

Formální jazyky

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...
```

Jazyk — množina (některých) slov tvořených symboly z dané abecedy

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

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**.

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 **podslovem** 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 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.

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 $\{a, b\}$.

- L_1 — množina všech slov obsahujících podslovo **baa**
- L_2 — množina všech slov se sudým počtem výskytů symbolu **b**

Pak

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

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

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

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 zřetězením 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).

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

Iterace jazyka L je jazyk

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

Poznámka:

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

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 $w^R = a_n a_{n-1} \cdots a_1$.

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 $<_L$.

Regulární výrazy

Regulární výrazy popisující jazyky nad abecedou Σ :

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

Příklad: abeceda $\Sigma = \{0, 1\}$

- Podle definice jsou 0 i 1 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.

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.

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.

Příklad: abeceda $\Sigma = \{0, 1\}$

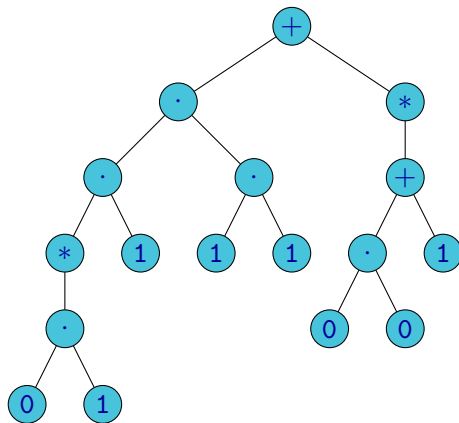
- Podle definice jsou 0 i 1 regulární výrazy.
- Protože 0 i 1 jsou regulární výrazy, je i $(0 + 1)$ regulární výraz.
- Protože 0 je regulární výraz, je i (0^*) regulární výraz.
- Protože $(0 + 1)$ i (0^*) jsou regulární výrazy, je i $((0 + 1) \cdot (0^*))$ regulární výraz.

Poznámka: Jestliže α je regulární výraz, zápisem $[\alpha]$ označujeme jazyk definovaný regulárním výrazem α .

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

Regulární výrazy

Strukturu regulárního výrazu si můžeme znázornit abstraktním syntaktickým stromem:



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

Popis (abstraktní) syntaxe regulárních výrazů pomocí Backus-Naurovy formy:

$$\alpha ::= \emptyset \mid \varepsilon \mid a \mid \alpha^* \mid \alpha \cdot \alpha \mid \alpha + \alpha$$

Formální definice sémantiky regulárních výrazů:

- $[\emptyset] = \emptyset$
- $[\varepsilon] = \{\varepsilon\}$
- $[a] = \{a\}$
- $[\alpha^*] = [\alpha]^*$
- $[\alpha \cdot \beta] = [\alpha] \cdot [\beta]$
- $[\alpha + \beta] = [\alpha] \cup [\beta]$

Regulární výrazy

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

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

Příklad: Místo

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

obvykle píšeme

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

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 ... jazyk tvořený jediným slovem 0

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 ... jazyk tvořený jediným slovem 0

01 ... jazyk tvořený jediným slovem 01

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 ... jazyk tvořený jediným slovem 0

01 ... jazyk tvořený jediným slovem 01

$0 + 1$... jazyk tvořený dvěma slovy 0 a 1

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 ... jazyk tvořený jediným slovem 0

01 ... jazyk tvořený jediným slovem 01

$0 + 1$... jazyk tvořený dvěma slovy 0 a 1

0^* ... jazyk tvořený slovy $\varepsilon, 0, 00, 000, \dots$

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 ... jazyk tvořený jediným slovem 0

01 ... jazyk tvořený jediným slovem 01

$0 + 1$... jazyk tvořený dvěma slovy 0 a 1

0^* ... jazyk tvořený slovy $\varepsilon, 0, 00, 000, \dots$

$(01)^*$... jazyk tvořený slovy $\varepsilon, 01, 0101, 010101, \dots$

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 ... jazyk tvořený jediným slovem 0

01 ... jazyk tvořený jediným slovem 01

$0 + 1$... jazyk tvořený dvěma slovy 0 a 1

0^* ... jazyk tvořený slovy $\varepsilon, 0, 00, 000, \dots$

$(01)^*$... jazyk tvořený slovy $\varepsilon, 01, 0101, 010101, \dots$

$(0 + 1)^*$... jazyk tvořený všemi slovy nad abecedou $\{0, 1\}$

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 ... jazyk tvořený jediným slovem 0

01 ... jazyk tvořený jediným slovem 01

$0 + 1$... jazyk tvořený dvěma slovy 0 a 1

0^* ... jazyk tvořený slovy $\varepsilon, 0, 00, 000, \dots$

$(01)^*$... jazyk tvořený slovy $\varepsilon, 01, 0101, 010101, \dots$

$(0 + 1)^*$... jazyk tvořený všemi slovy nad abecedou $\{0, 1\}$

$(0 + 1)^*00$... jazyk tvořený všemi slovy končícími 00

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 ... jazyk tvořený jediným slovem 0

01 ... jazyk tvořený jediným slovem 01

$0 + 1$... jazyk tvořený dvěma slovy 0 a 1

0^* ... jazyk tvořený slovy $\varepsilon, 0, 00, 000, \dots$

$(01)^*$... jazyk tvořený slovy $\varepsilon, 01, 0101, 010101, \dots$

$(0 + 1)^*$... jazyk tvořený všemi slovy nad abecedou $\{0, 1\}$

$(0 + 1)^*00$... jazyk tvořený všemi slovy končícími 00

$(01)^*111(01)^*$... jazyk tvořený všemi slovy obsahujícími podslovo 111 předcházené i následované libovolným počtem slov 01

$(0 + 1)^*00 + (01)^*111(01)^*$... jazyk tvořený všemi slovy, která buď končí 00 nebo obsahují podслово 111 předcházené i následované libovolným počtem slov 01

$(0 + 1)^*00 + (01)^*111(01)^*$... jazyk tvořený všemi slovy, která buď končí 00 nebo obsahují podslovo 111 předcházené i následované libovolným počtem slov 01

$(0 + 1)^*1(0 + 1)^*$... jazyk tvořený všemi slovy obsahujícími alespoň jeden symbol 1

$(0 + 1)^*00 + (01)^*111(01)^*$... jazyk tvořený všemi slovy, která buď končí 00 nebo obsahují podslovo 111 předcházené i následované libovolným počtem slov 01

$(0 + 1)^*1(0 + 1)^*$... jazyk tvořený všemi slovy obsahujícími alespoň jeden symbol 1

$0^*(10^*10^*)^*$... jazyk tvořený všemi slovy obsahujícími sudý počet symbolů 1

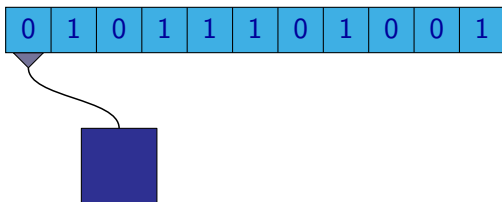
Konečné automaty

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.

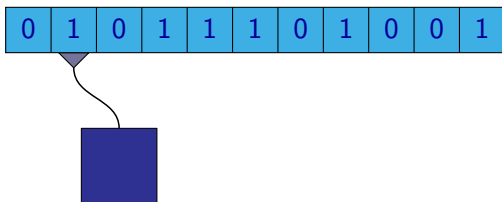


Rozpoznávání jazyka

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.

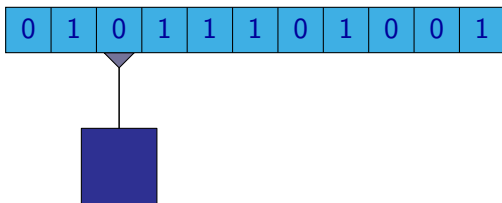


Rozpoznávání jazyka

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.

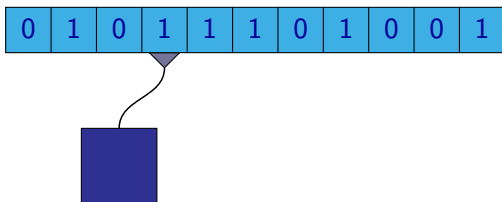


Rozpoznávání jazyka

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.

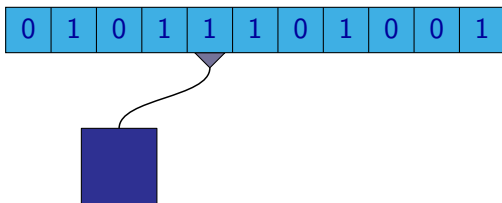


Rozpoznávání jazyka

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.

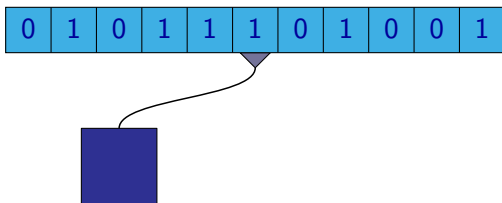


Rozpoznávání jazyka

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.

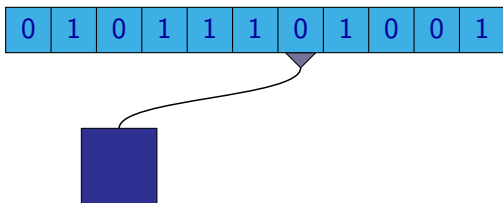


Rozpoznávání jazyka

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.

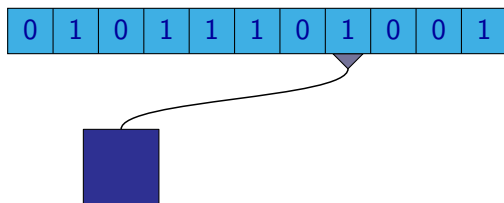


Rozpoznávání jazyka

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.

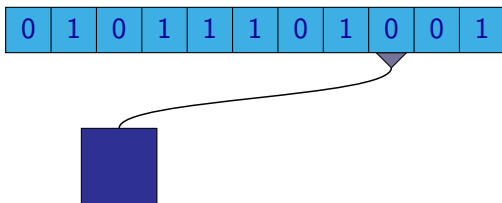


Rozpoznávání jazyka

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.

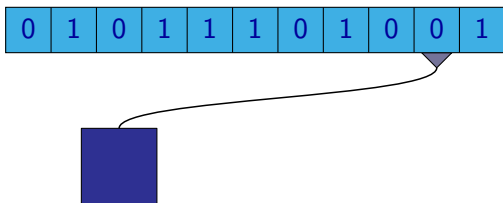


Rozpoznávání jazyka

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.

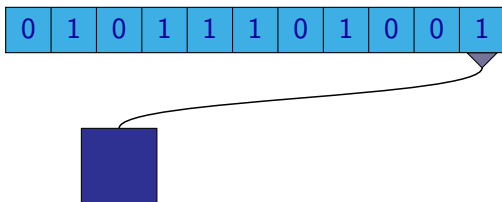


Rozpoznávání jazyka

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.

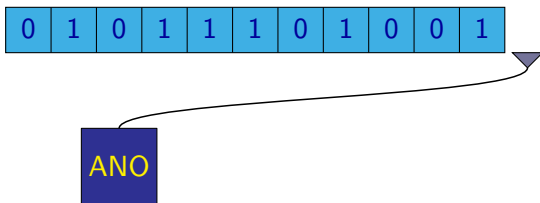


Rozpoznávání jazyka

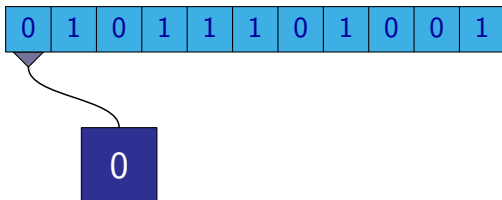
Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

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

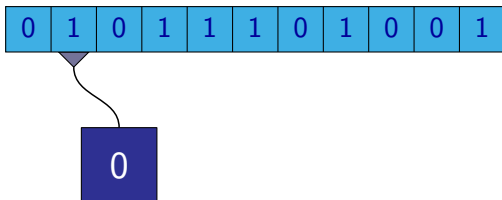
Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.



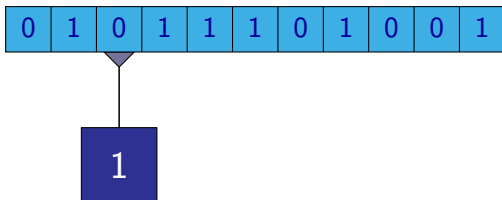
První nápad: Počítat počet výskytů symbolů 1.



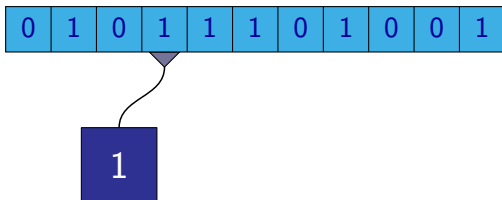
První nápad: Počítat počet výskytů symbolů 1.



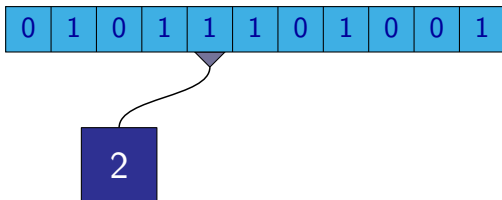
První nápad: Počítat počet výskytů symbolů 1.



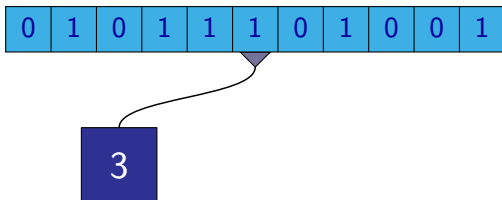
První nápad: Počítat počet výskytů symbolů 1.



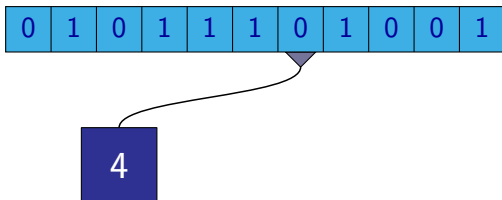
První nápad: Počítat počet výskytů symbolů 1.



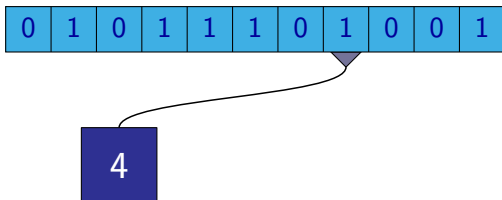
První nápad: Počítat počet výskytů symbolů 1.



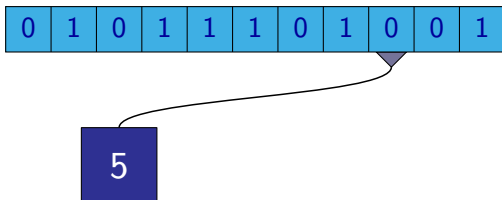
První nápad: Počítat počet výskytů symbolů 1.



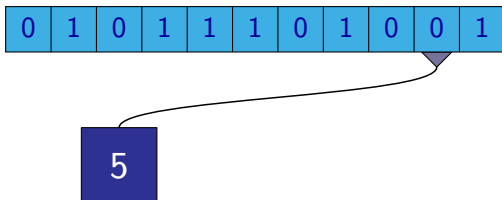
První nápad: Počítat počet výskytů symbolů 1.



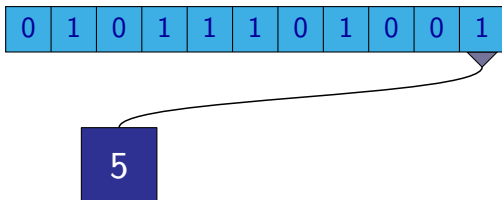
První nápad: Počítat počet výskytů symbolů 1.



První nápad: Počítat počet výskytů symbolů 1.



První nápad: Počítat počet výskytů symbolů 1.



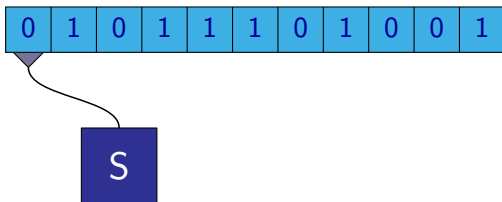
První nápad: Počítat počet výskytů symbolů 1.

0	1	0	1	1	1	0	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---

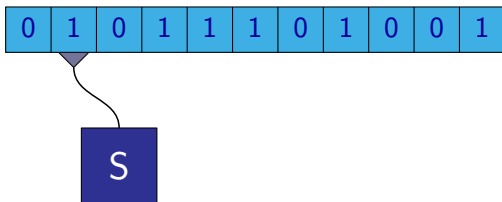
6

ANO – 6 je sudé číslo

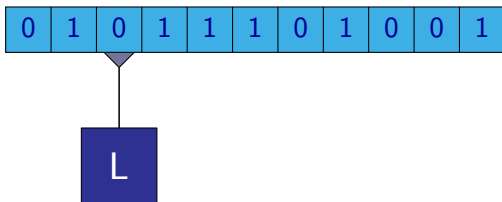
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



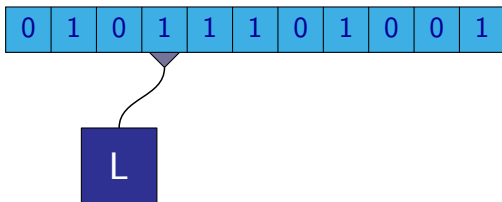
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



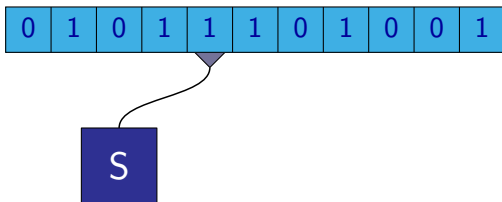
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



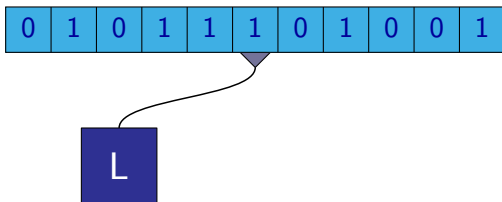
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



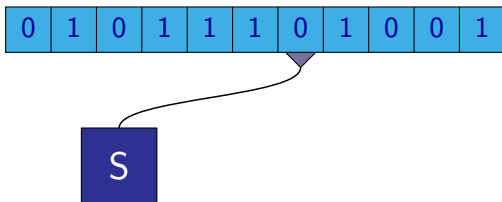
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



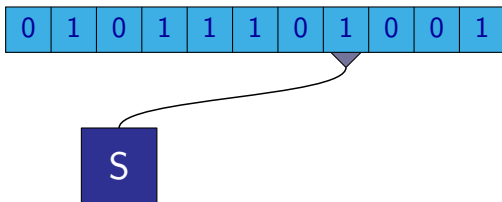
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



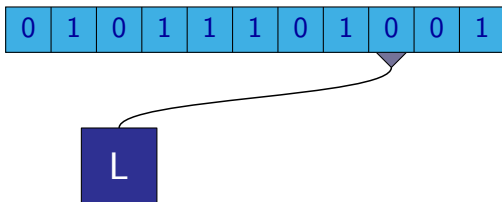
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



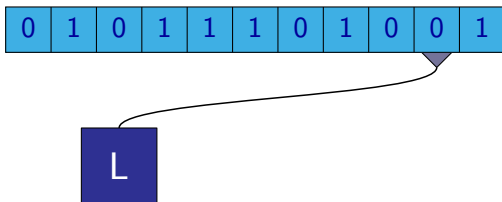
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



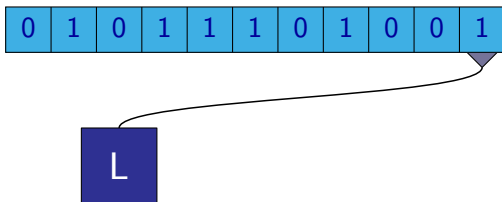
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



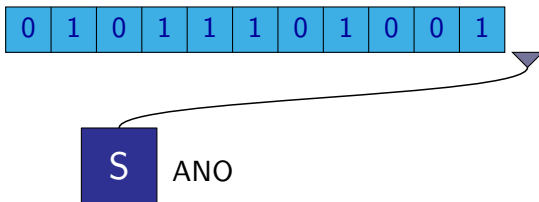
Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů **1** je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



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).



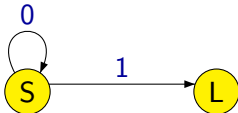
Chování tohoto zařízení můžeme popsat grafem:



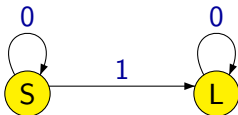
Chování tohoto zařízení můžeme popsat grafem:



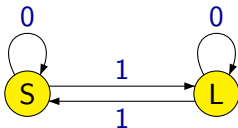
Chování tohoto zařízení můžeme popsat grafem:



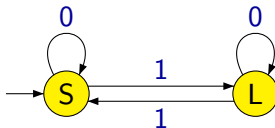
Chování tohoto zařízení můžeme popsat grafem:



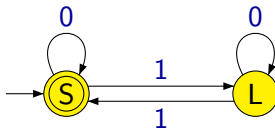
Chování tohoto zařízení můžeme popsat grafem:



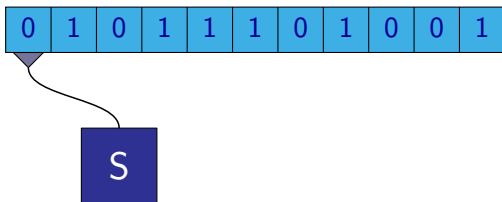
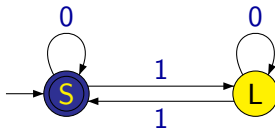
Chování tohoto zařízení můžeme popsat grafem:



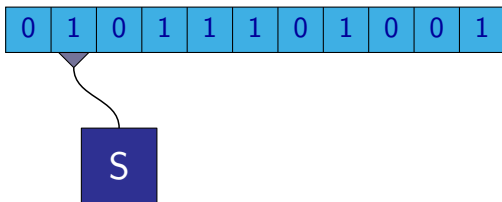
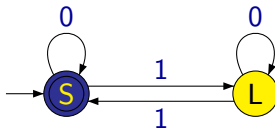
Chování tohoto zařízení můžeme popsat grafem:



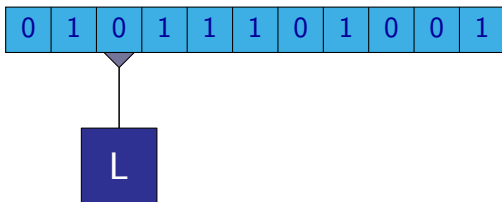
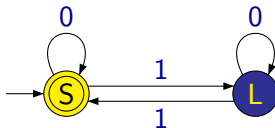
Chování tohoto zařízení můžeme popsat grafem:



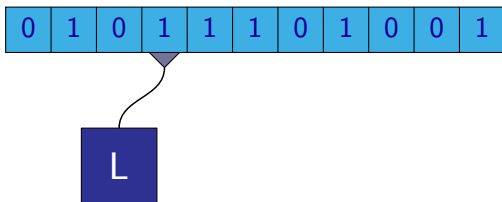
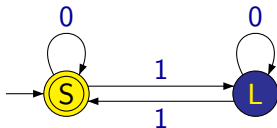
Chování tohoto zařízení můžeme popsat grafem:



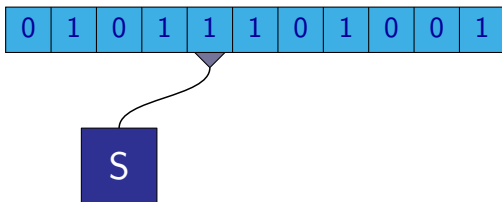
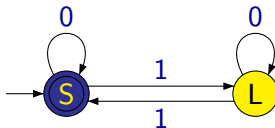
Chování tohoto zařízení můžeme popsat grafem:



Chování tohoto zařízení můžeme popsat grafem:

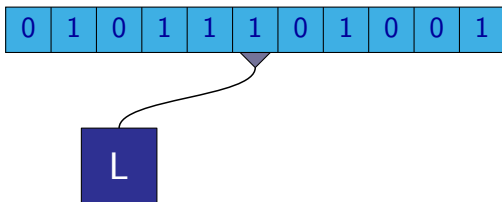
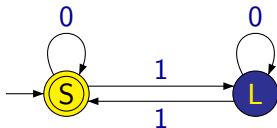


Chování tohoto zařízení můžeme popsat grafem:



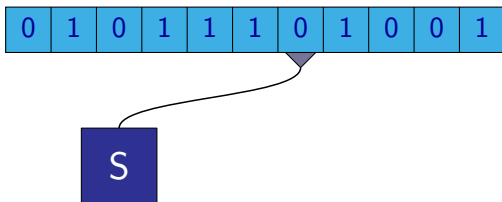
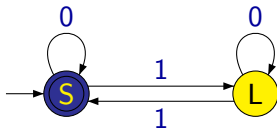
Rozpoznávání jazyka

Chování tohoto zařízení můžeme popsat grafem:

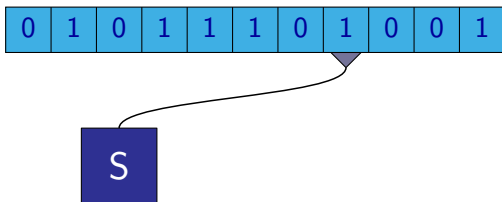
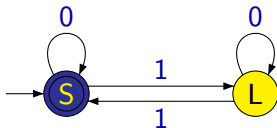


Rozpoznávání jazyka

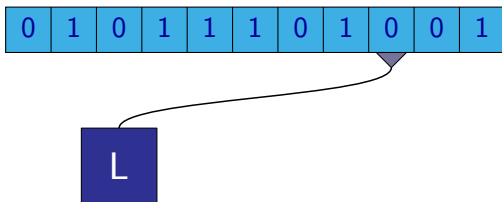
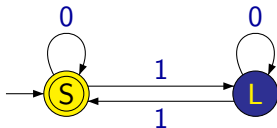
Chování tohoto zařízení můžeme popsat grafem:



Chování tohoto zařízení můžeme popsat grafem:

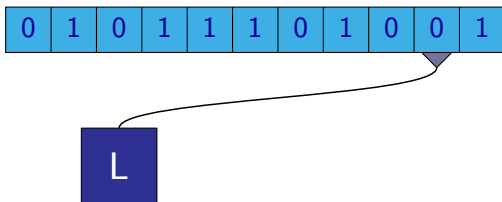
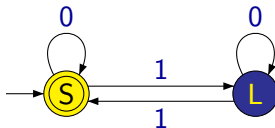


Chování tohoto zařízení můžeme popsat grafem:

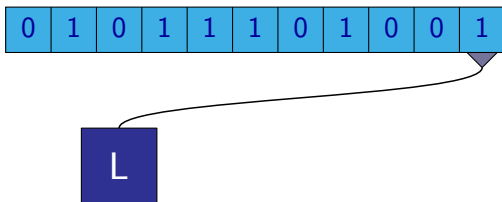
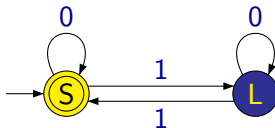


Rozpoznávání jazyka

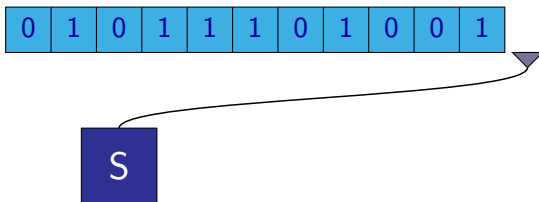
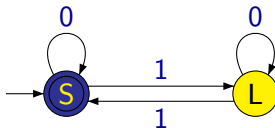
Chování tohoto zařízení můžeme popsat grafem:



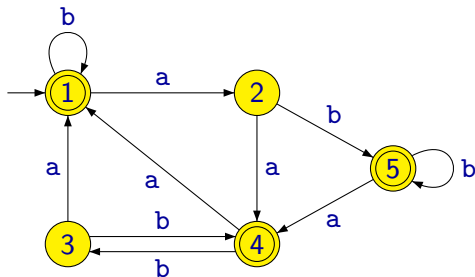
Chování tohoto zařízení můžeme popsat grafem:



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**.

Deterministický konečný automat

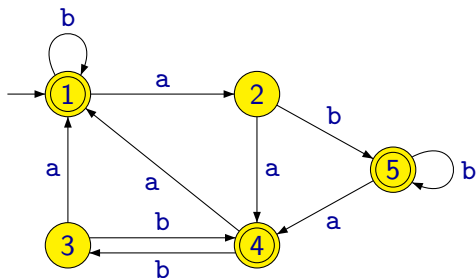
Formálně je **deterministický konečný automat (DKA)** definován jako pětice

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

kde:

- Q je neprázdna konečná množina **stavů**
- Σ je **abeceda** (neprázdna 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



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

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

- $q_0 = 1$

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

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

$$\delta(2, a) = 4 \quad \delta(2, b) = 5$$

$$\delta(3, a) = 1 \quad \delta(3, b) = 4$$

$$\delta(4, a) = 1 \quad \delta(4, b) = 3$$

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

Deterministický konečný automat

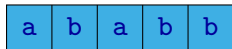
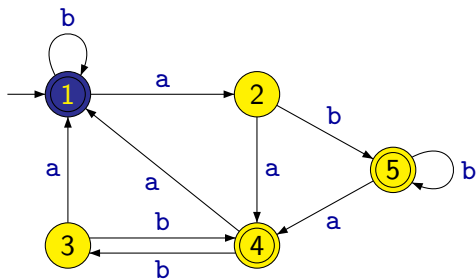
Místo zápisu

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

budeme raději používat stručnější tabulku nebo grafické znázornění:

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

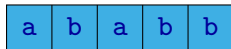
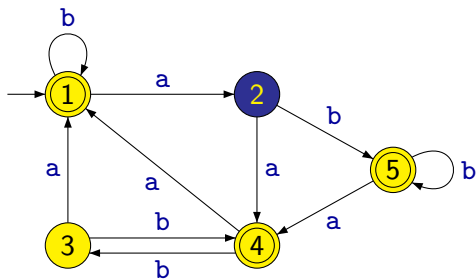
Deterministický konečný automat



1



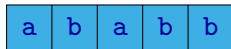
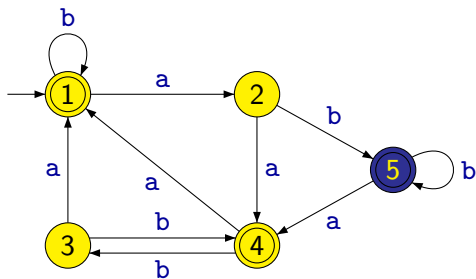
Deterministický konečný automat



$1 \xrightarrow{a} 2$

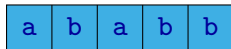
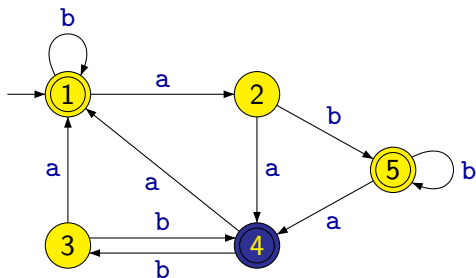


Deterministický konečný automat



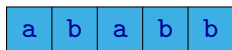
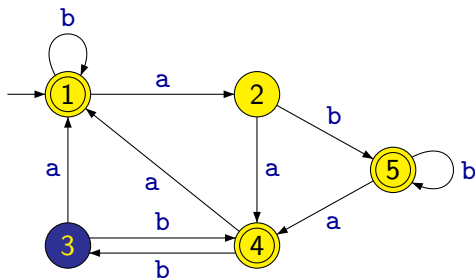
$1 \xrightarrow{a} 2 \xrightarrow{b} 5$

Deterministický konečný automat



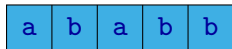
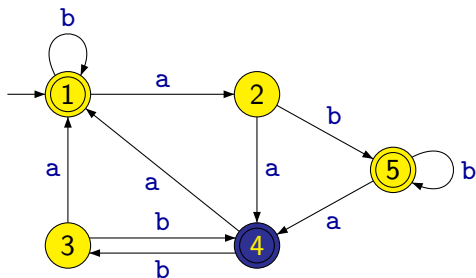
$1 \xrightarrow{a} 2 \xrightarrow{b} 5 \xrightarrow{a} 4$

Deterministický konečný automat



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

Deterministický konečný automat



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

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 $q \xrightarrow{w} q'$.

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

Relaci \longrightarrow můžeme formálně definovat následující induktivní definicí:

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

Deterministický konečný automat

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

Definice

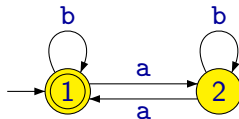
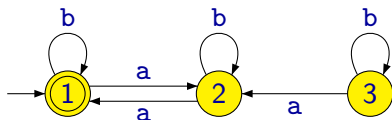
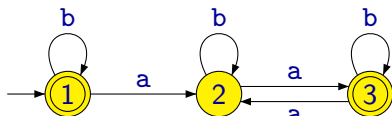
Jazyk rozpoznávaný (přijímaný) daným deterministickým konečným automatem $A = (Q, \Sigma, \delta, q_0, F)$, označovaný $L(A)$, je množina všech slov přijímaných tímto automatem, tj.

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

Definice

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

Ekvivalence automatů

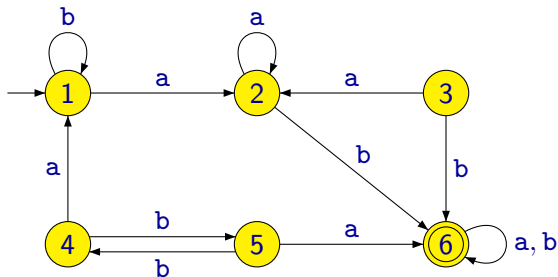


Všechny 3 automaty přijímají jazyk všech slov se sudým počtem a .

Definice

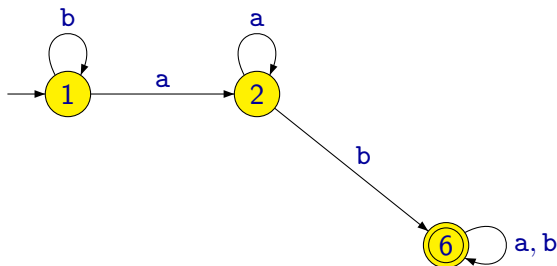
O konečných automatech A_1, A_2 řekneme, že jsou **ekvivalentní**, jestliže $L(A_1) = L(A_2)$.

Nedosažitelné stavy automatu



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

Nedosažitelné stavy automatu



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

Definice

Stav q konečného automatu $A = (Q, \Sigma, \delta, q_0, F)$ 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.