Úvod do teoretické informatiky

Martin Kot Zdeněk Sawa

Katedra informatiky, FEI,
Vysoká škola báňská – Technická universita Ostrava
17. listopadu 15, Ostrava-Poruba 708 33
Česká republika

11. února 2009

Přednášející

Garant předmětu:

Jméno: Ing. Zdeněk Sawa, Ph.D.

E-mail: zdenek.sawa@vsb.cz

Místnost: A1024

Další přednášející:

- Ing. Martin Kot
- doc. RNDr. Marie Duží, CSc.
- Mgr. Marek Menšík

WWW stránky

Webové stránky k předmětu naleznete na adrese:

http://www.cs.vsb.cz/sawa/uti

Na těchto stránkách najdete:

- Informace o předmětu
- Učební texty
- Slidy z přednášek
- Zadání příkladů na cvičení
- Zadání referátů
- Zadání prémiových příkladů
- Aktuální informace
- Animace

Požadavky

Povinné:

- **Zápočet** (35 bodů):
  - Dvě zápočtové písemky (po 10 bodech)
    - Budou se psát na přednáškách (pravděpodobně na 5. a 9. přednášce).
    - Doba trvání asi 25 minut.
  - Referát (15 bodů)

  Minimum pro získání zápočtu je 20 bodů.
- **Zkouška** (65 bodů)
  - Písemná zkouška skládající se ze tří částí, za každou je nutné získat nejméně 5 bodů.

Nepovinné (bonusové body):

- Vypracování prémiového příkladu (15 bodů)

Uznané zápočty

- Studenti, kteří předmět opakují a v loňském roce získali zápočet, musí e-mailem kontaktovat garanta předmětu (Z. Sawu) **nejpozději před konáním první zápočtové písemky** a oznámit, zda chtějí zápočet z loňska uznat nebo ne.
- Pokud budou mít uznaný zápočet z loňského roku, nebudou psát písemky ani vypracovávat referát.
- Kdo zápočet loni získal, ale nechce ho uznat (nebo včas nepožádá o uznání), musí absolvovat vše znovu (obě písemky i referát).
- Rovněž studenti, kteří absolvovali předmět už loni, ale zápočet nezískali, musí absolvovat vše znovu.

Náplň předmětu

Cílem tohoto předmětu je poskytnout studentům stručný úvod do následujících oblastí:

- **Formální jazyky a automaty**
- **Vyčíslitelnost a složitost**
- **Logika**

Výukové materiály

Hlavními výukovými texty jsou:

- prof. RNDr. Petr Jančar, CSc.
  Úvod do teoretické informatiky (učební text),
  VŠB-TU Ostrava, 2007.

  Poznámka: Pro zájemce s hlubším zájmem o problematiku je k dispozici i rozšířená verze tohoto textu určená pro studenty magisterského studia (pro předmět Teoretická informatika).

- doc. RNDr. Marie Duží, CSc.
  Matematická logika (učební text),
  VŠB-TU Ostrava, 2003.

Výukové materiály

Kromě výukových textů jsou k dispozici:

- **Slidy** z přednášek (na web budou doplňovány aktuální verze)
- **Animace** vytvořené M. Kotem, Z. Sawou a některými studenty v rámci diplomových prací
- **Zadání příkladů do cvičení**

Další literatura (pro zájemce)

- M. Sipser: *Introduction to the Theory of Computation*, PWS Publishing Company, 1997.
- D. Kozen: *Automata and Computability*, Undergraduate Text in Computer Science, Springer Verlag, 1997.
- Ch. Papadimitriou: *Computational Complexity*, Addison-Wesley, 1993.
- J. E. Hopcroft, R. Motwani, J. D. Ullman: *Introduction to Automata Theory, Languages, and Computation* (3rd Edition), Addison Wesley, 2006.
- J. Gruska: *Foundations of Computing*, International Thomson Computer Press, 1997.
- M. Huth, M. Ryan.: *Logic in Computer Science: Modelling and Reasoning about Systems*, Cambridge University Press, 2004.
- V. Švejdar: *Logika - neúplnost, složitost a nutnost*, Academia, 2002.

Další literatura (pro zájemce)

- Ding-Zhu Du, Ker-I Ko: *Problem Solving in Automata, Languages, and Complexity*, Wiley, 2001. Pozn.: v rámci sítě VŠB je tato publikace dostupná v elektronické podobě (jako PDF) na adrese `http://knihovna.vsb.cz/sluzby/e-knihy-wiley.htm`

Referáty

- Zadání budou zveřejněna v brzké době na webu k UTI.
- Čísla zadání přidělí studentům cvičící na cvičeních.
- U zadání bude uvedeno, do kdy má být referát odevzdán.
- Referát musí mít formu samostatně čitelného textu a musí být solidně zpracován po formální i jazykové stránce.
- Referát musí student odevzdat cvičícímu e-mailem jako soubor ve formátu PDF.
- Student je povinnen se s cvičícím domluvit na termínu, kdy mu referát odprezentuje. Odprezentování referátu je nutnou podmínkou pro získání bodů za referát.

Prémiové příklady

- Prémiové příklady budou postupně zveřejňovány v průběhu semestru na webu k předmětu.
- První dva studenti v ročníku, kteří daný příklad vyřeší, za něj mohou dastat až 15 bodů.
- Řešení posílejte e-mailem garantovi předmětu, ne cvičícím:
  - Zdeněk Sawa (zdenek.sawa@vsb.cz)
- Pro získání bodů je nutné poté referát garantovi předmětu odprezentovat (podobně jako běžné referáty).

Základní pojmy

Množiny

Množina – kolekce vzájemně odlišitelných objektů, které nazýváme jejími **prvky**.

- $x \in S$ – objekt x je prvkem množiny S
- $x \notin S$ – objekt x není prvkem množiny S

Jednou z možností, jak popsat množinu, je explicitně vyjmenovat všechny její prvky, např.:

$$S = \{1, 2, 3\}$$

Množina nemůže obsahovat prvek více než jednou a prvky množiny nejsou nijak seřazeny.

Množiny

Množiny A a B jsou si **rovny** ($A = B$), jestliže obsahují tytéž prvky.

Například

$$\{1, 2, 3\} = \{2, 1, 3\} = \{3, 2, 1\}$$

Množina neobsahující žádné prvky se nazývá **prázdná množina** a označuje se symbolem $\emptyset$.

Poznámka: Kromě množin se také někdy používají **multimnožiny**. Na rozdíl od množiny může multimnožina obsahovat více výskytů jednoho prvku.

$$M = \{1, 1, 1, 2, 3, 3, 3, 3, 3\}$$

Množiny

Příklady některých důležitých množin:

- $\mathbb{N}$ – množina všech **přirozených** čísel, tj. $\mathbb{N} = \{0, 1, 2, \ldots\}$,
- $\mathbb{N}_+$ – množina všech **kladných přirozených** čísel, tj. $\mathbb{N}_+ = \{1, 2, 3, \ldots\}$,
- $\mathbb{Z}$ – množina všech **celých** čísel, tj. $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$,
- $\mathbb{Q}$ – množina všech **racionálních** čísel (zlomky),
- $\mathbb{R}$ – množina všech **reálných** čísel.

Množiny

- $A \subseteq B$ – označuje, že A je **podmnožinou** B, tj.

$$\forall x(x \in A \Rightarrow x \in B)$$

(každý prvek množiny A patří rovněž do množiny B).

- $A \subset B$ – označuje, že A je **vlastní podmnožinou** B, tj.

$$A \subseteq B \ \wedge \ \exists x(x \in B \wedge x \notin A)$$

(tj. $A \subseteq B$, ale $A \neq B$).

Poznámka: Někdy se též používá zápis $A \subset B$ pro označení, že A je podmnožinou B (tj. připouští i možnost $A = B$), a zápis $A \subsetneq B$ pro označení, že A je vlastní podmnožinou B.

Množiny

Pro danou množinu A můžeme definovat množinu $B \subseteq A$ tvořenou těmi prvky množiny A, které mají určitou vlastnost (splňují nějakou podmínku) $\varphi(x)$.

$$B = \{x \in A \mid \varphi(x)\}$$

Příklad: Podmnožina X množiny přirozených čísel $\mathbb{N}$ tvořená těmi čísly, která dávají po dělení pěti zbytek dvě.

$$X = \{x \in \mathbb{N} \mid x \bmod 5 = 2\}$$

Množiny

Množinové operace:

- **Průnik** množin A a B je množina

$$A \cap B = \{x \mid x \in A \wedge x \in B\}$$

- **Sjednocení** množin A a B je množina

$$A \cup B = \{x \mid x \in A \vee x \in B\}$$

- **Rozdíl** množin A a B je množina

$$A - B = \{x \mid x \in A \wedge x \notin B\}$$

Poznámka: Pro rozdíl množin se též používá zápis $A \setminus B$.

Příklad: Jestliže $A = \{a, b, c, d\}$ a $B = \{b, c, e, f\}$, pak

$$A \cap B = \{b, c\}, \qquad A \cup B = \{a, b, c, d, e, f\}, \qquad A - B = \{a, d\}.$$

Množiny

Někdy jsou všechny množiny, které uvažujeme, podmnožinami nějaké jedné množiny U nazývané **universum**.

Příklad: Pokud se například bavíme o množinách přirozených čísel, pak je universem množina $\mathbb{N}$.

Pro dané universum U definujeme **doplněk** množiny A jako

$$\overline{A} = U - A$$

Pro libovolné množiny $A, B \subseteq U$ platí de Morganova pravidla:

$$\overline{A \cap B} = \overline{A} \cup \overline{B} \qquad \overline{A \cup B} = \overline{A} \cap \overline{B}$$

Množiny A a B jsou **disjunktní**, jestliže nemají žádný společný prvek, tj. jestliže $A \cap B = \emptyset$.

Množiny

Velikost dané množiny S se nazývá její **kardinalita** a označuje se $|S|$.

- V případě **konečné** množiny, je její kardinalita přirozené číslo odpovídající počtu jejích prvků, např. $|\emptyset| = 0$.
- Dvě (obecné) množiny mají stejnou kardinalitu, jesliže existuje bijekce (tj. vzájemně jednoznačné zobrazení) mezi jejich prvky.
- Množina S se nazývá **spočetná**, jestliže existuje bijekce mezi S a $\mathbb{N}$. Spočetné množiny jsou „nejmenší“ mezi nekonečnými množinami.
- Nekonečná množina, která není spočetná, se nazývá **nespočetná**.

Příklad: Množiny $\mathbb{N}$, $\mathbb{Z}$ a $\mathbb{Q}$ jsou spočetné, množina $\mathbb{R}$ je nespočetná.

Množiny

Množina všech podmnožin množiny S se nazývá **potenční množina** množiny S a označuje se zápisem $\mathcal{P}(S)$.

Příklad: Pokud $S = \{a, b, c\}$, pak

$$\mathcal{P}(S) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}.$$

Pokud je množina S konečná, pak $|\mathcal{P}(S)| = 2^{|S|}$.

Poznámka: Často se také používá pro označení potenční množiny místo $\mathcal{P}(S)$ výraz 2^S.

Množiny

Uspořádaná dvojice prvků a a b se označuje (a, b).

Na rozdíl od množiny u uspořádané dvojice záleží na pořadí prvků, (a, b) je něco jiného než (b, a).

Poznámka: Formálně je možno uspořádanou dvojici (a, b) pomocí množin např. takto:

$$(a, b) = \{a, \{a, b\}\}.$$

Analogicky můžeme definovat uspořádané trojice, čtveřice atd.

Kartézský součin množin A a B, označovaný $A \times B$, je množina všech uspořádaných dvojic, kde první prvek z dvojice patří do množiny A a druhý do množiny B:

$$A \times B = \{(a, b) \mid a \in A,\ b \in B\}$$

Příklad: $\{a, b\} \times \{a, b, c\} = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c)\}$

Množiny

Jestliže A a B jsou konečné množiny, pak $|A \times B| = |A| \cdot |B|$.

Kartézský součin n množin $A_1, A_2, \ldots, A_n$ je množina ***n*-tic**

$$A_1 \times A_2 \times \cdots \times A_n = \{(a_1, a_2, \ldots, a_n) \mid a_i \in A_i,\ i = 1, 2, \ldots, n\}$$

Jestliže všechna A_i jsou konečné množiny, platí

$$|A_1 \times A_2 \times \cdots \times A_n| = |A_1| \cdot |A_2| \cdots |A_n|$$

Místo kartézského součinu $A \times A \times \cdots \times A$, kde se množina A vyskytuje n krát, píšeme A^n.

Pro konečnou množinu A platí $|A^n| = |A|^n$.

Relace

Relace na množinách $A_1, A_2, \ldots, A_n$ je libovolná podmnožina kartézského součinu $A_1 \times A_2 \times \cdots \times A_n$.

Relace na n množinách se nazývá **n-ární** relace.

Jestliže $n = 2$, jedná se o **binární** relaci.
Jestliže $n = 3$, jedná se o **ternární** relaci.

V případě, že $A_1 = A_2 = \cdots = A_n$, hovoříme o **homogenní** relaci, v opačném případě o relaci **heterogenní**.

Když říkáme, že R je n-ární relace na množině A, máme tím na mysli, že $R \subseteq A^n$.

Relace

Příklad: Relace „menší než“ na množině přirozených čísel je množina

$$\{(a, b) \in \mathbb{N} \times \mathbb{N} \mid a < b\}$$

Poznámka: Jestliže $R \subseteq A \times B$ je binární relace, někdy místo $(a, b) \in R$ používáme infixový zápis a píšeme $a\ R\ b$.

Relace

Binární relace $R \subseteq A \times A$ je:

- **reflexivní**, jestliže pro všechna $a \in A$ platí $(a, a) \in R$,
- **ireflexivní**, jestliže pro všechna $a \in A$ platí $(a, a) \notin R$,
- **symetrická**, jestliže pro všechna $a, b \in A$ platí, že pokud $(a, b) \in R$, pak $(b, a) \in R$,
- **asymetrická**, jestliže pro všechna $a, b \in A$ platí, že pokud $(a, b) \in R$, pak $(b, a) \notin R$,
- **antisymetrická**, jestliže pro všechna $a, b \in A$ platí, že pokud $(a, b) \in R$ a $(b, a) \in R$, pak $a = b$,
- **tranzitivní**, jestliže pro všechna $a, b, c \in A$ platí, že pokud $(a, b) \in R$ a $(b, c) \in R$, pak $(a, c) \in R$.

Relace

Příklad:

- Relace "$=$" na $\mathbb{N}$ je reflexivní, symetrická, antisymetrická a tranzitivní, ale není ireflexivní ani asymetrická.
- Relace "$\leq$" na $\mathbb{N}$ je reflexivní, antisymetrická a tranzitivní, ale není ireflexivní, symetrická ani asymetrická.
- Relace "$<$" na $\mathbb{N}$ je ireflexivní, asymetrická, antisymetrická a tranzitivní, ale není reflexivní ani symetrická.

Relace

- **Reflexivní uzávěr** relace $R \subseteq A \times A$ je nejmenší reflexivní relace $R' \subseteq A \times A$ taková, že $R \subseteq R'$.

  Poznámka: Pojmem „nejmenší“ zde máme na mysli to, že neexistuje žádná reflexivní relace R'' taková, že $R \subseteq R'' \subset R'$.

- **Symetrický uzávěr** relace $R \subseteq A \times A$ je nejmenší symetrická relace $R' \subseteq A \times A$ taková, že $R \subseteq R'$.

- **Tranzitivní uzávěr** relace $R \subseteq A \times A$ je nejmenší tranzitivní relace $R' \subseteq A \times A$ taková, že $R \subseteq R'$.

- **Reflexivní a tranzitivní uzávěr** relace $R \subseteq A \times A$ je nejmenší relace $R' \subseteq A \times A$ taková, že $R \subseteq R'$ a R' je současně reflexivní i tranzitivní.

Ekvivalence

Binární relace R na množině A je **ekvivalence** právě tehdy, když je reflexivní, symetrická a tranzitivní.

Příklad: Následující relace $\equiv_5$ je ekvivalence

$$\equiv_5 = \{(a, b) \in \mathbb{Z}^2 \mid (a \bmod 5) = (b \bmod 5)\}$$

obecně pro libovolné $n > 0$ je relace $\equiv_n$ ekvivalence

$$\equiv_n = \{(a, b) \in \mathbb{Z}^2 \mid (a \bmod n) = (b \bmod n)\}$$

Ekvivalence

Jestliže R je ekvivalence na množině A, pak **třídou ekvivalence** prvku $a \in A$ je množina $[a]_R = \{b \in A \mid (a, b) \in R\}$, tj. množina všech prvků s ním ekvivalentních.

Mějme množinu A. Množina jejích podmnožin $\mathcal{A} = \{A_i \mid i \in I\}$ (pro nějakou indexovou množinu I) tvoří **rozklad** na množině A, jestliže:

- všechny množiny A_i jsou vzájemně disjunktní, tj. jestliže pro libovolné $A_i, A_j \in \mathcal{A}$ platí $A_i \cap A_j = \emptyset$ pokud $i \neq j$, a
- sjednocení množin z $\mathcal{A}$ je množina A, tj.

$$A = \bigcup_{A_i \in \mathcal{A}} A_i$$

Ekvivalence

Ekvivalence $R \subseteq A \times A$ definuje na A rozklad $\{\, [a]_R \mid a \in A \,\}$.

Naopak rozklad $\mathcal{A} = \{A_i \mid i \in I\}$ na množině A definuje ekvivalenci

$$R = \{(a, b) \subseteq A \times A \mid a, b \in A_i \text{ pro nějaké } A_i \in \mathcal{A}\}\,.$$

Příklad: Ekvivalence $\equiv_5$ definuje rozklad $\mathcal{A} = \{A_0, A_1, A_2, A_3, A_4\}$ na $\mathbb{N}$, kde

- $A_0 = \{\ldots, -15, -10, -5, 0, 5, 10, 15, \ldots\}$
- $A_1 = \{\ldots, -14, -9, -4, 1, 6, 11, 16, \ldots\}$
- $A_2 = \{\ldots, -13, -8, -3, 2, 7, 12, 17, \ldots\}$
- $A_3 = \{\ldots, -12, -7, -2, 3, 8, 13, 18, \ldots\}$
- $A_4 = \{\ldots, -11, -6, -1, 4, 9, 14, 19, \ldots\}$

Uspořádání

Binární relace R na množině A je **(částečné a neostré) uspořádání**, jestliže je reflexivní, tranzitivní a antisymetrická.

Binární relace R na množině A je **(částečné) ostré uspořádání**, jestliže je asymetrická a tranzitivní.
(Pozn.: Z toho, že je R asymetrická plyne, že je také ireflexivní a antisymetrická.)

Pro neostrá uspořádání se obvykle používají symboly jako $\leq$ a jemu podobné, pro ostrá uspořádání pak symboly jako $<$ a jemu podobné.

Uspořádání (ať už neostré či ostré) $R \subseteq A \times A$ je **úplné** (nebo také **lineární**), jestliže pro všechna $a, b \in A$ platí buď $(a, b) \in R$, $(b, a) \in R$ nebo $a = b$ (tj. pokud neexistují vzájemně nesrovnatelné prvky).

Uspořádání

Příklady:

- Relace "$\leq$" je úplné (neostré) uspořádání na množině $\mathbb{N}$ ($\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$).
- Relace "$<$" je ostré úplné uspořádání na množině $\mathbb{N}$ ($\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$).
- Relace "$\subseteq$" je částečné (neostré) uspořádání na množině $\mathcal{P}(X)$ (pro libovolnou množinu X).
- Relace "je dělitelem" je částečné (neostré) uspořádání na množině $\mathbb{N}_+$.
- Relace "$=$" je částečné (neostré) uspořádání na množině $\mathbb{N}$.

Ke každému neostrému uspořádání R na množině A existuje odpovídající ostré uspořádání $R' = R - \{(a, a) \mid a \in A\}$.

Naopak ke každému ostrému uspořádání S na množině A existuje odpovídající neostré uspořádání $S' = S \cup \{(a, a) \mid a \in A\}$

Uspořádání

Mějme libovolné neostré uspořádání $\leq$ na množině A.

- Prvek $a \in A$ je **minimální prvek** množiny A, jestliže v A neexistuje menší prvek než a (tj. z $x \leq a$ plyne $x = a$).
- Prvek $a \in A$ je **maximální prvek** množiny A, jestliže v A neexistuje větší prvek než a (tj. z $a \leq x$ plyne $a = x$).
- Prvek $a \in A$ je **nejmenší prvek** množiny A, jestliže je menší než všechny ostatní prvky v A (tj. pro každé $x \in A$ platí $a \leq x$).
- Prvek $a \in A$ je **největší prvek** množiny A, jestliže je větší než všechny ostatní prvky v A (tj. pro každé $x \in A$ platí $x \leq a$).

Uspořádání

- Prvek $a \in A$ je **infimum** množiny B (píšeme $a = \inf B$), jestliže a je největší ze všech prvků, které jsou menší než všechny prvky z B, tj. platí

$$(\forall x \in B)(a \leq x) \ \wedge \ (\forall b)((\forall x \in B)(b \leq x) \Rightarrow b \leq a)$$

- Prvek $a \in A$ je **supremum** množiny B (píšeme $a = \sup B$), jestliže a je nejmenší ze všech prvků, které jsou větší než všechny prvky z B, tj. platí

$$(\forall x \in B)(x \leq a) \ \wedge \ (\forall b)((\forall x \in B)(x \leq b) \Rightarrow a \leq b)$$

Funkce

Funkce f z množiny A do množiny B je binární relace $f \subseteq A \times B$ taková, že pro každé $a \in A$ existuje právě jedno $b \in B$ takové, že $(a, b) \in f$.

Množina A se nazývá **definiční obor** funkce f,

množina B se nazývá **obor hodnot** funkce f.

To, že f je funkce z množiny A do množiny B obvykle zapisujeme jako

$$f : A \to B$$

Místo $(a, b) \in f$ obvykle píšeme $b = f(a)$, neboť volbou prvku a je prvek b jednoznačně určen.

Funkce $f : A \to B$ tedy každému prvku z A přiřazuje právě jeden prvek z B.

Jestliže $b = f(a)$, říkáme, že a je **argumentem** funkce f a že b je **hodnotou** funkce f v bodě a.

Funkce

Výše uvedená definice se týká tzv. **totální** funkce, tj. funkce, jejíž hodnota je definovaná pro každou hodnotu argumentu.

Někdy má smysl uvažovat také tzv. **částečné (parciální) funkce**, tj. funkce, jejichž hodnota není pro některé hodnoty argumentu definována.

Formálně je částečná funkce $f : A \to B$ definována jako relace $f \subseteq A \times B$ taková, že pro každé $a \in A$ existuje nejvýše jedno $b \in B$ takové, že $(a, b) \in f$.

Poznámka: Pokud budeme mluvit o funkci a neuvedeme jinak, budeme mít vždy na mysli funkci totální.

Funkce

Konečná posloupnost (sekvence) délky n je funkce, jejímž definičním oborem je množina $\{0, 1, \ldots, n-1\}$.

Konečnou posloupnost obvykle zapisujeme tak, že vypíšeme její hodnoty:

$$f(0), f(1), \ldots, f(n-1)$$

Nekonečná posloupnost (sekvence) je funkce, jejímž definičním oborem je $\mathbb{N}$.

Nekonečnou posloupnost někdy zapisujeme tak, že uvedeme několik prvních prvků, za kterými následují tři tečky:

$$f(0), f(1), f(2), \ldots$$

Funkce

Jestliže definičním oborem funkce f je kartézský součin, obvykle vynecháváme jeden pár závorek v zápise argumentu funkce f.

Pokud například máme funkci $f : A_1 \times A_2 \times \cdots \times A_n \to B$, pak místo $b = f((a_1, a_2, \ldots, a_n))$ píšeme $b = f(a_1, a_2, \ldots, a_n)$.

Místo o funkci někdy též mluvíme o **operaci**.

n-ární operace je funkce f typu

$$f : A_1 \times A_2 \times \cdots \times A_n \to B$$

V případě, že $n = 2$, mluvíme o **binární** operaci.

- $f : A^n \to A$ – n-ární operace na množině A,
- $f : A \to A$ – unární operace na množině A,
- $f : A \times A \to A$ – binární operace na množině A.

Funkce

Mějme funkci $f : A^n \to A$. Množina $B \subseteq A$ je **uzavřená na operaci** f, jestliže z $a_1, a_2, \ldots, a_n \in B$ plyne, že $f(a_1, a_2, \ldots, a_n) \in B$.

Jestliže $f : A \to B$ je funkce a $b = f(a)$, pak někdy také říkáme, že b je **obrazem** a. Obrazem množiny $A' \subseteq A$ je množina

$$f(A') = \{b \in B \mid b = f(a) \text{ pro nějaké } a \in A'\}.$$

Funkce $f : A \to B$ je:

- **surjektivní** (je surjekcí, je zobrazením na), jestliže $f(A) = B$,
- emphinjektivní (je injekcí, je prostá), jestliže z $a \neq a'$ plyne $f(a) \neq f(a')$,
- **bijektivní** (je bijekcí, je vzájemně jednoznačným zobrazením), jestliže je současně surjektivní i injektivní.

Jestliže funkce f je bijekcí, pak funkce **inverzní** k funkci f, označovaná f^{-1}, je definována takto: $f^{-1}(b) = a$ právě když $f(a) = b$.

Funkce

Předpokládejme nyní funkci $f : A \times A \to A$.

- Funkce f je **asociativní**, jestliže pro libovolné prvky $a, b, c \in A$ platí
$$f(f(a, b), c) = f(a, f(b, c)) .$$
- Funkce f je **komutativní**, jestliže pro libovolné prvky $a, b \in A$ platí
$$f(a, b) = f(b, a) .$$
- Prvek $z \in A$ je **nulovým prvkem** vzhledem k funkci f, jestliže pro libovolné $a \in A$ platí
$$f(z, a) = f(a, z) = z .$$
- Prvek $e \in A$ je **jednotkovým prvkem** vzhledem k funkci f, jestliže pro libovolné $a \in A$ platí
$$f(e, a) = f(a, e) = a .$$

Poznámka: Dá se ukázat, že ke každé funkci existuje nejvýše jeden nulový a nejvýše jeden jednotkový prvek.

Jestliže k funkci f existuje jednotkový prvek e, pak $b \in A$ je **inverzním prvkem** k prvku $a \in A$ právě tehdy, když

$$f(a, b) = f(b, a) = e$$

Funkce

Pro funkce typu $f : A \times A \to A$ je často vhodnější používat infixovou notaci a používat jako název funkce nějaký speciální symbol.

Mějme například funkci

$$\otimes : A \times A \to A$$

Pak místo $\otimes(a, b)$ píšeme $a \otimes b$.

- Asociativita $\otimes$ pak znamená, že pro libovolné $a, b, c \in A$ platí

$$(a \otimes b) \otimes c = a \otimes (b \otimes c)$$

- a komutativita $\otimes$ znamená, že pro libovolné $a, b \in A$ platí

$$a \otimes b = b \otimes a$$

Příklad: Místo $+(x, y)$ píšeme $x + y$.

Formální jazyky

Teorie formálních jazyků – motivace

Příklady 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.

Poznámka: V teorii formálních jazyků se sekvencím znaků říká **slova**. Při programování máme na mysli například řetězce (stringy) nebo třeba soubory na disku apod.

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 $|w| = 5$.

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:

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

Zjevně pro libovolná slova v a w platí:

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

Pro libovolné slovo w také platí:

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

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 množinu všech textů tvořených ASCII znaky. Jestliže:

- L_1 je množina všech textů, ve kterých se vyskytuje sekvence znaků `FOO`, a
- L_2 je množina všech textů, ve kterých se vyskytuje sekvence znaků `BAR`,

pak

- $L_1 \cup L_2$ jsou všechny texty, ve kterých se vyskytuje `FOO` nebo `BAR`,
- $L_1 \cap L_2$ jsou všechny texty, ve kterých se vyskytuje `FOO` i `BAR`,
- $\overline{L_1}$ jsou všechny texty, ve kterých se nevyskytuje `FOO`,
- $L_1 - L_2$ jsou všechny texty, ve kterých se vyskytuje `FOO`, ale nevyskytuje `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: $w = \texttt{AHOJ}$ $w^R = \texttt{JOHA}$

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

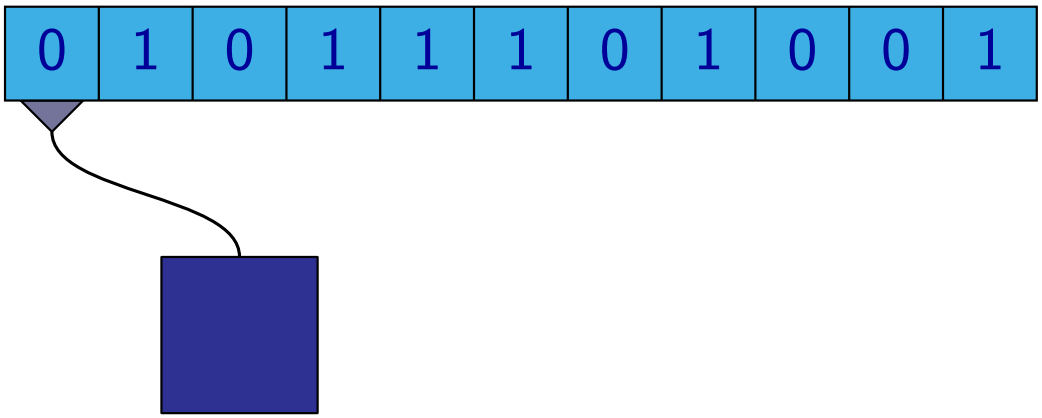
$L^R = \{ba, abaab, baaa\}$

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.

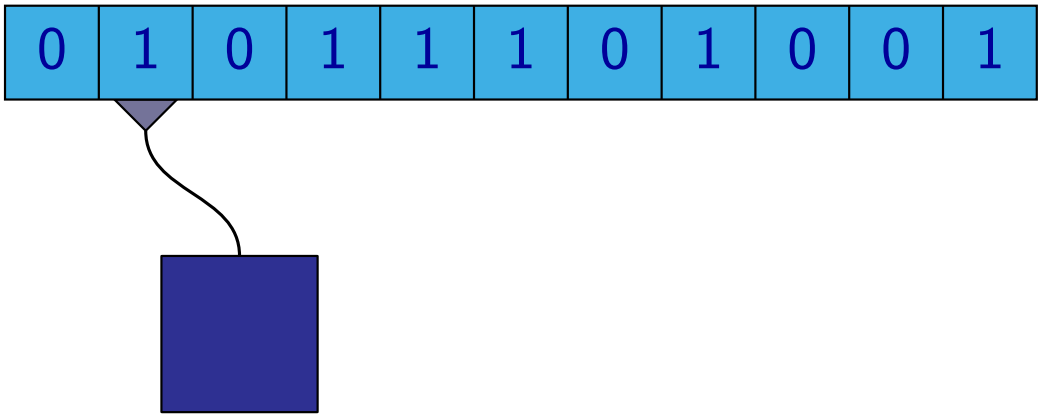


Rozpoznávání jazyka

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.

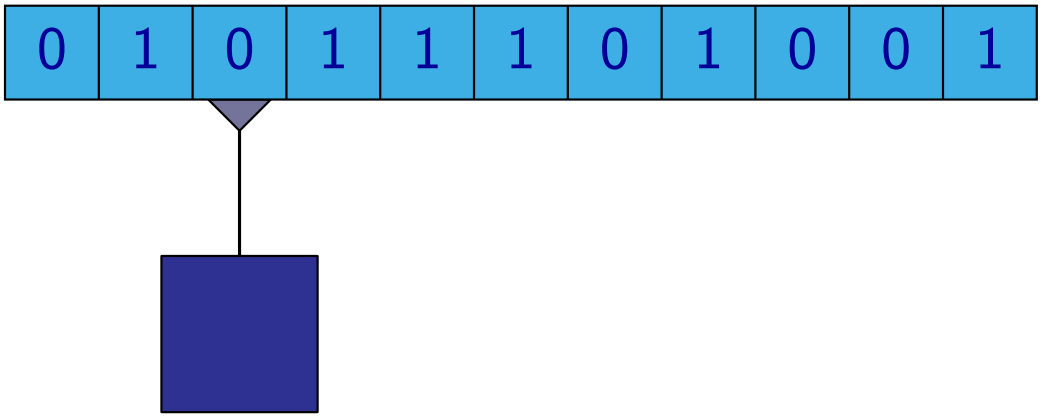


Rozpoznávání jazyka

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.

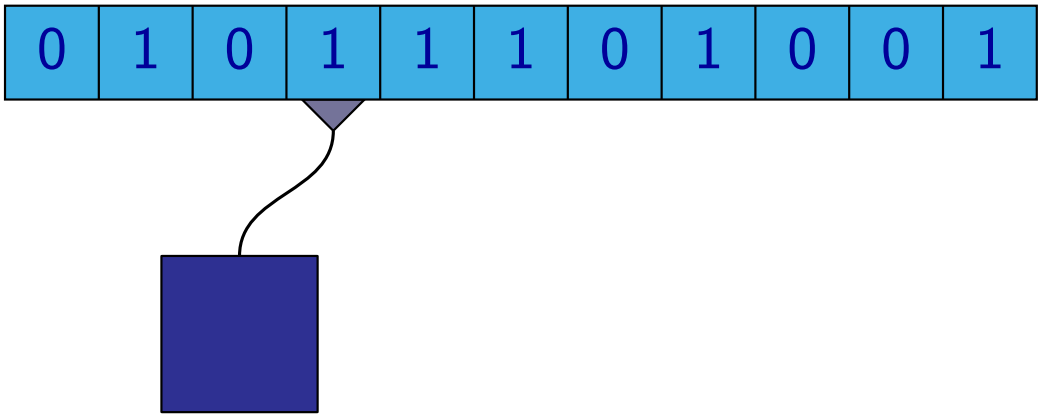


Rozpoznávání jazyka

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.

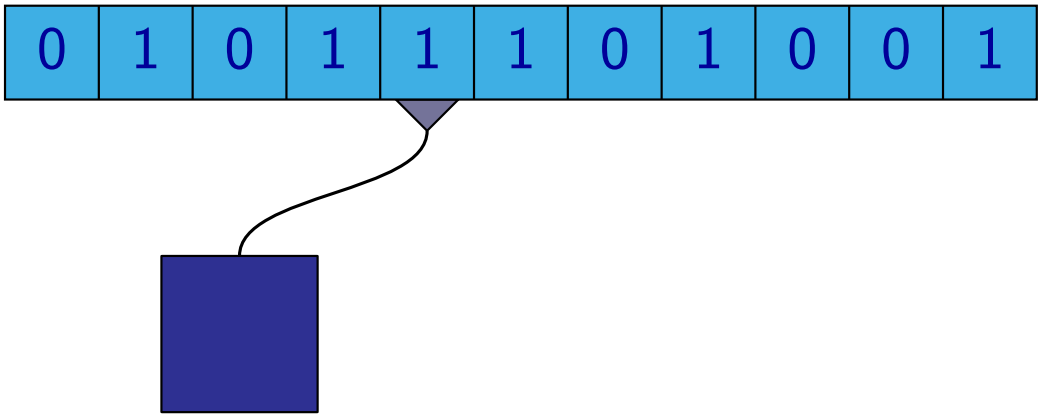


Rozpoznávání jazyka

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.

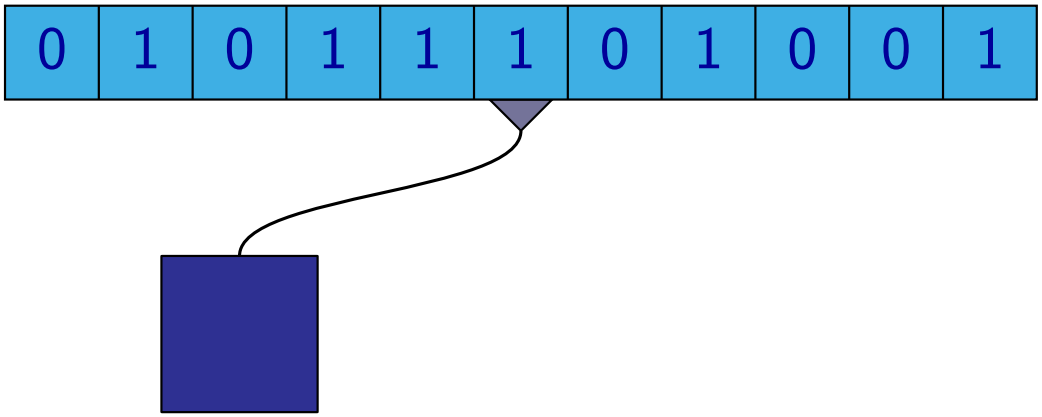


Rozpoznávání jazyka

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.

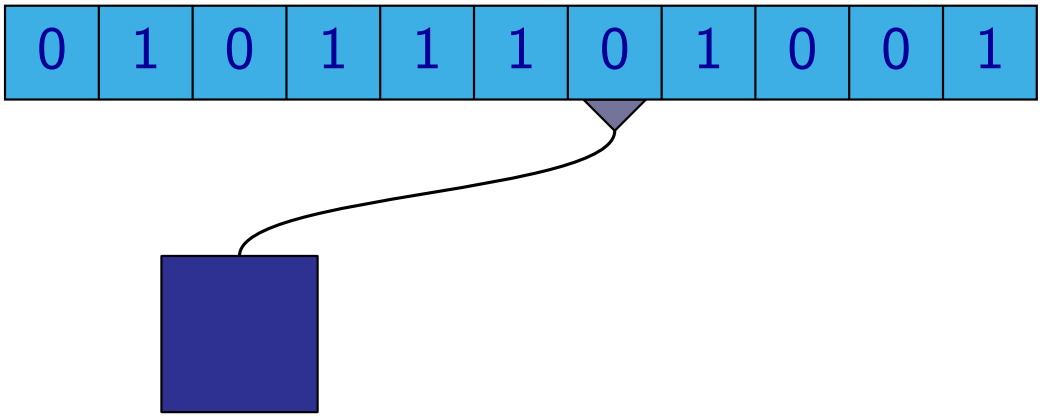


Rozpoznávání jazyka

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.

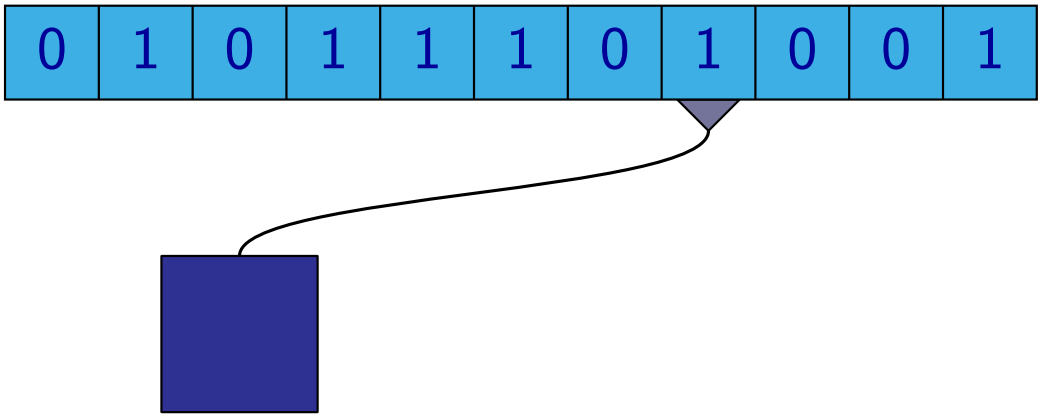


Rozpoznávání jazyka

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.

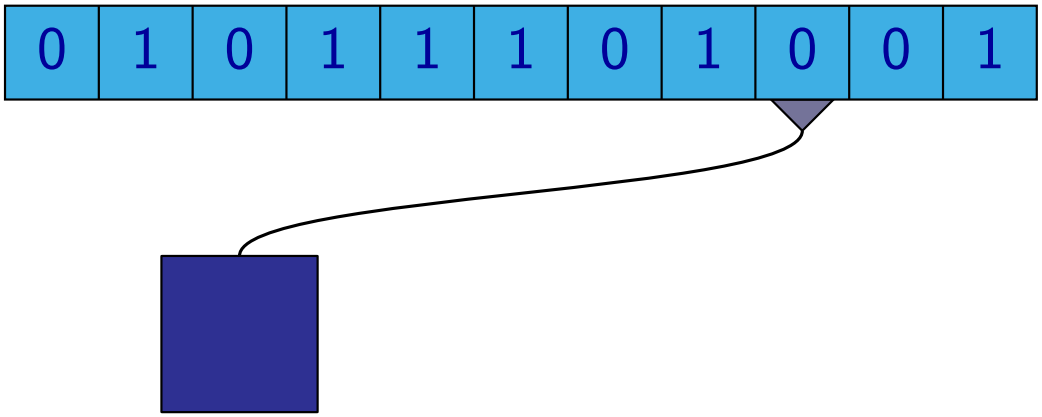


Rozpoznávání jazyka

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.

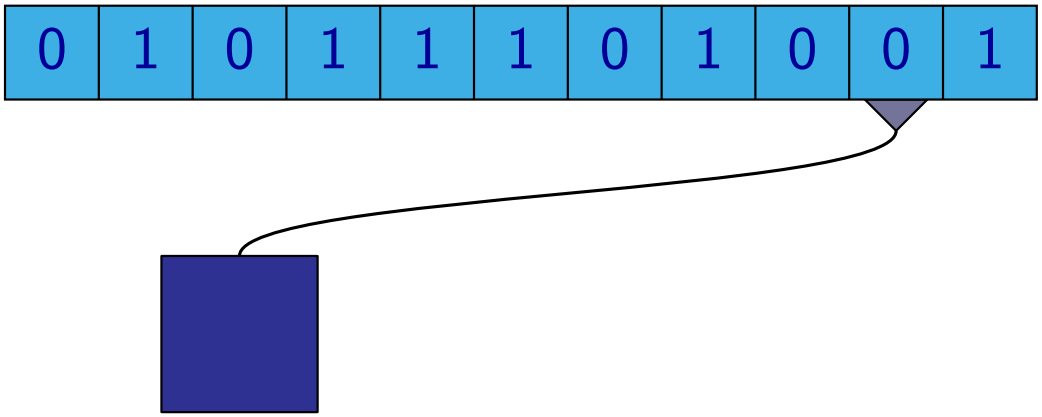


Rozpoznávání jazyka

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.

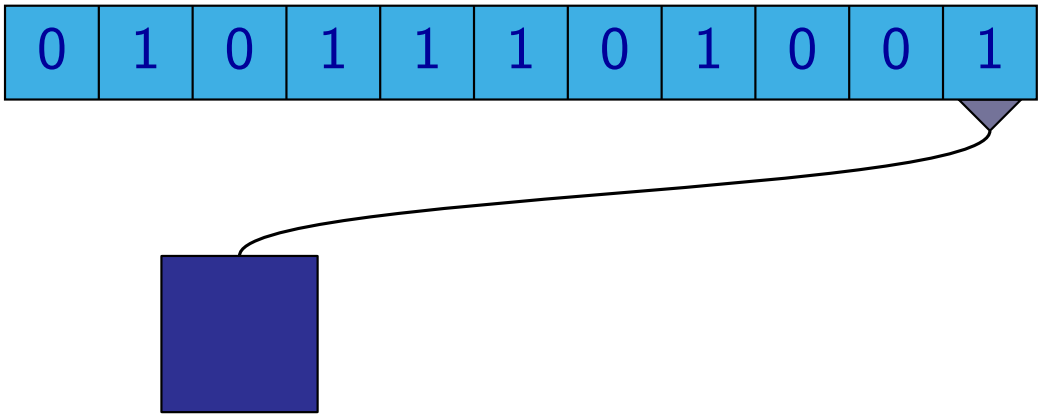


Rozpoznávání jazyka

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.

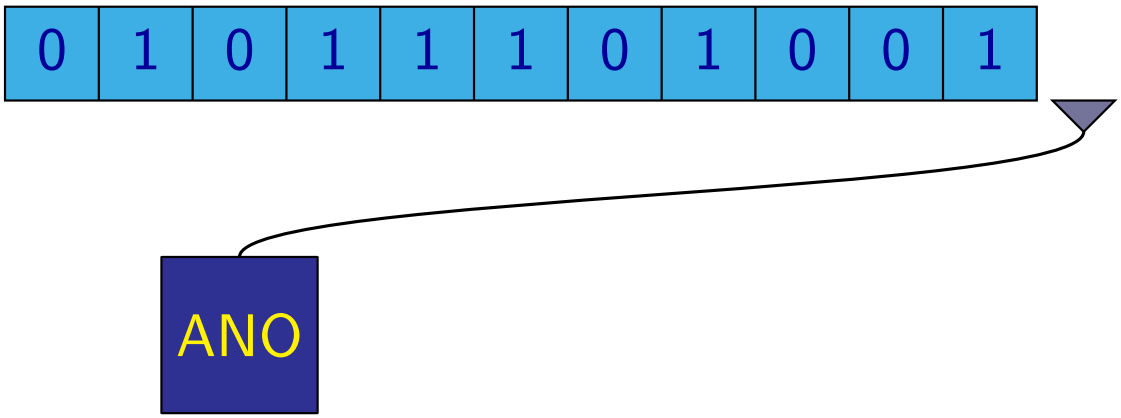


Rozpoznávání jazyka

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

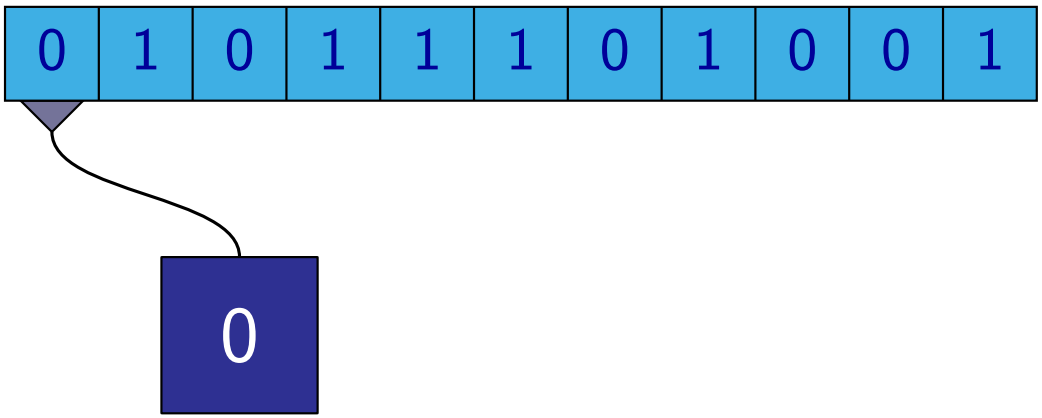
Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1.

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



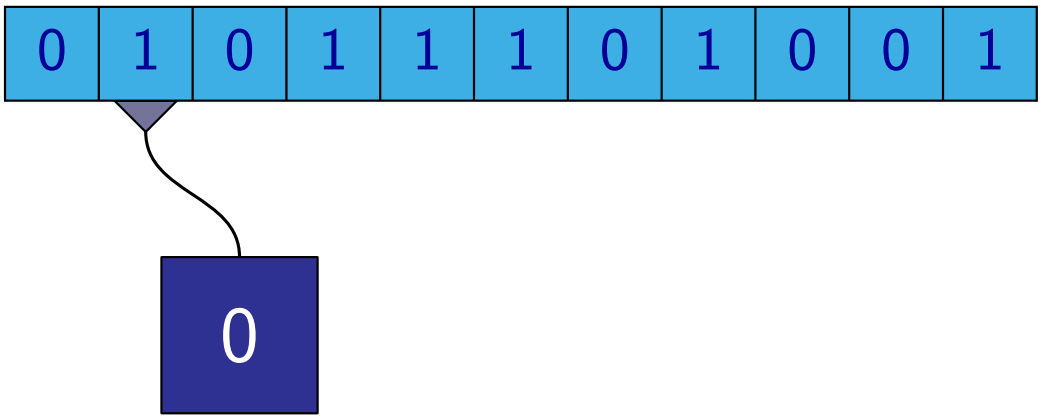
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



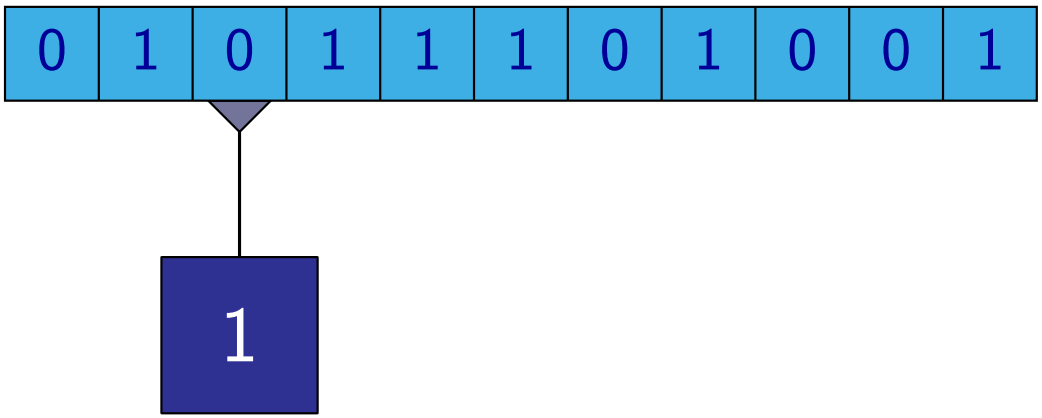
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



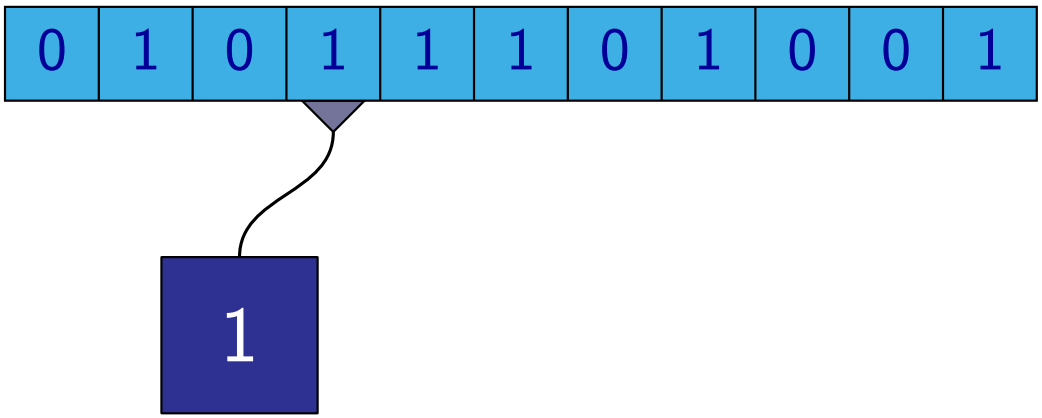
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



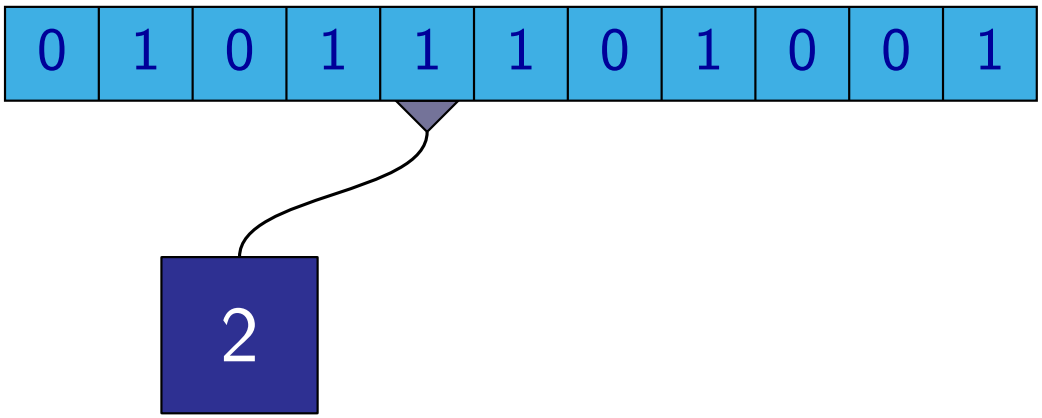
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



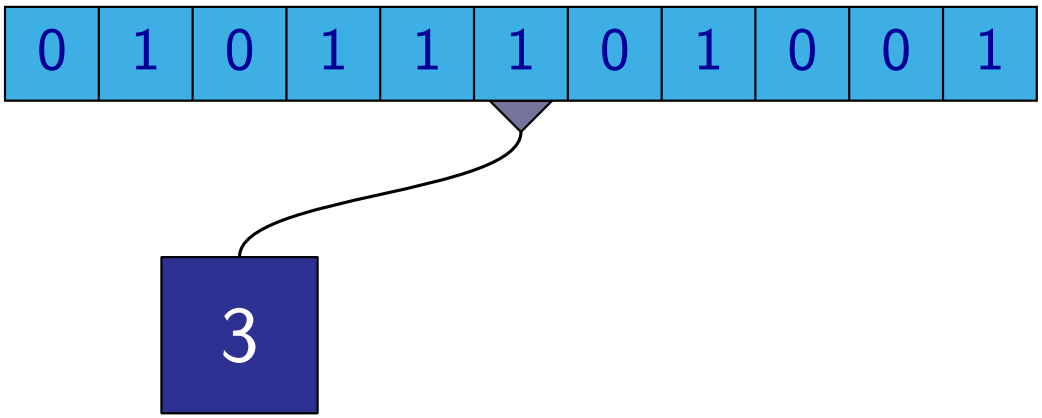
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



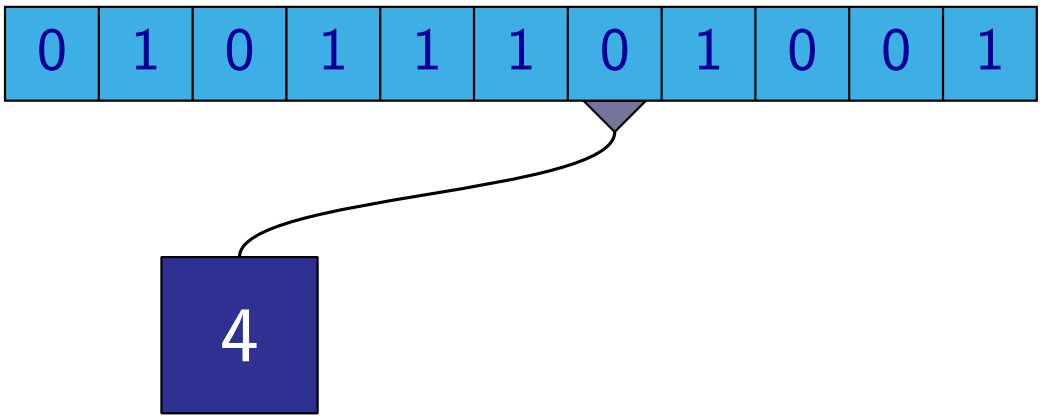
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



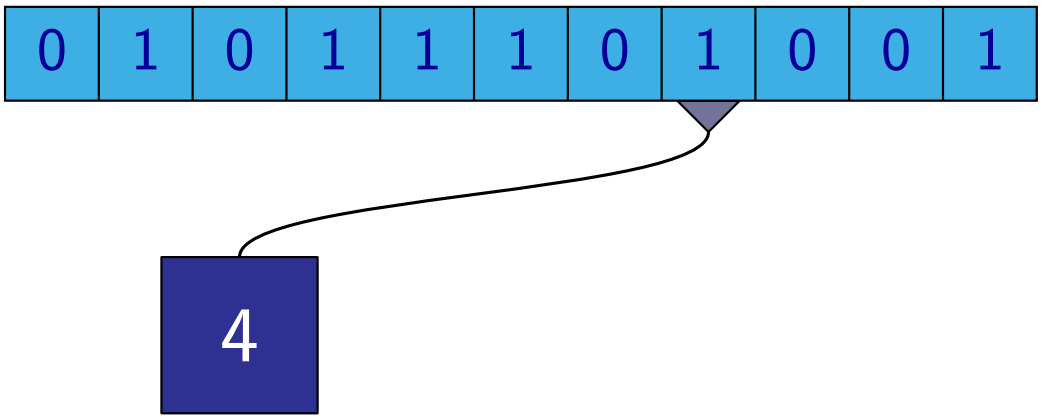
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



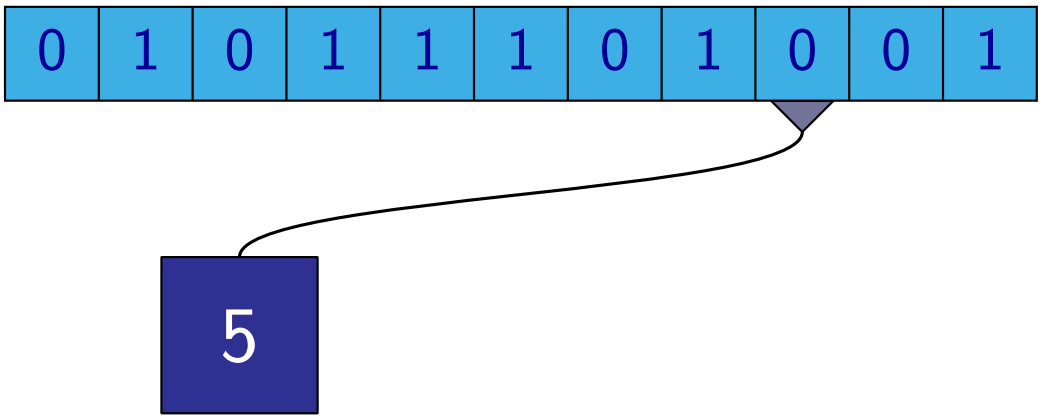
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



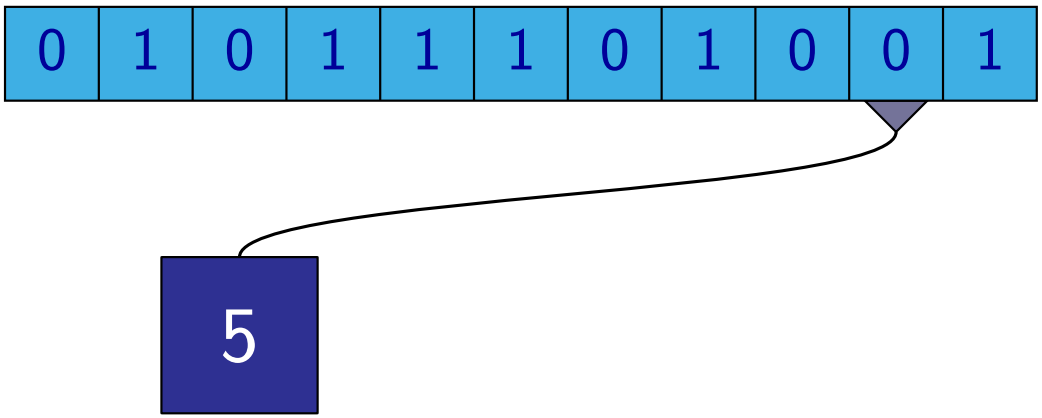
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



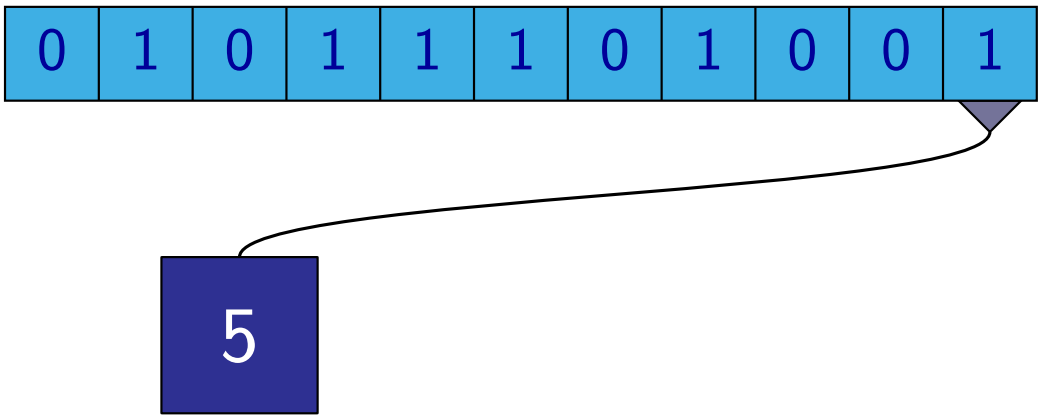
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



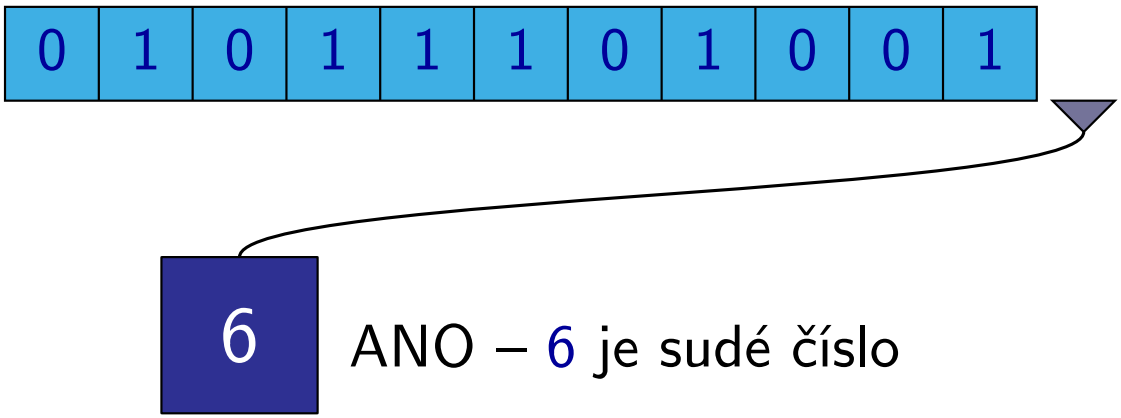
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



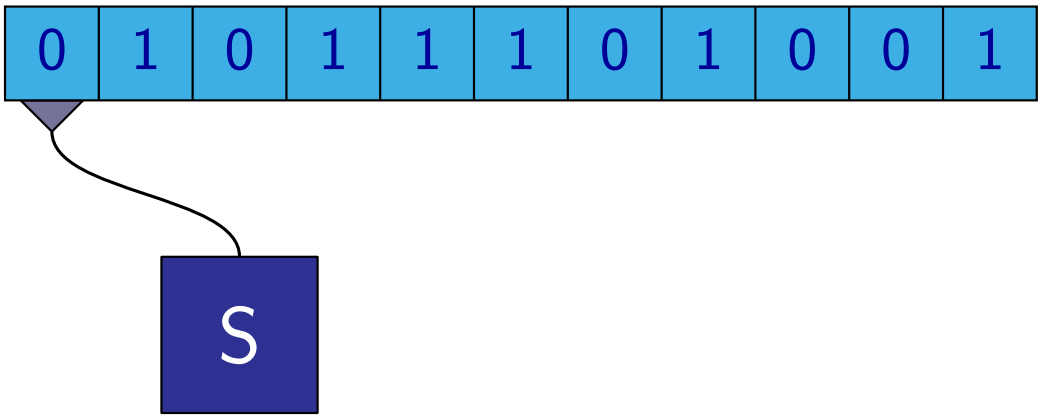
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



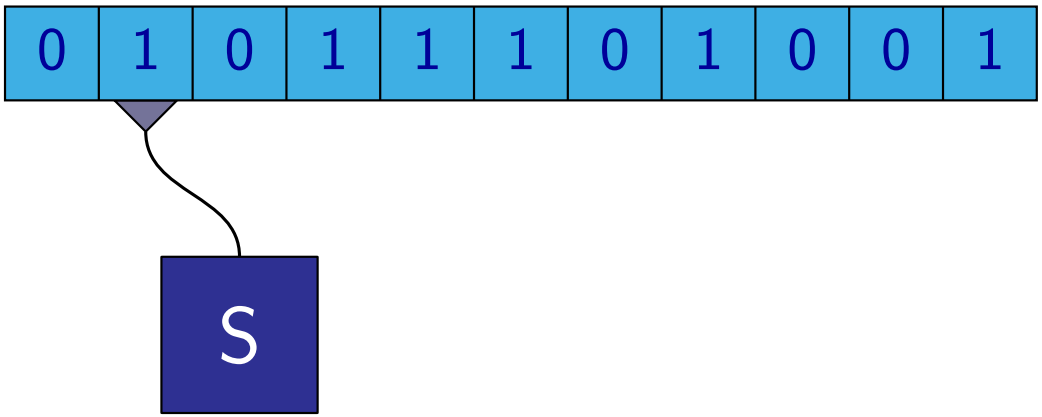
Rozpoznávání jazyka

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).



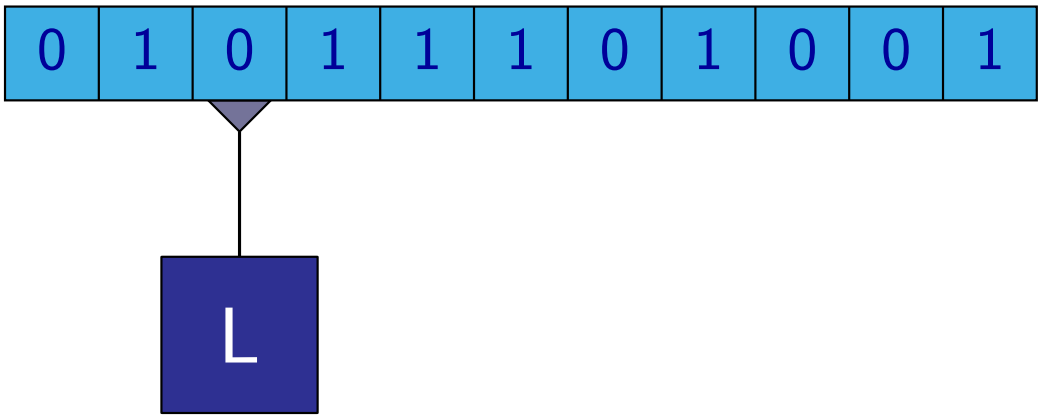
Rozpoznávání jazyka

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).



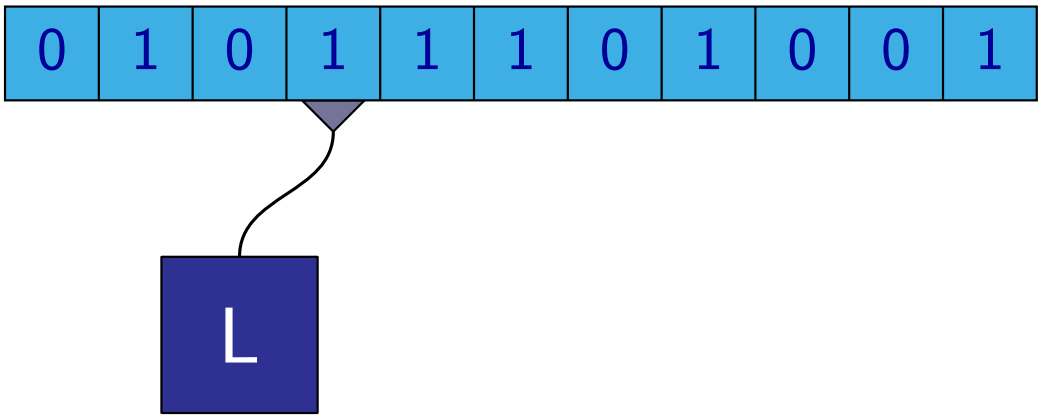
Rozpoznávání jazyka

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).



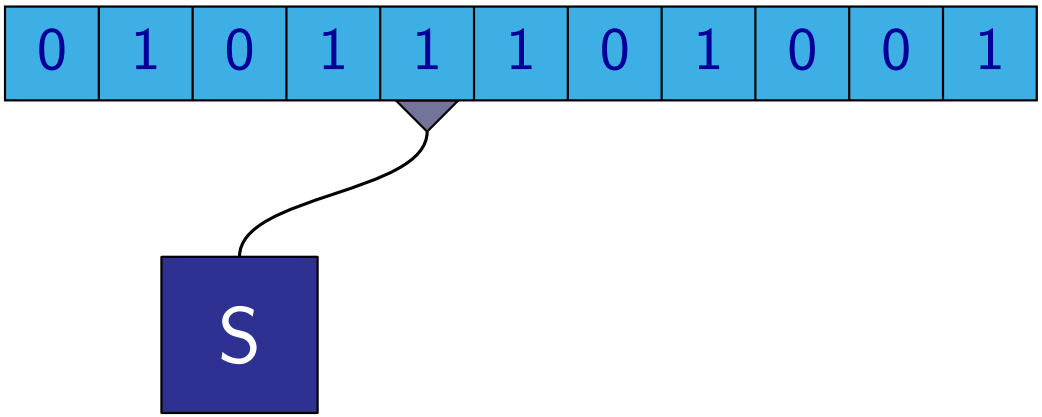
Rozpoznávání jazyka

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).



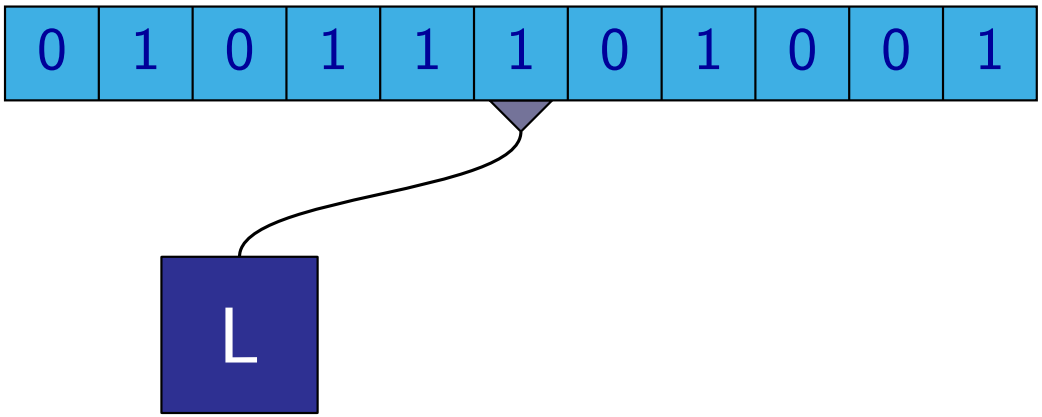
Rozpoznávání jazyka

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).



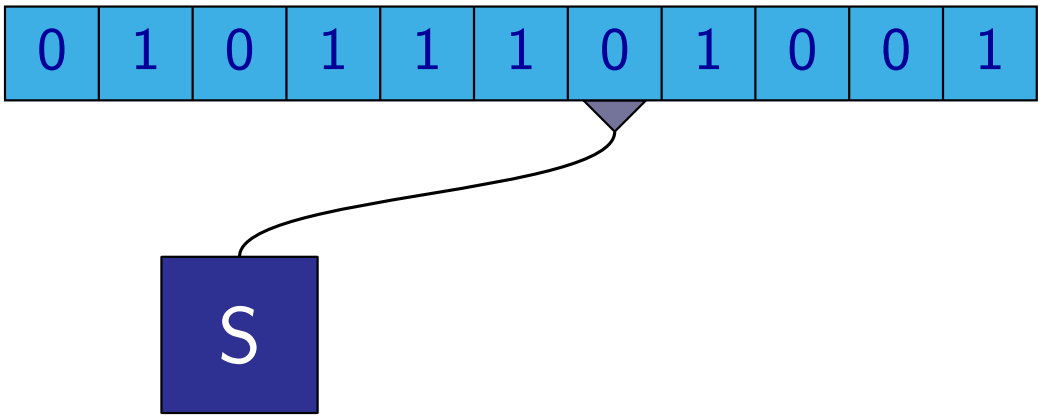
Rozpoznávání jazyka

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).



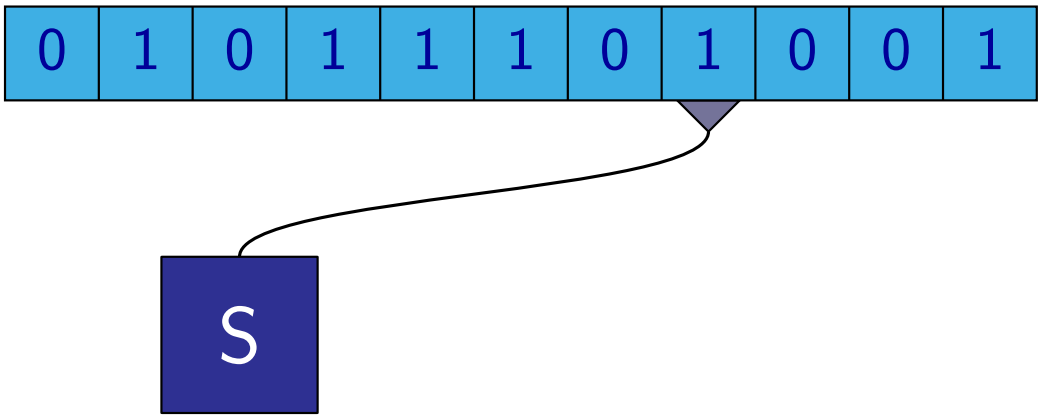
Rozpoznávání jazyka

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).



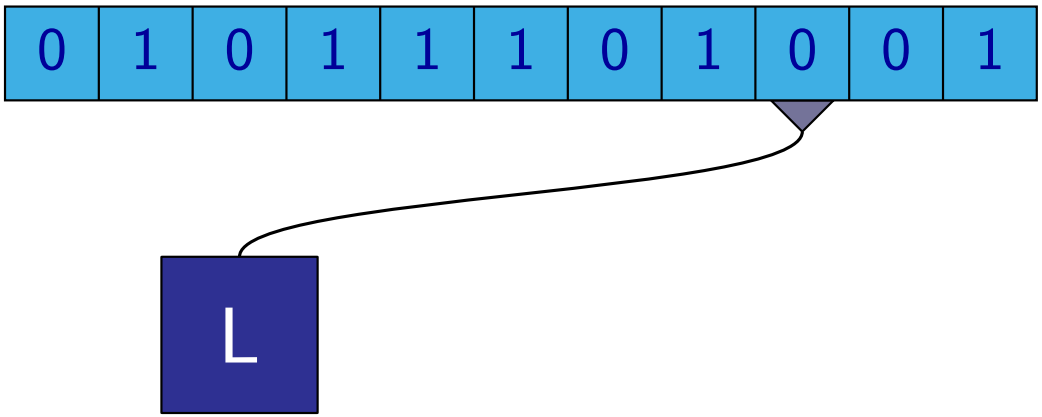
Rozpoznávání jazyka

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).



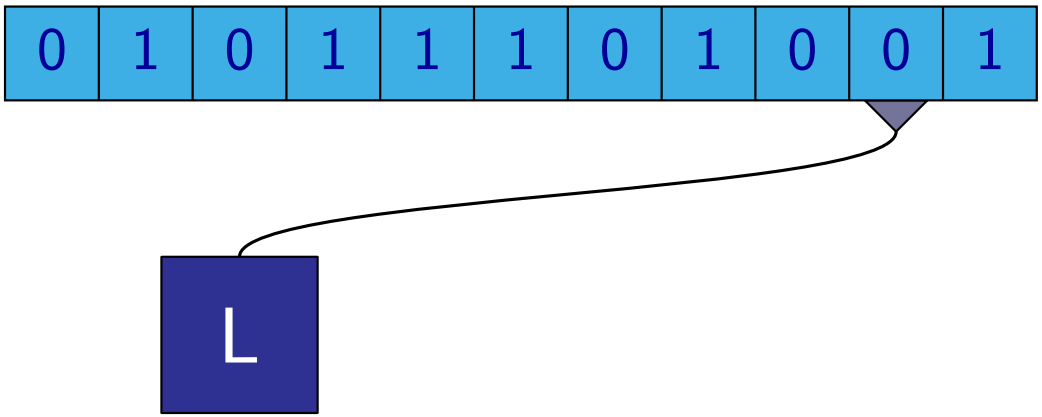
Rozpoznávání jazyka

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).



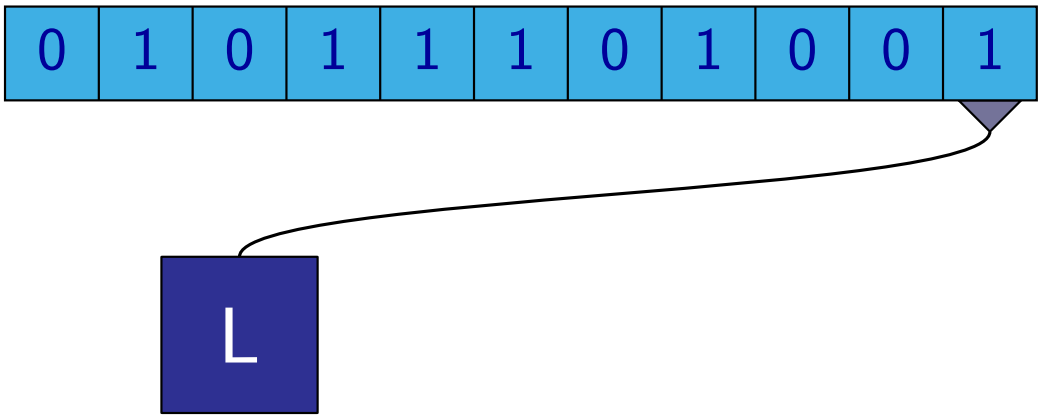
Rozpoznávání jazyka

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).



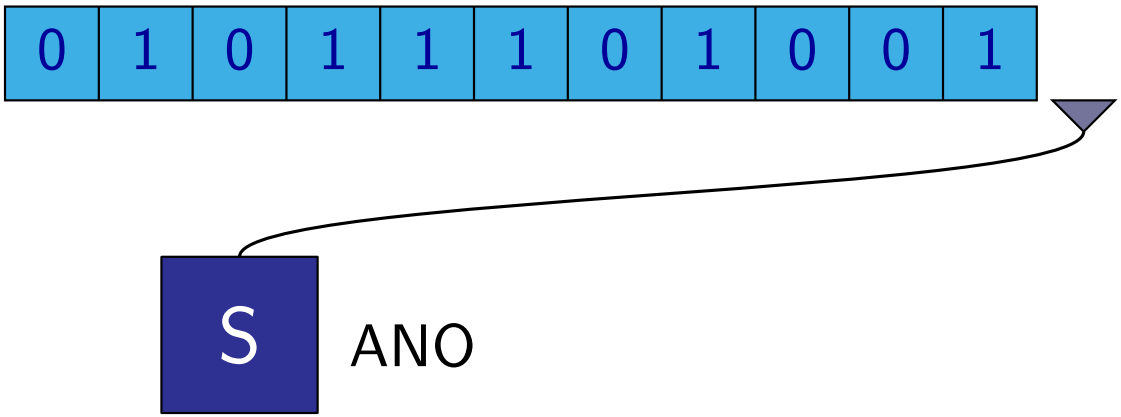
Rozpoznávání jazyka

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

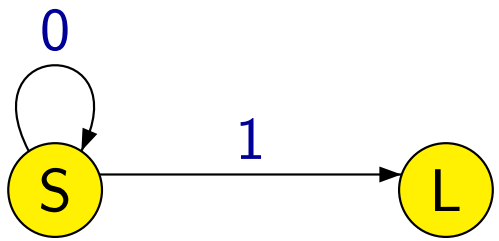
Chovaní tohoto zařízení můžeme popsat grafem:

Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:

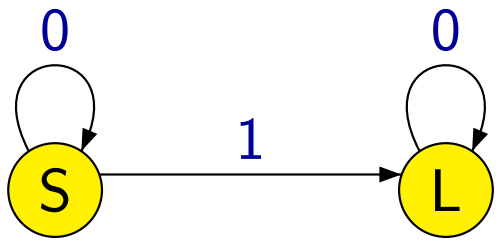
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



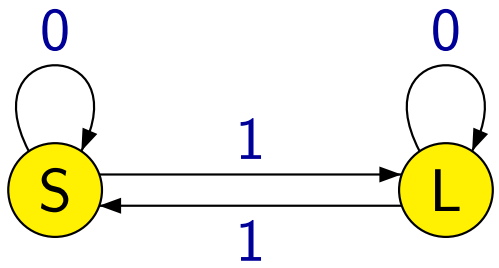
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



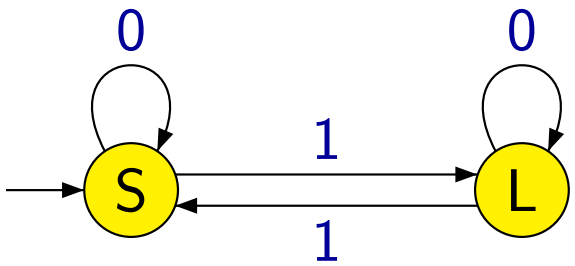
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



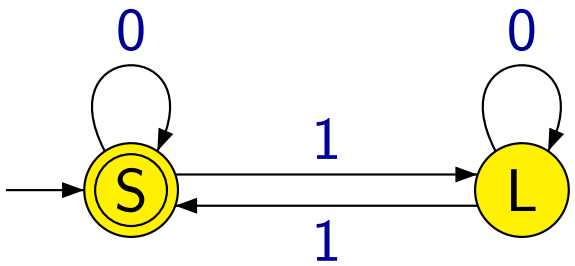
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



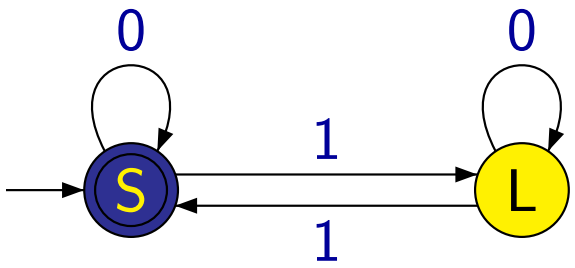
Rozpoznávání jazyka

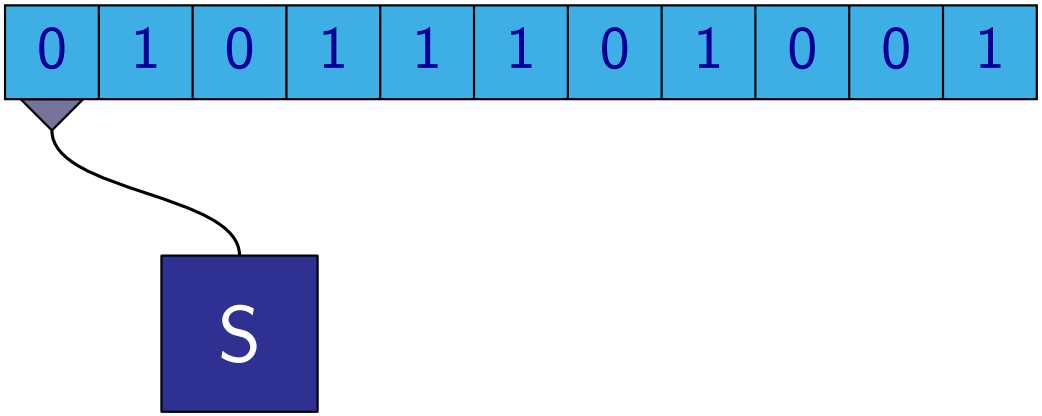
Chovaní tohoto zařízení můžeme popsat grafem:



Rozpoznávání jazyka

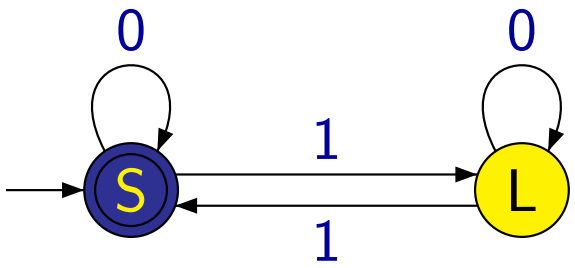
Chovaní tohoto zařízení můžeme popsat grafem:

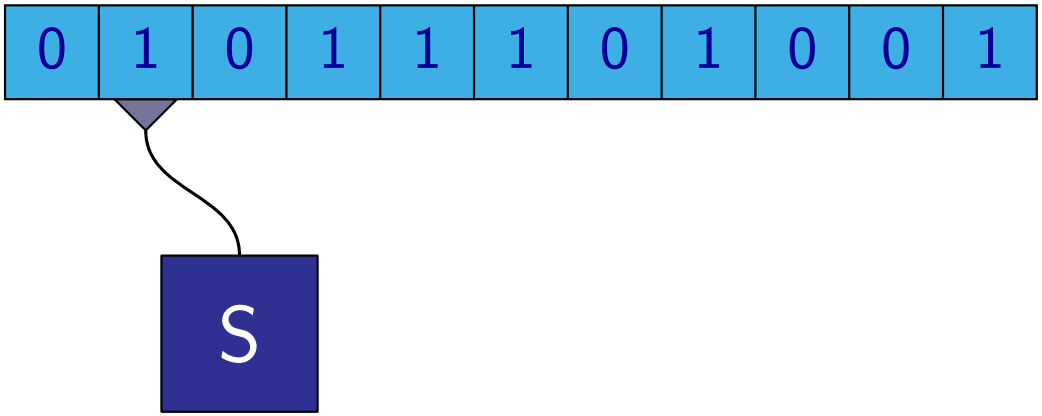




Rozpoznávání jazyka

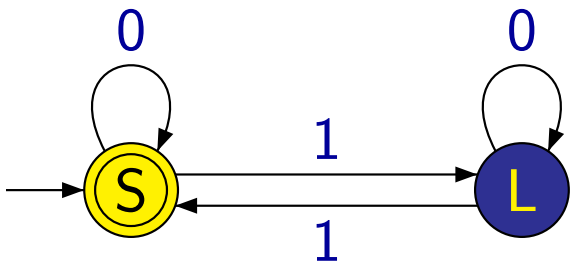
Chovaní tohoto zařízení můžeme popsat grafem:

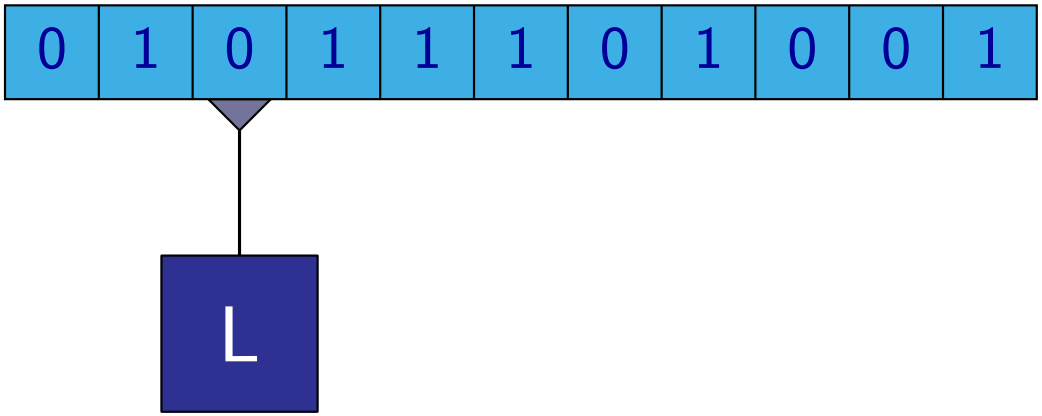




Rozpoznávání jazyka

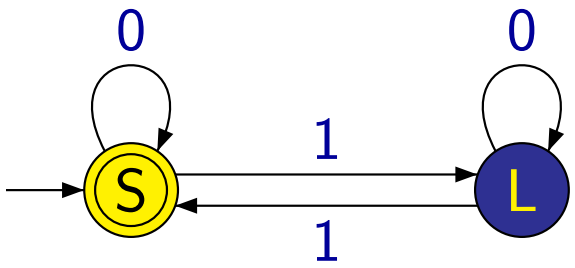
Chovaní tohoto zařízení můžeme popsat grafem:

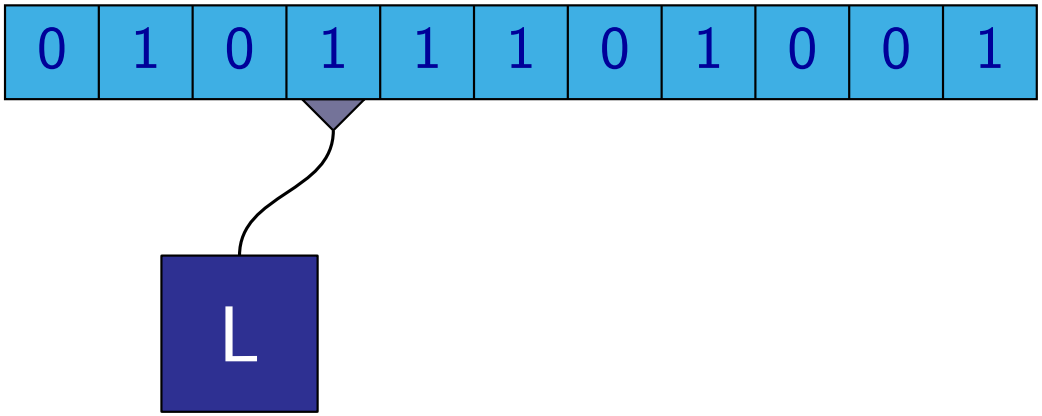




Rozpoznávání jazyka

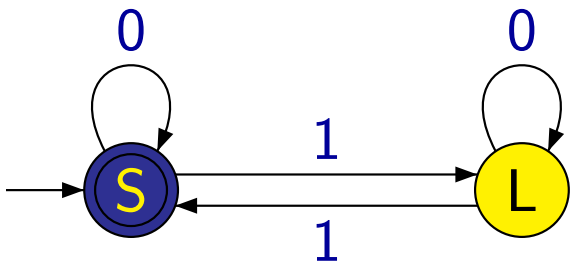
Chovaní tohoto zařízení můžeme popsat grafem:

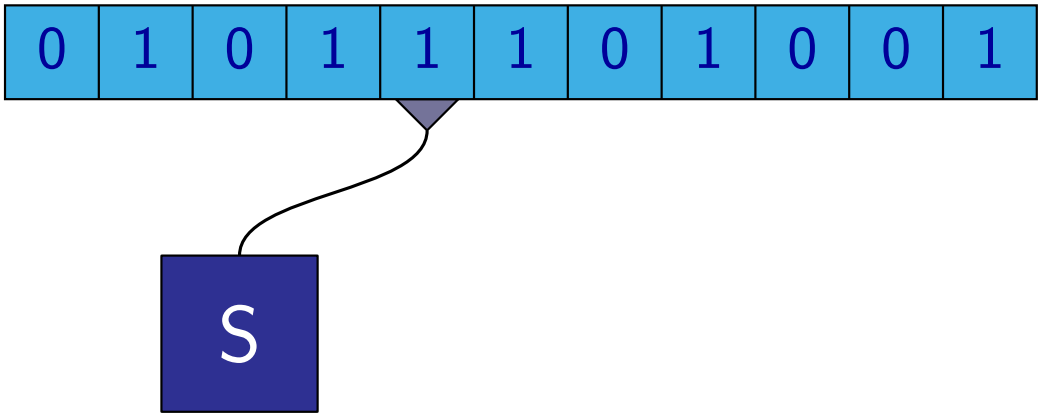




Rozpoznávání jazyka

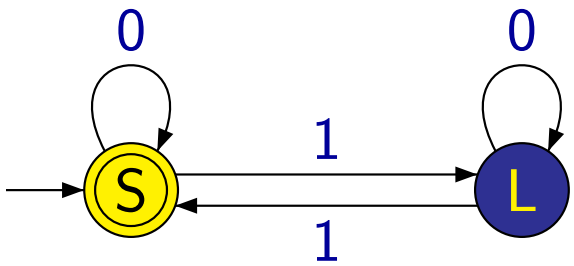
Chovaní tohoto zařízení můžeme popsat grafem:

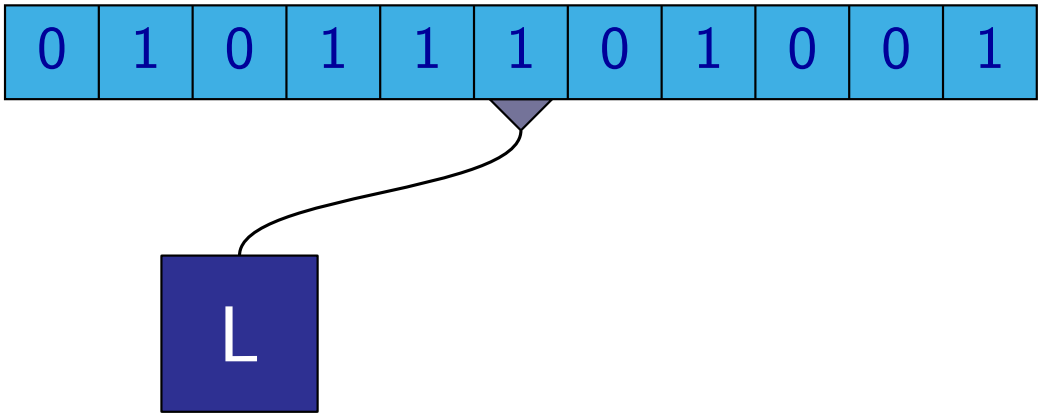




Rozpoznávání jazyka

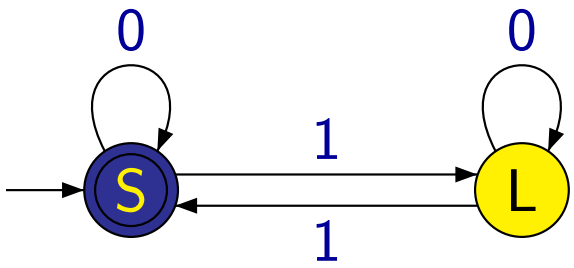
Chovaní tohoto zařízení můžeme popsat grafem:

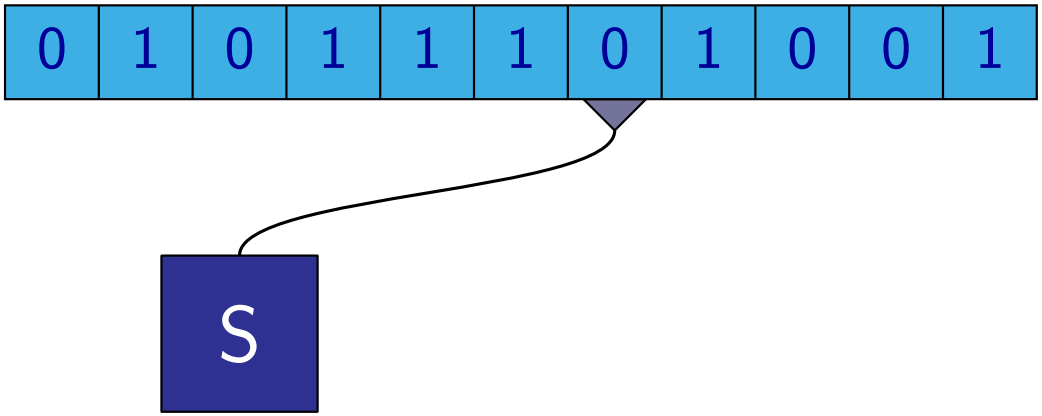




Rozpoznávání jazyka

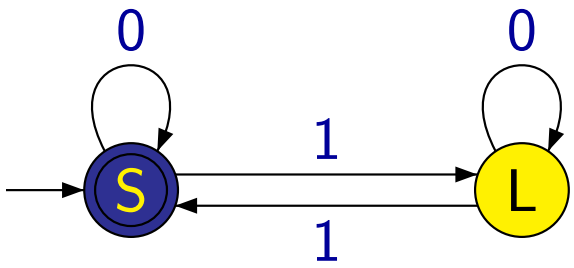
Chovaní tohoto zařízení můžeme popsat grafem:

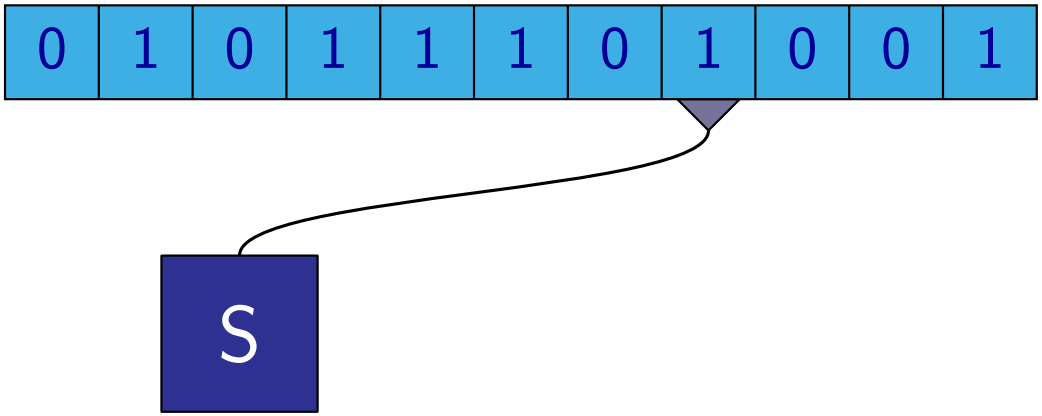




Rozpoznávání jazyka

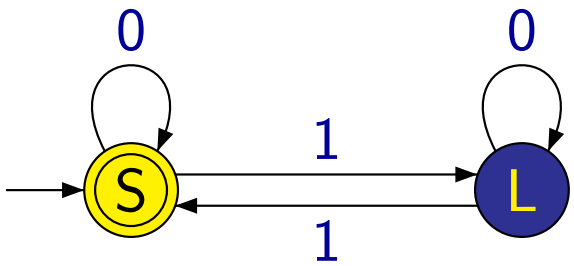
Chovaní tohoto zařízení můžeme popsat grafem:

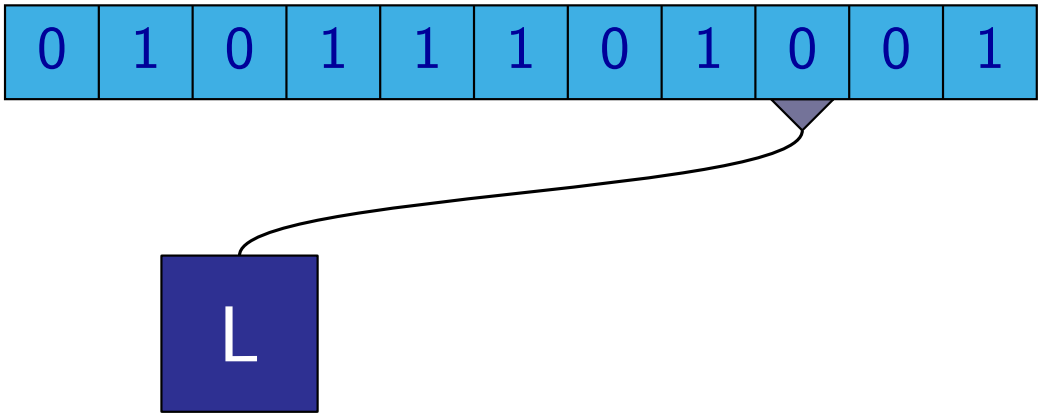




Rozpoznávání jazyka

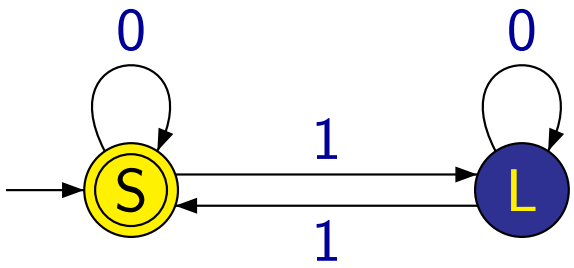
Chovaní tohoto zařízení můžeme popsat grafem:

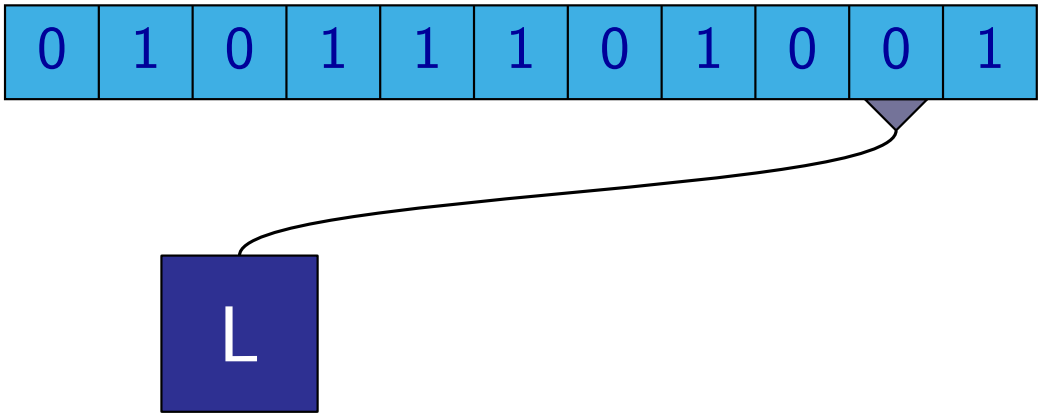




Rozpoznávání jazyka

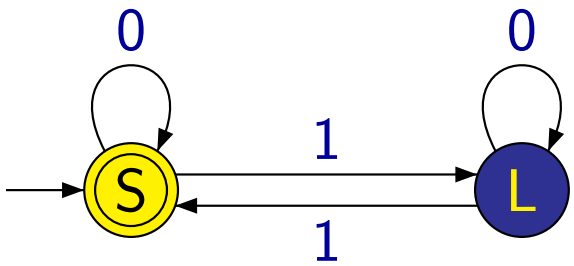
Chovaní tohoto zařízení můžeme popsat grafem:

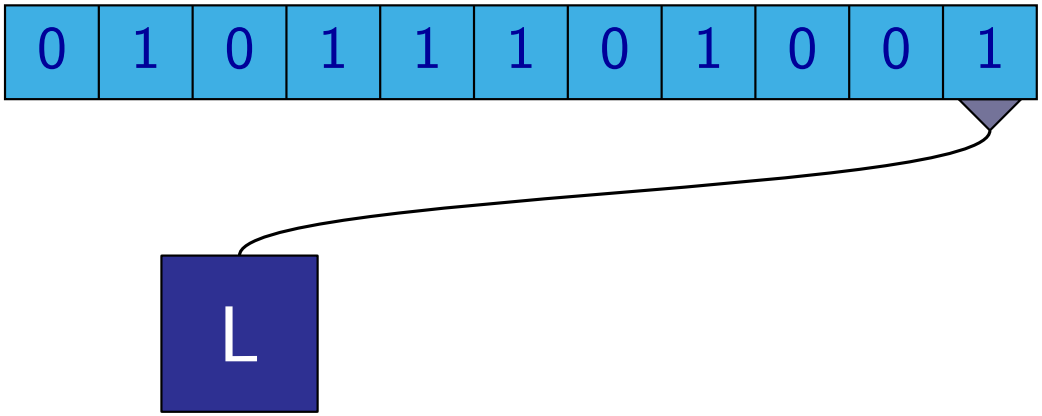




Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:





Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem: