Formální jazyky

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

Základní pojmy

Obecně se teorie formálních jazyků zabývá problémy, kde:

- Pracujeme se sekvencemi **znaků** (říkáme též **symbolů**).
- Znaky 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**.

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: $\mathtt{AHOJ}$ $\mathtt{ABRACADABRA}$ $\mathtt{ERROR}$

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, 314159, 666, 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*

Abeceda a slovo

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···
```

Práce s formálními jazyky

Když chceme nějaký jazyk popsat, máme několik možností:

- Můžeme vyjmenovat všechna jeho slova (což je ale použitelné jen pro 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.

Práce s formálními jazyky

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.: Někteří autoři používají pro označení prázdného slova místo ε řecké písmeno λ (lambda).)

$$|\varepsilon| = 0$$

Zřetězení slov

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.

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

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:

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

Zjevně pro libovolná slova v a w platí:

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

Pro libovolné slovo w také platí:

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

Prefixy, sufixy a podslova

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`, `abb`, `abaa`, `baab`, `abaab`.

Jazyk

Množinu všech slov tvořených symboly z abecedy Σ označujeme Σ^*.

Definice

(Formální) jazyk v abecedě Σ je nějaká libovolná podmnožina množiny Σ^*.

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.

Jazyk

Příklad 4: Uvažujme abecedu Σ tvořenou množinou všech Unicode znaků.

Množina všech komentářů v jazyce Java tvoří jazyk:

- jednořádkové komentáře začínající dvojicí znaků `//` a končící znakem konce řádku (nebo koncem souboru).
- víceřádkové komentáře začínající dvojicí znaků `/*` a končící dvojicí znaků `*/`, přičemž uvnitř se nesmí nacházet žádná další dvojice znaků `*/`.

Pokud bychom množinu všech jednořádkových komentářů označili jako jazyk L_1 a množinu všech víceřádkových komentářů jako jazyk L_2, můžeme množinu všech komentářů označit jako jazyk L, definovaný jako

$$L = L_1 \cup L_2$$

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

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.

Množinové operace na jazycích

Příklad:

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

- 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`.

Zřetězení jazyků

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:

$$abba \qquad abbab \qquad abbbbb \qquad baa \qquad baab \qquad babbb$$

Zřetězení jazyků

Příklad:
Pro nějaký programovací jazyk chceme definovat, jak mohou vypadat konstanty reprezentující čísla v plovoucí řádové čárce (floating-point), např.:

`1e1` `2.0` `.3` `0.0` `3.14` `1E-9` `4.5e137`

Abeceda $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ., e, E, +, -\}$

Zřetězení jazyků

Příklad:
Pro nějaký programovací jazyk chceme definovat, jak mohou vypadat konstanty reprezentující čísla v plovoucí řádové čárce (floating-point), např.:

`1e1` `2.0` `.3` `0.0` `3.14` `1E-9` `4.5e137`

Abeceda $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ., e, E, +, -\}$

Pokud zvolíme L_{num} jako množinu všech (neprázdných) slov tvořených pouze číslicemi, a $L_{dot} = \{.\}$, můžeme konstanty jako například `3467.982`, `3.141592` nebo `0.0` popsat takto:

$$L_{num} \cdot L_{dot} \cdot L_{num}$$

Zřetězení jazyků

Definice jazyka L všech možných konstant v plovoucí řádové čárce by pak mohla vypadat například takto:

$$\begin{aligned} L \quad = \quad & L_{num} \cdot L_{dot} \cdot (L_{num} \cup \{\varepsilon\}) \cdot (L_{exp} \cup \{\varepsilon\}) \quad \cup \\ & L_{dot} \cdot L_{num} \cdot (L_{exp} \cup \{\varepsilon\}) \quad \cup \\ & L_{num} \cdot L_{exp} \end{aligned}$$

kde

$$\begin{aligned} L_{num} \quad & - \quad \text{neprázdné sekvence číslic} \\ L_{dot} \quad & = \quad \{\,.\,\} \\ L_{exp} \quad & = \quad L_E \cdot L_{sign} \cdot L_{num} \\ L_E \quad & = \quad \{\mathtt{E}, \mathtt{e}\} \\ L_{sign} \quad & = \quad \{\varepsilon, +, -\} \end{aligned}$$

Iterace jazyka

Používáme následující zápis:

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

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

$$aaaaaa \quad aaaab \quad aabaa \quad aabb \quad baaaa \quad baab \quad bbaa \quad bbb$$

Definujeme

$$\begin{aligned} L^1 &= L \\ L^0 &= \{\varepsilon\} \end{aligned}$$

Iterace jazyka

Induktivní definice pro libovolné $k \geq 0$:

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

Definice

Iterace jazyka L je jazyk

$$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \cdots$$

tj. jazyk tvořený slovy vzniklými zřetězením libovolného počtu slov z jazyka L.

Příklad: $L = \{aa, b\}$

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

Iterace jazyka

Příklad: $L_{dig} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

$$L_{num} = L_{dig} \cdot L_{dig}^{*}$$

Poznámka: Používá se také následující značení:

$$L^{+} = L^{1} \cup L^{2} \cup L^{3} \cup L^{4} \cup \cdots$$

Řešení předchozího příkladu bychom tedy také mohli zapsat stručněji:

$$L_{num} = L_{dig}^{+}$$

Zrcadlový obraz

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

Zrcadlový obraz slova w značíme w^R.

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

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 $\Sigma = \{a_1, a_2, \ldots, a_n\}$, tak platí

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

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

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

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

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

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 $<_L$.

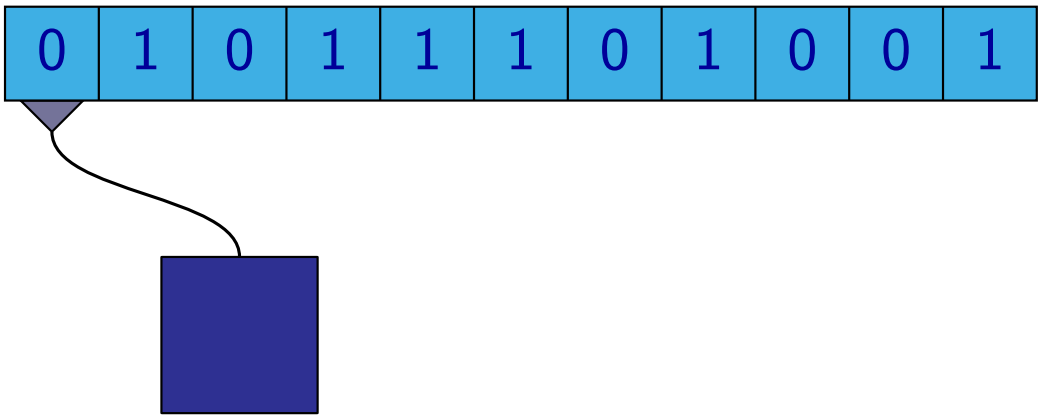
Konečné automaty

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.

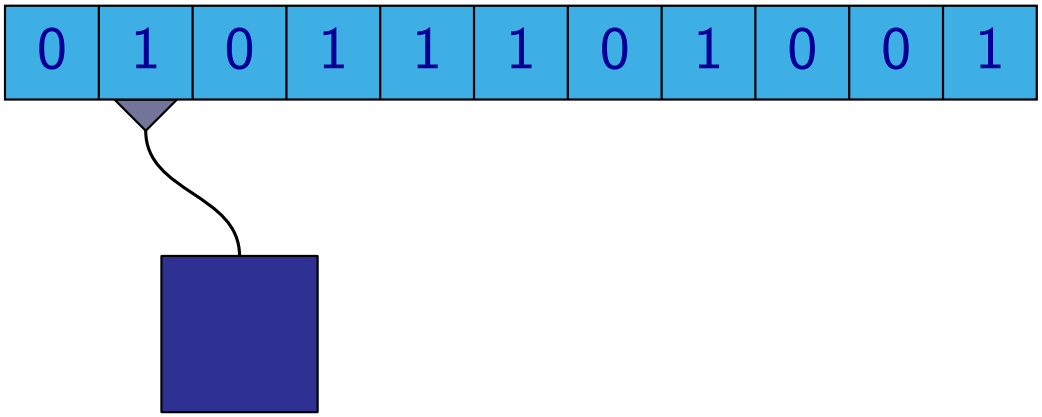


Rozpoznávání jazyka

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.

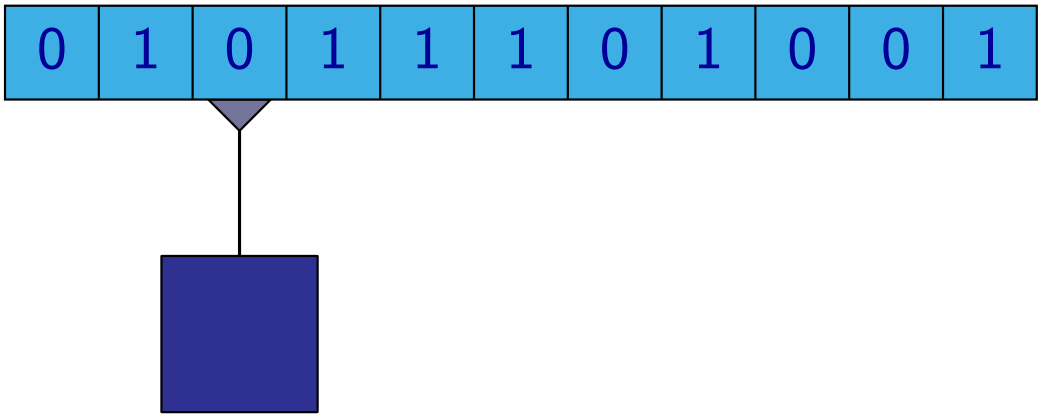


Rozpoznávání jazyka

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.

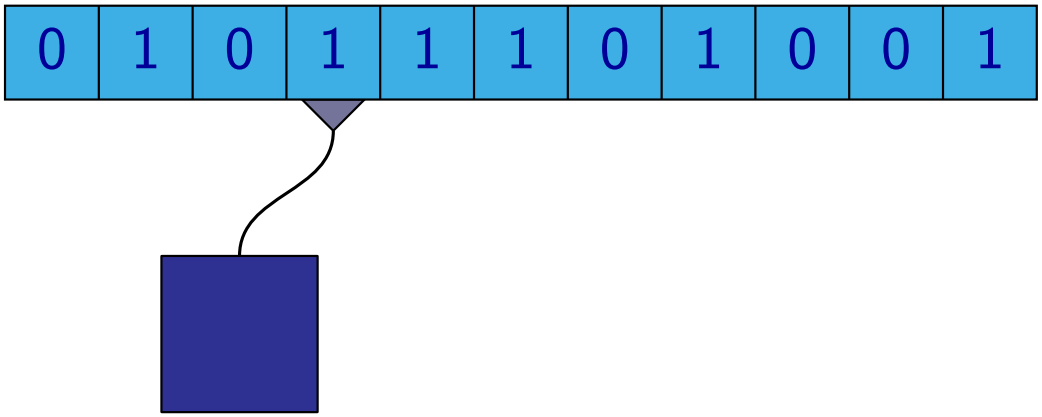


Rozpoznávání jazyka

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.

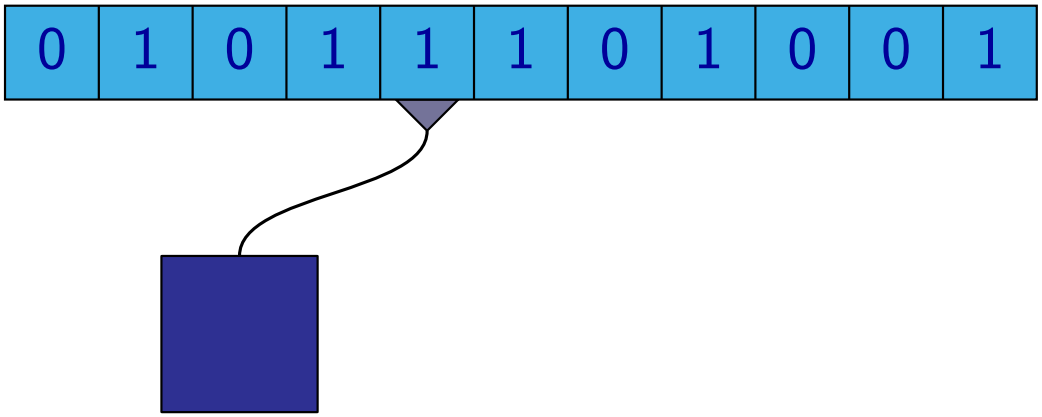


Rozpoznávání jazyka

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.

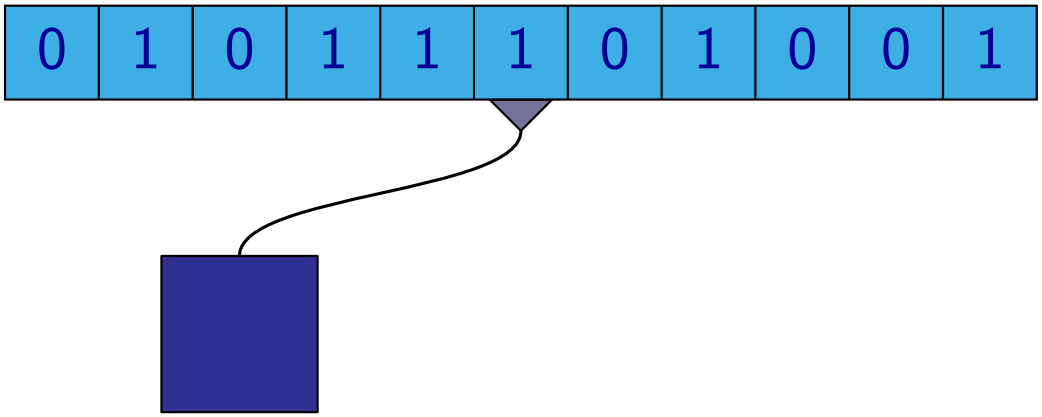


Rozpoznávání jazyka

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.

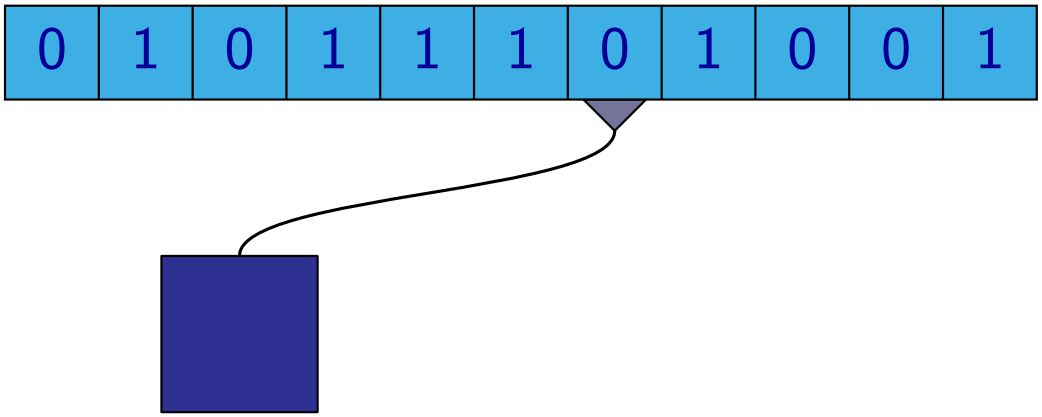


Rozpoznávání jazyka

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.

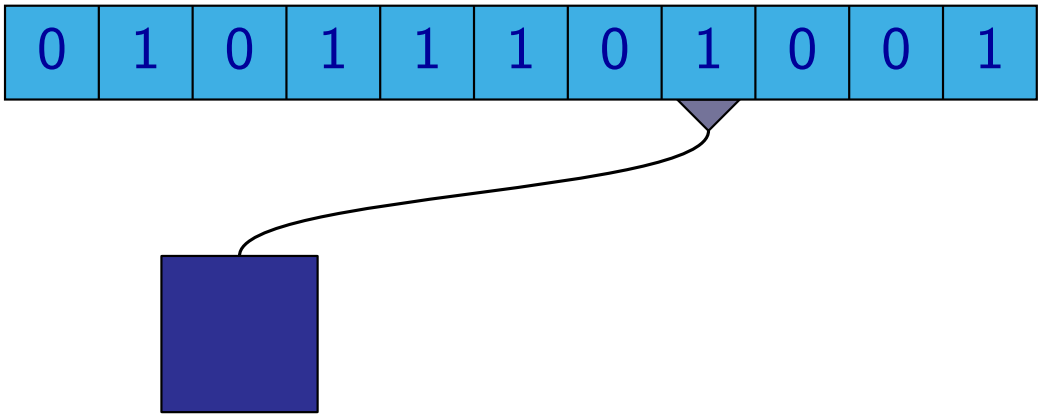


Rozpoznávání jazyka

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.

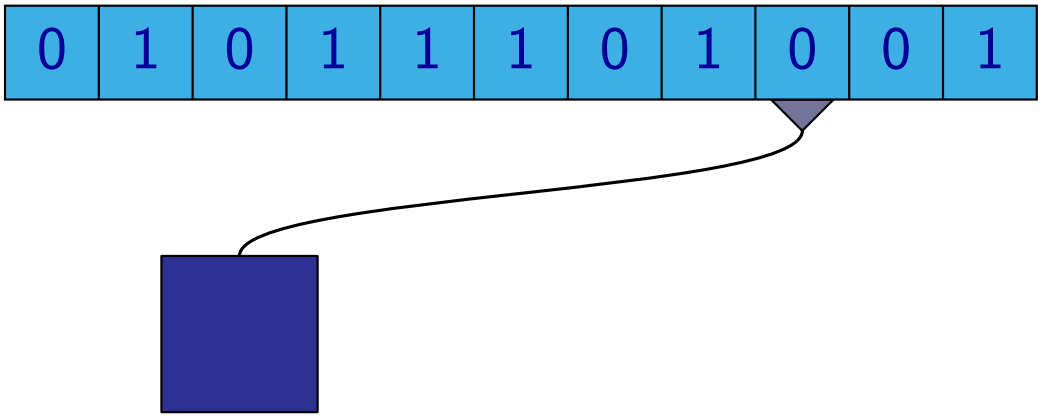


Rozpoznávání jazyka

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.

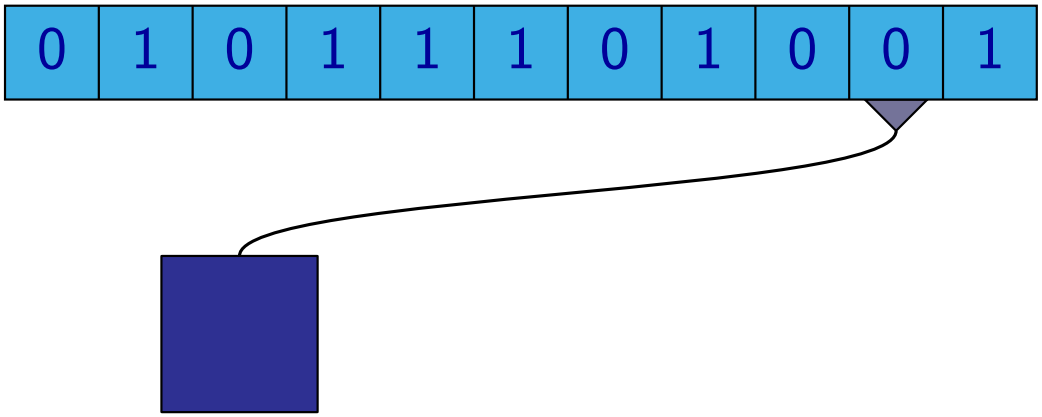


Rozpoznávání jazyka

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.

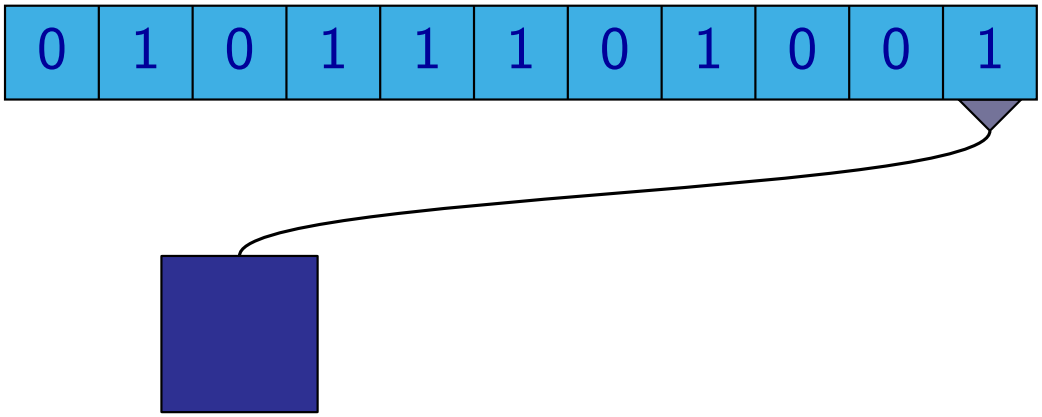


Rozpoznávání jazyka

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.

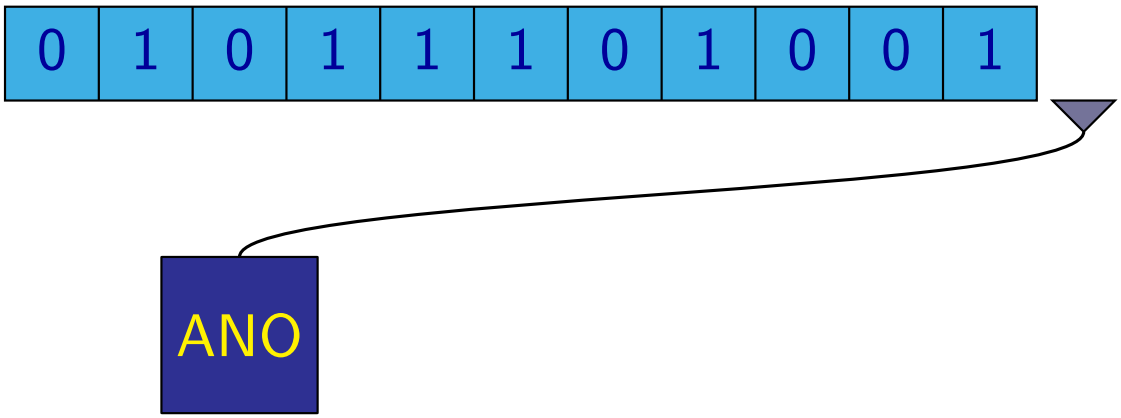


Rozpoznávání jazyka

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

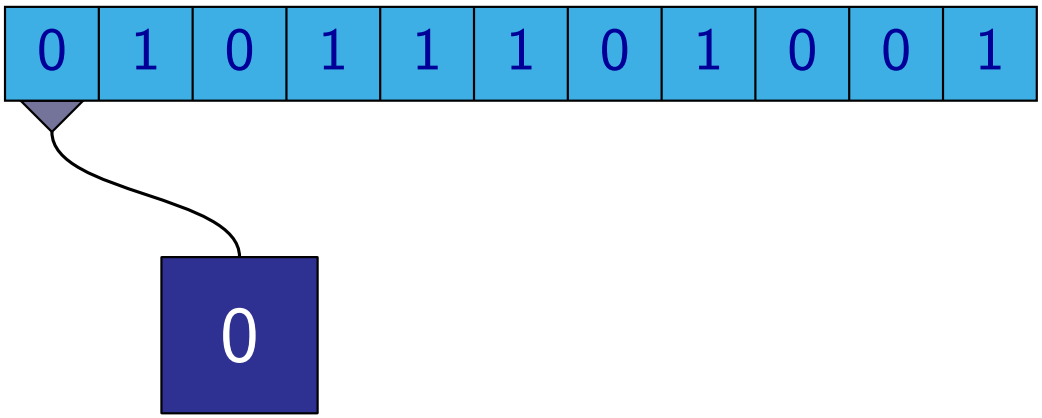
Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1.

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



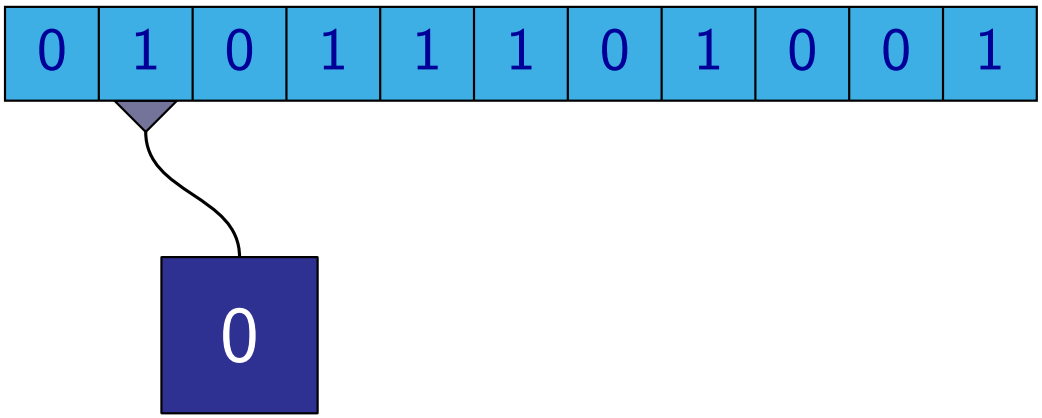
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



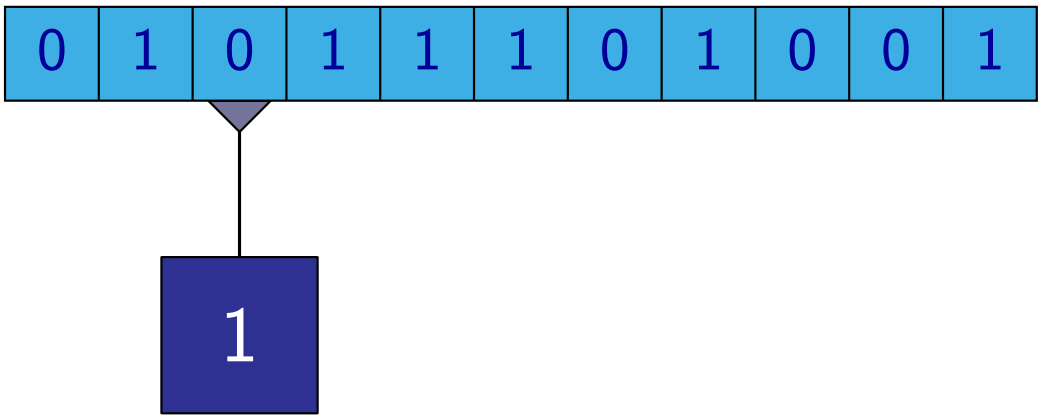
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



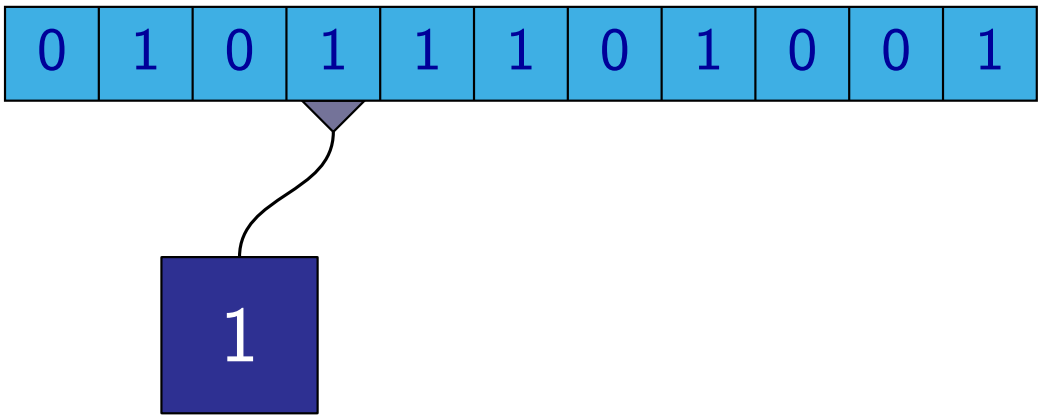
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



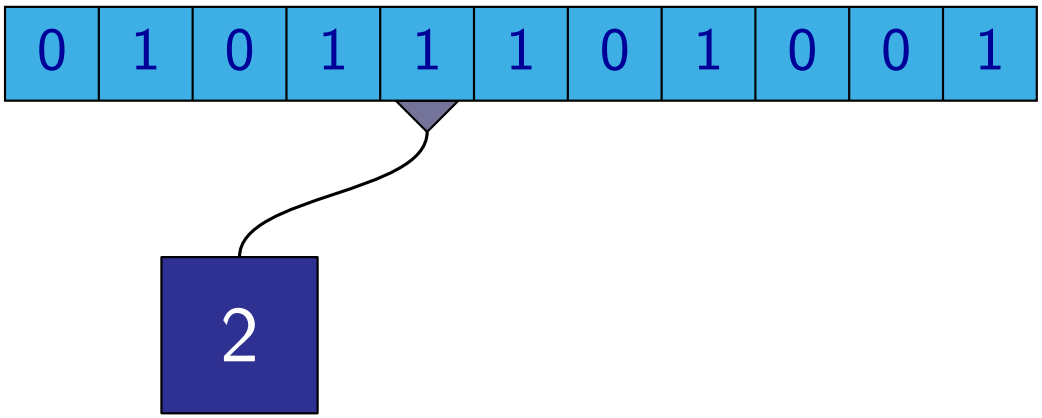
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



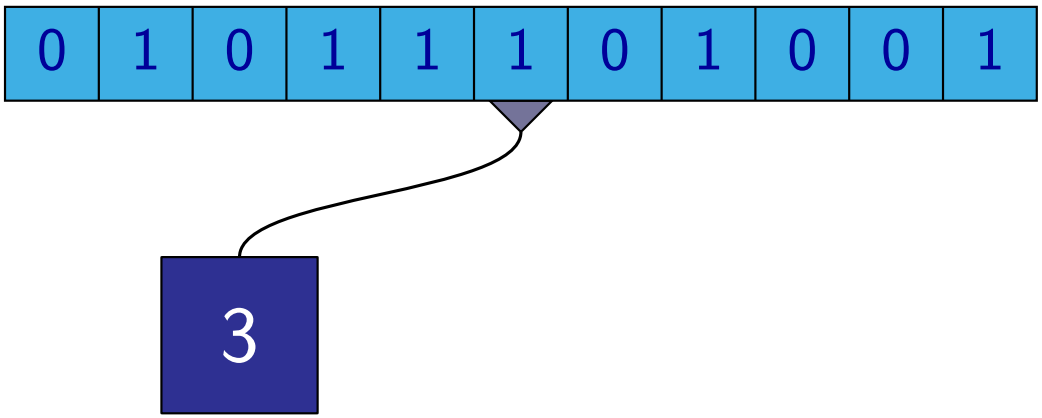
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



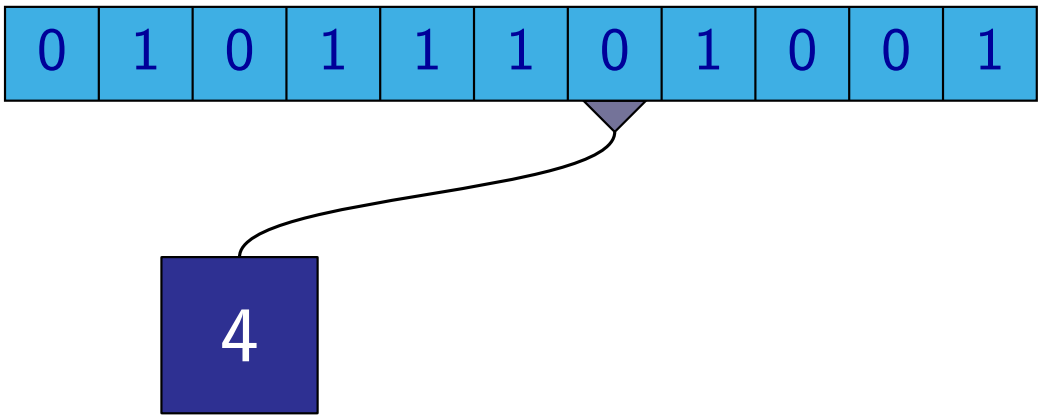
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



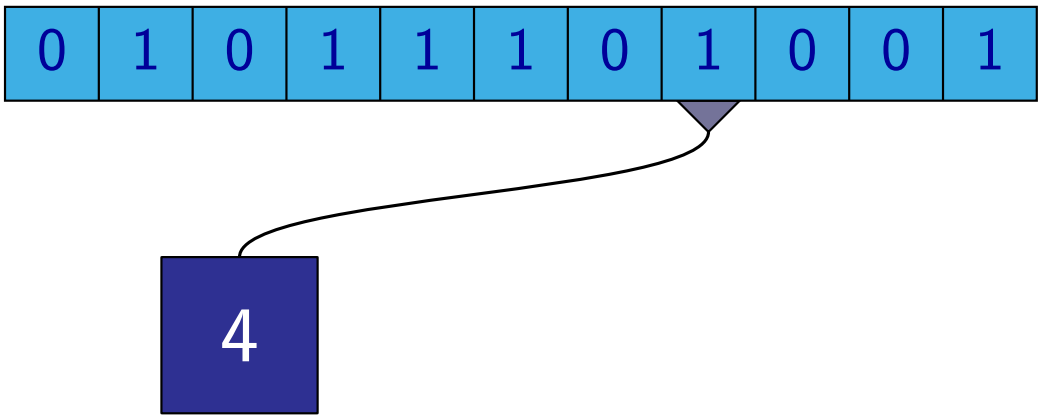
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



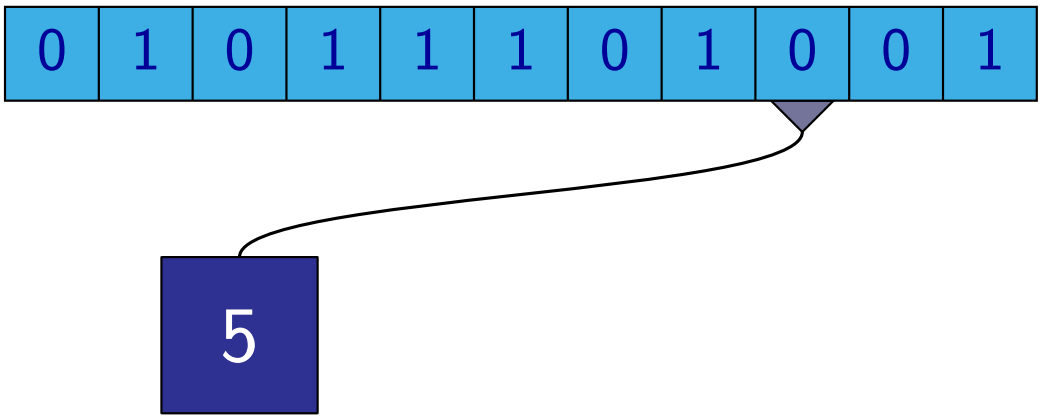
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



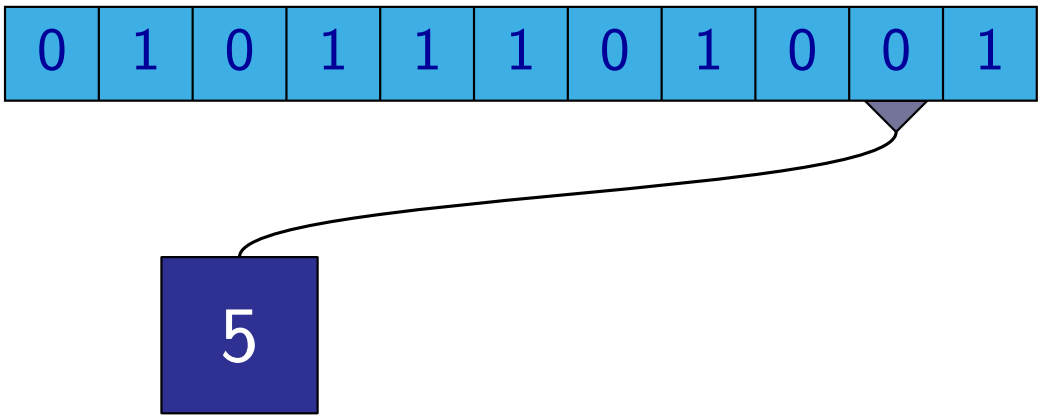
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



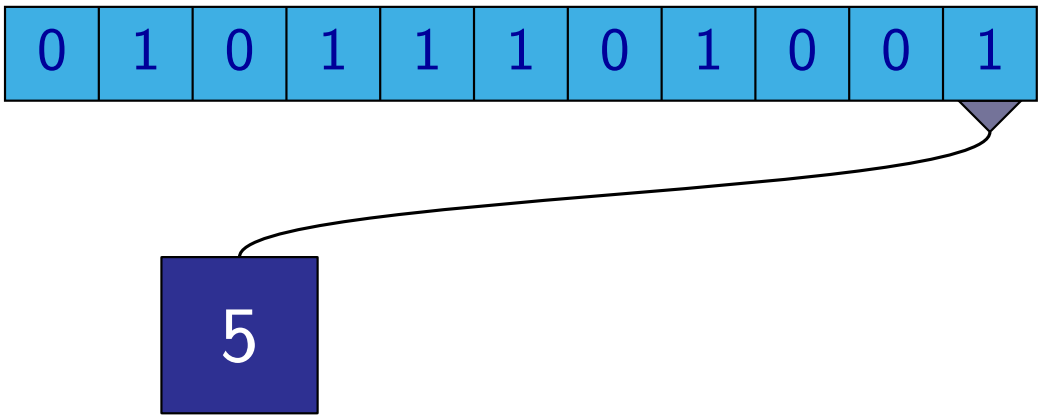
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



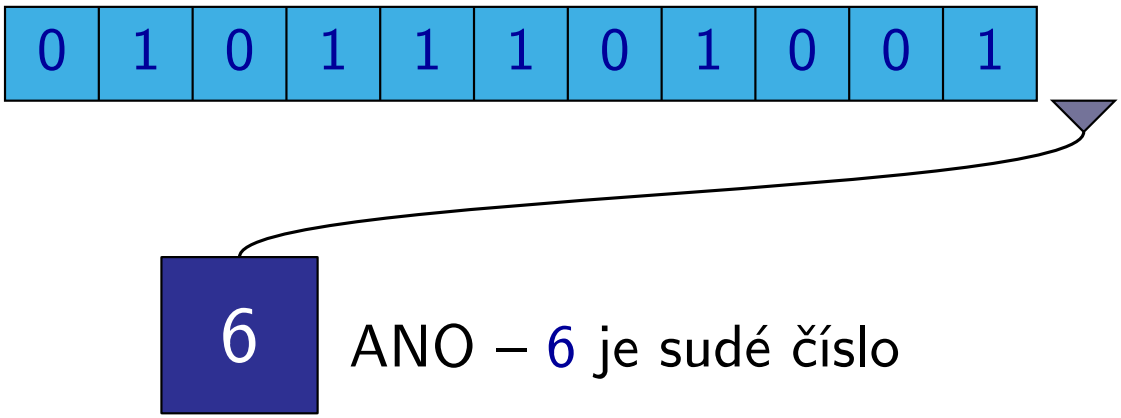
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



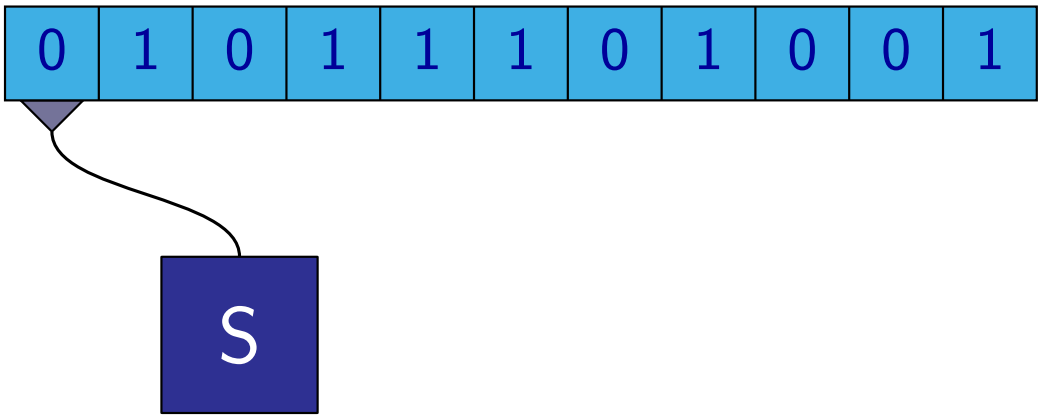
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



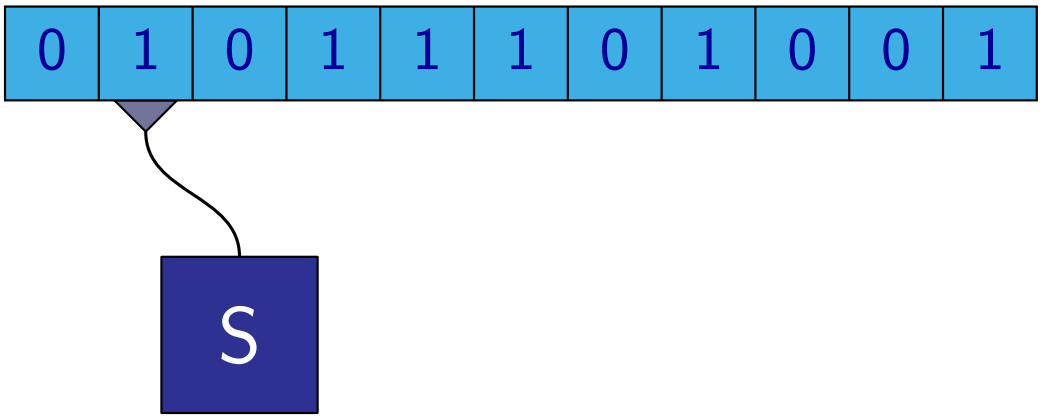
Rozpoznávání jazyka

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



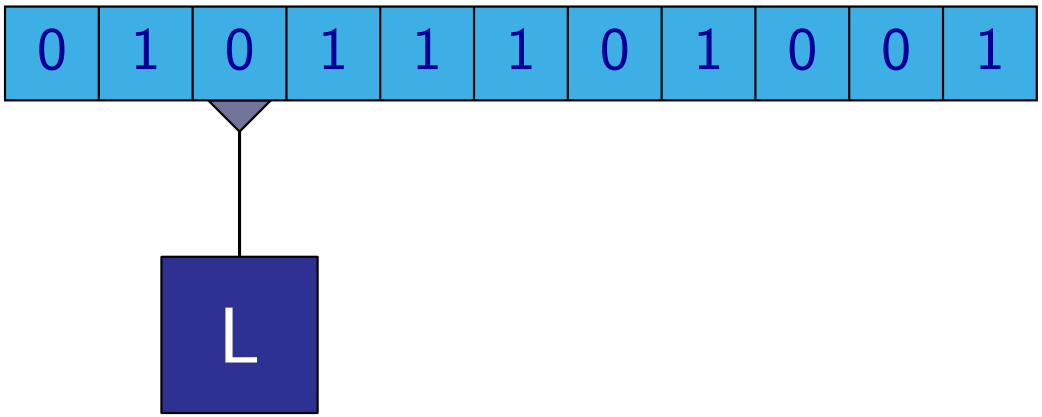
Rozpoznávání jazyka

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



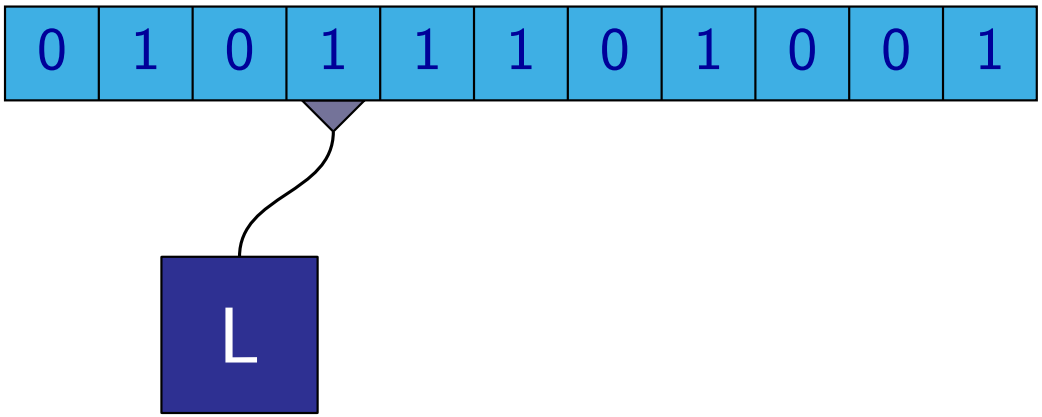
Rozpoznávání jazyka

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



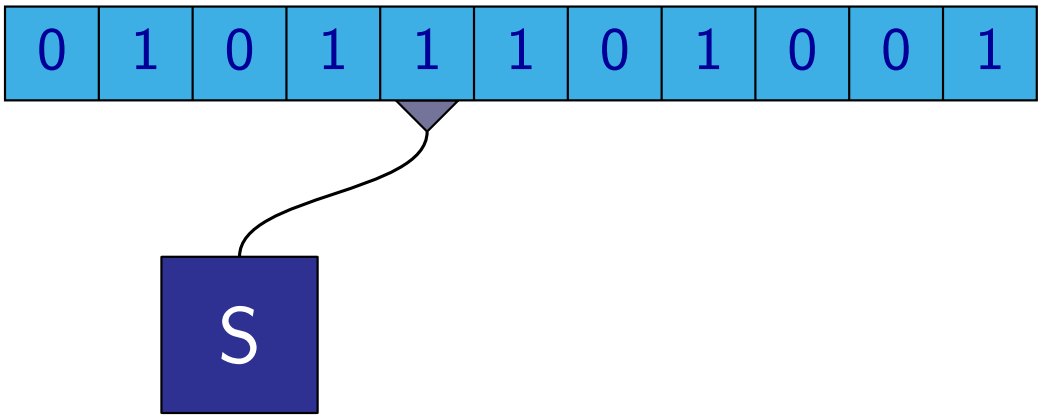
Rozpoznávání jazyka

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



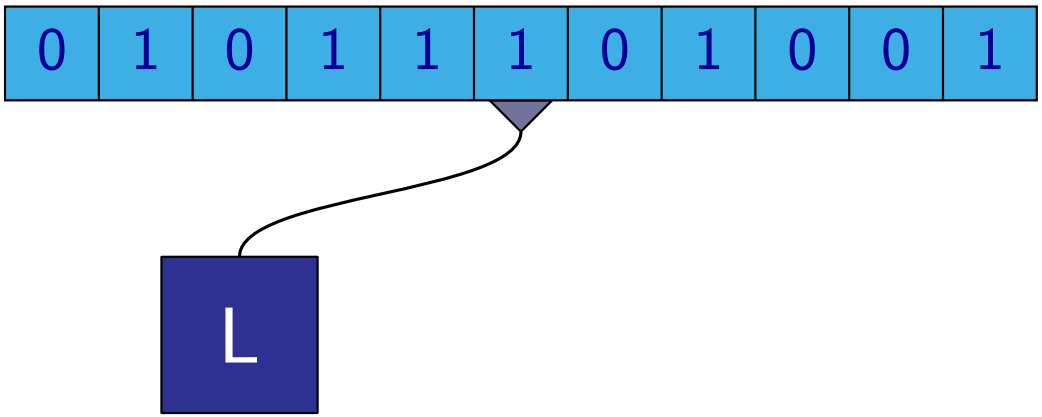
Rozpoznávání jazyka

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



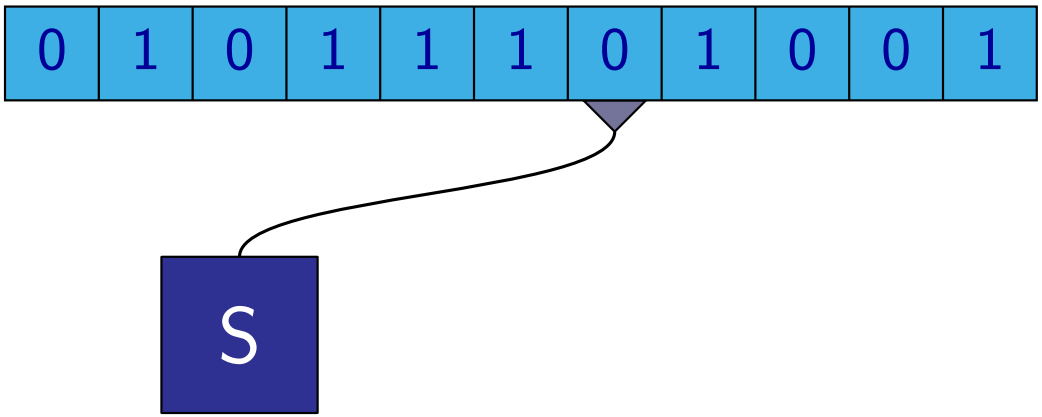
Rozpoznávání jazyka

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



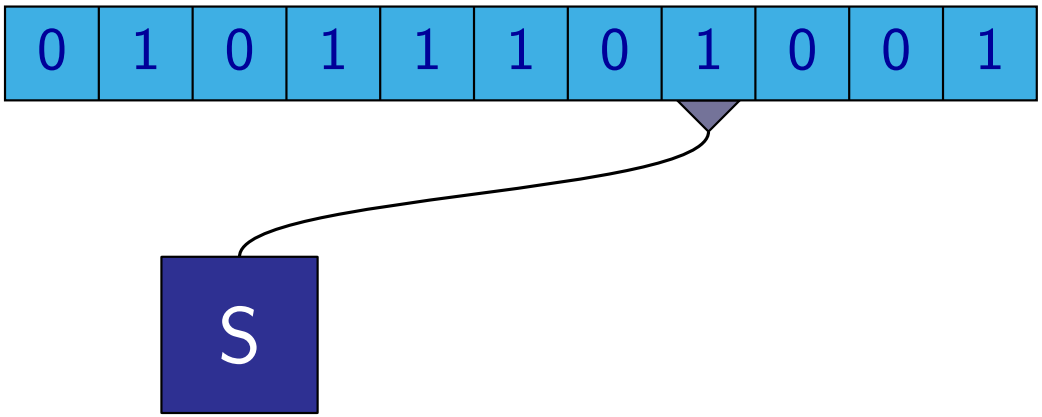
Rozpoznávání jazyka

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



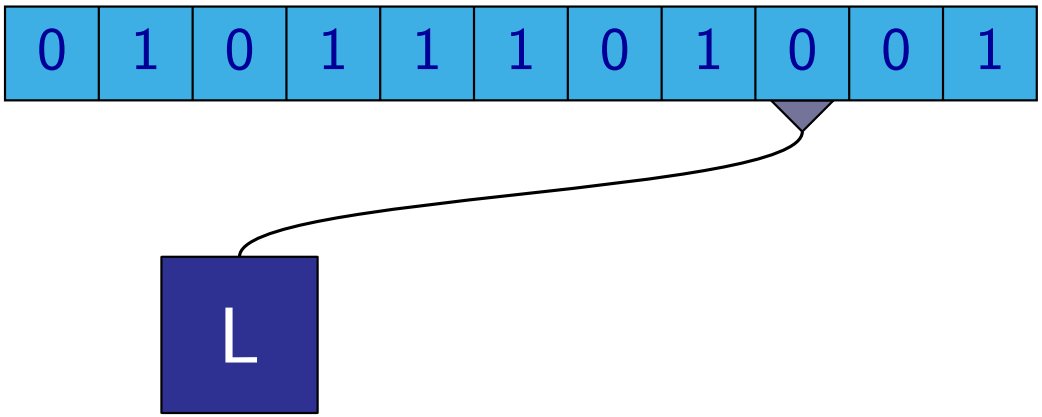
Rozpoznávání jazyka

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



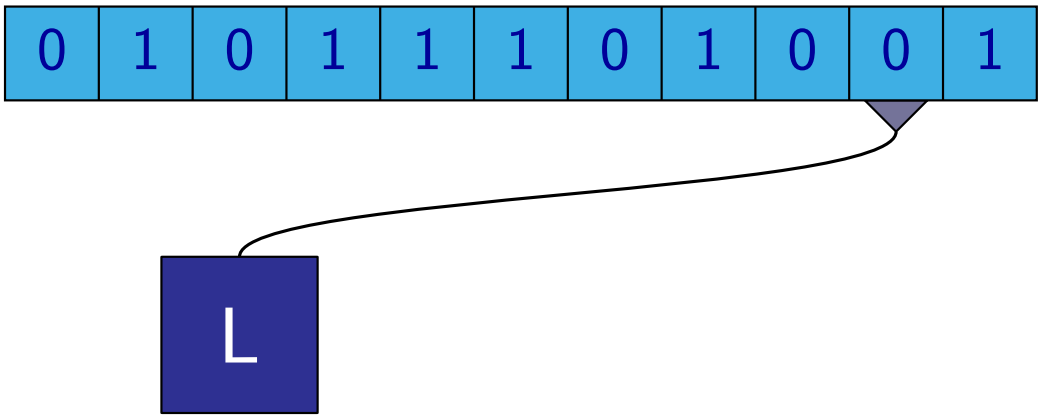
Rozpoznávání jazyka

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



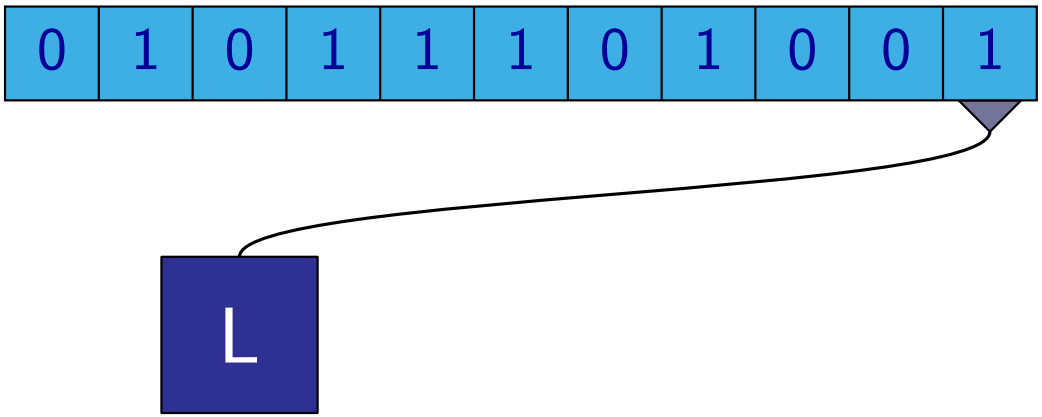
Rozpoznávání jazyka

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



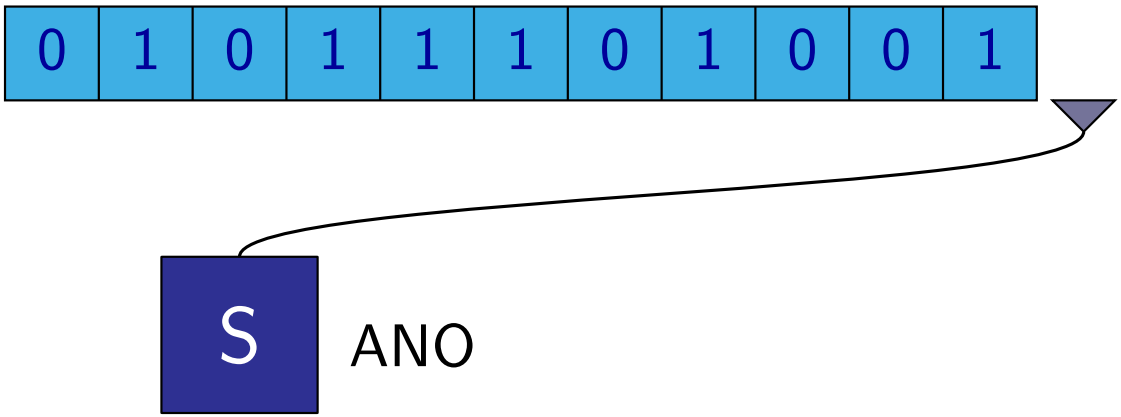
Rozpoznávání jazyka

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



Rozpoznávání jazyka

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



Rozpoznávání jazyka

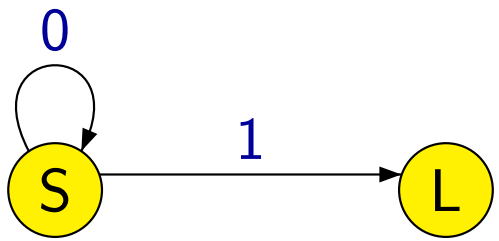
Chovaní tohoto zařízení můžeme popsat grafem:

Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:

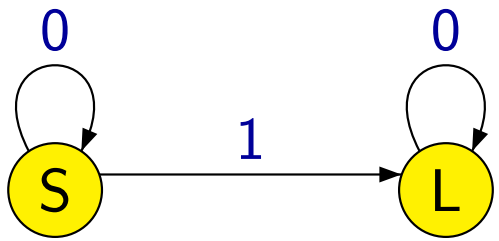
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



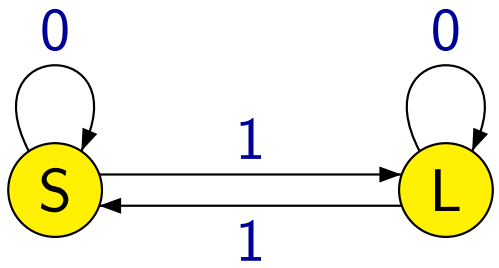
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



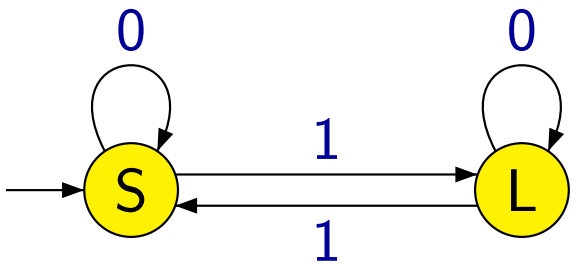
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



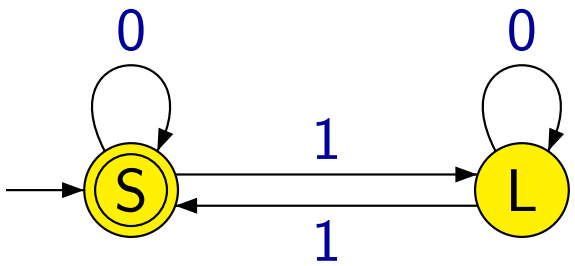
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



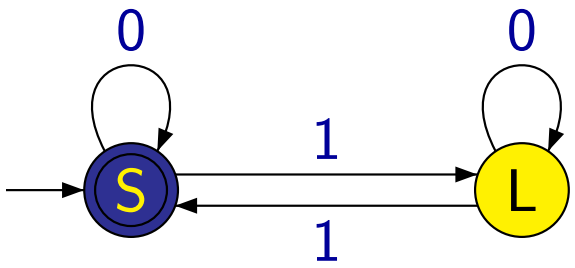
Rozpoznávání jazyka

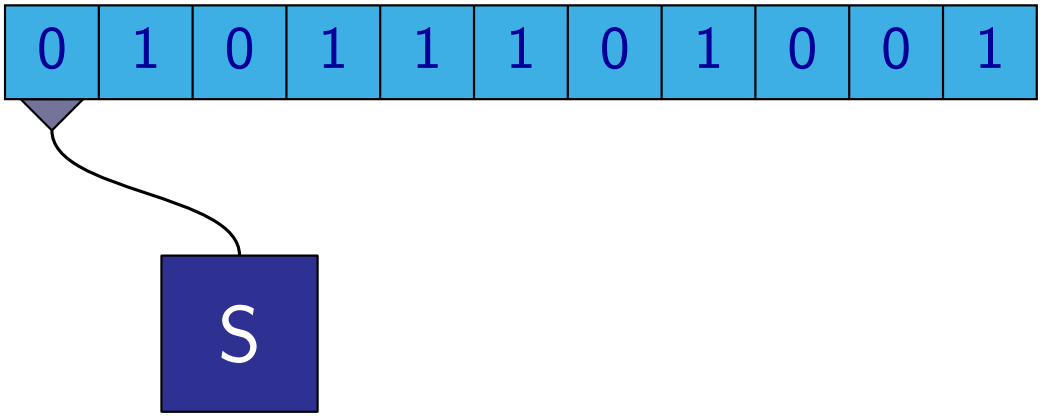
Chovaní tohoto zařízení můžeme popsat grafem:



Rozpoznávání jazyka

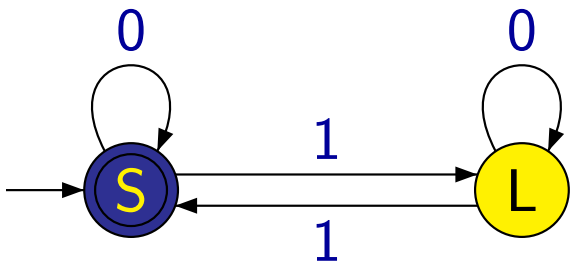
Chovaní tohoto zařízení můžeme popsat grafem:

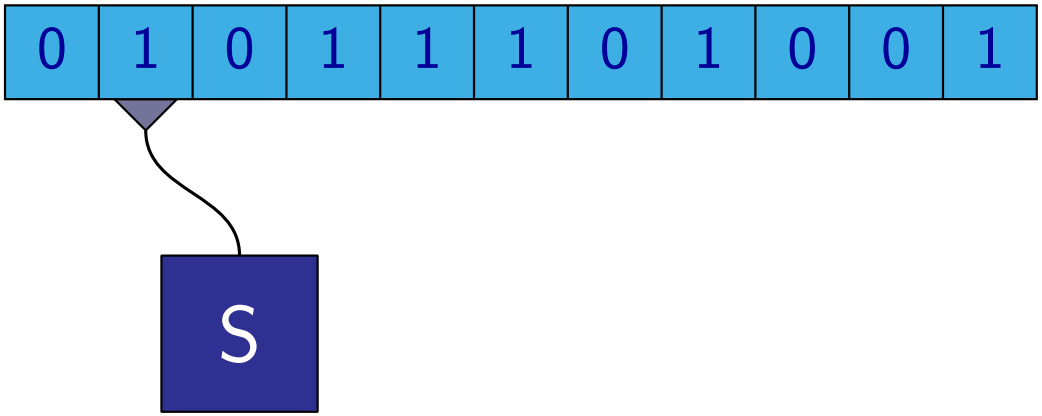




Rozpoznávání jazyka

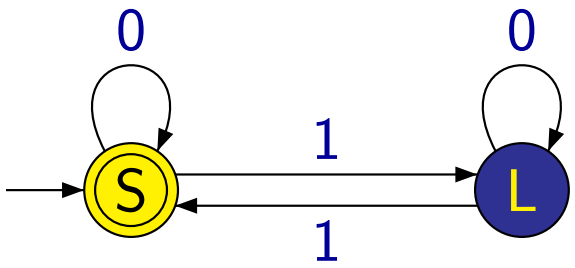
Chovaní tohoto zařízení můžeme popsat grafem:

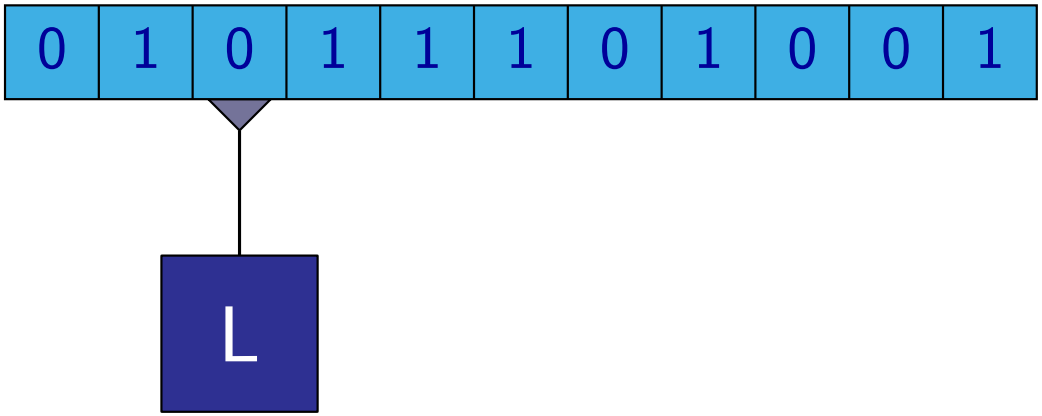




Rozpoznávání jazyka

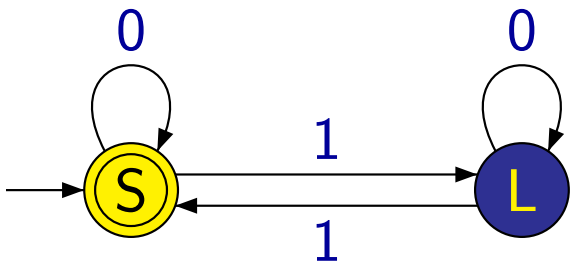
Chovaní tohoto zařízení můžeme popsat grafem:

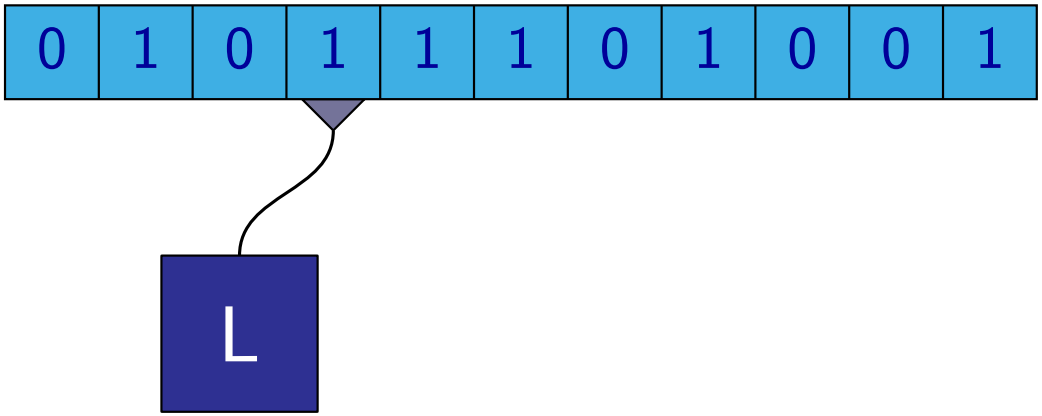




Rozpoznávání jazyka

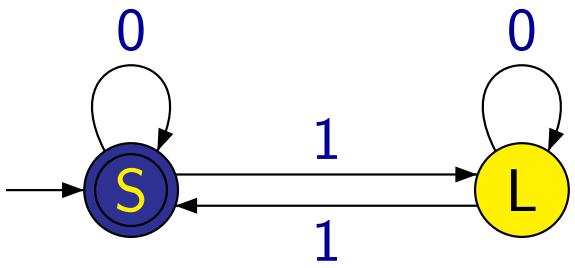
Chovaní tohoto zařízení můžeme popsat grafem:

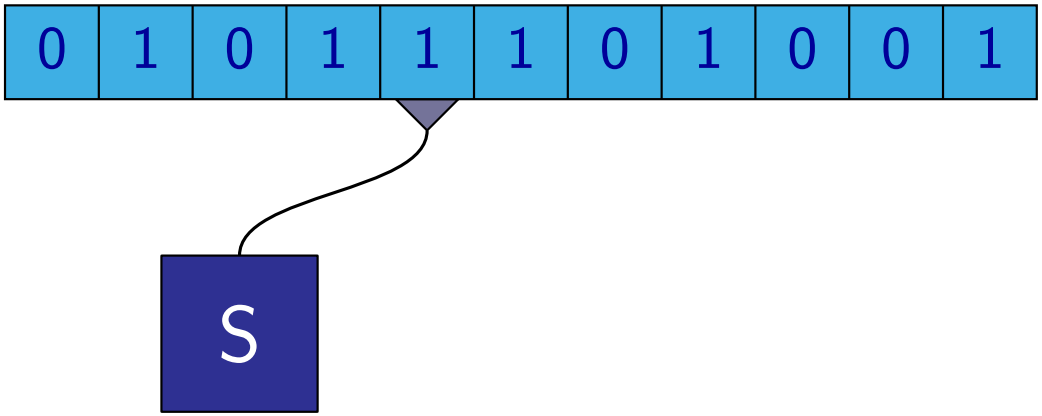




Rozpoznávání jazyka

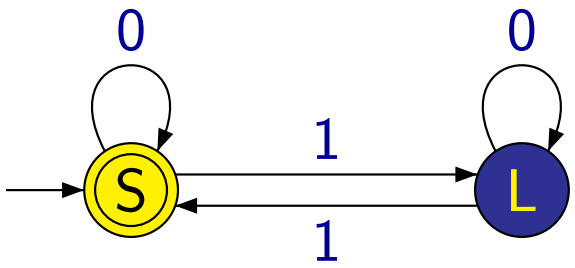
Chovaní tohoto zařízení můžeme popsat grafem:

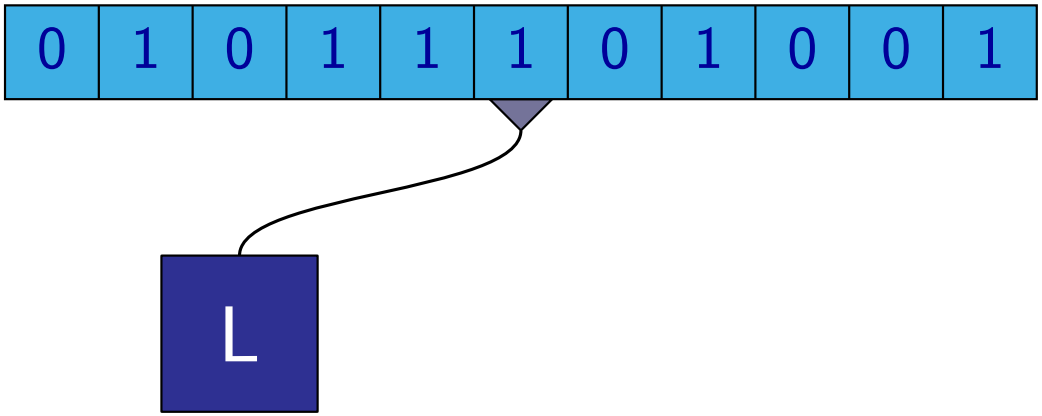




Rozpoznávání jazyka

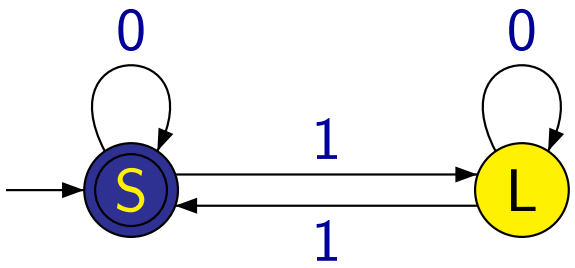
Chovaní tohoto zařízení můžeme popsat grafem:

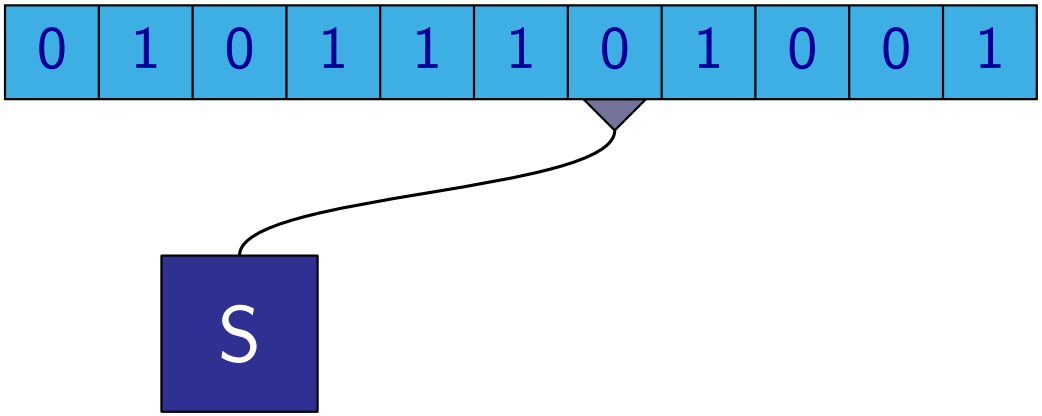




Rozpoznávání jazyka

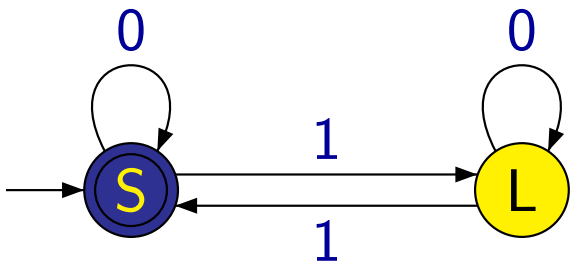
Chovaní tohoto zařízení můžeme popsat grafem:

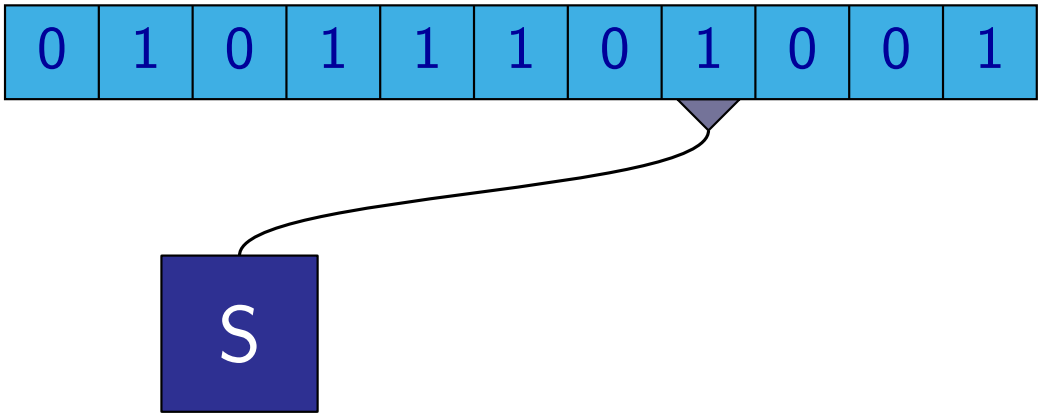




Rozpoznávání jazyka

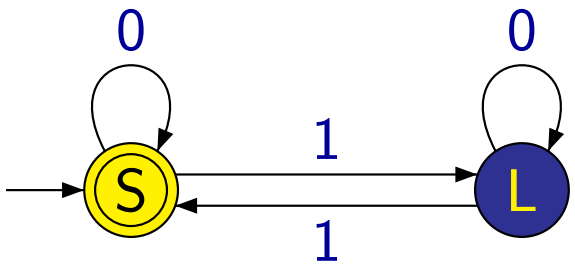
Chovaní tohoto zařízení můžeme popsat grafem:

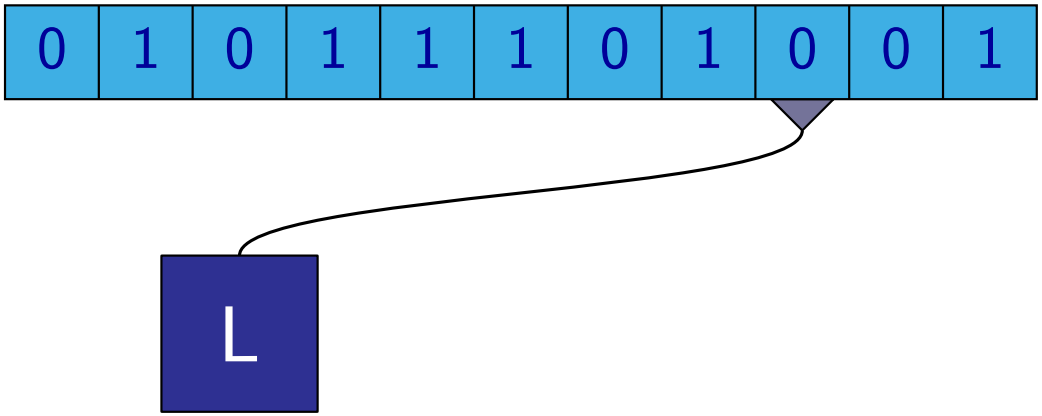




Rozpoznávání jazyka

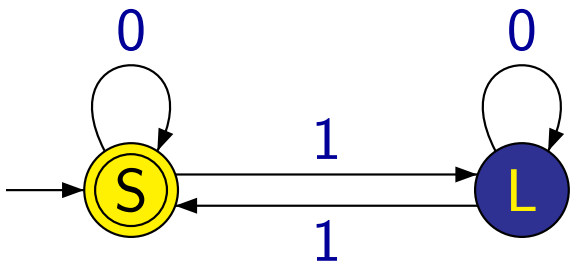
Chovaní tohoto zařízení můžeme popsat grafem:

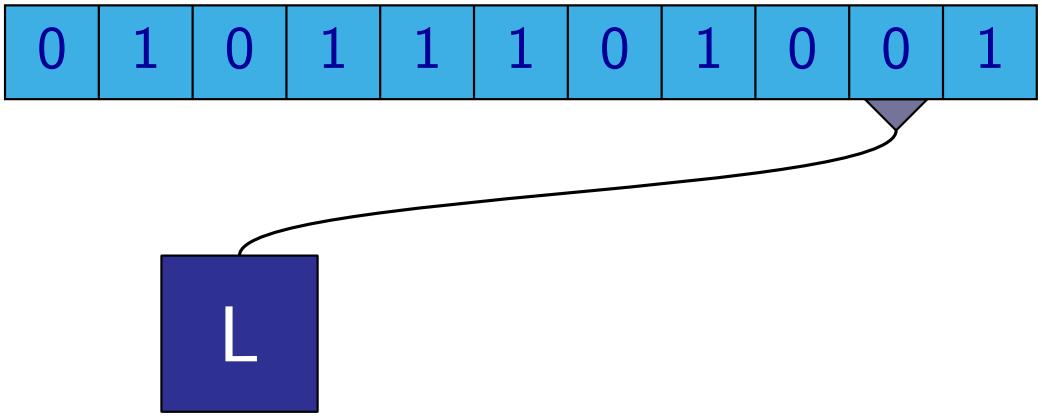




Rozpoznávání jazyka

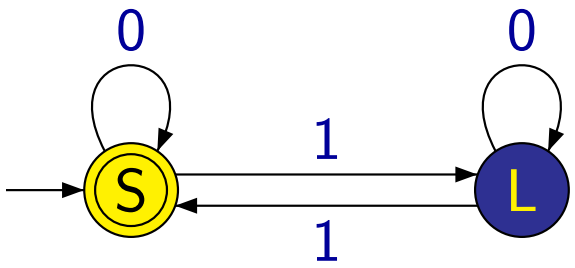
Chovaní tohoto zařízení můžeme popsat grafem:

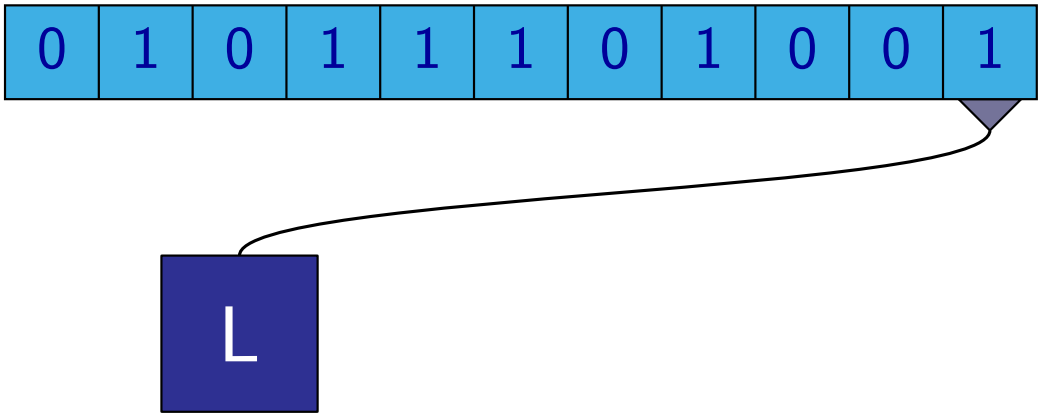




Rozpoznávání jazyka

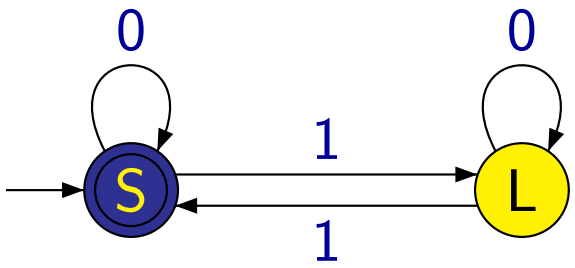
Chovaní tohoto zařízení můžeme popsat grafem:

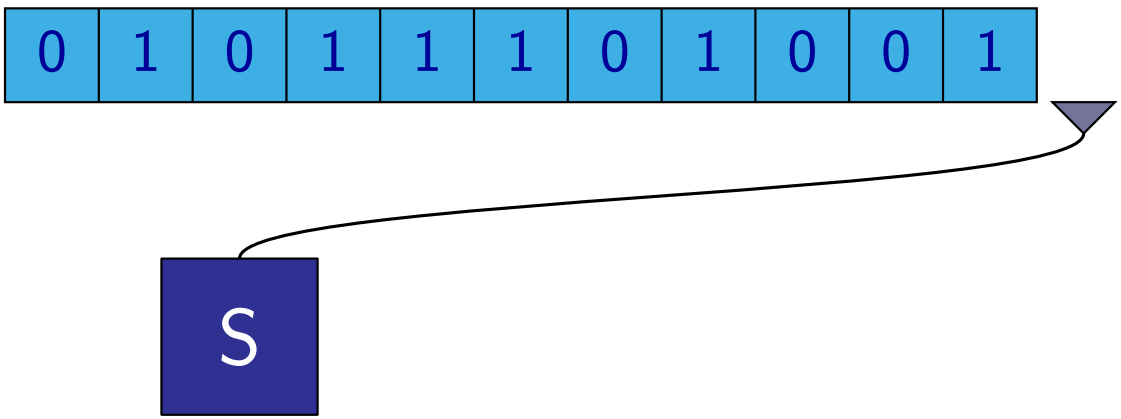




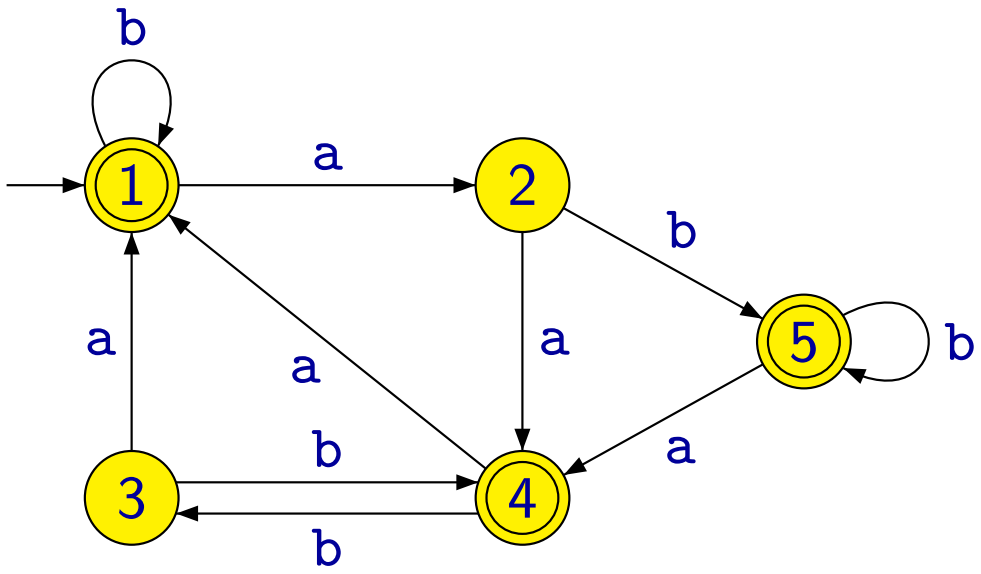
Rozpoznávání jazyka

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

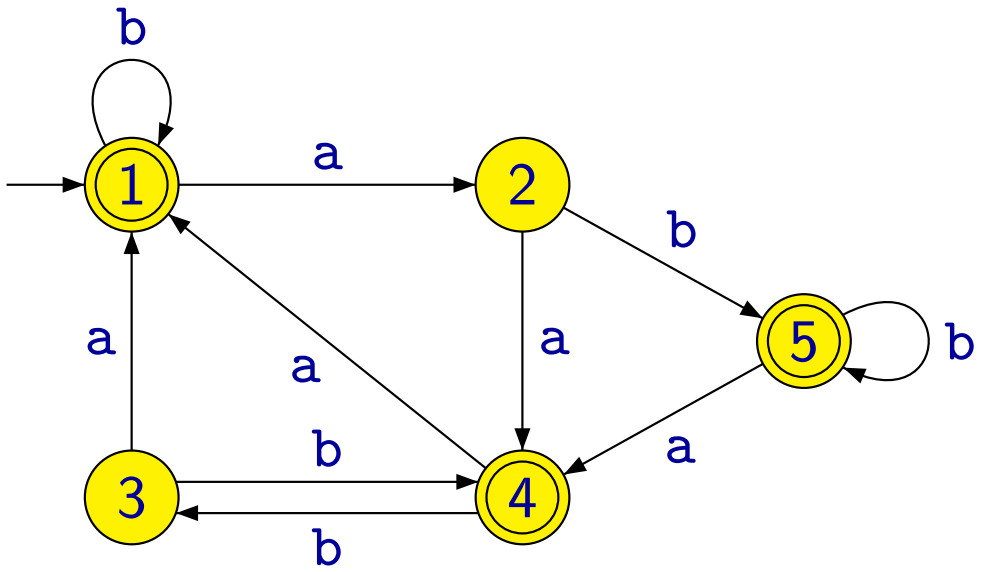
Formálně je **deterministický konečný automat** definován jako pětice

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

kde:

- Q je konečná množina **stavů**
- Σ je konečná **abeceda**
- $\delta : Q \times \Sigma \to Q$ 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
1	2	1
2	4	5
3	1	4
4	1	3
5	4	5

Deterministický konečný automat

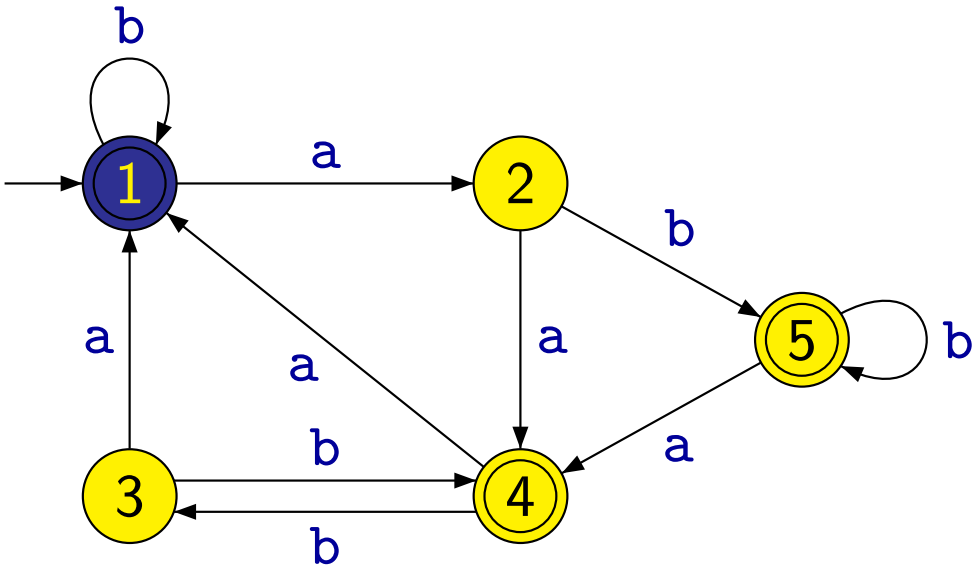
Místo zápisu

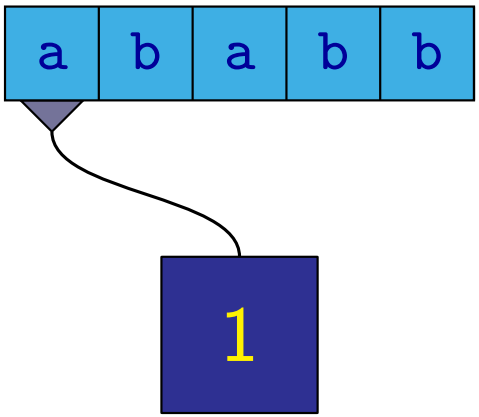
$$\begin{array}{ll} \delta(1,\mathtt{a}) = 2 & \delta(1,\mathtt{b}) = 1 \\ \delta(2,\mathtt{a}) = 4 & \delta(2,\mathtt{b}) = 5 \\ \delta(3,\mathtt{a}) = 1 & \delta(3,\mathtt{b}) = 4 \\ \delta(4,\mathtt{a}) = 1 & \delta(4,\mathtt{b}) = 3 \\ \delta(5,\mathtt{a}) = 4 & \delta(5,\mathtt{b}) = 5 \end{array}$$

budeme raději používat stručnější tabulku nebo grafické znázornění:

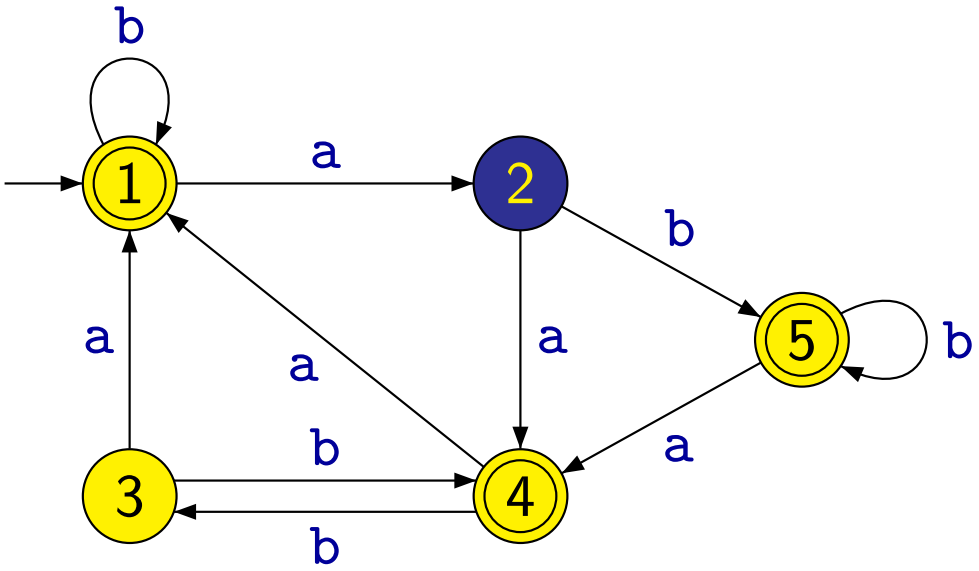
δ	a	b
$\leftrightarrow 1$	2	1
2	4	5
3	1	4
$\leftarrow 4$	1	3
$\leftarrow 5$	4	5

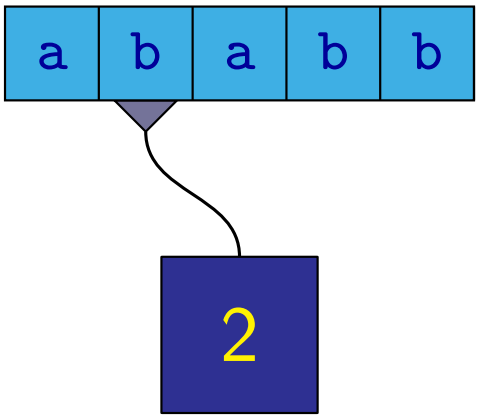
Deterministický konečný automat



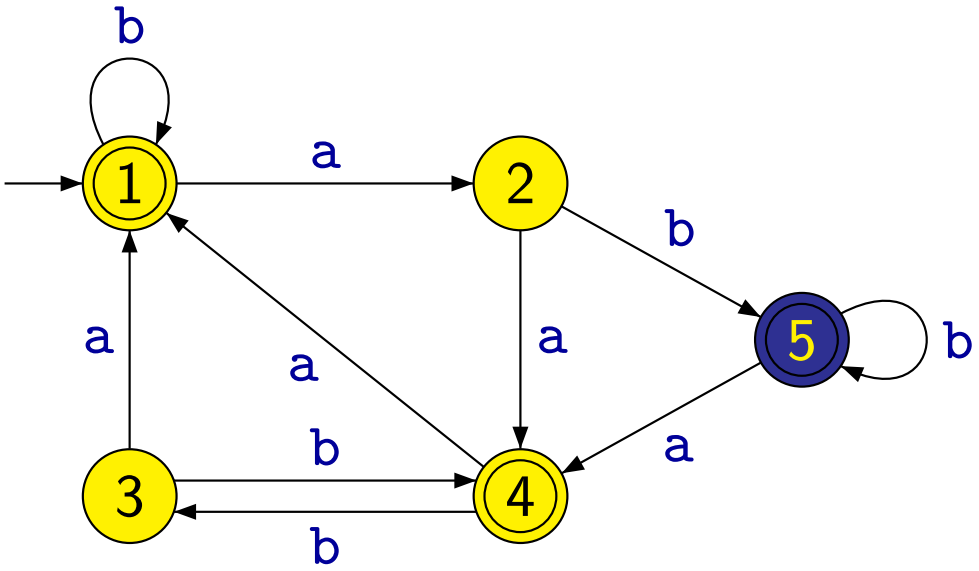


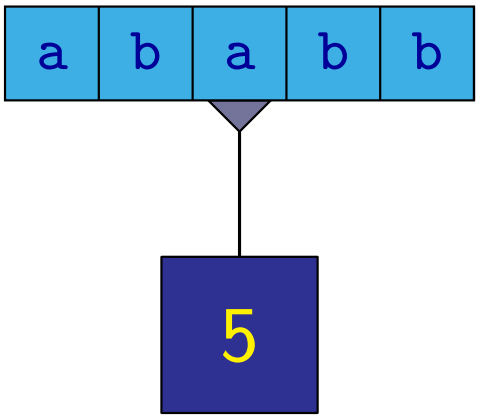
Deterministický konečný automat



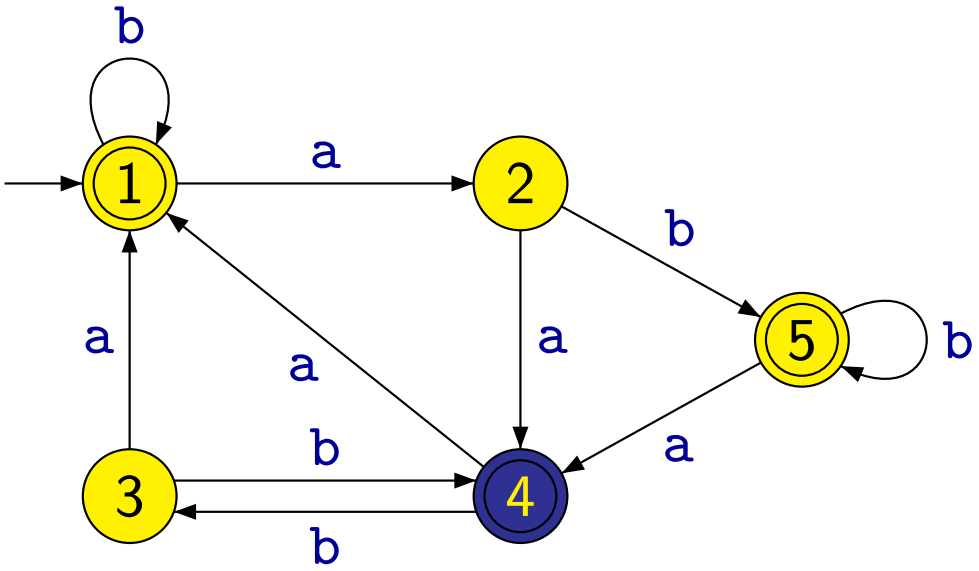


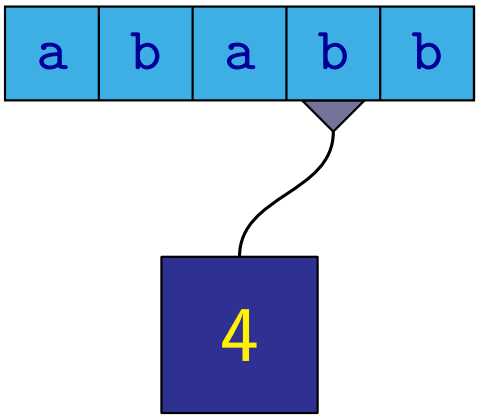
Deterministický konečný automat



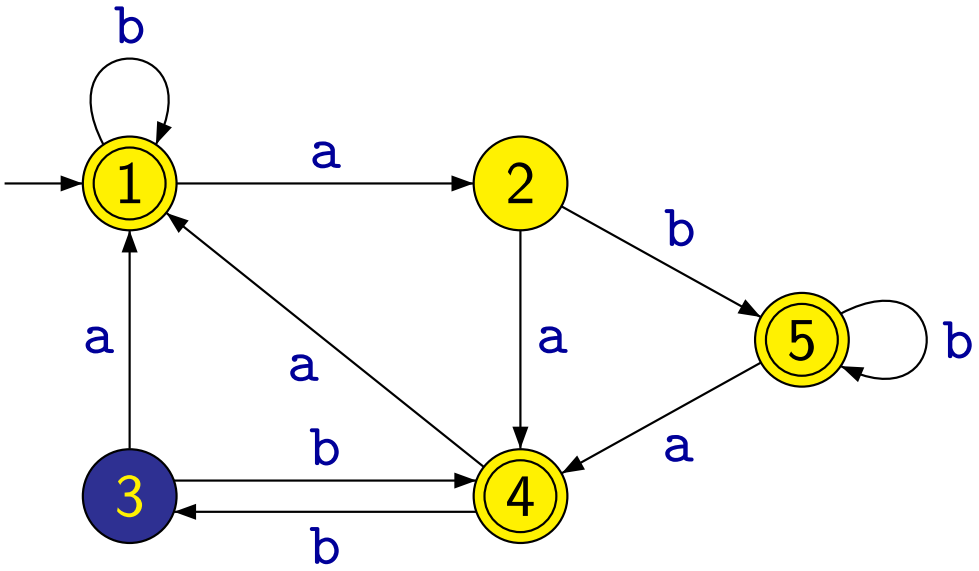


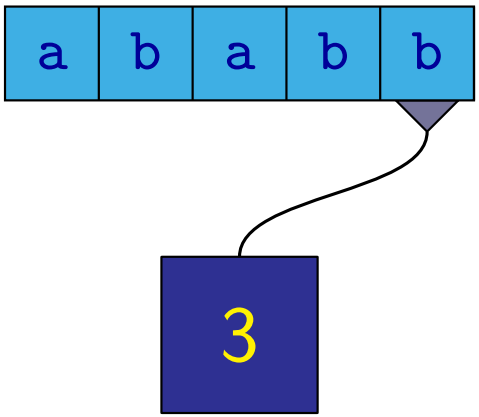
Deterministický konečný automat



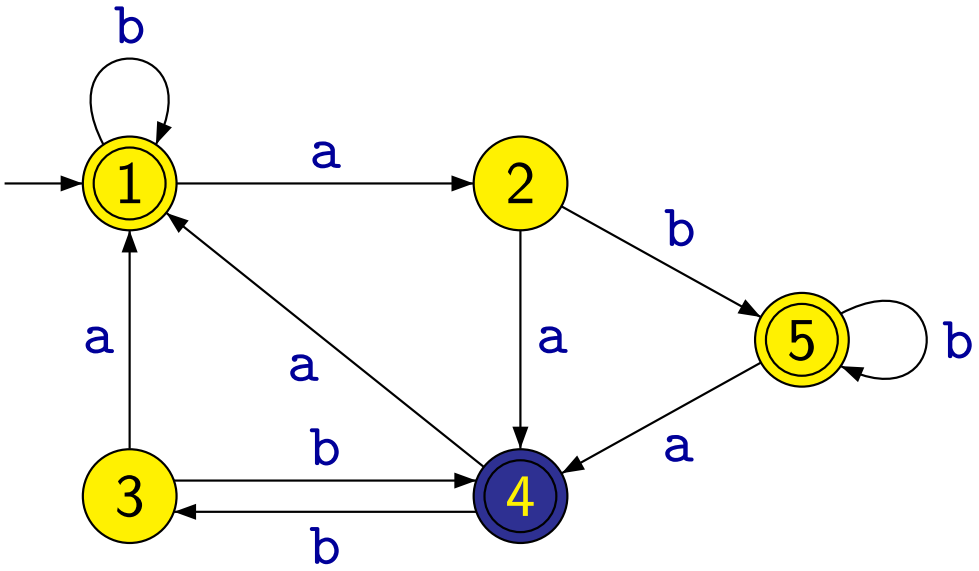


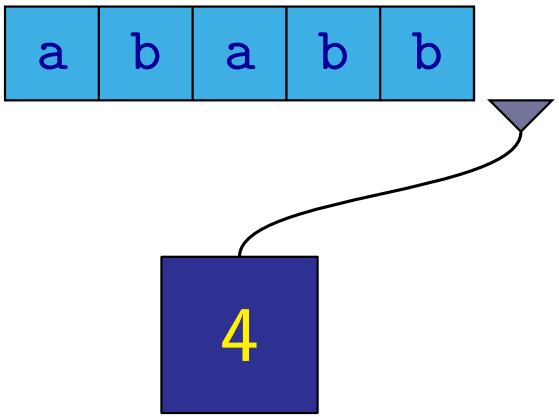
Deterministický konečný automat





Deterministický konečný automat





Deterministický konečný automat

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

Formálně můžeme konfiguraci definovat jako dvojici z množiny $Q \times \Sigma^*$.

Příklad: $(2, \texttt{babb})$ je konfigurace.

Na množině všech konfigurací můžeme definovat binární relaci $\vdash$ s následujícím významem: $C_1 \vdash C_2$ znamená, že automat může přejít jedním krokem z konfigurace C_1 do konfigurace C_2.

Příklad:

$$(2, \texttt{babb}) \vdash (5, \texttt{abb})$$

Formálně platí, že $(q, w) \vdash (q', w')$ právě 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: $(1, \texttt{ababb})$ je počáteční konfigurace.

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

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

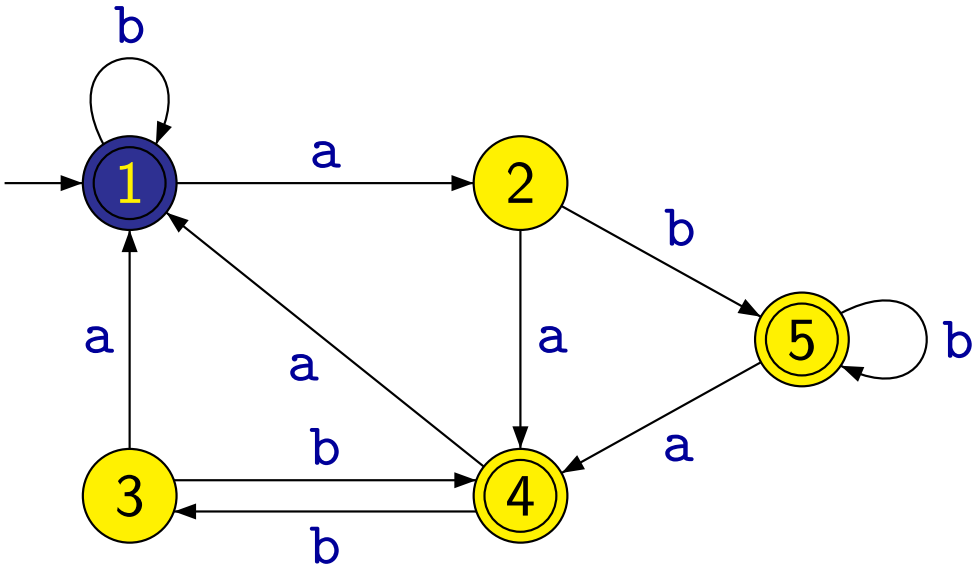
Definice

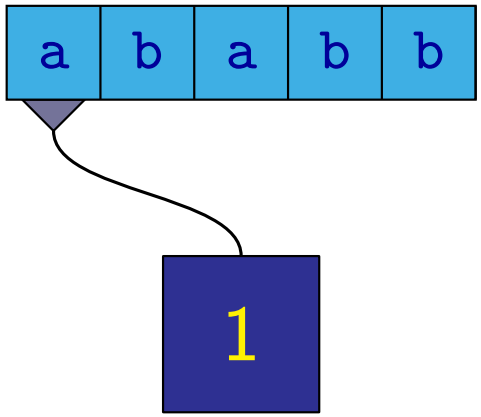
Výpočet automatu je posloupnost konfigurací

$$C_0, C_1, C_2, \cdots, C_k$$

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

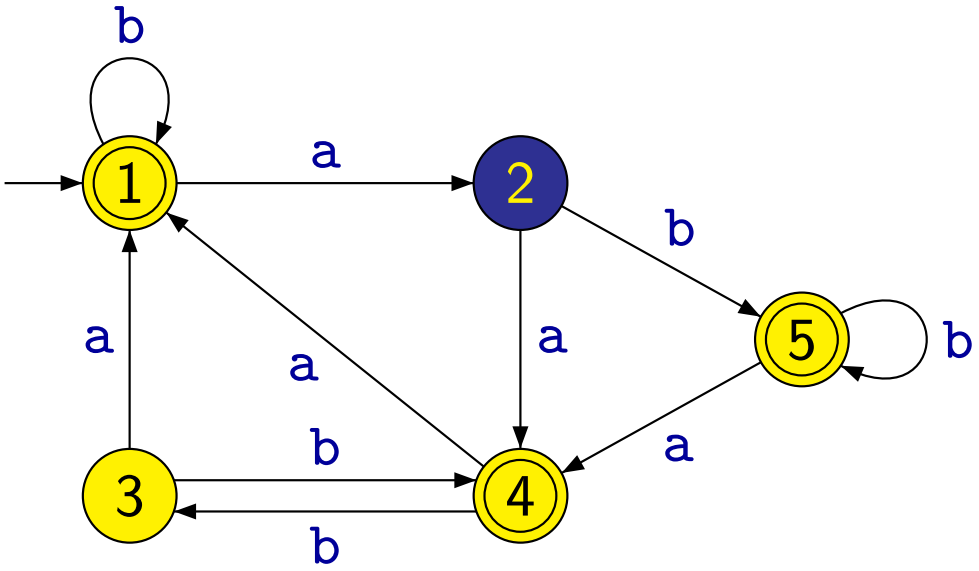
Deterministický konečný automat

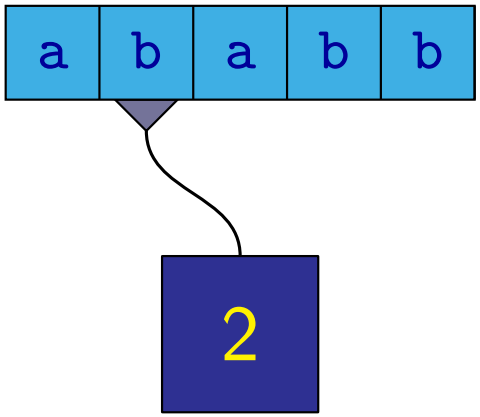




$(1, \mathtt{ababb})$

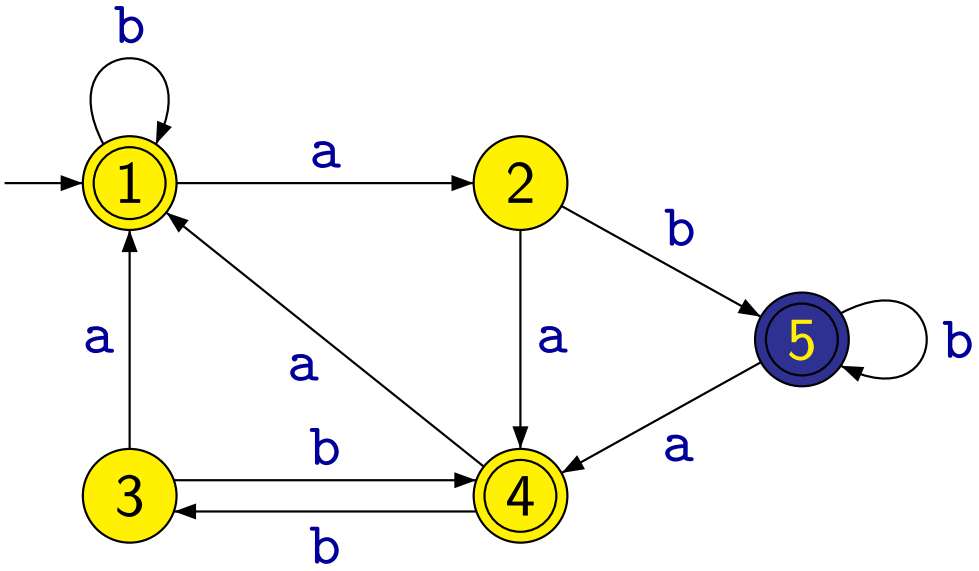
Deterministický konečný automat

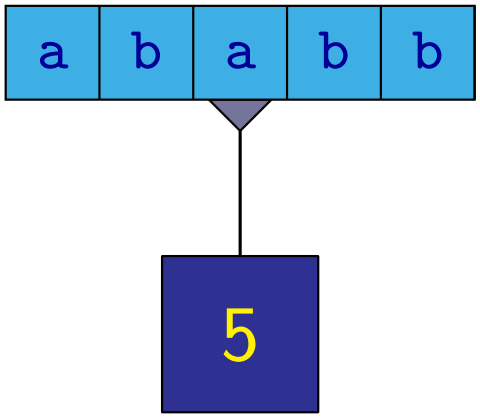




$(1, \mathtt{ababb}) \vdash (2, \mathtt{babb})$

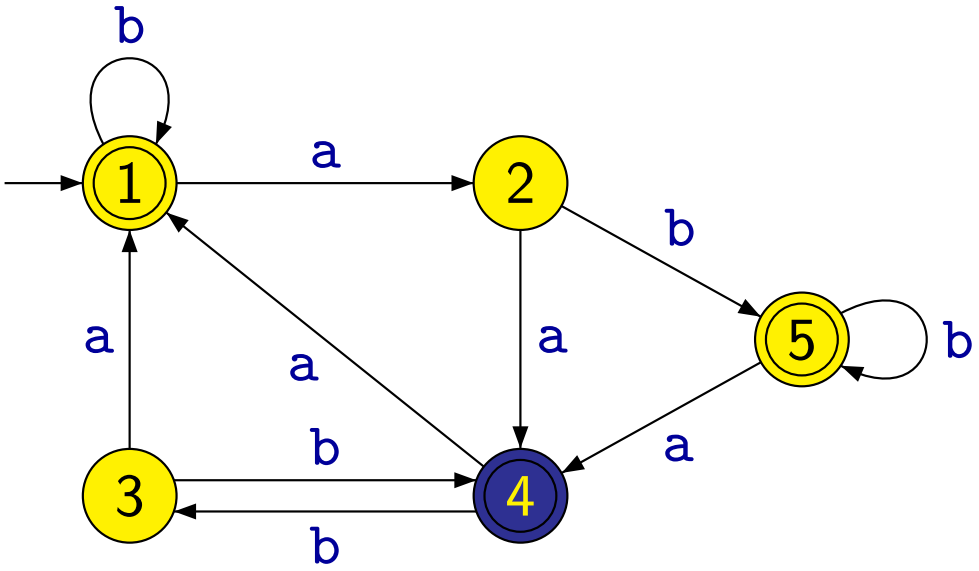
Deterministický konečný automat

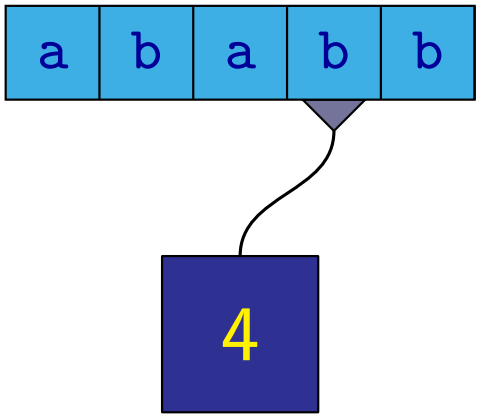




$(1, \mathtt{ababb}) \vdash (2, \mathtt{babb}) \vdash (5, \mathtt{abb})$

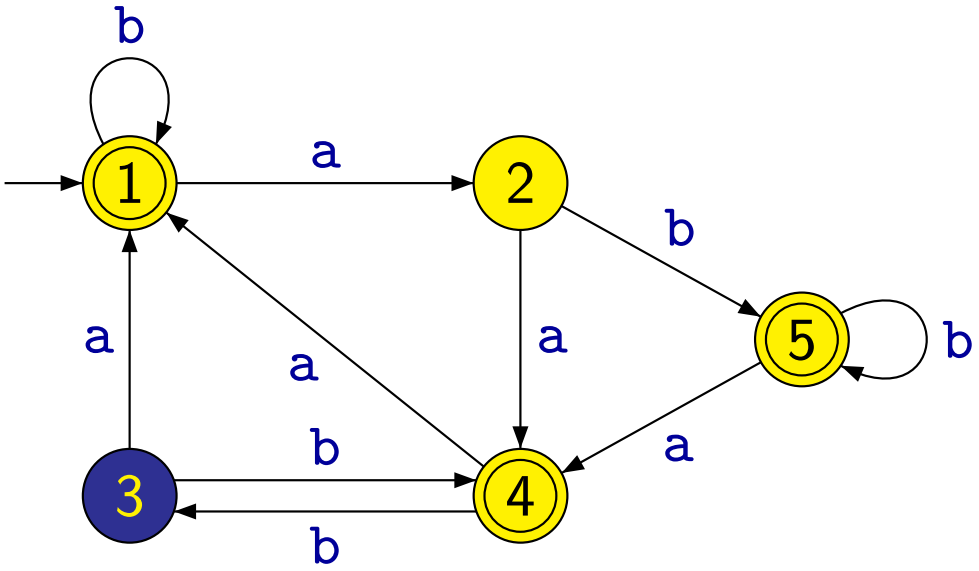
Deterministický konečný automat

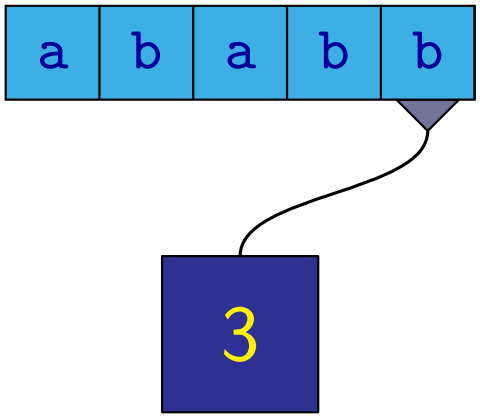




$(1, \mathtt{ababb}) \vdash (2, \mathtt{babb}) \vdash$
$(5, \mathtt{abb}) \vdash (4, \mathtt{bb})$

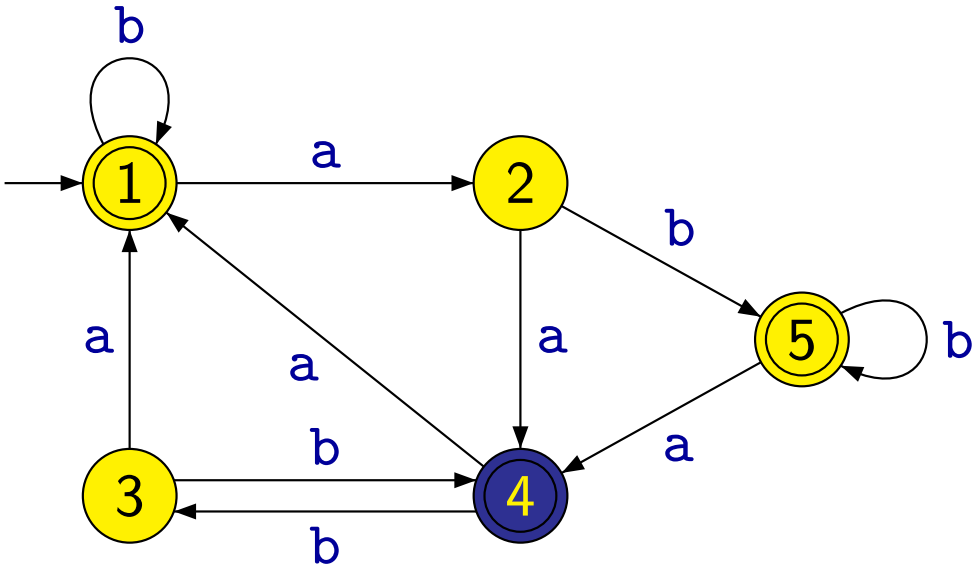
Deterministický konečný automat

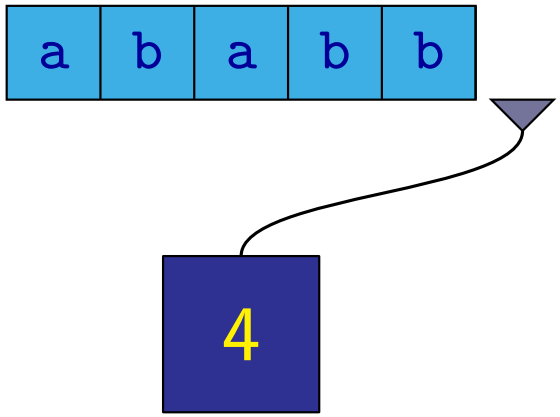




$(1, \mathtt{ababb}) \vdash (2, \mathtt{babb}) \vdash$
$(5, \mathtt{abb}) \vdash (4, \mathtt{bb}) \vdash$
$(3, \mathtt{b})$

Deterministický konečný automat





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

Deterministický konečný automat

Dále můžeme definovat relaci $\vdash^*$, jejíž význam je takový, 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

$$C_0, C_1, C_2, \ldots, C_k$$

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

Deterministický konečný automat

Definice

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

Definice

Automat **přijímá** slovo $w \in \Sigma^*$ právě tehdy, jestliže výpočet začínající v počáteční konfiguraci (q_0, w) skončí v přijímající koncové konfiguraci.

Poznámka: Formálně to můžeme definovat tak, že automat přijímá slovo $w \in \Sigma^*$ právě když $(q_0, w) \vdash^* (q, \varepsilon)$ pro nějaké $q \in F$.

Deterministický konečný automat

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 nazýváme **regulární** právě tehdy, když existuje konečný automat, který jej přijímá.