

Formální jazyky

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

Obecně se teorie formálních jazyků zabývá problémy, kde:

- Pracujeme se sekvencemi **znaků** (říkáme též **symbolů**).
- Znak patří do nějaké konečné **abecedy**.
- Musíme rozpoznávat ty sekvence znaků, které nějakou vlastnost mají a ty co ji nemají.

V teorii formálních jazyků se sekvencím znaků říká **slova**.

Množině slov z nějaké abecedy se říká **jazyk**.

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:

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

Slova v abecedě Σ : AHOJ ABRACADABRA ERROR

Příklad 2:

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

Slovo v abecedě Σ_2 : HELLO_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 $\Sigma_5 = \{a, b\}$: *aababb*, *abbabbba*, *aaab*

Příklad 6:

Abeceda Σ_6 je množina všech ASCII znaků.

Příklad slova:

```
class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, world!");  
    }  
}
```

```
class_HelloWorld_{ ← _ _ _ _ _ public _ static _ void _ main(Str...
```

Když chceme nějaký jazyk popsat, máme několik možností:

- Můžeme vyjmenovat všechna jeho slova (což je ale použitelné jen pro malé konečné jazyky).

Příklad: $L = \{aab, babba, aaaaaa\}$

- Můžeme specifikovat nějakou vlastnost, kterou mají právě ta slova, která do tohoto jazyka patří:

Příklad: Jazyk nad abecedou $\{0, 1\}$, obsahující všechna slova, ve kterých je počet výskytů symbolu 1 sudý.

V teorii formálních jazyků se používají především následující dva přístupy:

- Popsat (idealizovaný) stroj, zařízení, algoritmus, který rozpozná slova patřící do daného jazyka – vede k použití tzv. **automatů**.
- Popsat nějaký mechanismus umožňující generovat všechna možná slova patřící do daného jazyka – vede k tzv. **gramatikám** a **regulárním výrazům**.
(budeme se jim věnovat později)

Některé základní pojmy

Délka slova je počet znaků ve slově.

Například délka slova *abaab* je 5.

Délku slova w označujeme $|w|$.

Pokud tedy např. $w = abaab$, pak $|w| = 5$.

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

Pro slovo $w = ababb$ tedy platí $|w|_a = 2$ a $|w|_b = 3$.

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

Prázdné slovo se označuje řeckým písmenem ε (epsilon).

(Pozn.: V literatuře se pro označení prázdného slova někdy používá místo symbolu ε řecké písmeno λ (lambda).)

$$|\varepsilon| = 0$$

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

Například zřetězením slov **OST** a **RAVA** vznikne slovo **OSTRAVA**.

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

$$\text{OST} \cdot \text{RAVA} = \text{OSTRAVA}$$

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

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

což znamená, že při zápisu více zřetězení můžeme vypouštět závorky a psát například $w_1 \cdot w_2 \cdot w_3 \cdot w_4 \cdot w_5$ místo $(w_1 \cdot (w_2 \cdot w_3)) \cdot (w_4 \cdot w_5)$.

Zřetězení slov

Zřetězení není **komutativní**, tj. obecně pro dvojici slov u a v neplatí rovnost

$$u \cdot v = v \cdot u$$

Příklad:

$$\text{OST} \cdot \text{RAVA} \neq \text{RAVA} \cdot \text{OST}$$

Zjevně pro libovolná slova v a w platí:

$$|v \cdot w| = |v| + |w|$$

Pro libovolné slovo w také platí:

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

Definice

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

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

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

Příklad:

- Prefixy slova **abaab** jsou ϵ , **a**, **ab**, **aba**, **abaa**, **abaab**.
- Sufixy slova **abaab** jsou ϵ , **b**, **ab**, **aab**, **baab**, **abaab**.
- Podslova slova **abaab** jsou ϵ , **a**, **b**, **ab**, **ba**, **aba**, **baa**, **aab**, **abaa**, **baab**, **abaab**.

Množinu všech slov tvořených symboly z abecedy Σ označujeme 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 Java je jazyk v abecedě tvořené množinou všech Unicode znaků.

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

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 .

Doplňěk – $\overline{L_1}$ 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: Při operacích nad jazyky předpokládáme, že jazyky, se kterými operaci provádíme, používají tutéž abecedu Σ .

Formálně:

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

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

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

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

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

Příklad:

Uvažujme jazyky nad abecedou $\{A, B, \dots, Z\}$.

- L_1 — množina všech slov obsahujících podlovo **FOO**
- L_2 — množina všech slov obsahujících podlovo **BAR**

Pak

- $L_1 \cup L_2$ — množina všech slov obsahujících podslovo **FOO** nebo **BAR**,
- $L_1 \cap L_2$ — množina všech slov obsahujících podslova **FOO** i **BAR**,
- $\overline{L_1}$ — množina všech slov, která neobsahují podslovo **FOO**,
- $L_1 - L_2$ — množina všech slov, ve kterých se vyskytuje podslovo **FOO**, ale nevyskytuje podslovo **BAR**.

Definice

Zřetězení jazyků L_1 a L_2 je jazyk

$$L = \{uv \mid u \in L_1, v \in L_2\}$$

tj. jazyk všech slov, která začínají slovem z L_1 a pokračují slovem z L_2 .
Zřetězení jazyků L_1 a L_2 označujeme $L_1 \cdot L_2$.

Příklad:

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

Jazyk $L_1 \cdot L_2$ obsahuje slova:

abba *abbab* *abbbbb* *baa* *baab* *babbb*

Definice

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

Tj. $w \in L^*$ právě tehdy, když

$$\exists n \in \mathbb{N} : \exists w_1, w_2, \dots, w_n \in L : w = w_1 w_2 \cdots w_n$$

Příklad: $L = \{aa, b\}$

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

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).

Poznámka: Jeden pojem může mít více různých definic. Tyto definice mohou vypadat na první pohled rozdílně, přitom jsou ale vzájemně ekvivalentní ve smyslu, že popisují jednu a tutéž věc.

Jako příklad si ukážeme další dvě alternativní definice pojmu **iterace jazyka**, přičemž obě jsou ekvivalentní s dříve uvedenou definicí.

Iterace jazyka — alternativní definice

Nejprve definujeme pro jazyk L a číslo $k \in \mathbb{N}$ jazyk L^k :

$$L^0 = \{\varepsilon\}, \quad L^k = L^{k-1} \cdot L \quad \text{pro } k \geq 1$$

To znamená

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

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

aaaaaa aaaab aabaa aabb baaaa baab bbaa bbb

Alternativní definice 1

Iterace jazyka L je jazyk

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

Poznámka:

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

Alternativní definice 2

Iterace jazyka L je nejmenší (vzhledem k \subseteq) jazyk L' takový, že

- $\varepsilon \in L'$
- pro každé $u \in L'$ a $v \in L$ platí, že $uv \in L'$

Poznámka: To, že jazyk L' je nejmenší vzhledem k \subseteq znamená, že pro jakýkoliv jazyk L'' splňující výše uvedené podmínky platí $L' \subseteq L''$.

Poznámka: Používá se také zápis L^+ jako zkratka za $L \cdot L^*$, tj.

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

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

Zrcadlový obraz slova w značíme w^R .

Příklad: $w = \text{AHOJ}$ $w^R = \text{JOHA}$

Formálně můžeme definovat, že pro $w = a_1 a_2 \cdots a_n$ (kde $a_i \in \Sigma$) je $w^R = a_n a_{n-1} \cdots a_1$.

Alternativně můžeme definovat w^R jako $\text{rev}(w)$, kde $\text{rev} : \Sigma^* \rightarrow \Sigma^*$ je funkce definovaná induktivně takto:

- $\text{rev}(\varepsilon) = \varepsilon$
- pro $a \in \Sigma$ a $w \in \Sigma^*$ je $\text{rev}(aw) = \text{rev}(w)a$

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: $L = \{ab, baaba, aaab\}$
 $L^R = \{ba, abaab, baaa\}$

Uspořádání na slovech

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

$$a_1 < a_2 < \dots < a_n.$$

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

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

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

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

$$w_0, w_1, w_2, \dots$$

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

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

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 $<_L$.

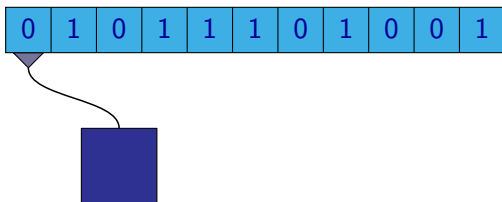
Konečné automaty

Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L , který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1 .

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

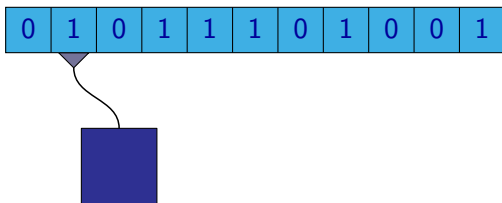


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L , který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1 .

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

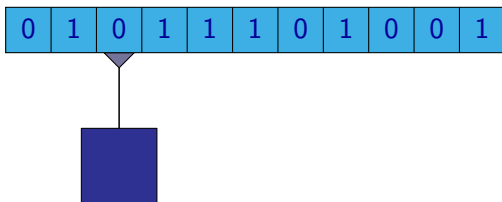


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L , který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1 .

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

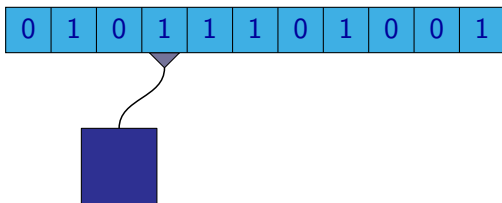


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L , který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1 .

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

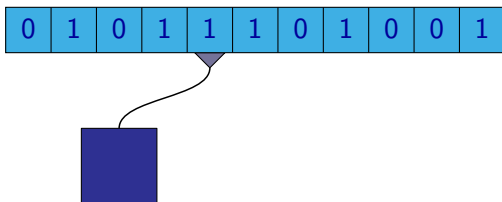


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L , který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1 .

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

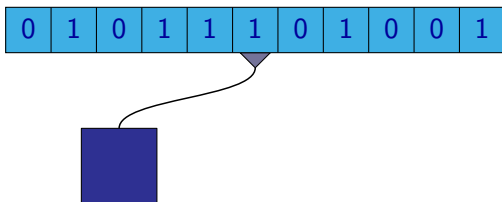


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L , který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1 .

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

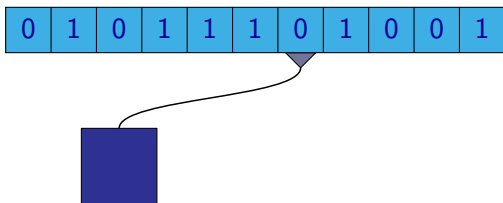


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L , který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1 .

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

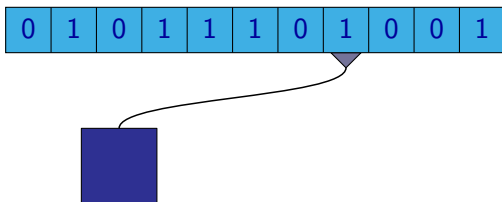


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L , který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1 .

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

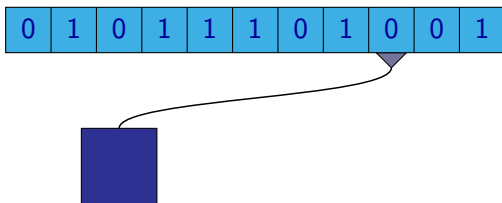


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L , který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1 .

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

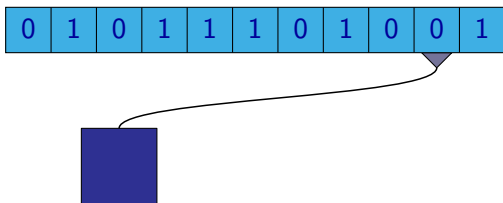


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L , který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1 .

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

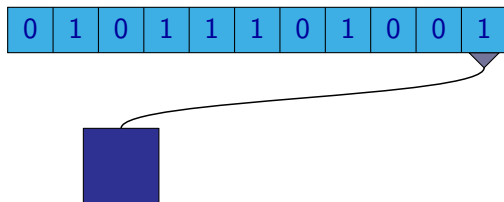


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L , který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1 .

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

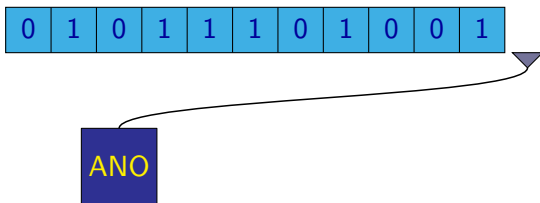


Rozpoznávání jazyka

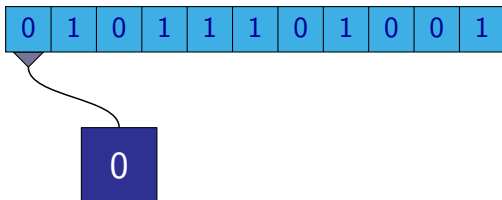
Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L , který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1 .

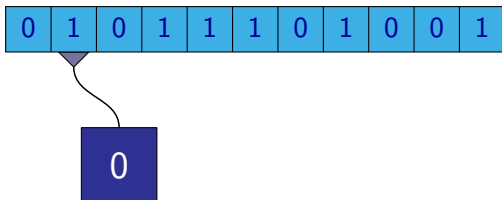
Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.



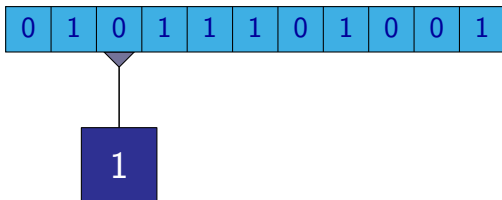
První nápad: Počítat počet výskytů symbolů 1.



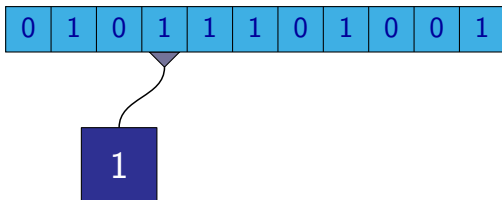
První nápad: Počítat počet výskytů symbolů 1.



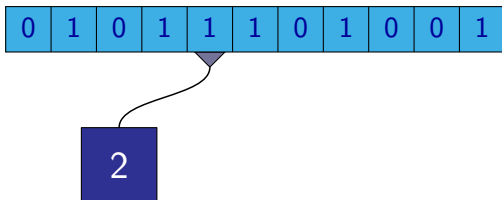
První nápad: Počítat počet výskytů symbolů 1.



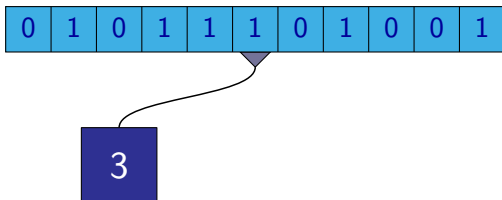
První nápad: Počítat počet výskytů symbolů 1.



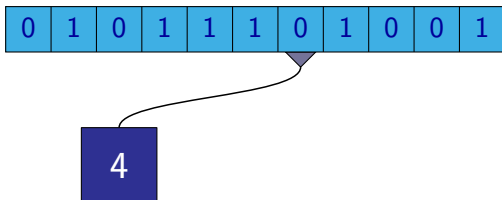
První nápad: Počítat počet výskytů symbolů 1.



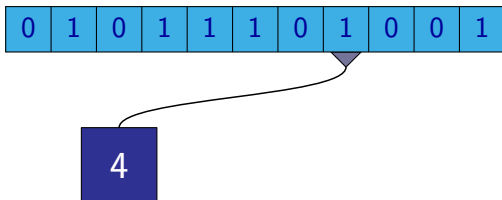
První nápad: Počítat počet výskytů symbolů 1.



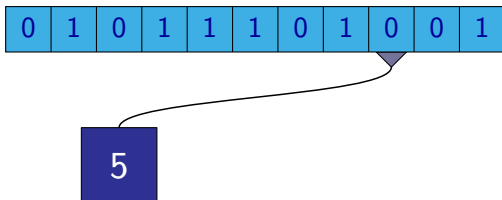
První nápad: Počítat počet výskytů symbolů 1.



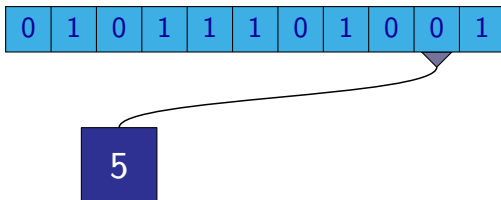
První nápad: Počítat počet výskytů symbolů 1.



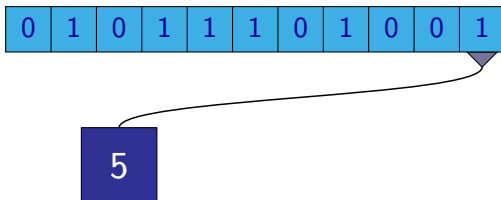
První nápad: Počítat počet výskytů symbolů 1.



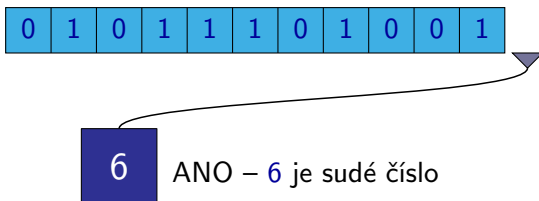
První nápad: Počítat počet výskytů symbolů 1.



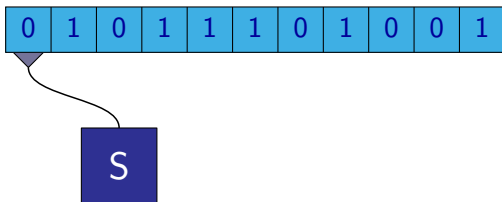
První nápad: Počítat počet výskytů symbolů 1.



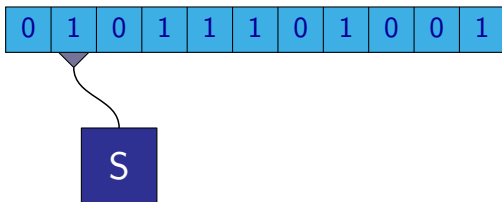
První nápad: Počítat počet výskytů symbolů 1.



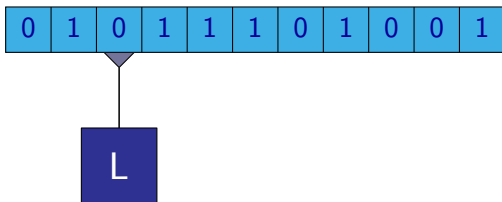
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



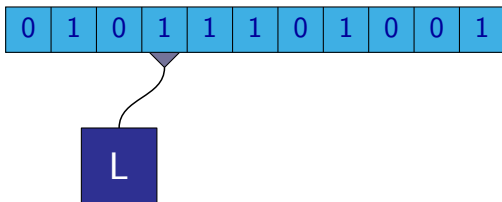
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



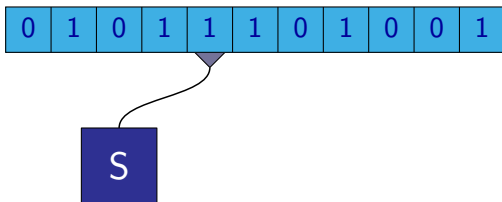
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



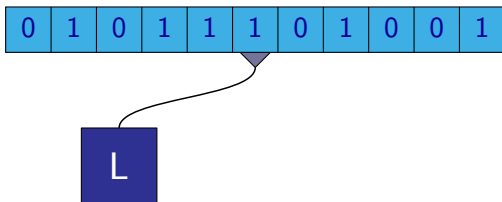
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



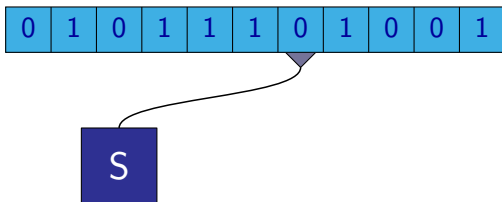
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



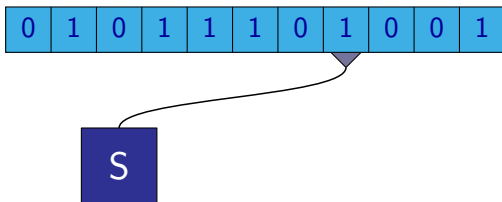
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



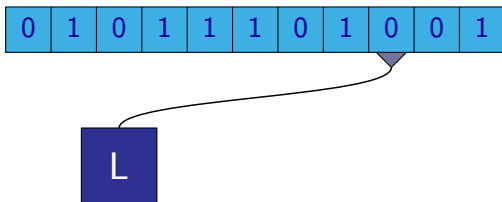
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



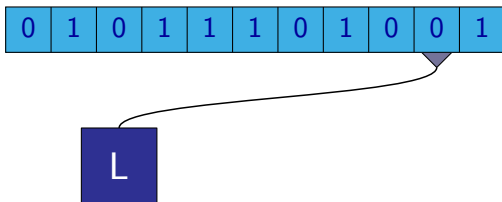
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



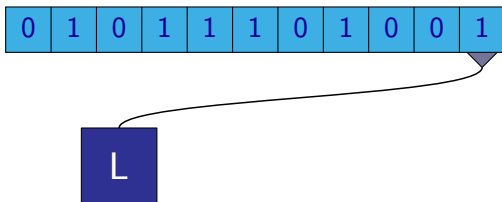
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



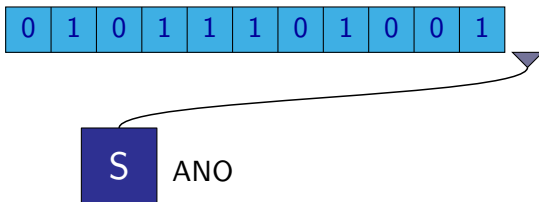
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



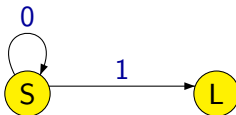
Chování tohoto zařízení můžeme popsat grafem:



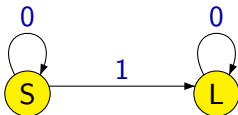
Chování tohoto zařízení můžeme popsat grafem:



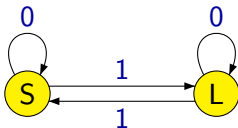
Chování tohoto zařízení můžeme popsat grafem:



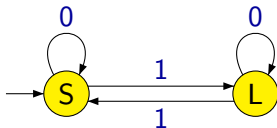
Chování tohoto zařízení můžeme popsat grafem:



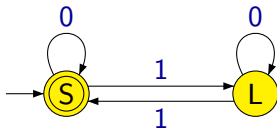
Chování tohoto zařízení můžeme popsat grafem:



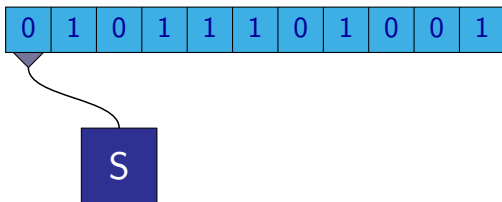
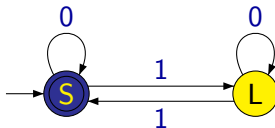
Chování tohoto zařízení můžeme popsat grafem:



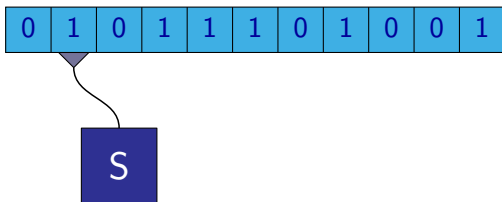
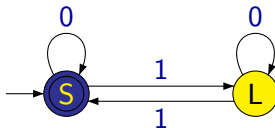
Chování tohoto zařízení můžeme popsat grafem:



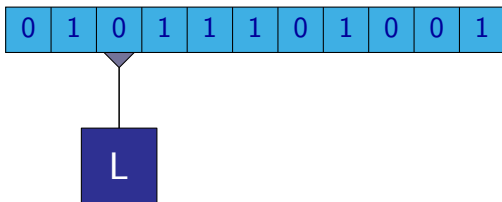
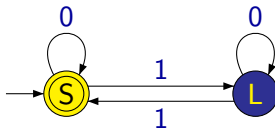
Chování tohoto zařízení můžeme popsat grafem:



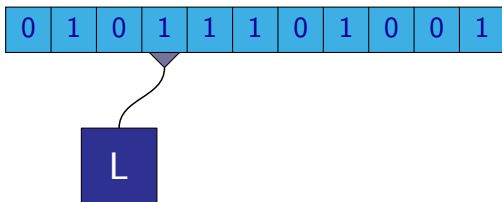
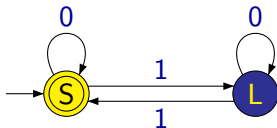
Chování tohoto zařízení můžeme popsat grafem:



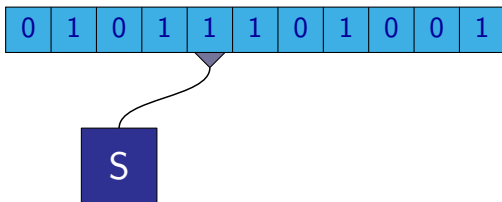
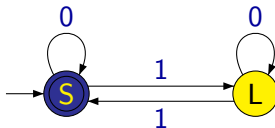
Chování tohoto zařízení můžeme popsat grafem:



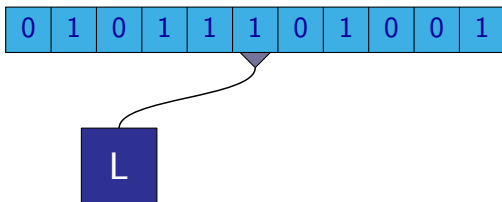
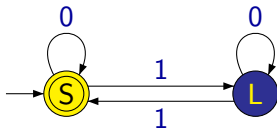
Chování tohoto zařízení můžeme popsat grafem:



Chování tohoto zařízení můžeme popsat grafem:

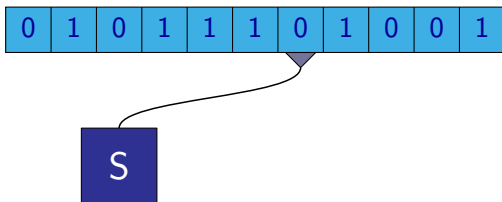
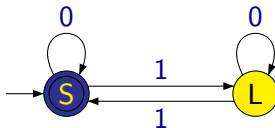


Chování tohoto zařízení můžeme popsat grafem:

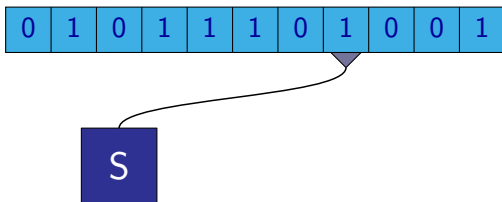
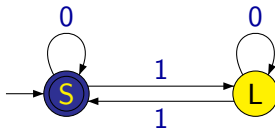


Rozpoznávání jazyka

Chování tohoto zařízení můžeme popsat grafem:

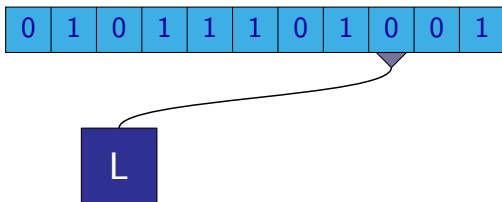
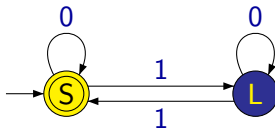


Chování tohoto zařízení můžeme popsat grafem:



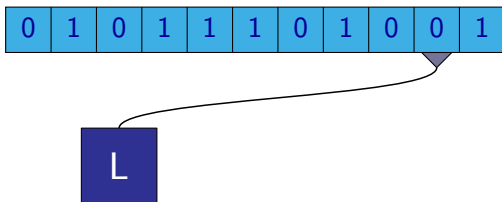
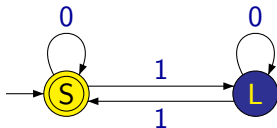
Rozpoznávání jazyka

Chování tohoto zařízení můžeme popsat grafem:



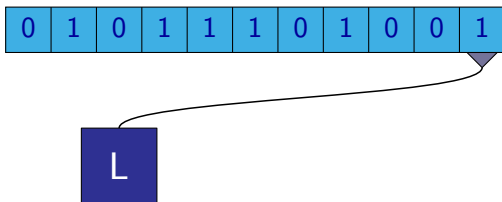
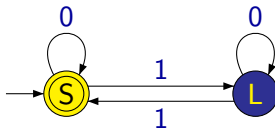
Rozpoznávání jazyka

Chování tohoto zařízení můžeme popsat grafem:

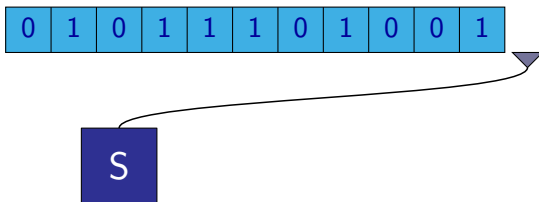
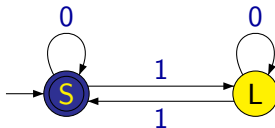


Rozpoznávání jazyka

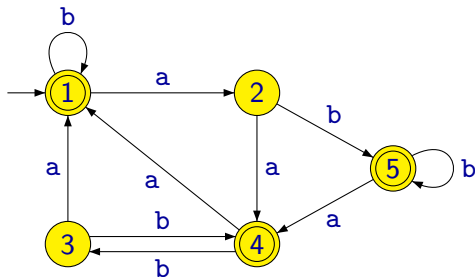
Chování tohoto zařízení můžeme popsat grafem:



Chování 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**.

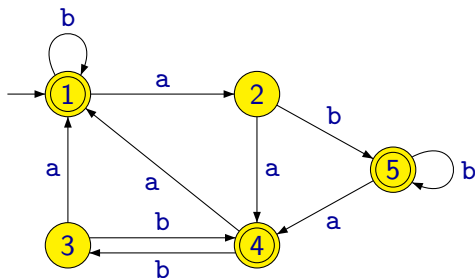
Formálně je **deterministický konečný automat** definován jako pětice

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

kde:

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

Deterministický konečný automat



- $Q = \{1, 2, 3, 4, 5\}$

- $\Sigma = \{a, b\}$

- $q_0 = 1$

- $F = \{1, 4, 5\}$

$$\delta(1, a) = 2$$

$$\delta(1, b) = 1$$

$$\delta(2, a) = 4$$

$$\delta(2, b) = 5$$

$$\delta(3, a) = 1$$

$$\delta(3, b) = 4$$

$$\delta(4, a) = 1$$

$$\delta(4, b) = 3$$

$$\delta(5, a) = 4$$

$$\delta(5, b) = 5$$

Deterministický konečný automat

Místo zápisu

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

budeme raději používat stručnější tabulku nebo grafické znázornění:

δ	a	b
1	2	1
2	4	5
3	1	4
4	1	3
5	4	5

Deterministický konečný automat

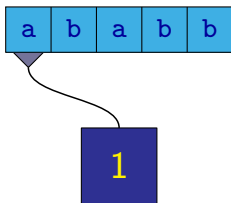
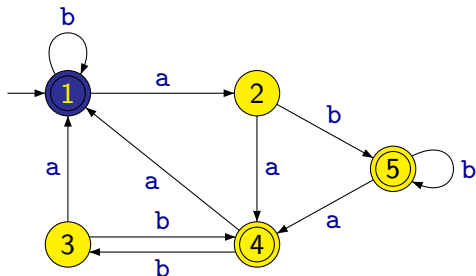
Místo zápisu

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

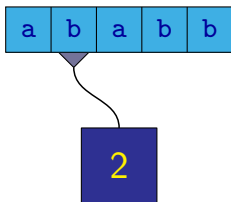
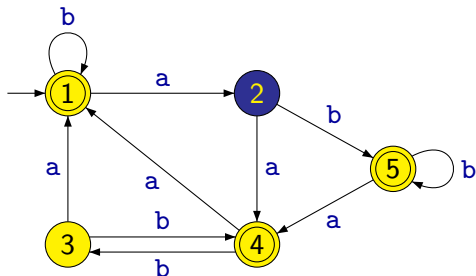
budeme raději používat stručnější tabulku nebo grafické znázornění:

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

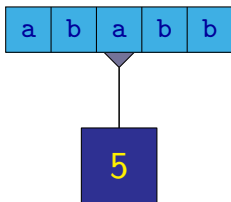
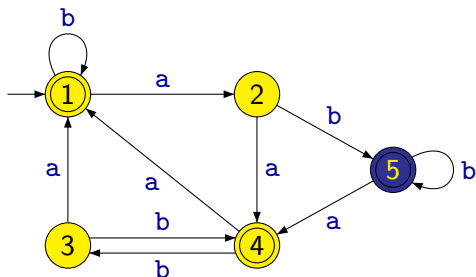
Deterministický konečný automat



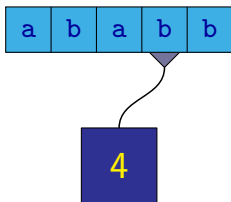
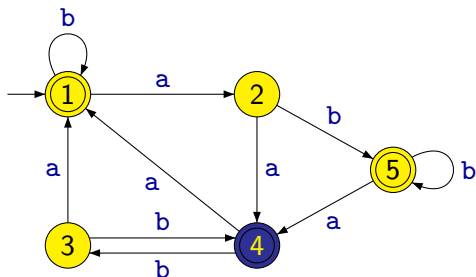
Deterministický konečný automat



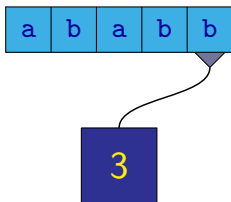
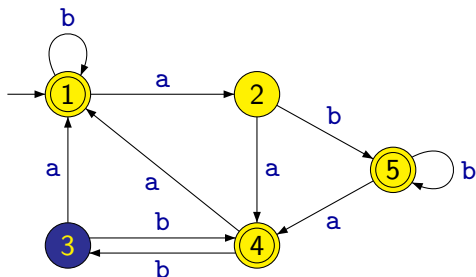
Deterministický konečný automat



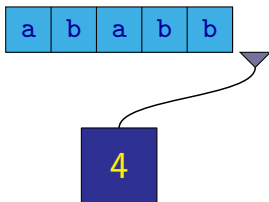
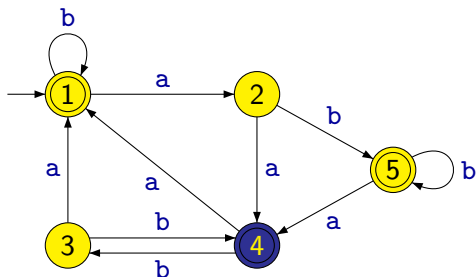
Deterministický konečný automat



Deterministický konečný automat



Deterministický konečný automat



Konfigurace konečného automatu A je dána stavem jeho řídicí jednotky a dosud nepřečteným obsahem pásky.

Množina všech konfigurací $Conf$ je $Q \times \Sigma^*$.

Příklad: $(2, babb) \in Conf$

Pro daný automat A můžeme na množině všech konfigurací definovat binární relaci $\vdash \subseteq Conf \times Conf$ s následujícím významem:

$C_1 \vdash C_2$ znamená, že automat A může přejít jedním krokem z konfigurace C_1 do konfigurace C_2 .

Příklad:

$$(2, babb) \vdash (5, abb)$$

Formálně platí, že $(q, w) \vdash (q', w')$ právě tehdy, když $w = aw'$ a $q' = \delta(q, a)$ pro nějaké $a \in \Sigma$.

Deterministický konečný automat

Konfigurace (q, w) se nazývá **počáteční konfigurace**, jestliže $q = q_0$.

Příklad: $(1, ababb)$ je počáteční konfigurace.

Konfigurace (q, w) se nazývá **koncová konfigurace**, jestliže $w = \varepsilon$.

Příklad: $(4, \varepsilon)$ je koncová konfigurace.

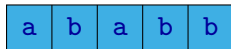
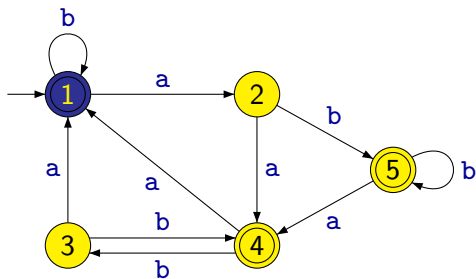
Definice

Výpočet automatu je posloupnost konfigurací

$$C_0, C_1, C_2, \dots, C_k$$

kde C_i jsou konfigurace, C_0 je počáteční konfigurace, C_k je koncová konfigurace a pro všechna $i \in \{1, 2, \dots, k\}$ platí, že $C_{i-1} \vdash C_i$.

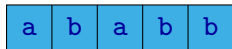
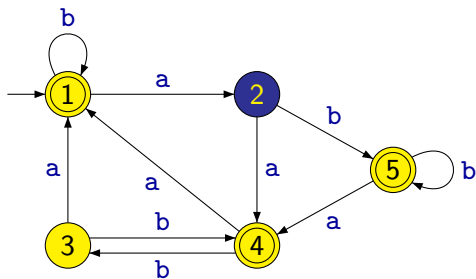
Deterministický konečný automat



(1, ababb)

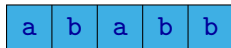
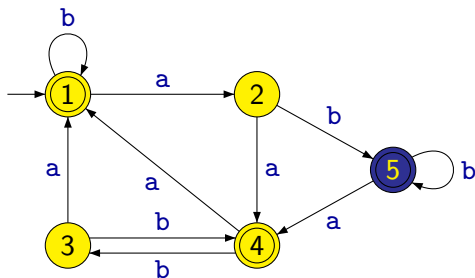


Deterministický konečný automat



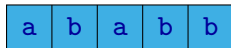
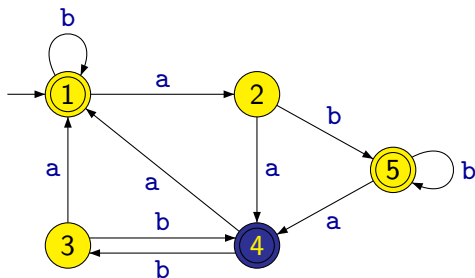
$(1, ababb) \vdash (2, babb)$

Deterministický konečný automat



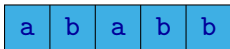
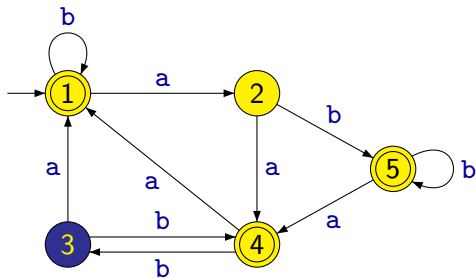
$(1, ababb) \vdash (2, babb) \vdash (5, abb)$

Deterministický konečný automat



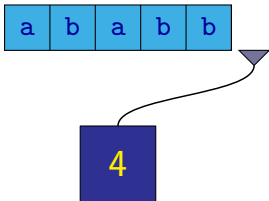
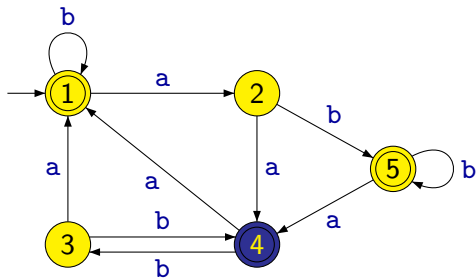
$(1, ababb) \vdash (2, babb) \vdash$
 $(5, abb) \vdash (4, bb)$

Deterministický konečný automat



$(1, ababb) \vdash (2, babb) \vdash$
 $(5, abb) \vdash (4, bb) \vdash$
 $(3, b)$

Deterministický konečný automat



$(1, ababb) \vdash (2, babb) \vdash$
 $(5, abb) \vdash (4, bb) \vdash$
 $(3, b) \vdash (4, \epsilon)$

Dále můžeme definovat pro daný automat A relaci

$$\vdash^* \subseteq \text{Conf} \times \text{Conf},$$

takovou, že $C \vdash^* C'$ platí právě tehdy, když automat může přejít nějakým libovolným (i nulovým) počtem kroků z konfigurace C do konfigurace C' .

Přesněji řečeno, $C \vdash^* C'$ platí právě tehdy, když existuje posloupnost konfigurací

$$C_0, C_1, C_2, \dots, C_k$$

kde $C_0 = C$, $C_k = C'$ a pro všechna $i \in \{1, 2, \dots, k\}$ platí, že $C_{i-1} \vdash C_i$.

Alternativně můžeme relaci \vdash^* definovat jako nejmenší relaci splňující následující podmínky:

- Pro každé $C \in Conf$ platí $C \vdash^* C$.
- Pro každé $C, C' \in Conf$ platí, že pokud $C \vdash C'$, pak $C \vdash^* C'$.
- Pro každé $C, C', C'' \in Conf$ platí, že pokud $C \vdash^* C'$ a $C' \vdash^* C''$, pak $C \vdash^* C''$.

Poznámka: Relace \vdash^* je **reflexivním a tranzitivním uzávěrem** relace \vdash , tj. je to nejmenší reflexivní a tranzitivní relace obsahující relaci \vdash .

Definice

Koncová konfigurace (q, ε) je **přijímající**, jestliže $q \in F$.

Definice

Automat $A = (Q, \Sigma, \delta, q_0, F)$ **přijímá** slovo $w \in \Sigma^*$ právě tehdy, když $(q_0, w) \vdash^* (q, \varepsilon)$ pro nějaké $q \in F$.

Tj. automat A přijímá slovo w právě tehdy, když výpočet automatu A začínající v počáteční konfiguraci (q_0, w) skončí v přijímající koncové konfiguraci.

Definice

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

$$L(A) = \{w \in \Sigma^* \mid (q_0, w) \vdash^* (q, \varepsilon), q \in F\}$$

Definice

Jazyk L je **regulární** právě tehdy, když existuje nějaký konečný automat, který jej přijímá.

Definice

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

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 $q \xrightarrow{w} q'$.

Tuto relaci 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 $q \xrightarrow{w} q''$ a $\delta(q'', a) = q'$.

Jazyk rozpoznávaný daným DKA $A = (Q, \Sigma, \delta, q_0, F)$ pak můžeme definovat (alternativně) takto:

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