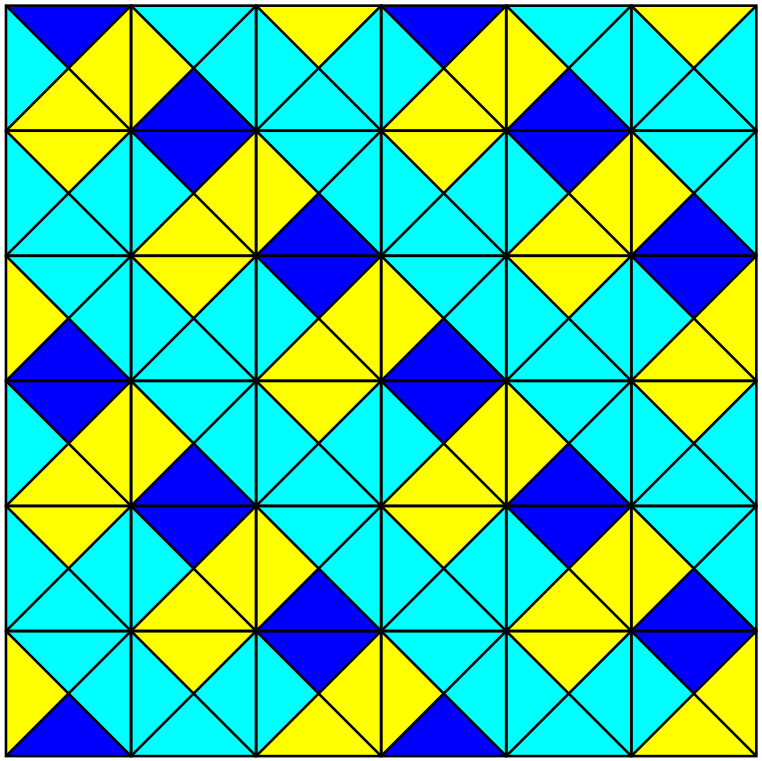
Otázka je, zda je možné použitím daných typů kachliček pokrýt každou libovolně velkou konečnou plochu tak, aby všechny kachličky spolu sousedily stejnými barvami.

Poznámka: Můžeme předpokládat, že máme v zásobě neomezené množství kachliček všech typů.

Kachličky není dovoleno otáčet.

Další nerozhodnutelné problémy





Další nerozhodnutelné problémy

Problém

Vstup: Uzavřená formule predikátové logiky (prvního řádu), ve které mohou být použity jako predikátové symboly pouze $=$ a $<$, jako funkční symboly pouze $+$ a $*$ a jako konstantní symboly pouze 0 a 1.

Otázka: Je daná formule pravdivá v oboru přirozených čísel (při přirozené interpretaci všech funkčních a predikátových symbolů)?

Příklad vstupu:

$$\forall x \exists y \forall z((x * y = z) \wedge (y + 1 = x))$$

Poznámka: Úzce souvisí s Gödelovou větou o neúplnosti.

Další nerozhodnutelné problémy

Je zajímavé, že analogický problém, kde ale místo přirozených čísel uvažujeme čísla reálná, je algoritmicky rozhodnutelný (i když popis daného algoritmu a důkaz jeho korektnosti jsou značně netriviální).

Rovněž pokud uvažujeme přirozená nebo celá čísla a stejné formule jako v předchozím případě, ale s tím, že v nich nesmí být použit funkční symbol $*$ (násobení), tak je problém algoritmicky rozhodnutelný.

Další nerozhodnutelné problémy

Pokud můžeme používat $*$, je ve skutečnosti nerozhodnutelný už velmi omezený případ:

Desátý Hilbertův problém

Vstup: Polynom $f(x_1, x_2, \ldots, x_n)$ vytvořený z proměnných $x_1, x_2, \ldots, x_n$ a celočíselných konstant.

Otázka: Existují přirozená čísla $x_1, x_2, \ldots, x_n$ taková, že $f(x_1, x_2, \ldots, x_n) = 0$?

Příklad vstupu: $5x^2y - 8yz + 3z^2 - 15$

Tj. ptáme se, zda

$$\exists x \exists y \exists z (5 * x * x * y + (-8) * y * z + 3 * z * z + (-15) = 0)$$

platí v oboru přirozených čísel.

Další nerozhodnutelné problémy

Také následující problém je algoritmicky nerozhodnutelný:

Problém

Vstup: Uzavřená formule φ predikátové logiky prvního řádu.

Otázka: Platí $\models \varphi$?

Poznámka: Zápis $\models \varphi$ znamená, že formule φ je logicky platná, tj. pravdivá v každé interpretaci.

Třídy složitosti

Složitost problémů

- Ukazuje se, že různé (algoritmické) problémy jsou různě těžké.
- Obtížnější jsou ty problémy, k jejichž řešení potřebujeme více času a paměti.
- Obtížnost problémů chceme nějak posuzovat, a to jak
  - absolutně – kolik času a kolik paměti potřebujeme k jejich řešení, tak
  - relativně – o kolik je jejich řešení obtížnější nebo naopak jednodušší oproti jiným problémům.
- Proč se u některých problémů nedaří nalézt efektivní algoritmy? Může vůbec nějaký efektivní algoritmus pro daný problém existovat?
- Kde přesně jsou limity toho, co je možné prakticky zvládnout?

Složitost problémů

Je potřeba rozlišovat **složitost algoritmu** a **složitost problému**.

Pokud například zkoumáme časovou složitost v nejhorším případě, mohli bychom neformálně říct:

- **složitost algoritmu** — funkce, která vyjadřuje, jaká bude pro daný algoritmus maximální doba výpočtu pro vstup velikosti n
- **složitost problému** — jaká je časová složitost „nejefektivnějšího“ algoritmu, který řeší daný problém

Zavedení pojmu „složitost problému“ ve výše uvedeném smyslu naráží na značné technické obtíže. Pojem „složitost problému“ se tedy jako takový nedefinuje, ale obchází se zavedením tzv. **tříd složitosti**.

Třídy složitosti

Třídy složitosti jsou podmnožiny množiny všech (algoritmických) **problémů**.

Daná konkrétní třída složitosti je vždy charakterizována nějakou vlastností, kterou mají problémy do ní patřící.

Typickým příkladem takové vlastnosti je vlastnost, že pro daný problém existuje nějaký algoritmus s určitým omezením (např. časové nebo prostorové složitosti):

- Do dané třídy pak patří všechny problémy, pro které takovýto algoritmus existuje.
- Naopak do ní nepatří problémy, pro které žádný takový algoritmus neexistuje.

Poznámka: V následujícím popisu se budeme soustředit prakticky jen na třídy **rozhodovacích** problémů.

Třídy složitosti

Definice

Pro libovolnou funkci $f : \mathbb{N} \to \mathbb{N}$ definujeme třídu $\mathcal{T}(f(n))$ jako třídu obsahující právě ty rozhodovací problémy, pro něž existuje algoritmus s časovou složitostí $O(f(n))$.

Příklad:

- $\mathcal{T}(n)$ – třída všech rozhodovacích problémů pro něž existuje algoritmus s časovou složitostí $O(n)$
- $\mathcal{T}(n^2)$ – třída všech rozhodovacích problémů pro než existuje algoritmus s časovou složitostí $O(n^2)$
- $\mathcal{T}(n \log n)$ – třída všech rozhodovacích problémů pro než existuje algoritmus s časovou složitostí $O(n \log n)$

Třídy složitosti

Definice

Pro libovolnou funkci $f : \mathbb{N} \to \mathbb{N}$ definujeme třídu $\mathcal{S}(f(n))$ jako třídu obsahující právě ty rozhodovací problémy, pro něž existuje algoritmus s prostorovou složitostí $O(f(n))$.

Příklad:

- $\mathcal{S}(n)$ – třída všech rozhodovacích problémů pro něž existuje algoritmus s prostorovou složitostí $O(n)$
- $\mathcal{S}(n^2)$ – třída všech rozhodovacích problémů pro než existuje algoritmus s prostorovou složitostí $O(n^2)$
- $\mathcal{S}(n \log n)$ – třída všech rozhodovacích problémů pro než existuje algoritmus s prostorovou složitostí $O(n \log n)$

Třídy složitosti

Poznámka:

Všimněte si, že u tříd $\mathcal{T}(f)$ a $\mathcal{S}(f)$ může to, které problémy do dané třídy patří, záviset na použitém výpočetním modelu (zda je to stroj RAM, jednopáskový Turingův stroj, vícepáskový Turingův stroj, ...).

Třídy složitosti

Pomocí tříd $\mathcal{T}(f(n))$ a $\mathcal{S}(f(n))$ můžeme definovat třídy PTIME a PSPACE jako

$$\text{PTIME} = \bigcup_{k \geq 0} \mathcal{T}(n^k) \qquad \text{PSPACE} = \bigcup_{k \geq 0} \mathcal{S}(n^k)$$

- PTIME je třída všech rozhodovacích problémů, pro které existuje algoritmus s polynomiální časovou složitostí, tj. s časovou složitostí $O(n^k)$, kde k je nějaká konstanta.
- PSPACE je třída všech rozhodovacích problémů, pro které existuje algoritmus s polynomiální prostorovou složitostí, tj. s prostorovou složitostí $O(n^k)$, kde k je nějaká konstanta.

Třídy složitosti

Poznámka: Vzhledem k tomu, že všechny (rozumné) výpočetní modely jsou schopné se navzájem simulovat tak, že při dané simulaci nevzroste počet kroků ani množství použité paměti víc než polynomiálně, není definice tříd PTIME a PSPACE závislá na použitém výpočetním modelu. Pro jejich zadefinování můžeme použít kterýkoliv výpočetní model.

Říkáme, že tyto třídy jsou **robustní** — jejich definice nezávisí na použitém výpočetním modelu.

Třídy složitosti

Analogicky můžeme zavést další třídy:

EXPTIME – množina všech rozhodovacích problémů, pro které existuje algoritmus s časovou složitostí $2^{O(n^k)}$, kde k je nějaká konstanta

EXPSPACE – množina všech rozhodovacích problémů, pro které existuje algoritmus s prostorovou složitostí $2^{O(n^k)}$, kde k je nějaká konstanta

LOGSPACE – množina všech rozhodovacích problémů, pro které existuje algoritmus s prostorovou složitostí $O(\log n)$

Poznámka: Místo $2^{O(n^k)}$ bychom mohli psát také $O(c^{n^k})$, kde c a k jsou nějaké konstanty.

Třídy složitosti

Při definici třídy LOGSPACE musíme přesněji specifikovat, co považujeme za prostorovou složitost algoritmu.

Uvažujeme například Turingův stroj, který pracuje se třemi páskami:

- **Vstupní páskou**, na které je na začátku výpočtu zapsán vstup. Z této pásky je možno pouze číst.
- **Pracovní páskou**, která je na začátku výpočtu prázdná. Z této pásky je možno číst i na ni zapisovat.
- **Výstupní páskou**, která je také na začátku výpočtu prázdná a na kterou je možno pouze zapisovat.

Množství použité paměti je pak definováno, jako počet použitých políček na pracovní pásce.

Třídy složitosti

Další příklady tříd složitosti:

2-EXPTIME – množina všech problémů, pro které existuje algoritmus s časovou složitostí $2^{2^{O(n^k)}}$, kde k je nějaká konstanta

2-EXPSPACE – množina všech problémů, pro které existuje algoritmus s prostorovou složitostí $2^{2^{O(n^k)}}$, kde k je nějaká konstanta

ELEMENTARY – množina všech problémů, pro které existuje algoritmus s časovou (či prostorovou) složitostí

$$2^{2^{\cdot^{\cdot^{\cdot^{2^{2^{O(n^k)}}}}}}}$$

kde k je konstanta a počet exponentů je omezen konstantou.

Vztahy mezi třídami složitosti

Pokud Turingův stroj provede m kroků, tak použije maximálně m políček na pásce.

Pokud tedy existuje pro nějaký problém algoritmus s časovou složitostí $O(f(n))$, má tento algoritmus prostorovou složitost (nejvýše) $O(f(n))$.

Je tedy zřejmé, že platí následující vztah.

Pozorování

Pro libovolnou funkci $f : \mathbb{N} \to \mathbb{N}$ platí $\mathcal{T}(f(n)) \subseteq \mathcal{S}(f(n))$.

Poznámka: Analogicky bychom mohli argumentovat například pro stroj RAM.

Vztahy mezi třídami složitosti

Z předchozího okamžitě plyne:

$$\begin{aligned} \text{PTIME} &\subseteq \text{PSPACE} \\ \text{EXPTIME} &\subseteq \text{EXPSPACE} \\ \text{2-EXPTIME} &\subseteq \text{2-EXPSPACE} \\ &\vdots \end{aligned}$$

Vzhledem k tomu, že polynomiální funkce rostou pomaleji než exponenciální a logaritmické pomaleji než polynomiální, zjevně platí:

$$\text{PTIME} \subseteq \text{EXPTIME} \subseteq \text{2-EXPTIME} \subseteq \cdots$$

$$\text{LOGSPACE} \subseteq \text{PSPACE} \subseteq \text{EXPSPACE} \subseteq \text{2-EXPSPACE} \subseteq \cdots$$

Vztahy mezi třídami složitosti

- Pro libovolná dvě reálná čísla ϵ_1 a ϵ_2 taková, že $0 \leq \epsilon_1 < \epsilon_2$, platí

$$\mathcal{S}(n^{\epsilon_1}) \subsetneq \mathcal{S}(n^{\epsilon_2})$$

- LOGSPACE $\subsetneq$ PSPACE
- PSPACE $\subsetneq$ EXPSPACE

- Pro libovolná dvě reálná čísla ϵ_1 a ϵ_2 taková, že $0 \leq \epsilon_1 < \epsilon_2$, platí

$$\mathcal{T}(n^{\epsilon_1}) \subsetneq \mathcal{T}(n^{\epsilon_2})$$

- PTIME $\subsetneq$ EXPTIME
- EXPTIME $\subsetneq$ 2-EXPTIME

Vztahy mezi třídami složitosti

Při zkoumání vztahů mezi třídami složitosti se ukazuje jako užitečný pojem **konfigurace**.

Konfigurací budeme rozumět celkový stav, ve kterém se během jednoho kroku nachází stroj, provádějící nějaký daný algoritmus.

- U Turingova stroje je konfigurace dána stavem jeho řídící jednotky, obsahem pásky (resp. pásek) a pozicí hlavy (resp. hlav).
- U stroje RAM je konfigurace dána obsahem paměti, obsahem všech registrů (včetně IP), obsahem vstupní a výstupní pásky a pozicemi čtecí a zapisovací hlavy.

Vztahy mezi třídami složitosti

Mělo by být jasné, že konfigurace (resp. jejich popisy) můžeme zapisovat jako slova v nějaké abecedě.

Navíc můžeme konfigurace zapisovat tak, že délka těchto slov bude zhruba stejná jako množství paměti použité algoritmem (tj. počet políček na pásce použitých Turingovým stojem, počet bitů paměti použitých strojem RAM apod.).

Poznámka: Pokud máme abecedu Σ, kde $|\Sigma| = c$, tak:

- Počet slov délky n je c^n, tj. $2^{\Theta(n)}$.
- Počet slov délky nejvýše n je

$$\sum_{i=0}^{n} c^n = \frac{c^{n+1} - 1}{c - 1}$$

tj. také $2^{\Theta(n)}$.

Vztahy mezi třídami složitosti

Je jasné, že během výpočtu korektního algoritmu se žádná konfigurace nemůže zopakovat, protože jinak by se algoritmus zacyklil a běžel by donekonečna.

Pokud tedy víme, že prostorová složitost nějakého algoritmu je $O(f(n))$, znamená to, že počet různých konfigurací dosažitelných během výpočtu je $2^{O(f(n))}$.

Protože se konfigurace během žádného výpočtu neopakují, je i časová složitost daného algoritmu maximálně $2^{O(f(n))}$.

Pozorování

Pro libovolnou funkci $f : \mathbb{N} \to \mathbb{N}$ platí, že pokud je nějaký problém P řešený algoritmem s prostorovou složitostí $O(f(n))$, pak časová složitost tohoto algoritmu je v $2^{O(f(n))}$.

Pokud je tedy problém P ve třídě $\mathcal{S}(f(n))$, pak je i ve třídě $\mathcal{T}(2^{c \cdot f(n)})$ pro nějaké $c > 0$.

Vztahy mezi třídami složitosti

Z předchozího plynou následující důsledky:

$$\begin{aligned} \text{LOGSPACE} &\subseteq \text{PTIME} \\ \text{PSPACE} &\subseteq \text{EXPTIME} \\ \text{EXPSPACE} &\subseteq \text{2-EXPTIME} \\ &\vdots \end{aligned}$$

Vztahy mezi třídami složitosti

Shrnutí:

$$\text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{EXPSPACE} \subseteq$$
$$\subseteq \text{2-EXPTIME} \subseteq \text{2-EXPSPACE} \subseteq \cdots \subseteq \text{ELEMENTARY}$$

- $\text{PTIME} \subsetneq \text{EXPTIME} \subsetneq \text{2-EXPTIME} \subsetneq \cdots$
- $\text{LOGSPACE} \subsetneq \text{PSPACE} \subsetneq \text{EXPSPACE} \subsetneq \text{2-EXPSPACE}, \subsetneq \cdots$

Horní a dolní odhady složitosti problémů

Horním odhadem složitosti problému rozumíme to, že složitost problému není vyšší než nějaká uvedená.

Většinou je to formulováno tak, že daný problém patří do nějaké určité třídy složitosti.

Příklady tvrzení, které se týkají horních odhadů složitosti:

- Problém dosažitenosti v grafu je v PTIME.
- Problém ekvivalence dvou regulárních výrazů je v EXPSPACE.

Pokud chceme zjistit nějaký horní odhad složitosti problému, stačí ukázat, že existuje algoritmus s danou složitostí.

Horní a dolní odhady složitosti problémů

Dolním odhadem složitosti problému rozumíme to, že složitost problému je alespoň taková jako nějaká uvedená.

Obecně je zjišťování (netriviálních) dolních odhadů složitosti problémů mnohem obtížnější než zjišťování horních odhadů.

Pro odvození dolního odhadu musíme totiž ukázat, že **každý** algoritmus řešící daný problém má danou složitost.

Horní a dolní odhady složitosti problémů

Problém „Třídění“

Vstup: Posloupnost prvků $a_1, a_2, \ldots, a_n$.

Výstup: Prvky $a_1, a_2, \ldots, a_n$ setříděné od nejmenšího po největší.

Dá se dokázat, že každý algoritmus, který řeší problém “Třídění” a na prvcích tříděné posloupnosti používá pouze operaci porovnávání (tj. nezkoumá obsah těchto prvků), má časovou složist v nejhorším případě v $\Omega(n \log n)$ (tj. pro každý takový algoritmus existují konstanty $c > 0$ a $n_0 \geq 0$ takové, že pro každé $n \geq n_0$ existuje vstup velikosti n, pro který provede algoritmus nejmémě $cn \log n$ operací).

Nedeterministické algoritmy a třídy složitosti

Nedeterminismus

Nedeterministický stroj RAM:

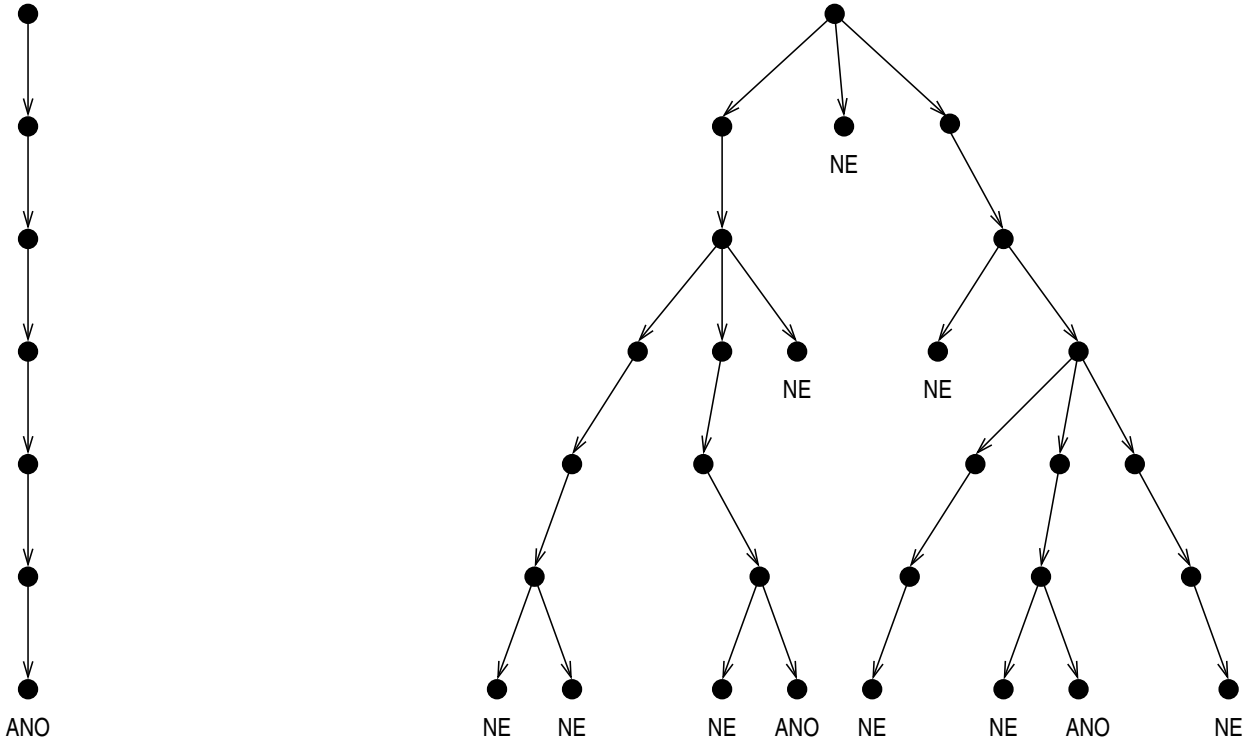
- Je definován velice podobně jako deterministický RAM.
- Navíc má instrukci

  nd_goto ℓ_1, ℓ_2

  která umožňuje stroji vybrat si jedno z možných pokračování.
- Pokud ze všech možných výpočtů takového stroje nad zadaným vstupem alespoň jeden skončí s odpovědí ANO, je odpověď ANO.
- Pokud všechny výpočty skončí s odpovědí NE, je odpověď NE.

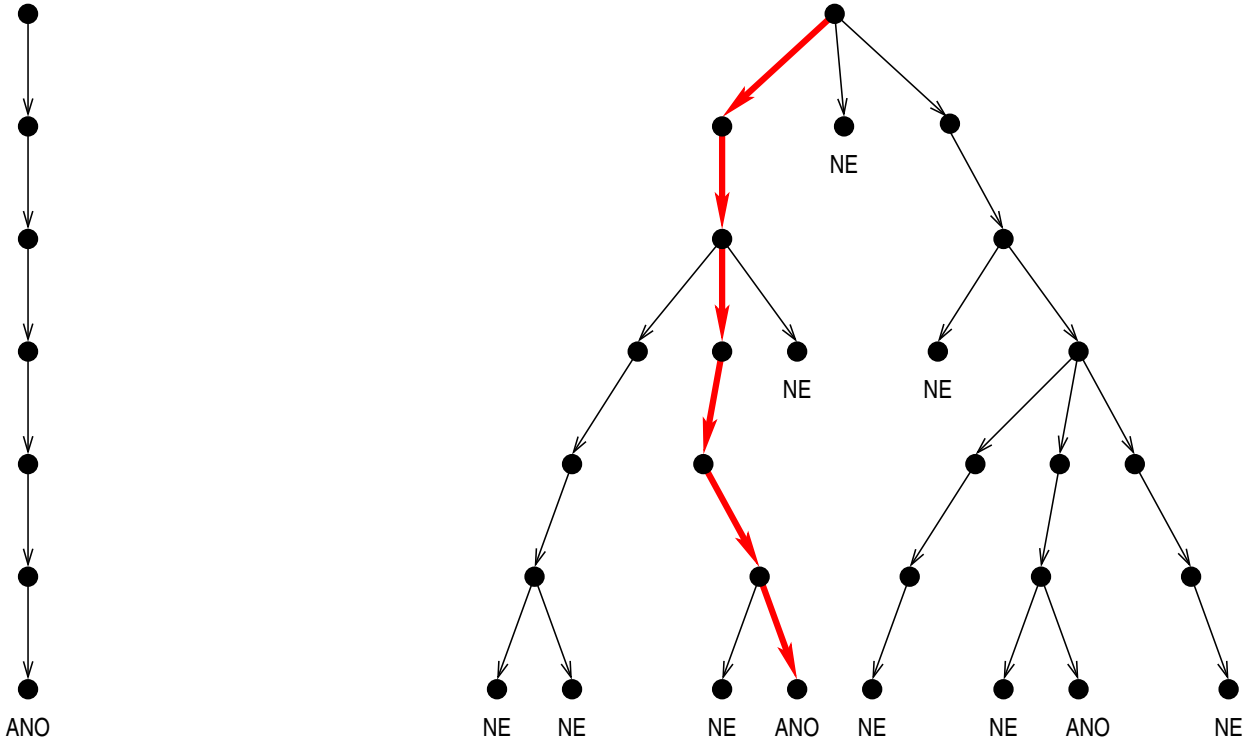
Podobně můžeme definovat nedeterministické verze jiných výpočetních modelů, např. nedeterministické Turingovy stroje.

Nedeterminismus



- Doba výpočtu nedeterministického stroje RAM (nebo jiného nedeterministického stroje) nad zadaným vstupem je definována jako délka nejdelšího možného výpočtu nad tímto vstupem.

Nedeterminismus



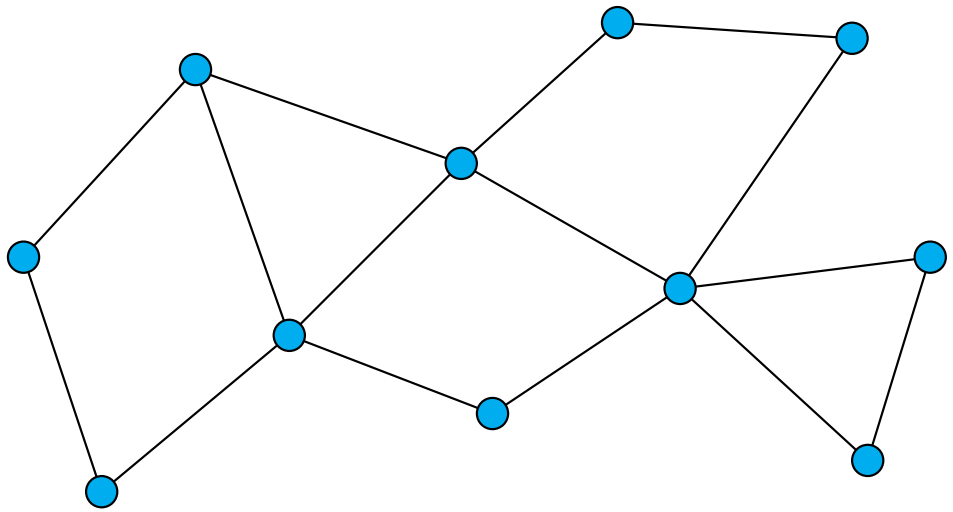
- Doba výpočtu nedeterministického stroje RAM (nebo jiného nedeterministického stroje) nad zadaným vstupem je definována jako délka nejdelšího možného výpočtu nad tímto vstupem.

Nedeterminismus

Problém „Barvení grafu k barvami“

Vstup: Neorientovaný graf G a přirozené číslo k.

Otázka: Je možné obarvit vrcholy grafu G k barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?



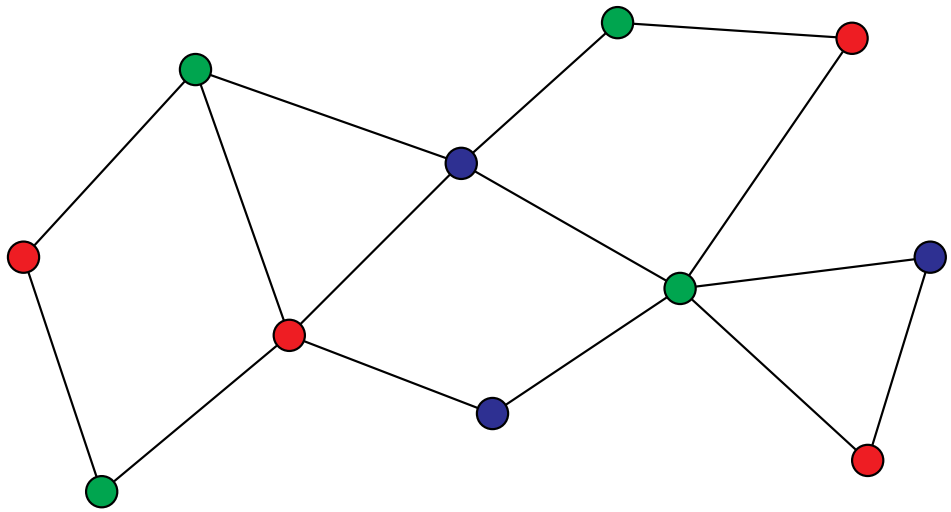
$k = 3$

Nedeterminismus

Problém „Barvení grafu k barvami“

Vstup: Neorientovaný graf G a přirozené číslo k.

Otázka: Je možné obarvit vrcholy grafu G k barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?



$k = 3$

Nedeterminismus

Problém „Barvení grafu k barvami“

Vstup: Neorientovaný graf G a přirozené číslo k.

Otázka: Je možné obarvit vrcholy grafu G k barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?

Nedeterministický algoritmus pracuje následovně:

1. Každému vrcholu grafu G nedeterministicky přiřadí jednu z k barev.
2. Projde všechny hrany grafu G a u každé z nich zkotroluje, že oba její koncové vrcholy jsou obarveny různými barvami. Pokud ne, skončí s odpovědí NE.
3. Pokud prošel všechny hrany a u všech byly koncové vrcholy obarveny různými barvami, skončí s odpovědí ANO.

Nedeterminismus

Problém „Isomorfismus grafů“

Vstup: Neorientované grafy $G_1 = (V_1, E_1)$ a $G_2 = (V_2, E_2)$.

Otázka: Jsou grafy G_1 a G_2 isomorfní?

Poznámka: Grafy G_1 a G_2 jsou isomorfní, jestliže existuje nějaká bijekce $f : V_1 \to V_2$ taková, že pro libovolné dva vrcholy $u, v \in V_1$ platí $(u, v) \in E_1$ právě když $(f(u), f(v)) \in E_2$.

Nedeterministický algoritmus pracuje následovně:

1. Nedeterministicky zvolí hodnoty funkce f pro všechny $v \in V_1$.
2. Deterministicky ověří, že f je bijekce a že pro všechny dvojice vrcholů je splněna výše uvedená podmínka.
3. Pokud je některá z podmínek porušena, skončí s odpovědí NE, v opačném případě s odpovědí ANO.

Nedeterminismus

- Z hlediska rozhodnutelnosti nepřináší nedeterministické algoritmy oproti deterministickým nic dalšího navíc:
  Pokud je nějaký problém možné řešit nedeterministickým strojem RAM nebo TS, tak je ho možné řešit i deterministickým, který postupně vyzkouší všechny možné výpočty nedeterministického stroje nad daným vstupem.
- Nedeterminismus má význam především při zkoumání výpočetní složitosti problémů.

Nedeterminismus

- Při výše uvedené přímočaré simulaci činnosti nedeterministického algoritmu pomocí deterministického, který systematicky zkouší všechny možné výpočty, je časová složitost deterministického algoritmu exponenciálně vyšší než u nedeterministického.
- Pro řadu problémů je zjevné, že pro ně existuje nedeterministický algoritmus s polynomiální časovou složitostí, ale není vůbec jasné, jestli pro ně existuje také deterministický algoritmus s polynomiální časovou složitostí.

Nedeterminismus

Na nedeterminismus můžeme nahlížet následujícími způsoby:

1. Ve chvíli, kdy má stroj nedeterministicky zvolit mezi několika možnostmi, tak „uhodne“, která z těchto možností povede k odpovědi ANO (pokud taková možnost existuje).

2. Ve chvíli, kdy má stroj nedeterministicky zvolit mezi několika možnostmi, rozdělí se do tolika kopií, kolik je těchto možností, a každá z těchto kopií pokračuje ve výpočtu odpovídající jedné z možností, přičemž pracují všechny paralelně.

   Odpověď je ANO právě tehdy, když alespoň jedna z kopií stroje odpoví ANO.

Ani jedno z toho není něco, co by se dalo efektivně realisticky implementovat.

Nedeterminismus

Další možný pohled na nedeterminismus:

- Druh algoritmu, který sice neřeší daný problém, ale s použitím dodatečné další informace — **svědka** (**witness**) — umí **ověřit**, že pro danou instanci je odpověď ANO.

  Předpokládejme, že v původním problému je vstupem nějaké x z množiny instancí In a otázka je, zda má dané x nějakou specifikovanou vlastnost P.

  Pro daný vstup x je dána množina **potenciálních svědků** $W(x)$, přičemž právě tehdy, když x má vlastnost P, tak existuje nějaký **skutečný svědek** $y \in W(x)$ toho, že x tuto vlastnost P skutečně má.

  Vezměme si **deterministický** algoritmus Alg, který jako vstup dostane dvojici (x, y) (kde $y \in W(x)$) a ověří, zda y je svědkem toho, že x má vlastnost P.

Nedeterminismus

Příklad: Problém „Barvení grafu k barvami“:

- *Vstup*: Neorientovaný graf $G = (V, E)$ a číslo k.
- *Potenciální svědci*: Všechna možná obarvení vrcholů grafu G s použím k barev, tj. všechny možné funkce $c : V \to \{1, \ldots, k\}$.
- *Skuteční svědci*: Taková obarvení c, kde pro každou hranu $(u, v) \in E$ platí, že $c(u) \neq c(v)$.

Nedeterminismus

- Ke každému takovému **deterministickému** algoritmu Alg, který pro danou dvojici (x, y) umí ověřit, zda y je svědkem toho, že x má vlastnost P, je možné snadno sestrojit odpovídající **nedeterministický** algoritmus, který řeší původní problém:
  - Pro dané $x \in In$ nejprve neterministicky vygeneruje potenciálního svědka $y \in W(x)$.
  - Použije algoritmus Alg jako podprogram k (deterministickému) ověření toho, zda je y skutečným svědkem.

Nedeterminismus

- Naopak ke každému **nedeterministickému** algoritmu můžeme snadno vytvořit **deterministický** algoritmus ověřující svědky:
  - Potenciálním svědkem bude posloupnost udávající pro jednotlivé kroky původního nedeterministického algoritmu, která možnost se má v daném kroku zvolit.
  - Deterministický algoritmus simuluje jeden konkrétní výpočet (jednu větev stromu) původního algoritmu, přičemž v krocích, kdy má na výběr z více možností, tak nehádá, ale postupuje podle toho, co je určeno v zadané posloupnosti.

Nedeterminismus

Zejména nás budou zajímat ty případy, kdy časová složitost algoritmu pro ověřování svědka je polynomiální vzhledem k velikosti vstupu x.

Mimo jiné to znamená, že daný svědek y, dosvědčující, že pro x je odpověď ANO, musí být polynomiálně velký.

Nedeterministickým algoritmem s polynomiální časovou složitostí se tedy dají řešit ty rozhodovací problémy, kde:

- pro daný vstup x existuje příslušný (polynomiálně velký) svědek právě tehdy, když pro x je odpověď ANO,
- je možné deterministickým algoritmem v polynomiálním čase ověřit, že daný potenciální svědek je skutečně svědkem.

Nedeterminismus

Mnohdy je existence takových polynomiálně velkých svědků a deterministických algoritmů, které je ověřují, očividná a je triviální ukázat, že existují — např. u problémů jako „Barvení grafu k barvami“, „Isomorfismus grafů“ nebo u následujícího problému:

Testování složenosti

Vstup: Přirozené číslo x.

Otázka: Je číslo x složené?

Poznámka: Číslo x je **složené**, když existují přirozená čísla a a b taková, že $a > 1$, $b > 1$ a $x = a \cdot b$.

Například číslo 15 je složené, protože $15 = 3 \cdot 5$.

Číslo $x \in \mathbb{N}$ je tedy složené, pokud $x > 1$ a x není prvočíslo.

Existence takových polynomiálně velkých svědků ale nutně neznamená, že je snadné je najít.

Nedeterminismus

U některých problémů může být ale ukázání existence takových polynomiálně velkých svědků, které je možné deterministiky v polynomiálním čase ověřovat, značně netriviálním výsledkem.

Příkladem je následující problém:

Testování prvočíselnosti

Vstup: Přirozené číslo x.

Otázka: Je číslo x prvočíslo?

S využitím různých netriviálních poznatků z teorie čísel se dá ukázat existence takových svědků i pro tento problém — svědci zde mají podobu poměrně komplikované rekurzivně definované datové struktury.

Poznámka: Tento výsledek ukázal V. Pratt v roce 1975.

Mnohem později bylo ukázáno, že „Testování prvočíselnosti“ je ve skutečnosti v PTIME (Agrawal–Kayal–Saxena, 2002).

Nedeterministické třídy složitosti

Definice

Pro funkci $f : \mathbb{N} \to \mathbb{N}$ rozumíme **třídou časové složitosti** $\mathcal{NT}(f)$ množinu těch rozhodovacích problémů, které jsou řešeny nedeterministickými RAMy s časovou složitostí v $O(f(n))$.

Definice

Pro funkci $f : \mathbb{N} \to \mathbb{N}$ rozumíme **třídou prostorové složitosti** $\mathcal{NS}(f)$ množinu těch rozhodovacích problémů, které jsou řešeny nedeterminictickými RAMy s prostorovou složitostí v $O(f(n))$.

Poznámka: Ve výše uvedených definicích mohou být samozřejmě místo strojů RAM uvedeny třeba Turingovy stroje či nějaký jiný výpočetní model.

Třída NPTIME

Definice

$$\text{NPTIME} = \bigcup_{k=0}^{\infty} \mathcal{NT}(n^k)$$

- NPTIME (někdy se píše jen NP) je třída všech problémů, pro které existuje nedeterministický algoritmus s polynomiání časovou složitostí.
- Do NPTIME tedy patří problémy, u kterých je možné pro daný vstup rychle ověřit, že odpověď je ANO, pokud nám ten, kdo nás o tom chce přesvědčit, dodá nějakou dodatečnou informaci.

Třídy NPSPACE, NEXPTIME, NEXPSPACE, ...

Podobně můžeme definovat další třídy složitosti:

NPSPACE – množina všech rozhodovacích problémů, pro které existuje nedeterministický algoritmus s polynomiání prostorovou složitostí

NEXPTIME – množina všech rozhodovacích problémů, pro které existuje nedeterministický algoritmus s časovou složitostí $2^{O(n^k)}$, kde k je nějaká konstanta

NEXPSPACE – množina všech rozhodovacích problémů, pro které existuje nedeterministický algoritmus s prostorovou složitostí $2^{O(n^k)}$, kde k je nějaká konstanta

NLOGSPACE – množina všech rozhodovacích problémů, pro které existuje nedeterministický algoritmus s prostorovou složitostí $O(\log n)$

Vztahy mezi třídami složitosti

Je zřejmé, že na deterministické algoritmy se můžeme dívat jako na speciální případ nedeterministických.

Očividně tedy platí:

$$\begin{aligned} \mathrm{LOGSPACE} &\subseteq \mathrm{NLOGSPACE} \\ \mathrm{PTIME} &\subseteq \mathrm{NPTIME} \\ \mathrm{PSPACE} &\subseteq \mathrm{NPSPACE} \\ \mathrm{EXPTIME} &\subseteq \mathrm{NEXPTIME} \\ \mathrm{EXPSPACE} &\subseteq \mathrm{NEXPSPACE} \\ &\vdots \end{aligned}$$

Vztahy mezi třídami složitosti

Rovněž je zřejmé, že jak u deterministických, tak u nedeterministických algoritmů, algoritmus během výpočtu nemůže použít řádově více buněk paměti, než kolik udělá kroků.

Prostorová složitost daného algoritmu je tedy vždy nejvýše taková, jaká je jeho časová složitost.

Z toho plyne:

$$\begin{aligned} \mathrm{PTIME} &\subseteq \mathrm{PSPACE} \\ \mathrm{NPTIME} &\subseteq \mathrm{NPSPACE} \\ \mathrm{EXPTIME} &\subseteq \mathrm{EXPSPACE} \\ \mathrm{NEXPTIME} &\subseteq \mathrm{NEXPSPACE} \\ &\vdots \end{aligned}$$

Vztahy mezi třídami složitosti

Vezměme si nějaký **nedeterministický** algoritmus s **časovou** složitostí $O(f(n))$.

Deteriministický algoritmus, který bude simulovat jeho činnost tím způsobem, že bude systematicky procházet všechny jeho výpočty (procházením stromu těchto výpočtů do hloubky), vystačí s následující pamětí:

- paměť, kde je uložena aktuální konfigurace simulovaného stroje — má velikost $O(f(n))$ (protože pokud tento simulovaný nedeterministický stroj udělá maximálně $O(f(n))$ kroků, tak jeho konfigurace budou používat nanejvýš $O(f(n))$ buněk paměti)
- paměť pro uložení zásobníku, který bude používat k tomu, aby se mohl vracet k předchozím konfiguracím — aby bylo možné z následující konfigurace α' obnovit předchozí konfiguraci α, stačí si uložit konstatní množství informace — jen to, co se při přechodu z α do α' změnilo

Vztahy mezi třídami složitosti

- Vzhledem k tomu, že délka větví je $O(f(n))$, množství potřebné paměti pro zásobník je $O(f(n))$.
- Celkově tedy deterministický algoritmus při této simulaci vystačí s množstvím paměti, které je nejvýše $O(f(n))$.

Z výše uvedeného tedy vyplývá:

$$\begin{aligned} \text{NPTIME} &\subseteq \text{PSPACE} \\ \text{NEXPTIME} &\subseteq \text{EXPSPACE} \\ &\vdots \end{aligned}$$

Vztahy mezi třídami složitosti

Vezměme si nějaký **nedeterministický** algoritmus s **prostorovou** složitostí $O(f(n))$:

- Připomeňme, že konfigurací velikosti nejvýše $O(f(n))$ je $O(c^{f(n)})$, kde c je nějaká konstanta, což můžeme psát jako $2^{O(f(n))}$.
- Počet kroků tohoto nedeterministického algoritmu v rámci jedné větve výpočtu tedy může být až $2^{O(f(n))}$.

  (Pozn.: Žádná konfigurace se během výpočtu nemůže zopakovat, protože jinak by mohly být výpočty nekonečné.)
- Simulace výše popsaným způsobem by tedy měla časovou složitost až $2^{2^{O(f(n))}}$.

Vztahy mezi třídami složitosti

Při simulaci můžeme postupovat, ale o něco chytřeji — představme si orientovaný graf, kde:

- **vrcholy** — všechny konfigurace simulovaného stroje, jejichž velikost je nejvýše $O(f(n))$
  — těchto konfigurací je $2^{O(f(n))}$
- **hrany** — mezi vrcholy, které reprezentují konfigurace α a α' vede hrany právě tehdy, když simulovaný stroj může přejít jedním krokem z konfigurace α do konfigurace α'
  — z každého vrcholu povede počet hran omezený shora nějakou kostantou — hran tedy bude také řádově $2^{O(f(n))}$

Stačí umět zjistit, zda ve výše uvedeném grafu existuje cesta z vrcholu, který odpovídá počáteční konfiguraci (pro daný vstup x), do některého vrcholu, který odpovídá koncové konfiguraci, kdy daný stroj dává odpověď Ano.

Vztahy mezi třídami složitosti

Pro zjištění existence takové cesty je možné použít libovolný algoritmus na procházení grafu — procházení do šířky, procházení do hloubky, . . . :

- Algoritmus si musí ukládat a značit, které konfigurace již navštívil. Další paměť potřebuje pro uložení fronty či zásobníku, apod.
- Časová i prostorová složitost tohoto algoritmu bude lineárně úměrná velikosti daného grafu, tj. $2^{O(f(n))}$.

Vztahy mezi třídami složitosti

Dostáváme tedy následující:

Činnost nedeterministického algoritmu, jehož prostorová složitost je $O(f(n))$, je možné simulovat deterministickým algoritmem, jehož časová složitost je $2^{O(f(n))}$.

Z toho vyplývá:

$$\begin{aligned} \text{NLOGSPACE} &\subseteq \text{PTIME} \\ \text{NPSPACE} &\subseteq \text{EXPTIME} \\ \text{NEXPSPACE} &\subseteq \text{2-EXPTIME} \\ &\vdots \end{aligned}$$

Vztahy mezi třídami složitosti

Uvažujeme opět nějaký **nedeterministický** algoritmus s **prostorovou** složitostí $O(f(n))$. Teď nám ale pro změnu půjde o co nejmenší **prostorovou** složitost simulujícího deterministického algoritmu.

Věta (Savitch, 1970)

Činnost nedeterministického algoritmu s prostorovou složitostí $O(f(n))$ je možné simulovat deterministickým algoritmem s prostorovou složitostí $O(f(n)^2)$.

Myšlenka důkazu:

- Opět si představme výše popsaný graf konfigurací, který má $2^{O(f(n))}$ vrcholů (i hran).
- Algoritmus bude zjišťovat, zda existuje cesta z počáteční konfigurace do některé přijímající konfigurace.

Vztahy mezi třídami složitosti

Základem bude rekurzivní funkce $F(\alpha, \alpha', i)$, která pro libovolné zadané konfigurace α a α' a číslo $i \in \mathbb{N}$ zjistí, zda ve výše uvedeném grafu existuje cesta z α do α' délky nejvýše 2^i:

- Pokud je $i = 0$, zjistí, zda existuje cesta z α do α' délky nejvýše 1:
  - buď je to cesta délky 0, tj. $\alpha = \alpha'$,
  - nebo je to cesta délky 1, tj. je možné přejít z α do α' jedním krokem
- Pokud je $i > 0$, bude systematicky probírat všechny možné konfigurace α'' a testovat, jestli:
  - existuje cesta délky nejvýše $2^i/2$ z α do α'' — zavolá rekurzivně $F(\alpha, \alpha'', i-1)$
  - existuje cesta délky nejvýše $2^i/2$ z α'' do α' — zavolá rekurzivně $F(\alpha'', \alpha', i-1)$

  Pokud obojí vrátí TRUE, vrátí TRUE, jinak pokračuje zkoušením dalšího α''.

Vztahy mezi třídami složitosti

Analýza prostorové složitosti daného algoritmu:

- V rámci jednoho rekurzivního volání funkce F je třeba mít uložené:
  - tři konfigurace α, α', α'' — všechny jsou velikosti $O(f(n))$
  - hodnotu čísla i, které je řádově $O(f(n))$ — proto na jeho uložení stačí zhruba $O(\log F(n))$ bitů
  - další pomocné proměnné, jejichž hodnoty jsou proti velikosti výše uvedených položek zanedbatelné
- Množství paměti potřebné v rámci jednoho rekurzivního volání je tedy $O(f(n))$.
- Hloubka zanoření rekurze je také $O(f(n))$.
- Celková prostorová složitost daného algoritmu je tedy $O(f(n)^2)$.

Vztahy mezi třídami složitosti

Z výše uvedené věty vyplývá:

$$\begin{aligned} \mathrm{NPSPACE} &\subseteq \mathrm{PSPACE} \\ \mathrm{NEXPSPACE} &\subseteq \mathrm{EXPSPACE} \\ &\vdots \end{aligned}$$

Spolu s triviálními fakty, že $\mathrm{PSPACE} \subseteq \mathrm{NPSPACE}$, $\mathrm{EXPSPACE} \subseteq \mathrm{NEXPSPACE}$, . . . nám to tedy dává:

$$\begin{aligned} \mathrm{PSPACE} &= \mathrm{NPSPACE} \\ \mathrm{EXPSPACE} &= \mathrm{NEXPSPACE} \\ &\vdots \end{aligned}$$

Poznámka: Všimněte si, že z výše uvedeného **nevyplývá**, že by muselo platit $\mathrm{LOGSPACE} = \mathrm{NLOGSPACE}$.

Vztahy mezi třídami složitosti

Celkově tak dostáváme následující **hierarchii tříd složitosti**:

$$\begin{aligned}
&\mathrm{LOGSPACE} \subseteq \mathrm{NLOGSPACE} \subseteq \\
&\subseteq \mathrm{PTIME} \subseteq \mathrm{NPTIME} \subseteq \mathrm{PSPACE} = \mathrm{NPSPACE} \subseteq \\
&\subseteq \mathrm{EXPTIME} \subseteq \mathrm{NEXPTIME} \subseteq \mathrm{EXPSPACE} = \mathrm{NEXPSPACE} \subseteq \\
&\vdots
\end{aligned}$$

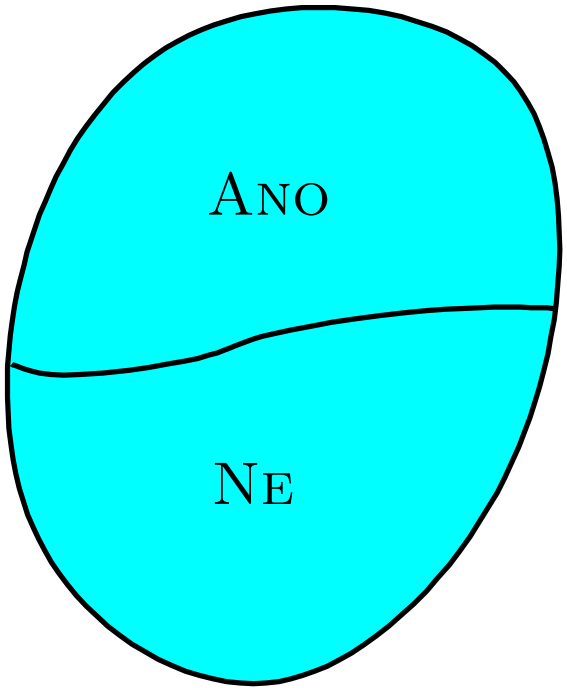
NP-úplné problémy

Polynomiální převody mezi problémy

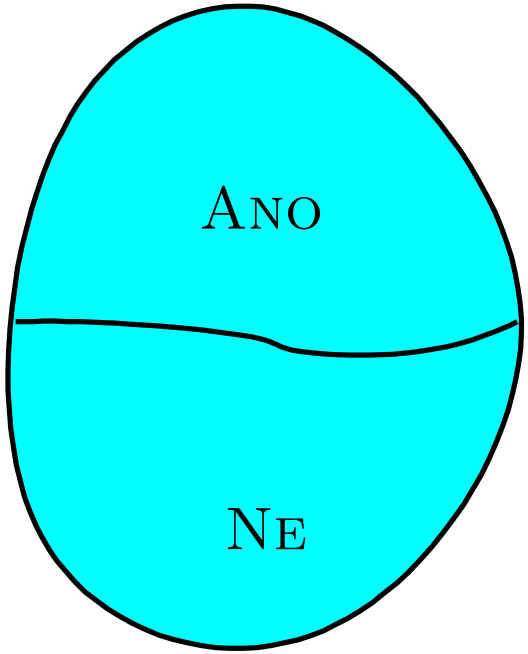
Problém P_1 je **polynomiálně převeditelný** na problém P_2, jestliže existuje algoritmus Alg s polynomiální časovou složitostí, který převádí problém P_1 na problém P_2.

Polynomiální převody mezi problémy

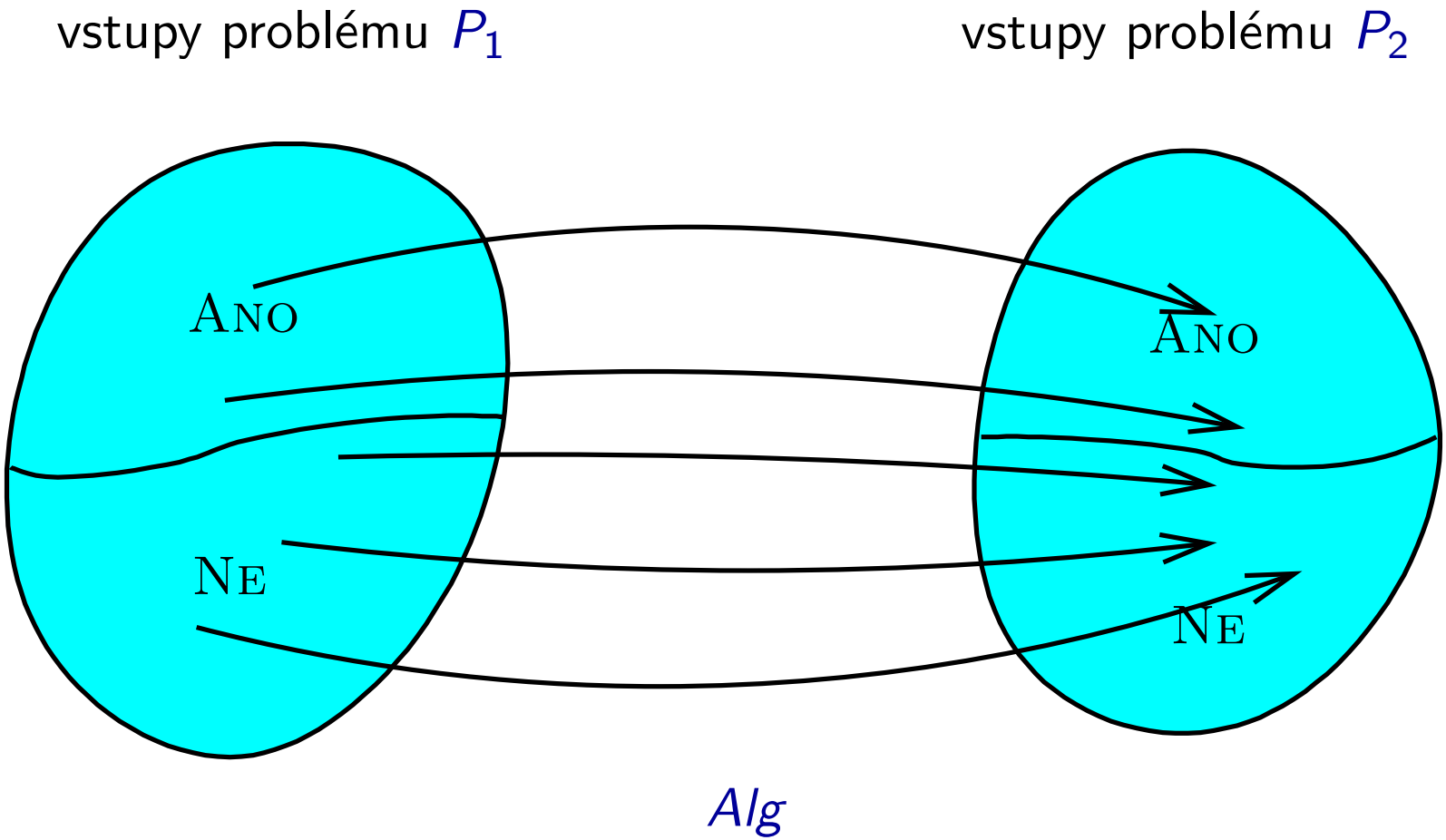
vstupy problému P_1



vstupy problému P_2



Polynomiální převody mezi problémy



Polynomiální převody mezi problémy

Řekněme, že problém P_1 je polynomiálně převeditelný na problém P_2, tj. existuje polynomiální algoritmus Alg realizující tento převod.

Pokud pro problém P_2 existuje polynomiální algoritmus, pak i pro problém P_1 existuje polynomiální algoritmus.

Řešení problému P_1 pro vstup w:

- Zavoláme Alg se vstupem w, vrátí nám hodnotu $Alg(w)$.
- Zavoláme algoritmus řešící problém P_2 se vstupem $Alg(w)$. Hodnotu, kterou nám vrátí, vypíšeme jako výsledek.

Z toho plyne:

Pokud neexistuje polynomiální algoritmus pro problém P_1, tak neexistuje ani polynomiální algoritmus pro problém P_2.

Polynomiální převody mezi problémy

Existuje velká skupina algoritmických problémů označovaných jako **NP-úplné** problémy, které:

- patří do třídy NPTIME, tj. jsou řešitelné v polynomiálním čase **nedeterministickým** algoritmem
- jsou tedy řešitelné v exponenciálním čase
- není pro ně znám žádný algoritmus s polynomiální časovou složitostí
- na druhou stranu není ani dokázáno, že daný pro daný problém nemůže algoritmus s polynomiální časovou složitostí existovat
- jsou všechny navzájem polynomiálně převeditelné

Poznámka: Toto není definice NP-úplných problémů. Ta bude uvedena později.

Problém SAT

Typickým příkladem NP-úplného problému je problém SAT:

SAT (splnitelnost booleovských formulí)

Vstup: Booleovská formule φ.

Otázka: Je φ splnitelná?

Příklad:

Formule $\varphi_1 = x_1 \wedge (\neg x_2 \vee x_3)$ je splnitelná:
např. při ohodnocení v, kde $v(x_1) = 1$, $v(x_2) = 0$, $v(x_3) = 1$, je formule φ_1 pravdivá.

Formule $\varphi_2 = (x_1 \wedge \neg x_1) \vee (\neg x_2 \wedge x_3 \wedge x_2)$ není splnitelná:
je nepravdivá při každém ohodnocení v.

Problém 3-SAT

3-SAT je varianta problému SAT, ve které se omezujeme na formule určitého speciálního typu:

3-SAT

Vstup: Formule φ v konjunktivní normální formě, kde každá klauzule obsahuje právě 3 literály.

Otázka: Je φ splnitelná?

Problém 3-SAT

Připomenutí některých pojmů:

- **Literál** je formule tvaru x nebo $\neg x$, kde x je atomický výrok.
- **Klauzule** je disjunkce literálů.

  Příklady: $x_1 \vee \neg x_2$ $\neg x_5 \vee x_8 \vee \neg x_{15} \vee \neg x_{23}$ x_6
- Formule je v **konjunktivní normální formě (KNF)**, jestliže je konjunkcí klauzulí.

  Příklad: $(x_1 \vee \neg x_2) \wedge (\neg x_5 \vee x_8 \vee \neg x_{15} \vee \neg x_{23}) \wedge x_6$

V případě problému 3-SAT tedy vyžadujeme, aby formule φ byla v KNF a navíc, aby každá klauzule obsahovala právě tři literály.

Příklad:

$(x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_3 \vee x_3) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4) \wedge (x_2 \vee \neg x_3 \vee x_4)$

Problém 3-SAT

Následující formule je splnitelná:

$$(x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_3 \vee x_3) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4) \wedge (x_2 \vee \neg x_3 \vee x_4)$$

Je pravdivá např. při ohodnocení v, kde

$$v(x_1) = 0$$
$$v(x_2) = 1$$
$$v(x_3) = 0$$
$$v(x_4) = 1$$

Naproti tomu následující formule není splnitelná:

$$(x_1 \vee x_1 \vee x_1) \wedge (\neg x_1 \vee \neg x_1 \vee \neg x_1)$$

Polynomiální převody mezi problémy

Ukážeme si příklad polynomiálního převodu problému 3-SAT na problém nezávislé množiny (IS).

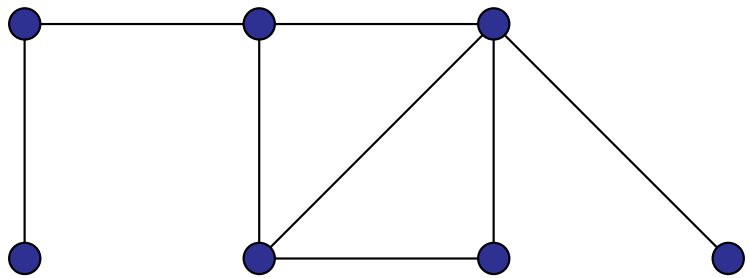
Poznámka: Jak 3-SAT, tak IS jsou příklady NP-úplných problémů.

Problém nezávislé množiny (IS)

Problém nezávislé množiny (IS)

Vstup: Neorientovaný graf G, číslo k.

Otázka: Existuje v grafu G nezávislá množina velikosti k?



$k = 4$

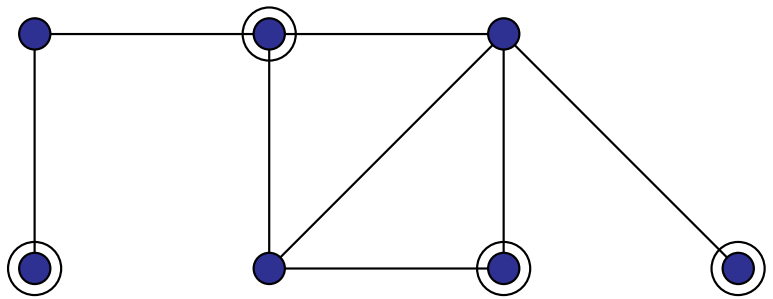
Poznámka: **Nezávislá množina** v grafu je podmnožina vrcholů grafu taková, že žádné dva vrcholy z této podmnožiny nejsou spojeny hranou.

Problém nezávislé množiny (IS)

Problém nezávislé množiny (IS)

Vstup: Neorientovaný graf G, číslo k.

Otázka: Existuje v grafu G nezávislá množina velikosti k?

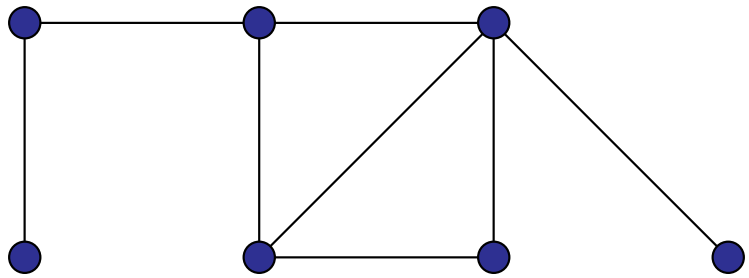


$k = 4$

Poznámka: **Nezávislá množina** v grafu je podmnožina vrcholů grafu taková, že žádné dva vrcholy z této podmnožiny nejsou spojeny hranou.

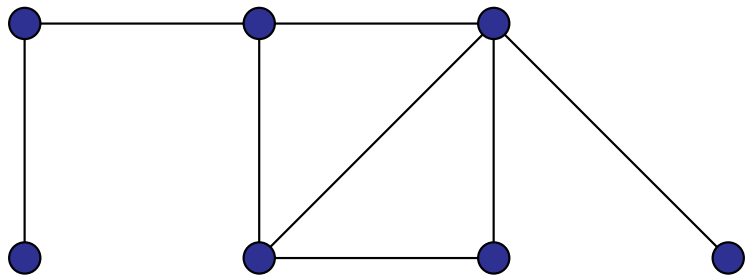
Problém nezávislé množiny (IS)

Příklad instance, kde je odpověď ANO:



$k = 4$

Příklad instance, kde je odpověď NE:



$k = 5$

Převod 3-SAT na IS

Popíšeme (polynomiální) algoritmus, který bude mít následující vlastnosti:

- **Vstup:** Libovolná instance problému 3-SAT, tj. formule φ v konjunktivní normální formě, kde každá klauzule obsahuje právě tři literály.
- **Výstup:** Instance problému IS, tj. neorientovaný graf G a číslo k.
- Navíc bude pro libovolný vstup (tj. pro libovolnou formuli φ ve výše uvedeném tvaru) zaručeno následující:

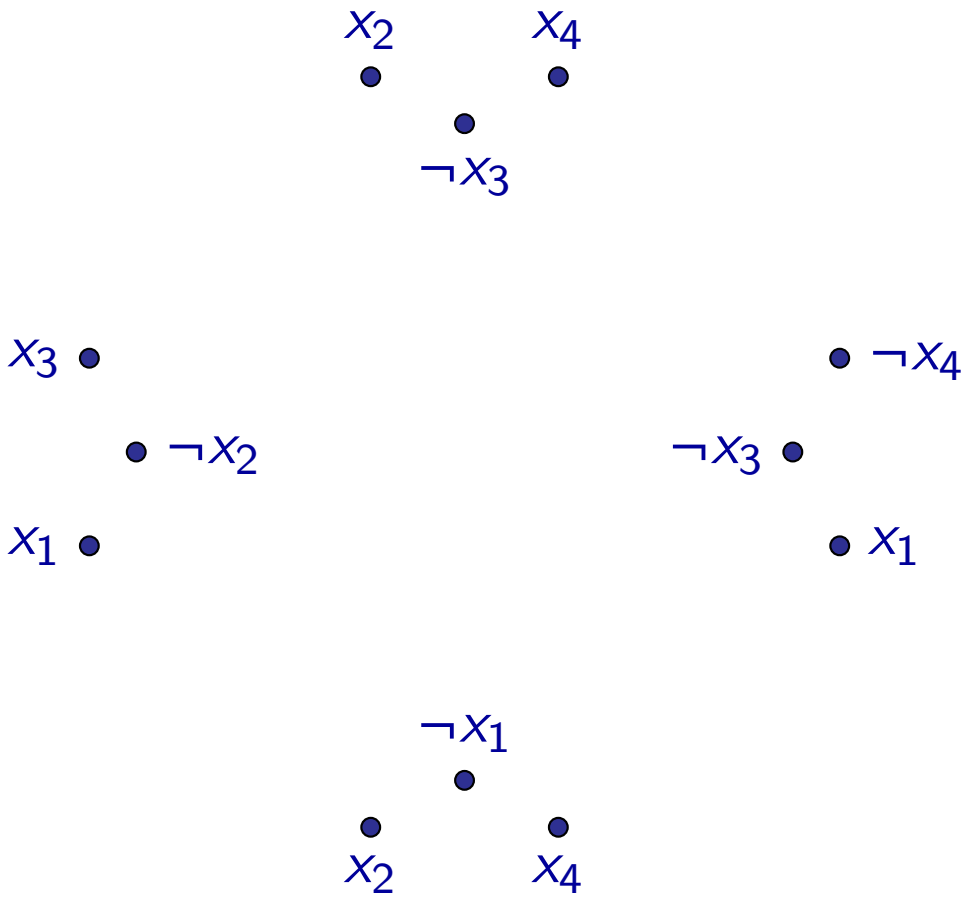
  V grafu G bude existovat nezávislá množina velikosti k právě tehdy, když formule φ bude splnitelná.

Převod 3-SAT na IS

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$$

Převod 3-SAT na IS

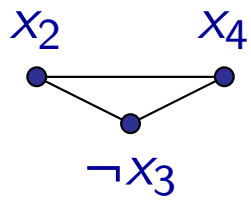
$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$$

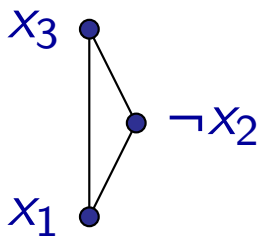


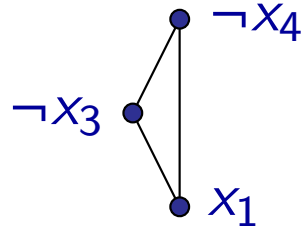
Pro každý výskyt literálu přidáme do grafu jeden vrchol.

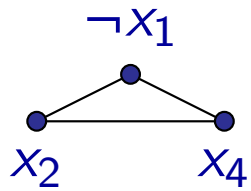
Převod 3-SAT na IS

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$$





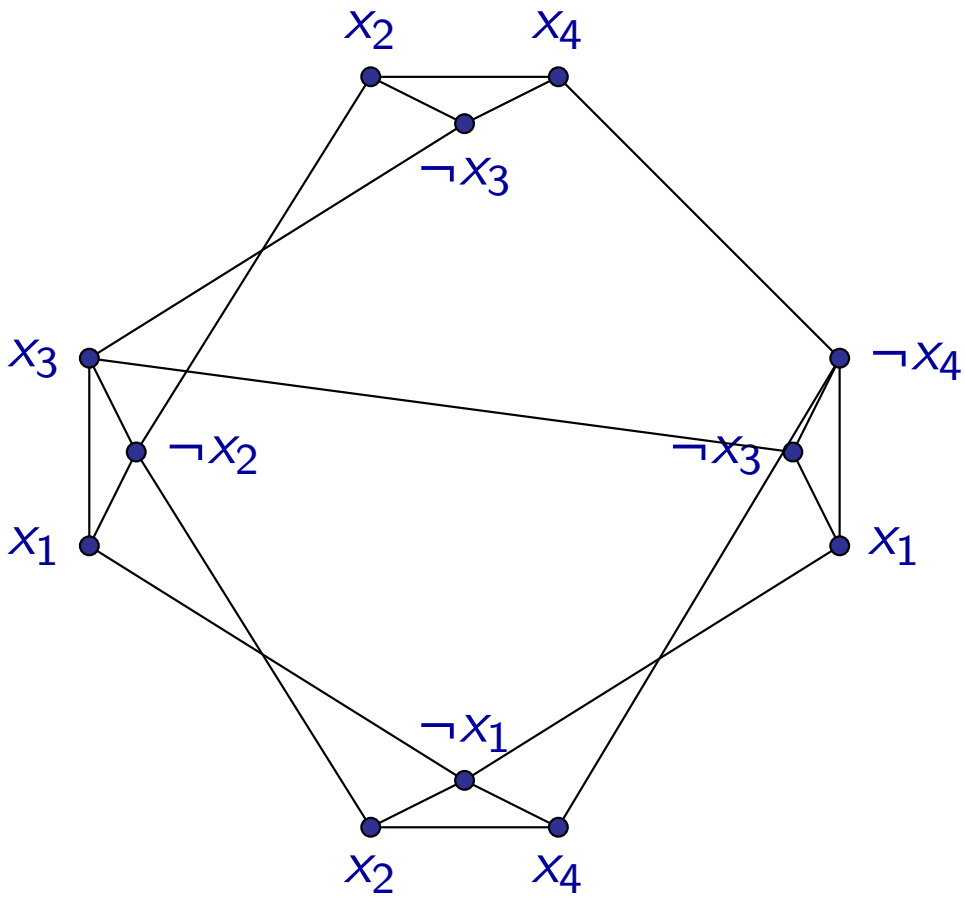




Vrcholy odpovídající výskytům literálů patřícím do stejné klauzule spojíme hranami.

Převod 3-SAT na IS

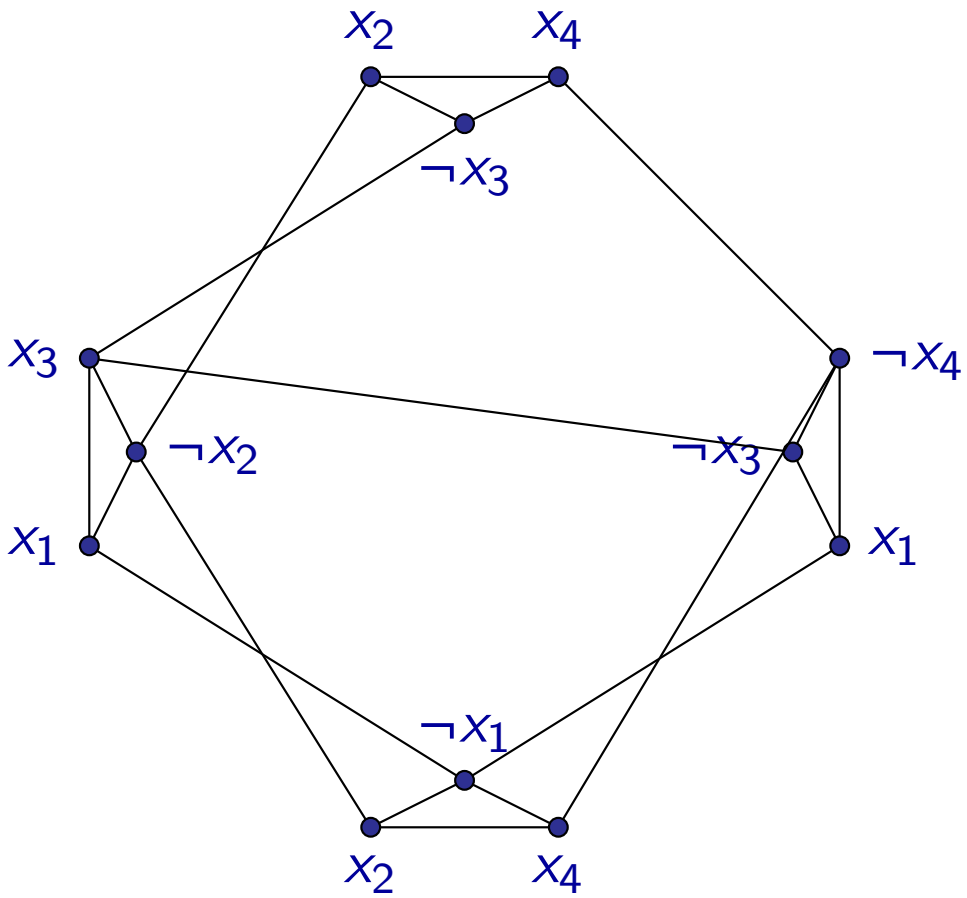
$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$



Dvojice vrcholů odpovídající literálům x_i a $\neg x_i$ spojíme hranami.

Převod 3-SAT na IS

$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$

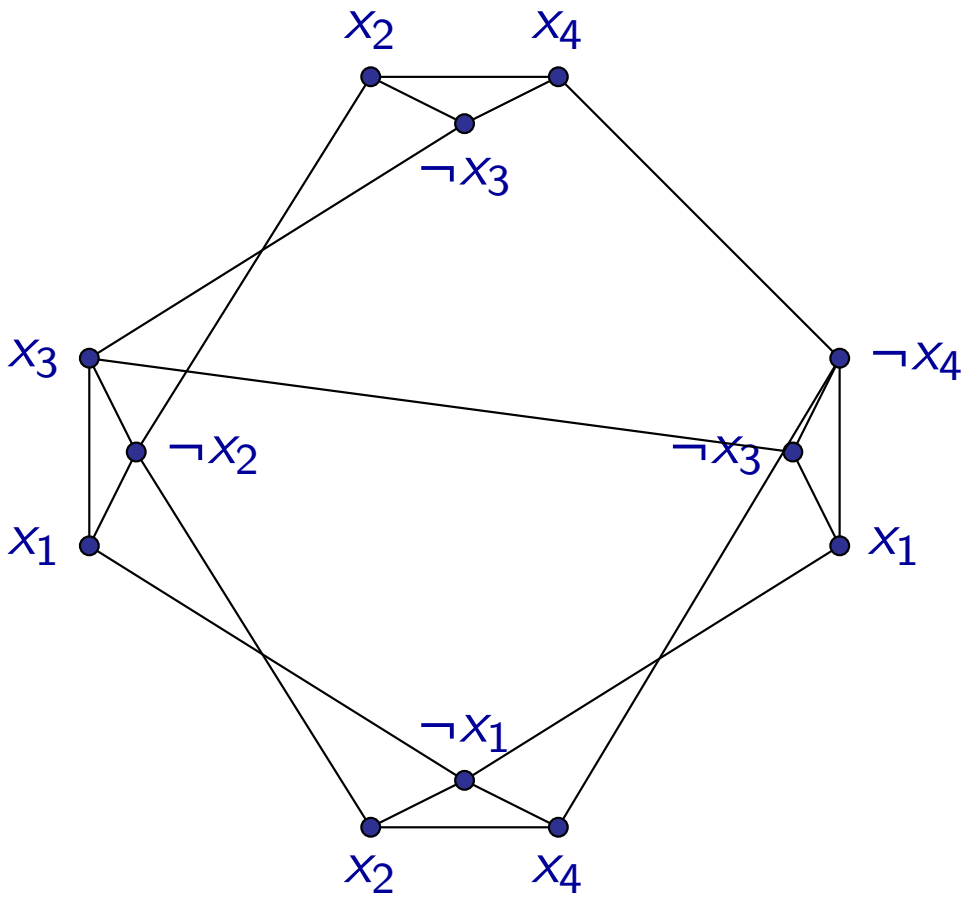


$k = 4$

Číslo k položíme rovno počtu klauzulí.

Převod 3-SAT na IS

$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$



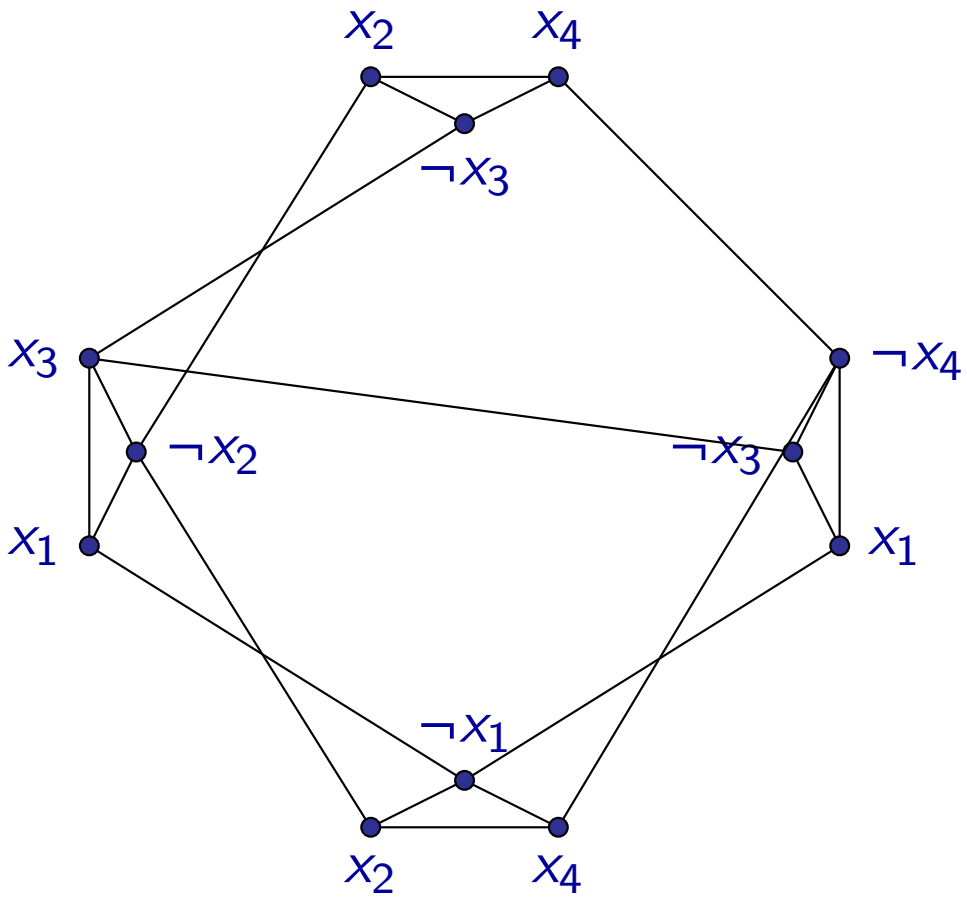
$k = 4$

Vytvořený graf a číslo k vydá algoritmus jako výstup.

Převod 3-SAT na IS

$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$

$v(x_1) = 1$
$v(x_2) = 1$
$v(x_3) = 0$
$v(x_4) = 1$



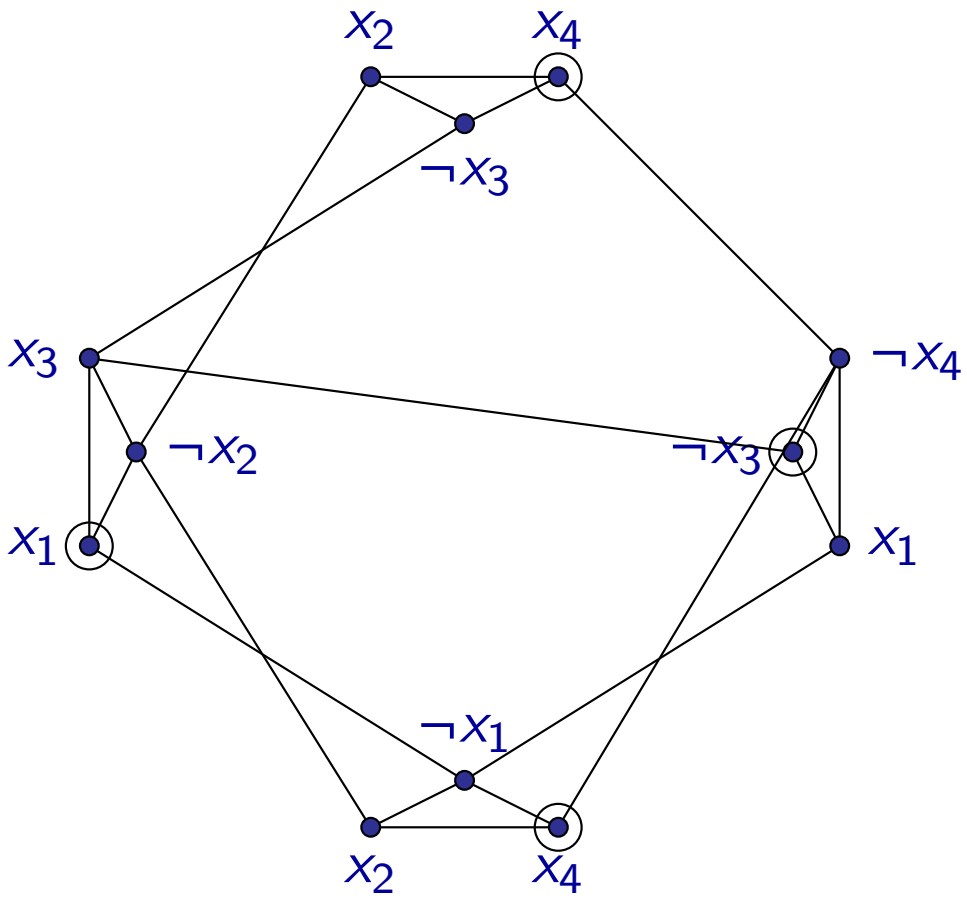
$k = 4$

Jestliže je formule φ splnitelná, existuje ohodnocení v, při kterém má v každé klauzuli alespoň jeden literál hodnotu 1.

Převod 3-SAT na IS

$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$

$v(x_1) = 1$
$v(x_2) = 1$
$v(x_3) = 0$
$v(x_4) = 1$



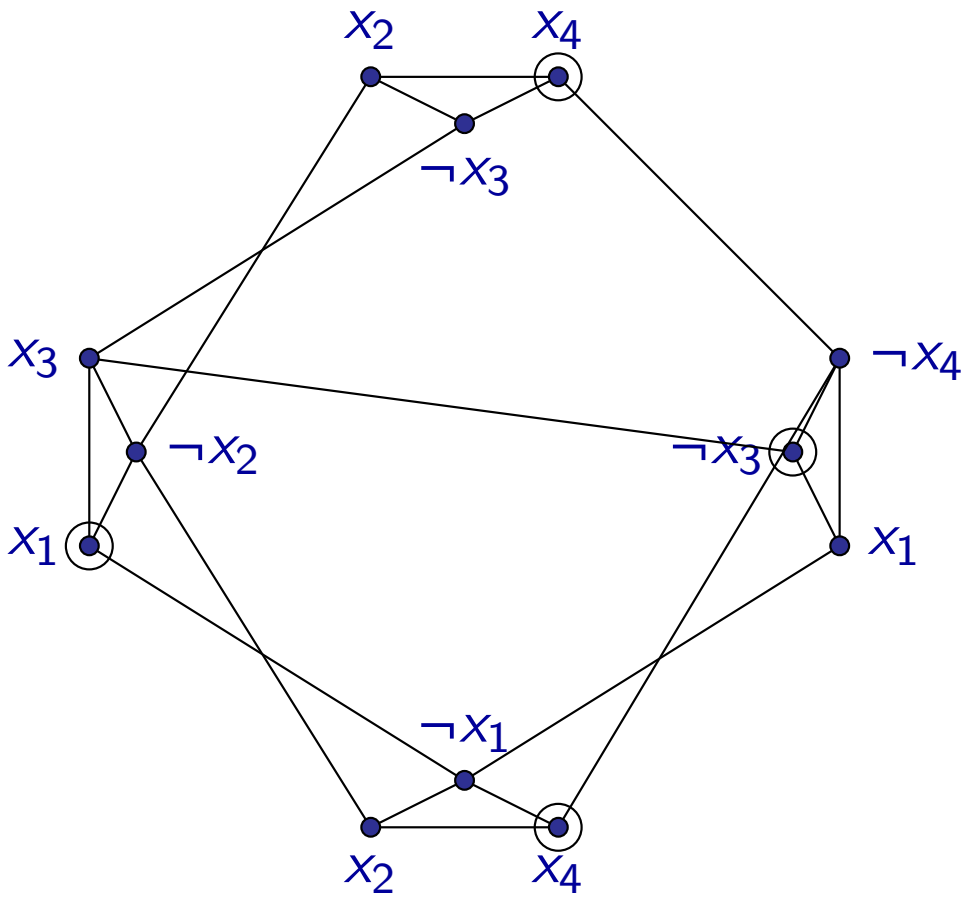
$k = 4$

Z každé klauzule vybereme jeden literál, který má při ohodnocení v hodnotu 1, a do nezávislé množiny přidáme odpovídající vrchol.

Převod 3-SAT na IS

$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$

$v(x_1) = 1$
$v(x_2) = 1$
$v(x_3) = 0$
$v(x_4) = 1$



$k = 4$

Lehce ověříme, že vybrané vrcholy tvoří nezávislou množinu.

Převod 3-SAT na IS

Vybrané vrcholy tvoří nezávislou množinu, protože:

- Z každé trojice vrcholů odpovídající jedné klauzuli byl vybrán jen jeden vrchol.
- Nemohly být současně vybrány vrcholy označené x_i a $\neg x_i$.
  (Při daném ohodnocení v má hodnotu 1 jen jeden z nich.)

Převod 3-SAT na IS

Na druhou stranu, pokud v grafu G existuje nezávislá množina velikosti k, musí určitě splňovat následující vlastnosti:

- Z každé trojice vrcholů odpovídající jedné klauzuli musí být vybrán nejvýše jeden vrchol.

  Protože je ale klauzulí k a je vybráno k vrcholů, musí být z každé takové trojice vybrán právě jeden.

- Nemohly být současně vybrány vrcholy označené x_i a $\neg x_i$.

Ohodnocení tedy zvolíme podle vybraných vrcholů, protože z předchozího vyplývá, že nehrozí, že by neexistovalo.
(Zbylým proměnným přiřadíme libovolné hodnoty.)

Při daném ohodnocení má formule φ určitě hodnotu 1, neboť v každé klauzuli má hodnotu 1 alespoň jeden literál.

Převod 3-SAT na IS

Popsaný algoritmus je určitě polynomiální:

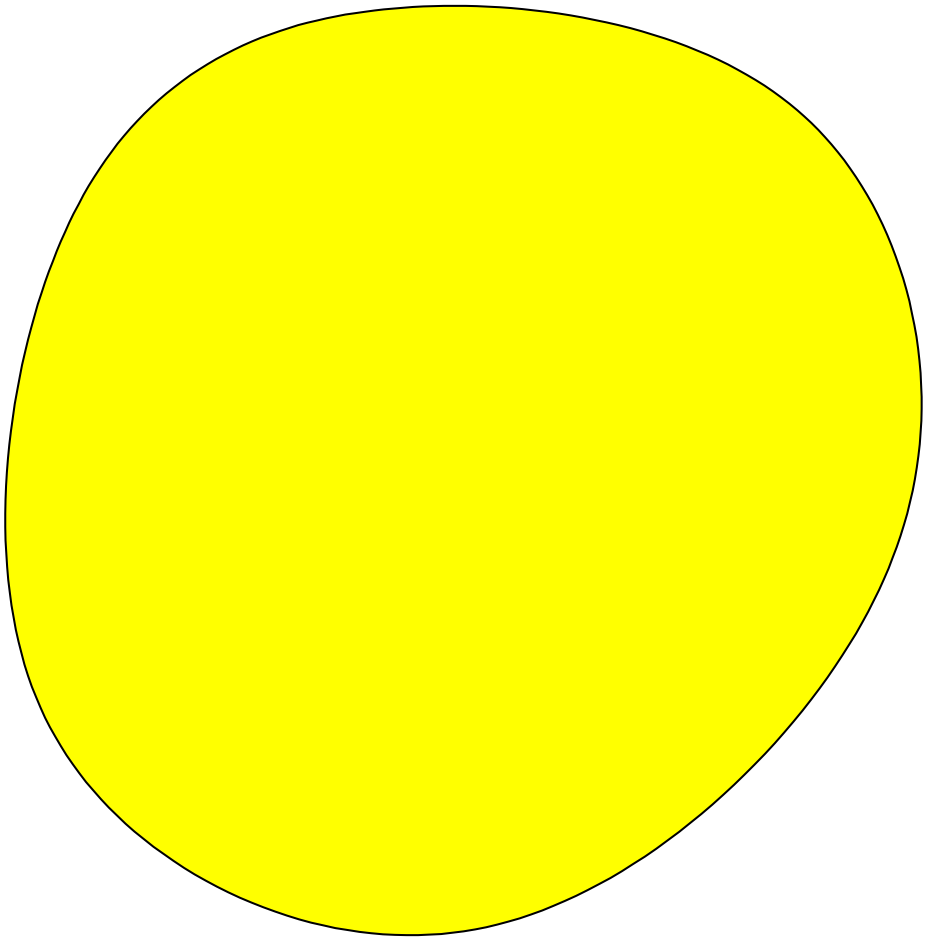
Graf G a číslo k je možné zkonstruovat k formuli φ v čase $O(n^2)$, kde n je velikost formule φ.

Navíc jsme viděli, že ve zkonstruovaném grafu G existuje nezávislá množina velikosti k právě tehdy, když formule φ je splnitelná.

Popsaný algoritmus tedy ukazuje, že problém 3-SAT je polynomiálně převeditelný na problém IS.

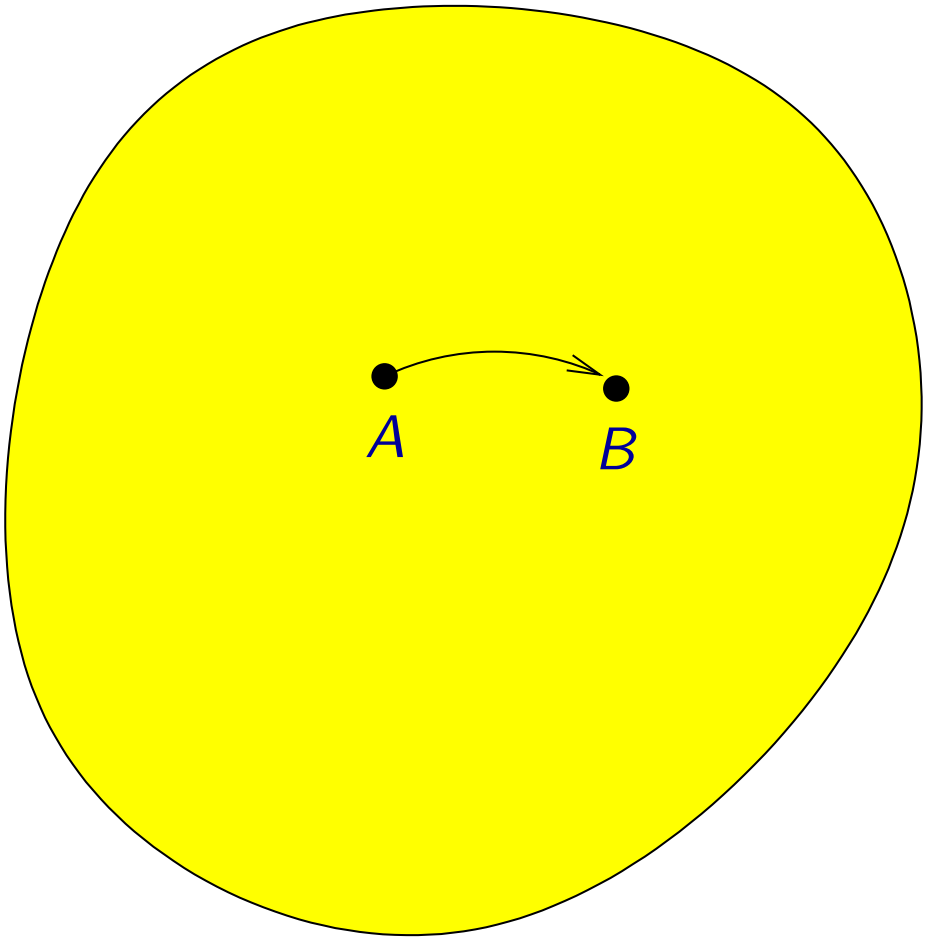
NP-úplné problémy

Vezměme si množinu všech možných rozhodovacích problémů.



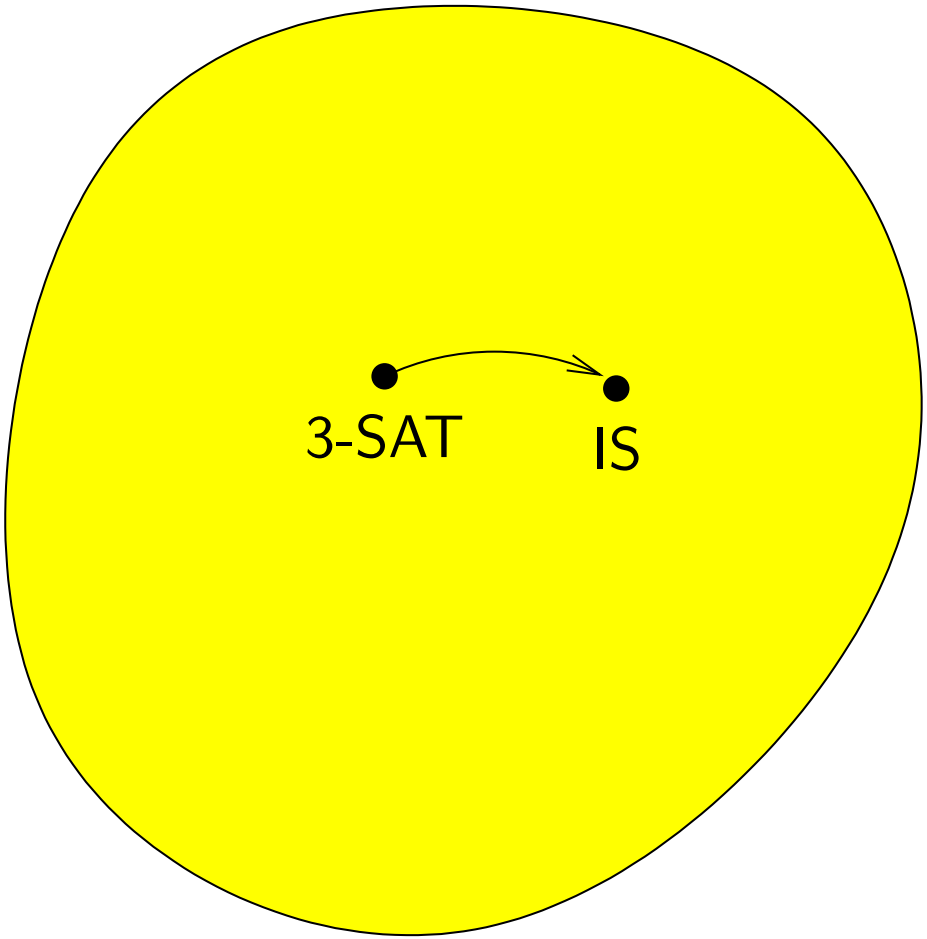
NP-úplné problémy

Šipkou si znázorníme, že problém A je polynomiálně převeditelný na problém B.



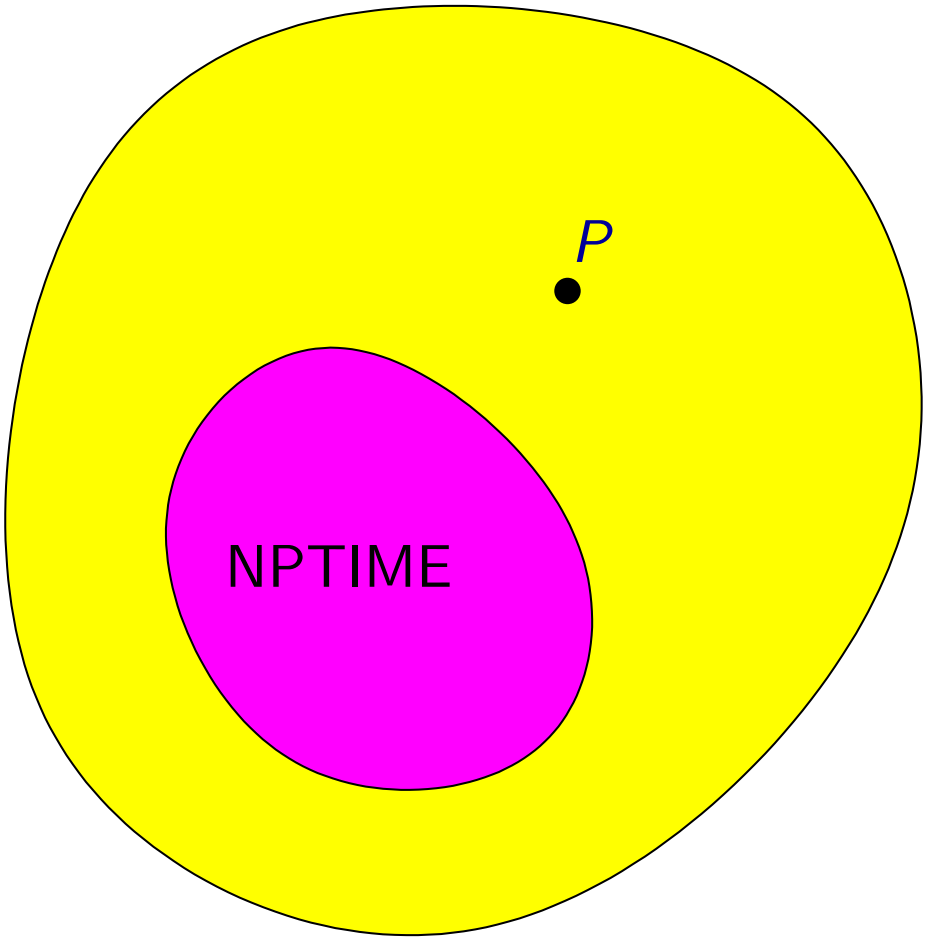
NP-úplné problémy

Například problém 3-SAT je polynomiálně převeditelný na problém IS.



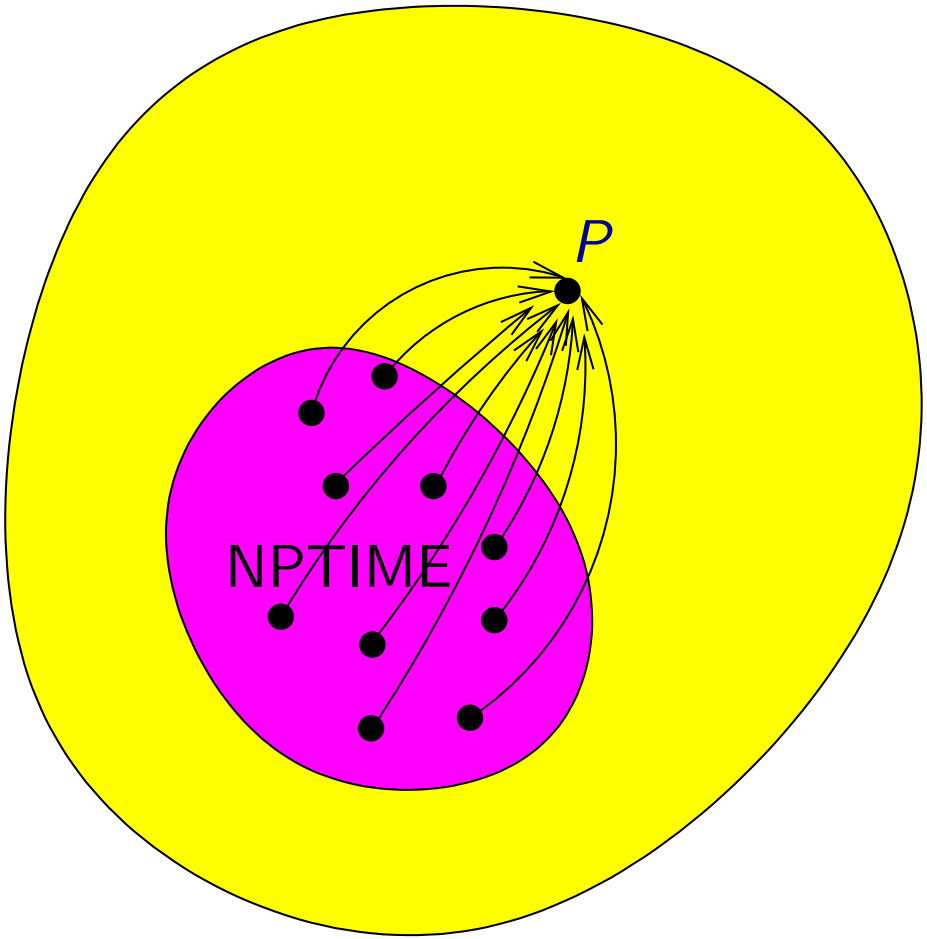
NP-úplné problémy

Vezměme si nyní třídu NPTIME a nějaký problém P.



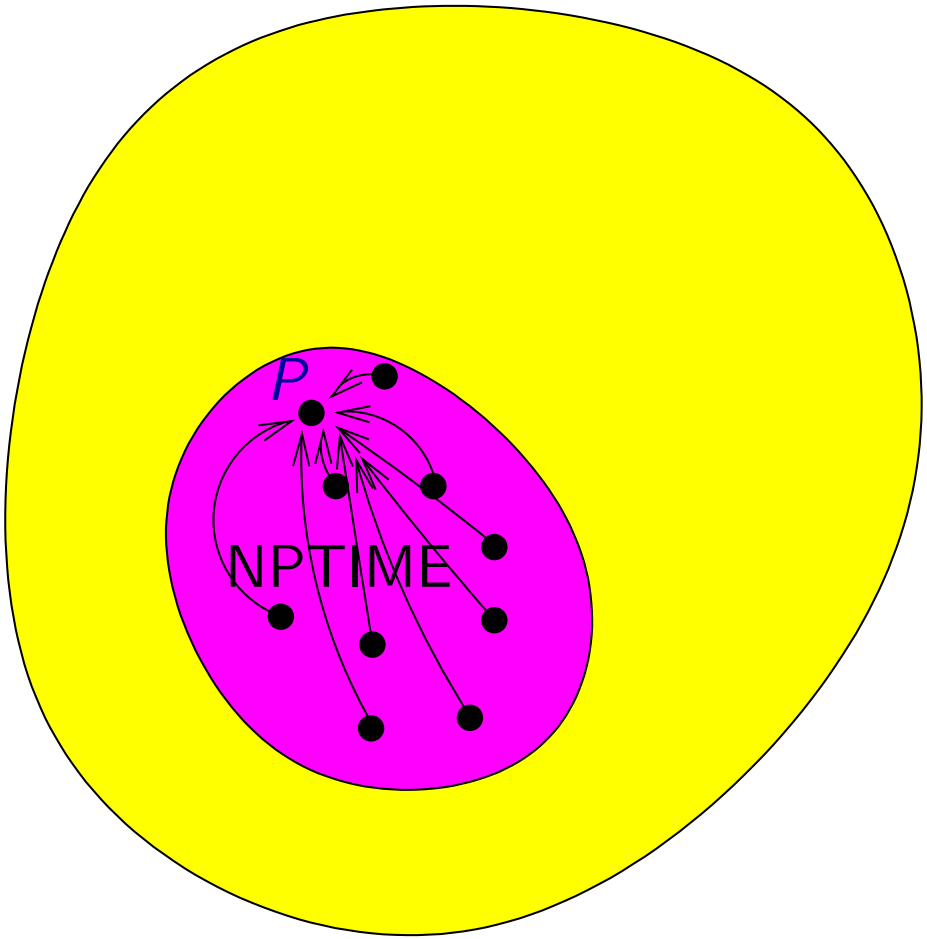
NP-úplné problémy

Problém P je NP-**těžký**, jestliže každý problém z NPTIME je polynomiálně převeditelný na P.



NP-úplné problémy

Problém P je NP-**úplný**, jestliže je NP-těžký a navíc sám patří do třídy NPTIME.



NP-úplné problémy

Pokud bychom pro nějaký NP-těžký problém P nalezli polynomiální algoritmus, získali bychom tím polynomiální algoritmus pro každý problém P' z NPTIME:

- Na vstup problému P' bychom nejprve aplikovali algoritmus realizující polynomiální převod z P' na P.
- Na vytvořenou instanci problému P bychom aplikovali polynomiální algoritmus řešící problém P a výsledek bychom vrátili jako odpověď pro danou instanci problému P'.

V takovém případě by tedy platilo PTIME = NPTIME, neboť pro každý problém z NPTIME by existoval polynomiální (deterministický) algoritmus.

NP-úplné problémy

Na druhou stranu, pokud existuje alespoň jeden problém z NPTIME, pro který neexistuje polynomiální algoritmus, tak z předchozího plyne, že pro žádný NP-těžký problém nemůže existovat polynomiální algoritmus.

Zda platí první nebo druhá možnost, je otevřený problém.

NP-úplné problémy

Není těžké si rozmyslet následující:

Pokud je problém A polynomiálně převeditelný na problém B a problém B je polynomiálně převeditelný na problém C, pak problém A je polynomiálně převeditelný na problém C.

Pokud tedy o nějakém problému P víme, že je NP-těžký a že P je polynomiálně převeditelný na problém P', pak víme, že i problém P' je NP-těžký.

NP-úplné problémy

Věta (Cook, 1971)

Problém SAT je NP-úplný.

Dá se ukázat, že SAT je polynomiálně převeditelný na 3-SAT a viděli jsme, že 3-SAT je polynomiálně převeditelný na IS.

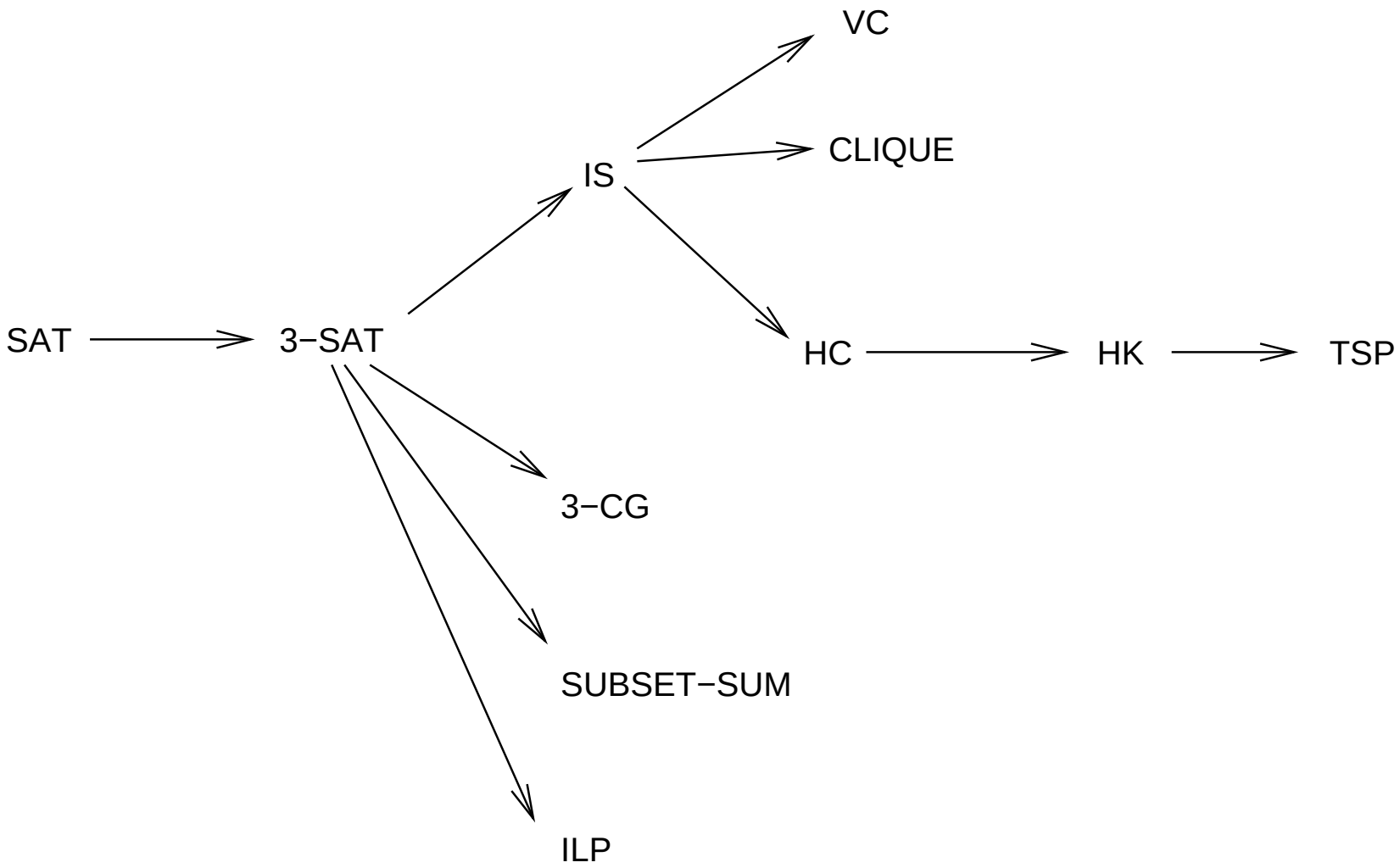
Z toho plyne, že problémy 3-SAT a IS jsou NP-těžké.

Není také těžké ukázat, že 3-SAT i IS patří do třídy NPTIME.

Problémy 3-SAT i IS jsou NP-úplné.

NP-úplné problémy

Polynomiálními převody z již známých NP-úplných problémů se dá ukázat NP-obtížnost celé řady různých dalších problémů:



Příklady některých NP-úplných problémů

Ze zatím uvedených problémů jsou NP-úplné následující tři problémy:

- SAT (splnitelnost booleovských formulí)
- 3-SAT
- IS — problém nezávislé množiny (independent set)

Na následujících slidech jsou uvedeny některé další NP-úplné problémy:

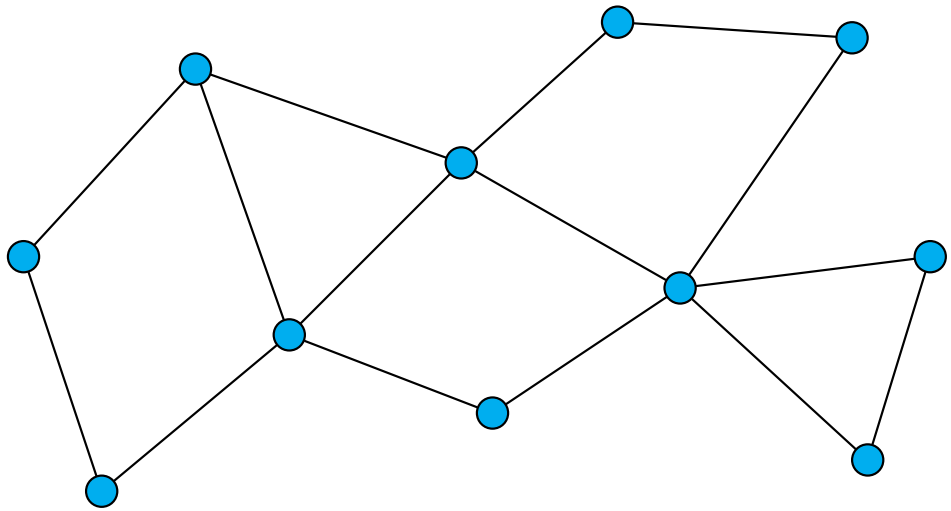
- CG — vrcholové barvení grafu (pozn.: je NP-úplný i ve speciálním případě, kdy máme právě 3 barvy)
- VC — vrcholové pokrytí grafu (vertex cover)
- CLIQUE — problém kliky
- HC — problém Hamiltonovského cyklu
- HK — problém Hamiltonovské kružnice
- TSP — problém obchodního cestujícího (traveling salesman problem)
- SUBSET-SUM
- ILP — celočíselné lineární programování (integer linear programming)

Barvení grafu

Vstup: Neorientovaný graf G, přirozené číslo k.

Otázka: Lze vrcholy grafu G obarvit k barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?

Příklad: $k = 3$

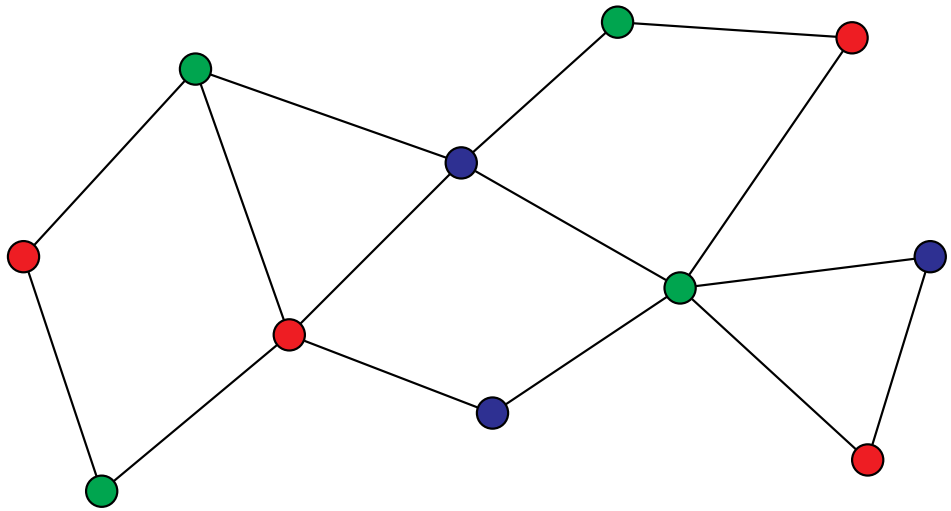


Barvení grafu

Vstup: Neorientovaný graf G, přirozené číslo k.

Otázka: Lze vrcholy grafu G obarvit k barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?

Příklad: $k = 3$



Odpověď: ANO

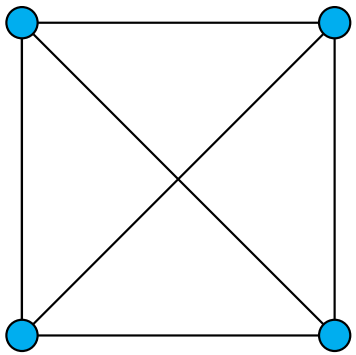
Barvení grafu

Barvení grafu

Vstup: Neorientovaný graf G, přirozené číslo k.

Otázka: Lze vrcholy grafu G obarvit k barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?

Příklad: $k = 3$

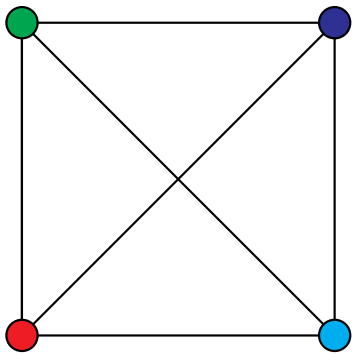


Barvení grafu

Vstup: Neorientovaný graf G, přirozené číslo k.

Otázka: Lze vrcholy grafu G obarvit k barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?

Příklad: $k = 3$



Odpověď: NE

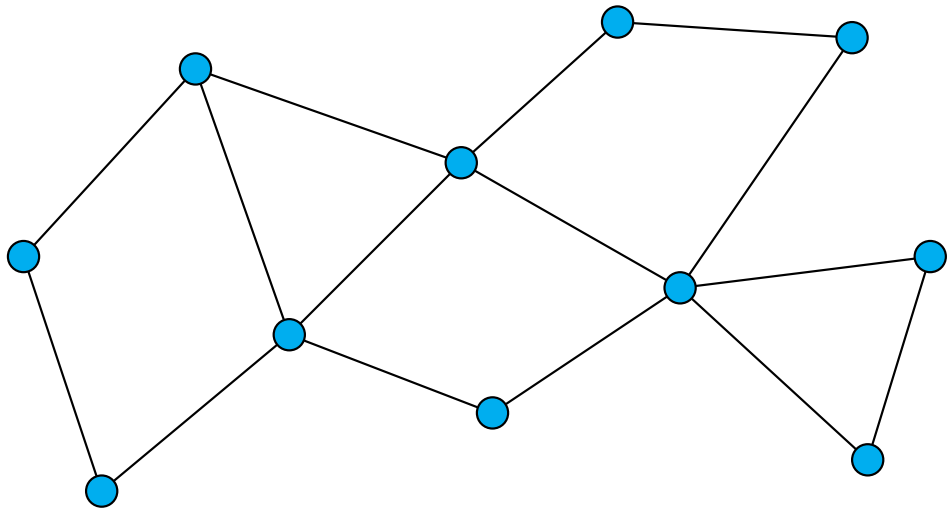
VC – Vrcholové pokrytí

VC – vrcholové pokrytí (vertex cover)

Vstup: Neorientovaný graf G a přirozené číslo k.

Otázka: Existuje v grafu G množina vrcholů velikosti k taková, že každá hrana má alespoň jeden svůj vrchol v této množině?

Příklad: $k = 6$



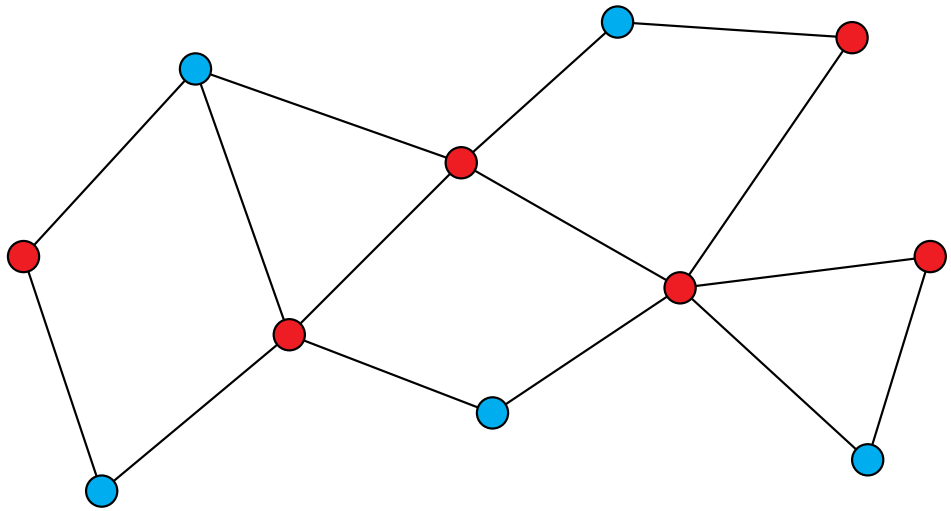
VC – Vrcholové pokrytí

VC – vrcholové pokrytí (vertex cover)

Vstup: Neorientovaný graf G a přirozené číslo k.

Otázka: Existuje v grafu G množina vrcholů velikosti k taková, že každá hrana má alespoň jeden svůj vrchol v této množině?

Příklad: $k = 6$



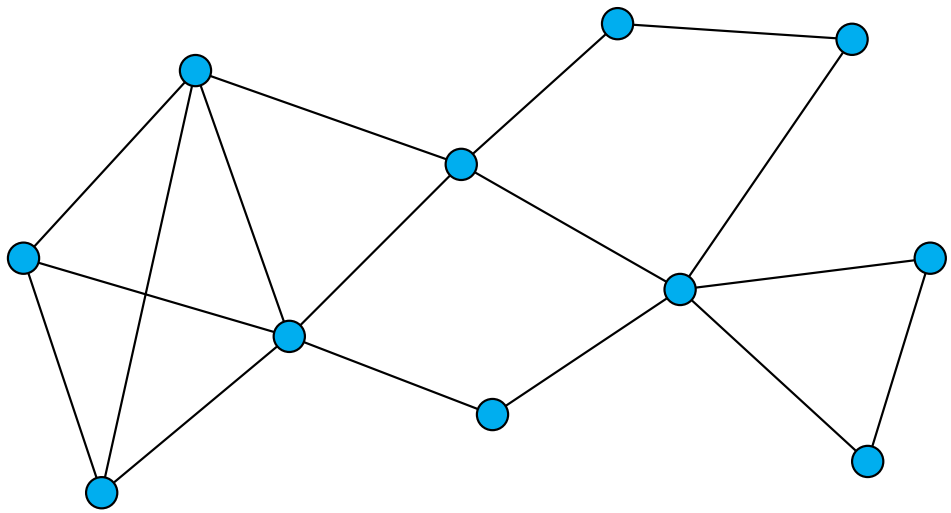
Odpověď: Ano

CLIQUE – problém kliky

Vstup: Neorientovaný graf G a přirozené číslo k.

Otázka: Existuje v grafu G množina vrcholů velikosti k taková, že každé dva vrcholy této množiny jsou spojeny hranou?

Příklad: $k = 4$

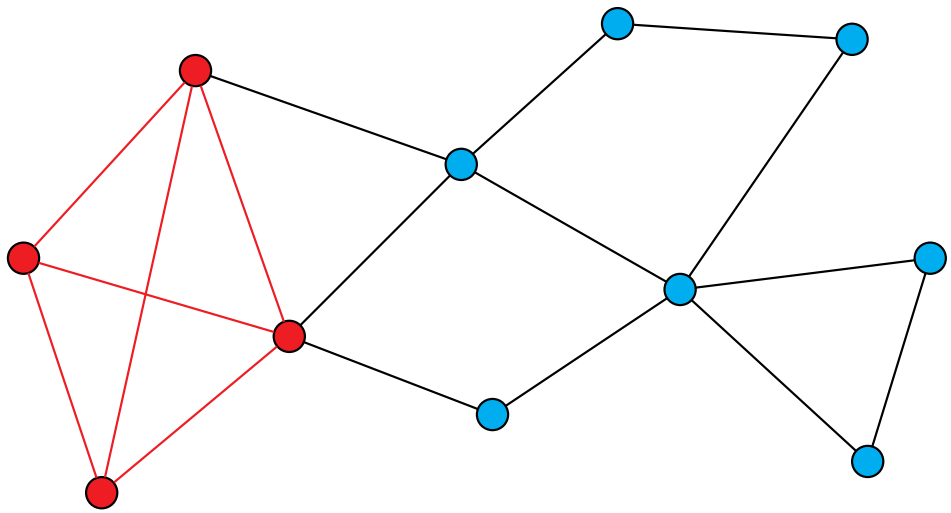


CLIQUE – problém kliky

Vstup: Neorientovaný graf G a přirozené číslo k.

Otázka: Existuje v grafu G množina vrcholů velikosti k taková, že každé dva vrcholy této množiny jsou spojeny hranou?

Příklad: $k = 4$



Odpověď: ANO

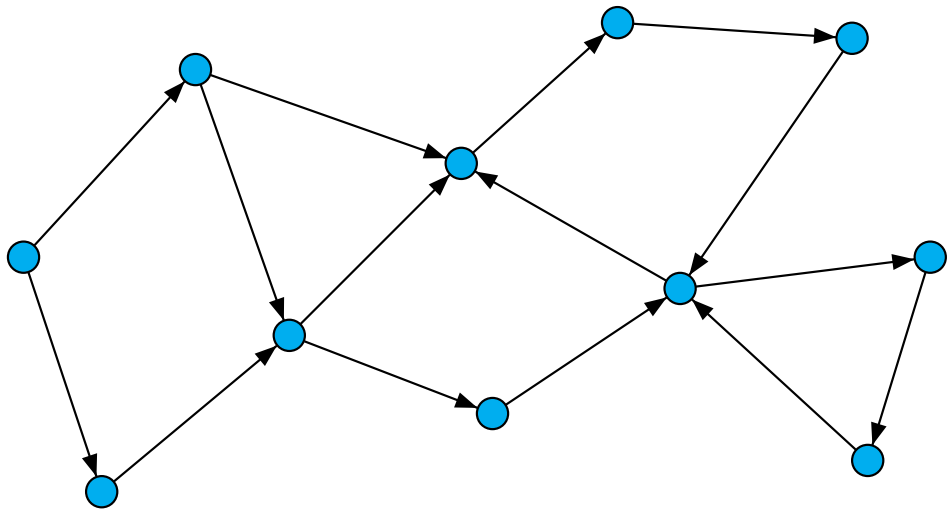
Hamiltonovský cyklus

HC – Problém „Hamiltonovský cyklus“

Vstup: Orientovaný graf G.

Otázka: Existuje v grafu G Hamiltonovský cyklus (orientovaný cyklus procházející každým vrcholem právě jednou)?

Příklad:



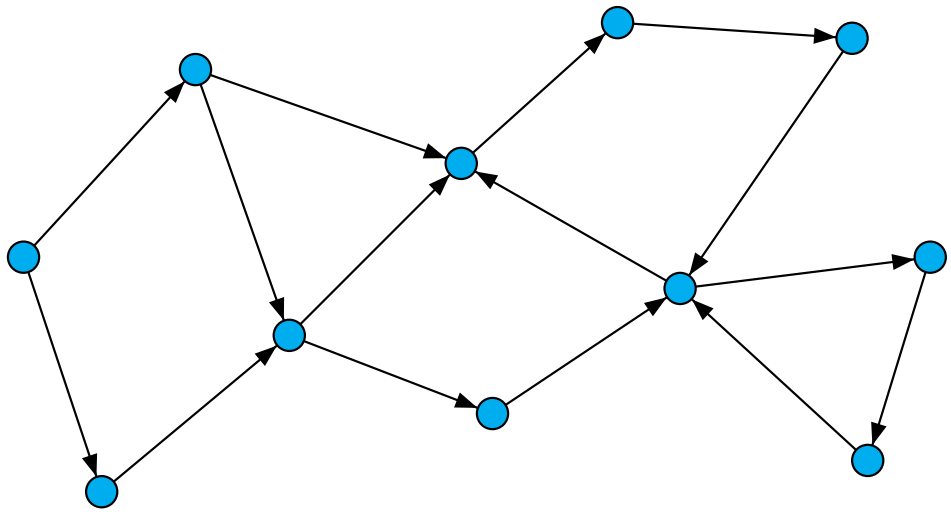
Hamiltonovský cyklus

HC – Problém „Hamiltonovský cyklus“

Vstup: Orientovaný graf G.

Otázka: Existuje v grafu G Hamiltonovský cyklus (orientovaný cyklus procházející každým vrcholem právě jednou)?

Příklad:



Odpověď: NE

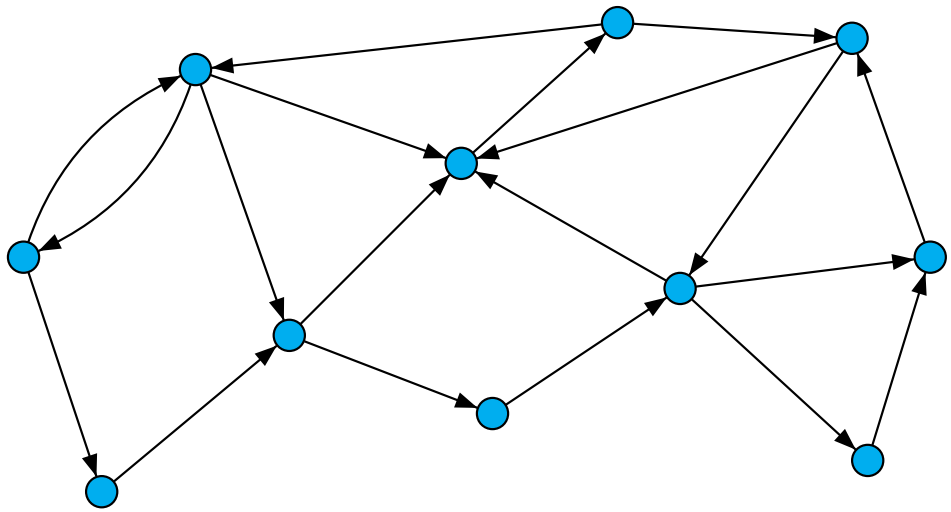
Hamiltonovský cyklus

HC – Problém „Hamiltonovský cyklus“

Vstup: Orientovaný graf G.

Otázka: Existuje v grafu G Hamiltonovský cyklus (orientovaný cyklus procházející každým vrcholem právě jednou)?

Příklad:



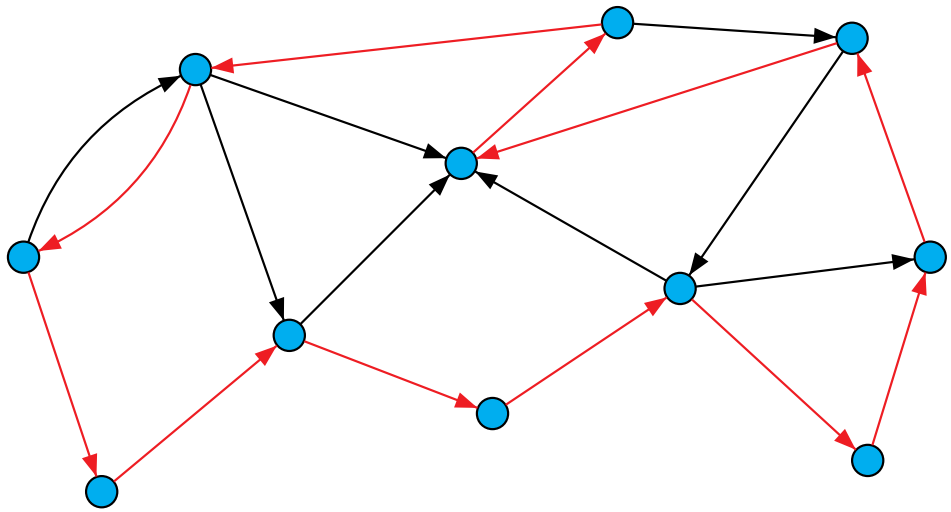
Hamiltonovský cyklus

HC – Problém „Hamiltonovský cyklus“

Vstup: Orientovaný graf G.

Otázka: Existuje v grafu G Hamiltonovský cyklus (orientovaný cyklus procházející každým vrcholem právě jednou)?

Příklad:



Odpověď: Ano

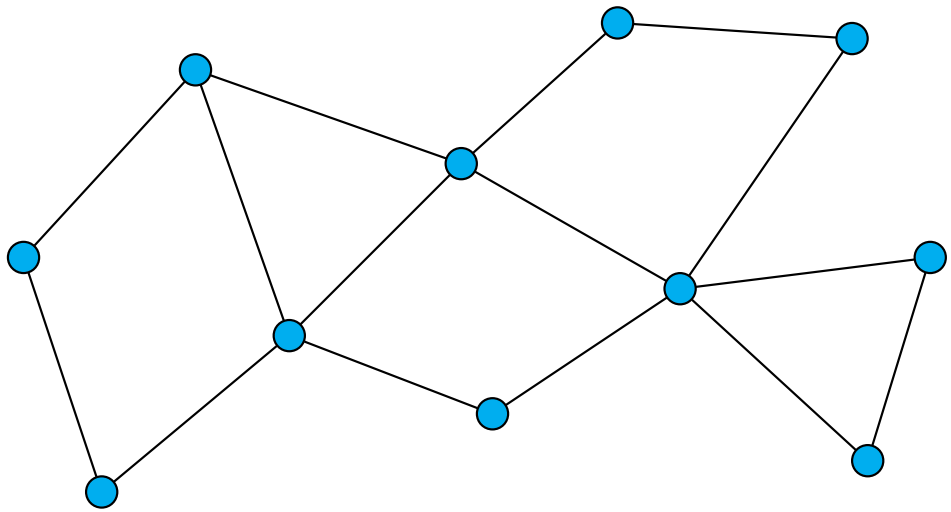
Hamiltonovská kružnice

HK – Problém „Hamiltonovská kružnice“

Vstup: Neorientovaný graf G.

Otázka: Existuje v grafu G Hamiltonovská kružnice (neorientovaný cyklus procházející každým vrcholem právě jednou)?

Příklad:



Odpověď: NE

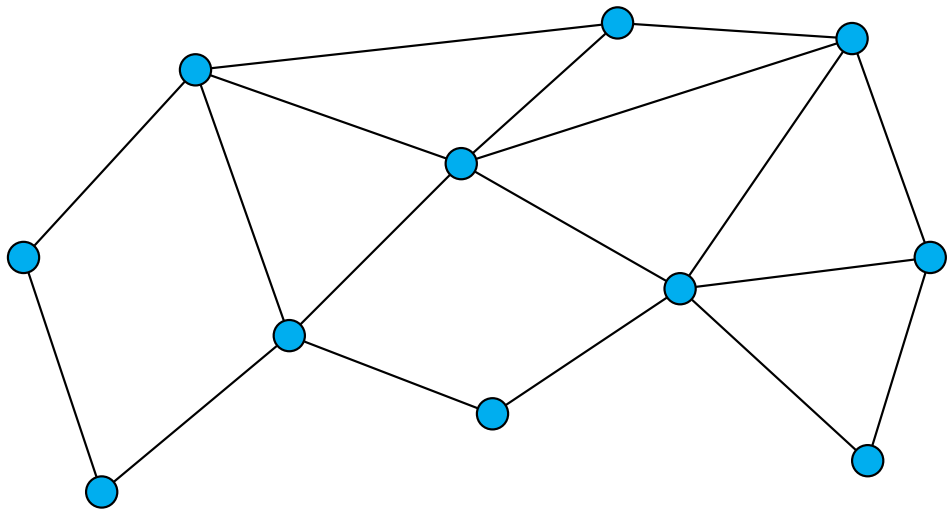
Hamiltonovská kružnice

HK – Problém „Hamiltonovská kružnice“

Vstup: Neorientovaný graf G.

Otázka: Existuje v grafu G Hamiltonovská kružnice (neorientovaný cyklus procházející každým vrcholem právě jednou)?

Příklad:



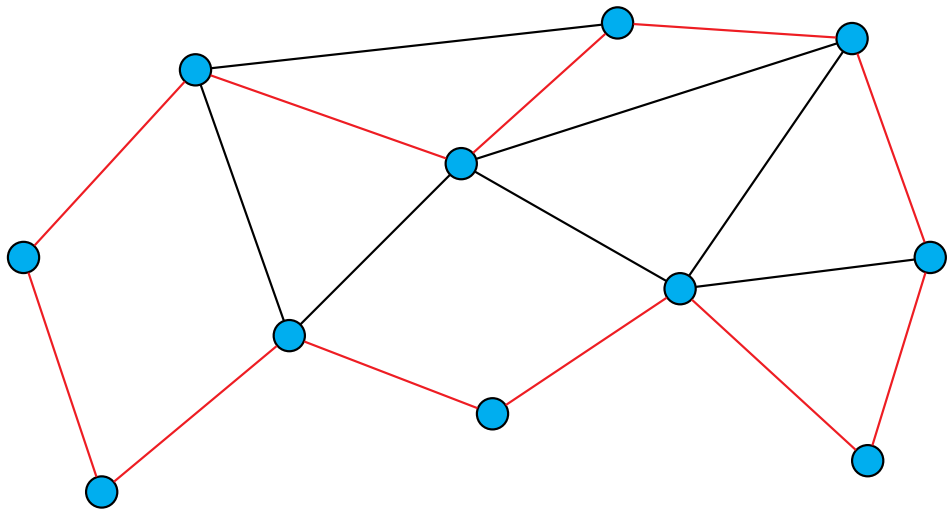
Hamiltonovská kružnice

HK – Problém „Hamiltonovská kružnice“

Vstup: Neorientovaný graf G.

Otázka: Existuje v grafu G Hamiltonovská kružnice (neorientovaný cyklus procházející každým vrcholem právě jednou)?

Příklad:



Odpověď: Ano

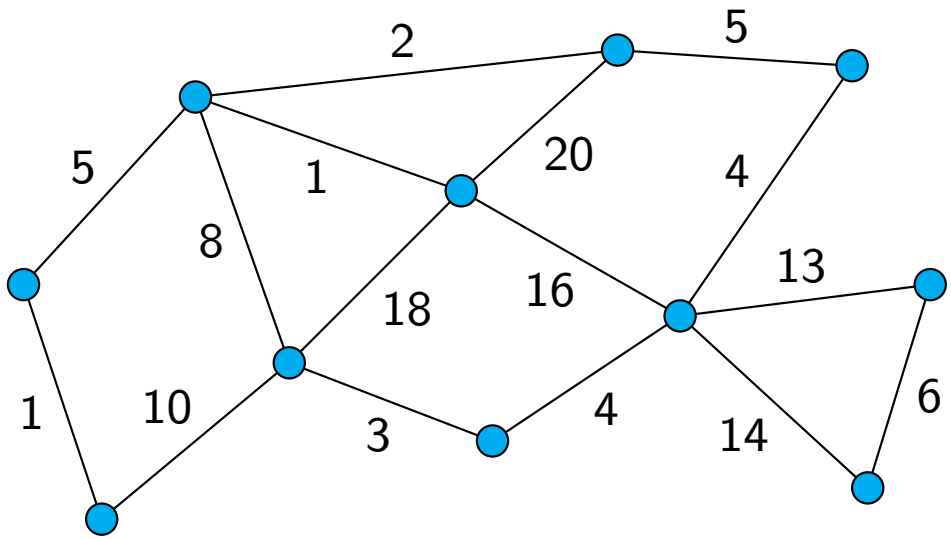
Problém obchodního cestujícího

TSP - Problém „obchodního cestujícího“

Vstup: Neorientovaný graf G s hranami ohodnocenými přirozenými čísly a číslo k.

Otázka: Existuje v grafu G uzavřená cesta procházející všemi vrcholy takový, že součet délek hran na této cestě (včetně opakovaných) je maximálně k?

Příklad: $k = 70$



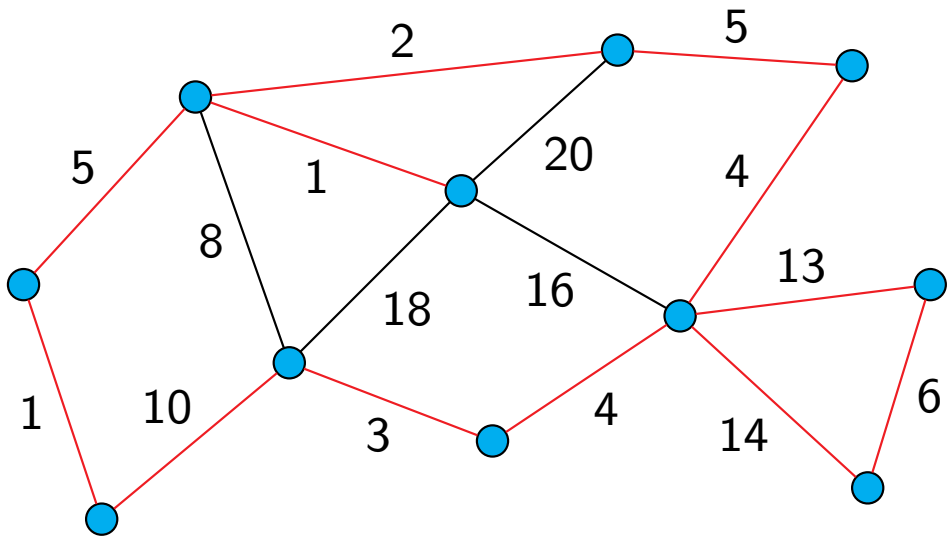
Problém obchodního cestujícího

TSP - Problém „obchodního cestujícího“

Vstup: Neorientovaný graf G s hranami ohodnocenými přirozenými čísly a číslo k.

Otázka: Existuje v grafu G uzavřená cesta procházející všemi vrcholy takový, že součet délek hran na této cestě (včetně opakovaných) je maximálně k?

Příklad: $k = 70$



Odpověď: ANO, protože byla nalezena cesta se součtem 69.

SUBSET-SUM

Problém SUBSET-SUM

Vstup: Sekvence přirozených čísel $a_1, a_2, \ldots, a_n$ a přirozené číslo s.

Otázka: Existuje množina $I \subseteq \{1, 2, \ldots, n\}$ taková, že $\sum_{i \in I} a_i = s$?

Jinak řečeno, ptáme se zda z dané (multi)množiny čísel je možné vybrat podmnožinu, jejíž součet je s.

Příklad: Pro vstup tvořený čísly $3, 5, 2, 3, 7$ a číslem $s = 15$ je odpověď ANO, neboť $3 + 5 + 7 = 15$.

Pro vstup tvořený čísly $3, 5, 2, 3, 7$ a číslem $s = 16$ je odpověď NE, neboť žádná podmnožina těchto čísel nedává součet 16.

SUBSET-SUM

Poznámka:
Pořadí čísel $a_1, a_2, \ldots, a_n$ na vstupu není důležité.

Všimněte si však určitého rozdílu oproti tomu, kdybychom problém formulovali tak, že vstupem je množina $\{a_1, a_2, \ldots, a_n\}$ a číslo s — v množině se čísla neopakují, zatímco v sekvenci se může totéž číslo vyskytnout vícekrát.

SUBSET-SUM

Problém SUBSET-SUM je speciálním případem **problému batohu** (knapsack problem):

Knapsack problem

Vstup: Sekvence dvojic přirozených čísel $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$ a dvě přirozená čísla s a t.

Otázka: Existuje množina $I \subseteq \{1, 2, \ldots, n\}$ taková, že $\sum_{i \in I} a_i \leq s$ a $\sum_{i \in I} b_i \geq t$?

SUBSET-SUM

Neformálně můžeme problém batohu formulovat takto:

Máme n předmětů, kde i-tý předmět váží a_i gramů a má cenu b_i Kč. Do batohu se vejdou předměty o maximální celkové váze s gramů.

Otázka zní, zda můžeme z předmětů vybrat podmnožinu, která by vážila maximálně s gramů a měla celkovou cenu alespoň t Kč.

Poznámka:
Zde jsme problém batohu formulovali jako rozhodovací problém.

Běžnější je formulovat tento problém jako optimalizační problém, kde je cílem najít takovou množinu $I \subseteq \{1, 2, \ldots, n\}$, kde hodnota $\sum_{i \in I} b_i$ je maximální, přičemž ovšem musí být dodržena podmínka $\sum_{i \in I} a_i \leq s$, tj. vybrat předměty s maximální celkovou cenou tak, aby nebyla překročena kapacita batohu.

SUBSET-SUM

To, že SUBSET-SUM je speciálním případem problému batohu, vidíme z následující jednoduché konstrukce:

Řekněme, že $a_1, a_2, \ldots, a_n$, s_1 je instance problému SUBSET-SUM. Je očividné, že pro instanci problému batohu, kde máme sekvenci $(a_1, a_1), (a_2, a_2), \ldots, (a_n, a_n)$, $s = s_1$ a $t = s_1$, je odpověď stejná jako pro původní instanci SUBSET-SUM.

SUBSET-SUM

Pokud chceme studovat složitost problémů jako jsou SUBSET-SUM nebo problém batohu, je dobré si nejprve ujasnit, co považujeme za velikost vstupu.

Asi nejpřirozenější je definovat velikost vstupu jako celkový počet bitů, který potřebujeme k zápisu instance.

Musíme však určit, jakým způsobem jsou na vstupu zadána přirozená čísla – zda binárně (případně v jiné číselné soustavě o základu alespoň 2, např. desítkové nebo šestnáctkové) nebo unárně.

- Pokud počítáme velikost vstupu jako celkový počet bitů při použití **binárního** zápisu čísel, tak pro problém SUBSET-SUM není znám polynomiální algoritmus.
- Pokud počítáme velikost vstupu jako celkový počet bitů při použití **unárního** zápisu, tak existuje pro problém SUBSET-SUM algoritmus s polynomiální časovou složitostí.

ILP – celočíselné lineární programování

Problém ILP (celočíselné lineární programování)

Vstup: Celočíselná matice A a celočíselný vektor b.

Otázka: Existuje celočíselný vektor x, takový že $Ax \leq b$?

Příklad instance problému:

$$A = \begin{pmatrix} 3 & -2 & 5 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix} \qquad b = \begin{pmatrix} 8 \\ -3 \\ 5 \end{pmatrix}$$

Ptáme se tedy, zda existuje celočíselné řešení následující soustavy nerovnic:

$$\begin{aligned} 3x_1 - 2x_2 + 5x_3 &\leq 8 \\ x_1 + x_3 &\leq -3 \\ 2x_1 + x_2 &\leq 5 \end{aligned}$$

Jedním z řešení soustavy

$$\begin{aligned} 3x_1 - 2x_2 + 5x_3 &\leq 8 \\ x_1 + x_3 &\leq -3 \\ 2x_1 + x_2 &\leq 5 \end{aligned}$$

je například $x_1 = -4$, $x_2 = 1$, $x_3 = 1$, tj.

$$x = \begin{pmatrix} -4 \\ 1 \\ 1 \end{pmatrix}$$

neboť

$$\begin{aligned} 3 \cdot (-4) - 2 \cdot 1 + 5 \cdot 1 &= -9 \leq 8 \\ -4 + 1 &= -3 \leq -3 \\ 2 \cdot (-4) + 1 &= -7 \leq 5 \end{aligned}$$

Pro tuto instanci je tedy odpověď Ano.

ILP – celočíselné lineární programování

Poznámka: Analogický problém, kdy se pro danou soustavu lineálních nerovnic ptáme, zda existuje její řešení v oboru **reálných** čísel, je možné řešit v polynomiálním čase.

PSPACE-úplné problémy, EXPTIME-úplné problémy, . . .

- Problém P je PSPACE-**těžký**, jestliže je každý problém P' z PSPACE polynomiálně převeditelný na problém P.
- Problém P je PSPACE-**úplný**, jestliže je PSPACE-těžký a navíc sám patří do třídy PSPACE.

- Problém P je EXPTIME-**těžký**, jestliže je každý problém P' z EXPTIME polynomiálně převeditelný na problém P.
- Problém P je EXPTIME-**úplný**, jestliže je EXPTIME-těžký a navíc sám patří do třídy EXPTIME.

$\vdots$

PSPACE-úplné problémy, EXPTIME-úplné problémy, ...

Obecně pro libovolnou třídu složitosti $\mathcal{C}$ můžeme zavést třídy **$\mathcal{C}$-těžkých** a **$\mathcal{C}$-úplných** problémů:

Definice

- Problém P je **$\mathcal{C}$-těžký**, jestliže je každý problém P' ze třídy $\mathcal{C}$ polynomiálně převeditelný na problém P.
- Problém P je **$\mathcal{C}$-úplný**, jestliže je $\mathcal{C}$-těžký a navíc sám patří do třídy $\mathcal{C}$.

Kromě NP-úplných problémů tak máme PSPACE-úplné problémy, EXPTIME-úplné problémy, EXPSPACE-úplné problémy, 2-EXPTIME-úplné problémy, ...

Obecně se dá říci, že $\mathcal{C}$-úplné problémy jsou vždy ty nejtěžší problémy v dané třídě $\mathcal{C}$.

PTIME-úplné problémy, NL-úplné problémy, . . .

Poznámka: Výše uvedeným způsobem zavedené pojmy $\mathcal{C}$-těžkých a $\mathcal{C}$-úplných problémů, kdy byl v definici použit pojem **polynomiální převoditelnosti**, nedávají příliš smysl pro třídu PTIME a další třídy, které jsou jejími podmnožinami (jako třeba NLOGSPACE).

Pro takové třídy se zavádí pojmy $\mathcal{C}$-těžké a $\mathcal{C}$-úplné problémy podobným způsobem jako v předchozích definicích, ale místo polynomiální redukcí se používají, tzv. **logspace redukce**:

- algoritmus realizují daný převod musí být deterministický a mít logaritmickou prostorovou složitost

Tímto způsobem se zavádí například:

- PTIME-úplné a PTIME-těžké problémy
- NLOGSPACE-úplné a NLOGSPACE-těžké problémy (většinou se označují kratším názvem jako NL-úplné a NL-těžké)

Příklad NL-úplného problému

Typický příklad NL-úplného problému:

Dosažitelnost v grafu

Vstup: Orientovaný graf G a dva jeho vrcholy s a t.

Otázka: Existuje v grafu G cesta z vrcholu s do vrcholu t?

Příklad PTIME-úplného problému

Typický příklad PTIME-úplného problému:

Circuit Value Problem

Vstup: Acyklický booleovský obvod C skládající se z hradel a vodičů a booleovské hodnoty $x_1, x_2, \ldots, x_n$ na vstupech tohoto obvodu.

Otázka: Bude na výstupu obvodu C při daných hodnotách vstupů hodnota 1 ?

Příklady PSPACE-úplných problémů

Typickým příkladem PSPACE-úplného problému je problém **kvantifikovaných booleovských formulí** — **QBF** (Quantified Boolean Formulas):

QBF

Vstup: Kvantifikovaná booleovská formule tvaru

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdots \exists x_{n-1} \forall x_n : \varphi,$$

kde φ je (běžná) booleovská formule obsahující proměnné $x_1, x_2, \ldots, x_n$.

Otázka: Je daná formule pravdivá?

Příklady PSPACE-úplných problémů

EqNFA

Vstup: Nedeterministické konečné automaty $\mathcal{A}_1$ a $\mathcal{A}_2$.

Otázka: Je $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$?

Univerzalita NKA

Vstup: Nedeterministický konečný automat $\mathcal{A}$.

Otázka: Je $\mathcal{L}(\mathcal{A}) = \Sigma^*$?

Příklady PSPACE-úplných problémů

EqRE

Vstup: Regulární výrazy α_1 a α_2.

Otázka: Je $\mathcal{L}(\alpha_1) = \mathcal{L}(\alpha_2)$?

Univerzalita RV

Vstup: Regulární výraz α.

Otázka: Je $\mathcal{L}(\alpha) = \Sigma^*$?

Příklady PSPACE-úplných problémů

Uvažujme následující hru, kterou hrají dva hráči na orientovaném grafu G:

- Hráči střídavě přesunují po vrcholech grafu G jeden hrací kámen.
- Při tazích se označují vrcholy, které již byly kamenem navštíveny.
- Začíná se na specifikovaném vrcholu v_0.
- Řekněme, že kámen je momentálně na vrcholu v. Hráč, který je na tahu, vybere vrchol v' takový, že existuje hrana z v do v' a vrchol v' nebyl dosud navštíven.
- Hráč, který nemůže táhnout, prohrál a jeho protivník vyhrál.

Generalized Geografy

Vstup: Orientovaný graf G s vyznačeným počátečním vrcholem v_0.

Otázka: Má hráč, který táhne jako první, vyhrávající strategii ve hře hrané na grafu G, kde se začíná ve vrcholu v_0 ?

Příklady EXPTIME-úplných problémů

Typický příklad EXPTIME-úplného problému:

Vstup: Turingův stroj $\mathcal{M}$, slovo w a číslo k zapsané binárně.

Otázka: Zastaví se výpočet stroje $\mathcal{M}$ nad slovem w do k kroků?
(Tj. udělá stroj $\mathcal{M}$ při výpočtu nad slovem w
nejvýše k kroků?)

Příklady EXPTIME-úplných problémů

Další příklady EXPTIME-úplných problémů jsou například zobecněné varianty her jako jsou šachy, dáma nebo Go, hrané na hrací ploše libovolné velikosti (např. šachovnice velikosti $n \times n$):

- vstupem je pozice v dané hře (např. v šachu konkrétní rozestavení figurek na šachovnici a informace, který hráč je na tahu)
- otázka je, zda má hráč, který je momentálně na tahu, v dané pozici vyhrávající strategii

Příklady EXPSPACE-úplných problémů

Regulární výrazy s mocněním jsou definovány podobně jako běžné regulární výrazy, ale kromě operátorů $+$, $\cdot$ a * mohou navíc obsahovat unární operátor ^{2} s následujícím významem:

- α^2 je zkratkou pro $\alpha \cdot \alpha$.

Následující dva problémy jsou EXPSPACE-úplné:

Vstup: Regulární výrazy s mocněním α_1 a α_2.
Otázka: Je $\mathcal{L}(\alpha_1) = \mathcal{L}(\alpha_2)$?

Vstup: Regulární výraz s mocněním α.
Otázka: Je $\mathcal{L}(\alpha) = \Sigma^*$?

Presburgerova aritmetika

Příklad problému, který je sice rozhodnutelný, ale má velkou výpočetní složitost:

Problém

Vstup: Uzavřená formule predikátové logiky (prvního řádu), ve které mohou být použity jako predikátové symboly pouze $=$ a $<$, jako funkční symbol pouze $+$ a jako kostantní symboly pouze 0 a 1.

Otázka: Je daná formule pravdivá v oboru přirozených čísel (při přirozené interpretaci všech funkčních a predikátových symbolů)?

Pro tento problém je znám deterministický algoritmus s časovou složitostí $2^{2^{2^{O(n)}}}$ a je rovněž známo, že každý nedeterministický algoritmus řešící tento problém, musí mít časovou složitost nejméně $2^{2^{\Omega(n)}}$.