

# Automaty a Gramatiky

Poznámky z přednášek

Letní semestr 2020/2021

Viktor Soukup, Lukáš Salak

# Obsah

<b>1 První přednáška</b>	<b>2</b>
<b>2 Druhá přednáška</b>	<b>4</b>
<b>3 Třetí přednáška</b>	<b>8</b>
<b>4 Čtvrtá přednáška</b>	<b>11</b>
<b>5 Pátá přednáška</b>	<b>14</b>
<b>6 Šestá přednáška</b>	<b>17</b>
6.1 Formální (generativní) gramatiky, bezkontextové gramatiky . . . . .	17
<b>7 Sedmá přednáška</b>	<b>22</b>
7.1 Zásobníkový automat . . . . .	22
<b>8 Osmá přednáška</b>	<b>25</b>
<b>9 Devátá přednáška</b>	<b>28</b>
<b>10 Desátá přednáška</b>	<b>30</b>
10.1 Uzávěrová vlastnosti . . . . .	30
10.2 Uzávěrové vlastnosti deterministických CFL . . . . .	33
<b>11 Jedenáctá přednáška</b>	<b>34</b>
11.1 Jak charakterizovat bezkontextové jazyky? . . . . .	34
11.2 Od zásobníkového automatu ke gramatice CFG . . . . .	35
11.3 Greibachové normální forma . . . . .	35
11.4 Turingovy stroje . . . . .	36
<b>12 Dvanáctá přednáška</b>	<b>37</b>
<b>13 Dvanáctá přednáška</b>	<b>39</b>
13.1 Lineárně omezené automaty . . . . .	39
13.2 Rekurzivní jazyky . . . . .	40
13.3 Jazyk, který není rekurzivně spočetný . . . . .	40
13.4 Kódování . . . . .	41

# 1 První přednáška

**Poznámka** (Chomského hierarchie): Automaty a gramatiky - dva způsoby popisu:

Turingovy stroje	$\leftrightarrow$	gramatiky Typu 0
lineárně omezené automaty	$\leftrightarrow$	kontextové gramatiky, monotónní gramatiky
zásobníkové automaty	$\leftrightarrow$	bezkontextové gramatiky
konečné automaty (DFA, NFA, $\lambda$ NFA)	$\leftrightarrow$	regulární jazyky

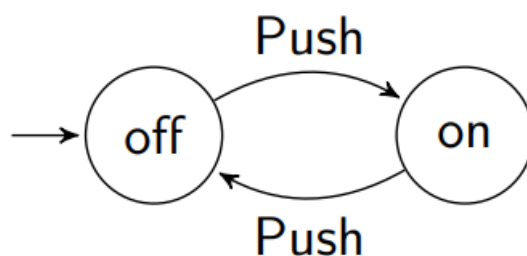
Nejjednodušší jsou nejnižší, turingův stroj je nejkomplikovanější. Každá gramatika odpovídá nějaké třídě automatů.

Proč to řešíme?

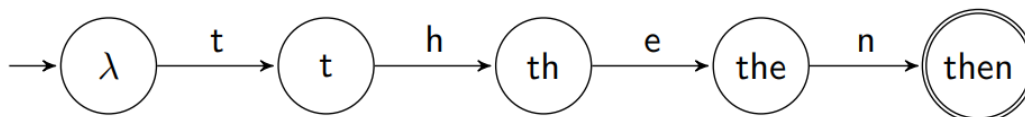
- zpracování přirozeného jazyka,
- překladače (lexikální, syntaktická analýza...),
- návrh, popis, verifikace hardware...
- hledání výskytu slova v textu (grep),
- verifikace systémů s konečně mnoha stavy

**Příklad:**

1. Návrh a verifikace integrovaných obvodů, např. Konečný automat modelující spínač on/off



2. Lexikální analýza, např. Konečný automat rozpoznávající slovo *then*



**Definice** (Deterministický konečný automat (DFA)):  $A = (Q, \Sigma, \delta, q_0, F)$  sestává z:

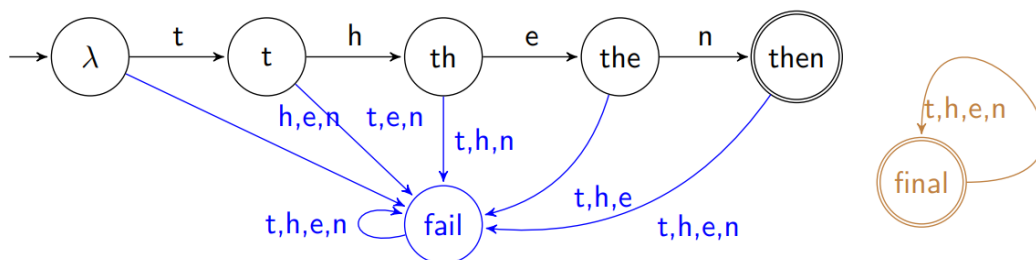
1. konečné množiny **stavů**, zpravidla značíme  $Q$
2. konečné neprázdné množiny **vstupních symbolů (abecedy)**, značíme  $\Sigma$
3. **přechodové funkce** zobrazení  $Q \times X \rightarrow Q$ , značíme  $\delta$ , která bude reprezentovaná hranami grafu
4. **počátečního stavu**  $q_0 \in Q$ , vede do něj šipka 'odnikud'

5. neprázdné **množiny koncových (přijímajících) stavů** (final states)  $F \subseteq Q$ , označených dvojíým kruhem či šipkou 'ven'.

**Poznámka:**

Pokud pro některou dvojici stavu a písmene není definovaný přechod, přidáme nový stav *fail* a přechodovou funkci doplníme na totální přidáním šipek do *fail*.

Pokud je množina  $F$  prázdná, přidáme do ní i  $Q$  nový stav *final* do kterého vedou jen přechody z něj samotného  $\forall s \in \Sigma : \delta(\text{final}, s) = \text{final}$ .



**Příklad:**

Automat  $A$  přijímající  $L = x01y : x, y \in \{0, 1\}^*$ .

Automat  $A = (\{q_0, q_1, q_2\}, 0, 1, \delta, q_0, q_1)$

Reprezentujeme stavovým diagramem (grafem), pomocí tabulky nebo stavovým stromem

**Definice** (Abeceda, slova, jazyky): Mějme neprázdnou množinu symbolů  $\Sigma$ .

- **Slovo** je konečná (i prázdná) posloupnost symbolů  $s \in \Sigma$ , **prázdné slovo** se značí  $\lambda$  nebo  $\epsilon$
- **Množinu všech slov v abecedě**  $\Sigma$  značíme  $\Sigma^*$
- množinu všech neprázdných slov v abecedě značíme  $\Sigma^+$
- **jazyk**  $L \subseteq \Sigma^*$  je množina slov v abecedě  $\Sigma$

**Definice** (Operace na  $\Sigma^*$ ):

1. **zřetězení slov**  $u.v$  nebo  $uv$
2. **mocnina** (počet opakování)  $u^n (u^0 = \lambda, u^1 = u, u^{n+1} = u^n.u)$
3. **délka slova**  $|u| (|\lambda| = 0, |auto| = 4)$
4. **počet výskytů**  $s \in \Sigma$  ve slově  $u$  značíme  $|u|_s (|zmrzlina|_z = 2)$ .

**Definice** (Rozšířená přechodová funkce): Mějme přechodovou funkci  $\delta : Q \times \Sigma \rightarrow Q$ .

Rozšířenou přechodovou funkci  $\delta^* : Q \times \Sigma^* \rightarrow Q$  (tranzitivní uzávěr  $\delta$ ) definujeme induktivně:

1.  $\delta^*(q, \lambda) = q$ ,
2.  $\delta^*(q, wx) = \delta(\delta^*(q, w), x)$  pro  $x \in \Sigma, w \in \Sigma^*$ .

**Poznámka:** Pokud se v textu objeví  $\delta$  aplikované na slova, míní se tím  $\delta^*$ .

**Definice** (Jazyk rozpoznávaný (přijímaný, akceptovaný) konečným automatem): Jazykem rozpoznávaným konečným automatem  $A = (Q, \Sigma, \delta, q_0, F)$  nazveme jazyk  $L(A) = \{w : w \in \Sigma^* \& \delta^*(q_0, w) \in F\}$ .

- Slovo  $w$  je přijímáno automatem  $A$ , právě když  $w \in L(A)$ .
- Jazyk  $L$  je rozpoznatelný konečným automatem, jestliže existuje konečný automat  $A$  takový, že  $L = L(A)$ .
- Třidu jazyků rozpoznatelných konečnými automaty označíme  $\mathcal{F}$ , nazveme **regulární jazyky**.

**Věta** (Iterační (pumping) lemma pro regulární jazyky): *Mějme regulární jazyk  $L$ . Pak existuje konstanta  $n \in \mathbb{N}$  (závislá na  $L$ ) tak, že každé  $w \in L; |w| \geq n$  můžeme rozdělit na tři části,  $w = xyz$ , že:*

1.  $y \neq \lambda$
2.  $|xy| \leq n$
3.  $\forall k \in \mathbb{N}_0$ , slovo  $xy^kz$  je také v  $L$ .

**Důkaz:**

- Mějme regulární jazyk  $L$ , pak existuje DFA  $A$  s  $n$  stavy, že  $L = L(A)$ .
- Vezměme libovolné slovo  $a_1a_2a_3 \dots a_m = w \in L$  délky  $m \geq n, a_i \in \Sigma$ .
- Definujeme:  $\forall i p_i = \delta^*(q_0, a_1a_2 \dots a_i)$ . Platí  $p_0 = q_0$ .
- Máme  $n + 1 p_i$  a  $n$  stavů, některý se opakuje, vezměme první takový, t. j.  $(\exists i, j : 0 \leq i < j \leq n \wedge p_i = p_j)$ .
- Definujeme  $x = a_1a_2 \dots a_i, y = a_{i+1}a_{i+2} \dots a_j, z = a_{j+1}a_{j+2} \dots a_m$ , t.j.  $w = xyz, y \neq \lambda, |xy| \leq n$ .
- pak  $y^k$  můžeme opakovat libovolněkrát a vstup je také akceptovaný.

□

**Příklad** (Aplikace pumping lemmatu): TODO

## 2 Druhá přednáška

**Definice** (Kongruence): Mějme konečnou abecedu  $\Sigma$  a relaci ekvivalence  $\sim$  na  $\Sigma^*$  (reflexivní, symetrická, tranzitivní). Potom:

1.  $\sim$  je pravá kongruence, jestliže  $(\forall u, v, w \in \Sigma^*) u \sim v \implies uw \sim vw$ .
2. je konečného indexu, jestliže rozklad  $\Sigma^* / \sim$  má konečný počet tříd.
3. Třidu kongruence  $\sim$  obsahující slovo  $u$  značíme  $[u]_\sim$ , resp.  $[u]$ .

**Věta** (Myhill-Nerodova Věta): *Nechť  $L$  je jazyk nad konečnou abecedou  $\Sigma$ . Potom následující tvrzení jsou ekvivalentní:*

1.  $L$  je rozpoznatelný konečným automatem,
2.  $\exists$  pravá kongruence  $\sim$  konečného indexu nad  $\Sigma^*$  tak, že  $L$  je sjednocením jistých tříd rozkladu  $\Sigma^* / \sim$ .

**Důkaz:**

1.  $\implies$  2.; t.j. automat  $\implies$  pravá kongruence konečného indexu
  - definujeme  $u \sim v \equiv \delta^*(q_0, u) = \delta^*(q_0, v)$ .

- je to ekvivalence (reflexivní, symetrická, tranzitivní)
- je to pravá kongruence (z definice  $\delta^*$ )
- má konečný index (konečně mnoho stavů)

$$L = \{w | \delta^*(q_0, w) \in F\} = \bigcup_{q \in F} \{w | \delta^*(q_0, w) = q\} = \bigcup_{q \in F} [w | \delta^*(q_0, w) = q]_{\sim}.$$

2.  $\implies$  1.; t.j. pravá kongruence konečného indexu  $\implies$  automat

- abeceda automatu nazveme  $\Sigma$
- za stavy  $Q$  volíme třídy rozkladu  $\Sigma^* / \sim$
- počáteční stav  $q_0 \equiv [\lambda]_{\sim}$
- koncové stavy  $F = \{c_1, \dots, c_n\}$ , kde  $L = \bigcup_{i=1, n} c_i$
- přechodová funkce  $\delta([u], x) = [ux]$  (je korektní z def. pravé kongruence).
- $L(A) = L$

$$w \in L \Leftrightarrow w \in \bigcup_{i=1, n} c_i \Leftrightarrow w \in c_1 \vee \dots \vee w \in c_n \Leftrightarrow [w] = c_1 \vee \dots \vee [w] = c_n \Leftrightarrow [w] \in F \Leftrightarrow w \in L(A)$$

□

**Příklad:** Sestrojte automat přijímající jazyk

$$L = \{w | w \in a, b^* \& |w|_a = 3k + 2\},$$

t. j. obsahuje  $3k + 2$  symbolů  $a$ .

1.  $|u|_x$  značí počet symbolů  $x$  ve slově  $u$
2. definujeme  $u \sim v \equiv (|u|_a \bmod 3 = |v|_a \bmod 3)$
3. třídy ekvivalence 0, 1, 2
4.  $L$  odpovídá třídě 2
5.  $a$  - přechody do následující třídy
6.  $b$  - přechody zachovávající třídu

## 28. slide, doplň obrázek

**Příklad** (Neregulární pumpovatelný jazyk): Ne-regulární jazyk, který lze pumpovat

Jazyk  $L = \{u | u = a^+ b^i c^i \vee u = b^i c^j\}$  není regulární (Myhill-Nerodova věta), ale vždy lze pumpovat první písmeno.

1. Předpokládejme, že  $L$  je regulární
2.  $\implies$  pak  $\exists$  pravá kongruence  $\sim_L$  konečného indexu  $m$ ,  $L$  je sjednocení některých tříd  $\Sigma^* / \sim_L$
3. vezmeme množinu slov  $S = \{ab, abb, abbb, \dots, ab^{m+1}\}$
4. existují dvě slova  $i \neq j$ , která padnou do stejné třídy  
 $i \neq j \quad ab^i \sim ab^j$   
 přidáme  $c^i \quad ab^i c^i \sim ab^j c^i$   $\sim$  je kongruence  
 spor  $ab^i c^i \in L \& ab^j c^i \notin L \quad s' L$  je sjednocení některých tříd  $\Sigma^* / \sim_L$ .

**Definice** (Dosažitelný stav): Mějme DFA  $A = (Q, \Sigma, \delta, q_0, F)$  a  $q \in Q$ . Řekneme, že stav je dosažitelný, jestliže  $\exists w \in \Sigma^* : \delta^*(q_0, w) = q$ .

**Příklad:** Algoritmus na hledání dosažitelných stavů : DFS (důkaz asi není nutný)

**Definice** (Automatový homomorfismus): Necht'  $A_1, A_2$  jsou DFA. Řekneme, že zobrazení  $h : Q_1 \rightarrow Q_2$  je (automatovým) homomorfismem, jestliže:

$$\begin{array}{ll} h(q_{0_1}) = q_{0_2} & \text{'stejně' počáteční stavy} \\ h(\delta_1(q, x)) = \delta_2(h(q), x) & \text{'stejně' přechodové funkce} \\ q \in F_1 \Leftrightarrow h(q) \in F_2 & \text{'stejně' koncové stavy.} \end{array}$$

Homomorfismus prostý a na nazýváme isomorfismus.

**Definice** (Ekvivalence automatů): Dva konečné automaty  $A, B$  nad stejnou abecedou  $\Sigma$  jsou ekvivalentní, jestliže rozpoznávají stejný jazyk, t. j.  $L(A) = L(B)$ .

**Věta** (Věta o ekvivalenci automatů): *Existuje-li homomorfismus konečných automatů  $A_1$  do  $A_2$ , pak jsou  $A_1$  a  $A_2$  ekvivalentní.*

**Důkaz:**

1. Pro libovolné slovo  $w \in \Sigma^*$  konečnou iterací

$$h(\delta_1^*(q, w)) = \delta_2^*(h(q), w)$$

2. dále

$$\begin{aligned} w \in L(A_1) &\Leftrightarrow \delta_1^*(q_{0_1}, w) \in F_1 \\ &\Leftrightarrow h(\delta_1^*(q_{0_1}, w)) \in F_2 \\ &\Leftrightarrow \delta_2^*(h(q_{0_1}, w)) \in F_2 \\ &\Leftrightarrow \delta_2^*(q_{0_2}, w) \in F_2 \\ &\Leftrightarrow w \in L(A_2) \end{aligned}$$

□

**Definice** (Ekvivalence stavů): Říkáme, že stavy  $p, q \in Q$  konečného automatu  $A$  jsou ekvivalentní, pokud:

1. Pro všechna vstupní slova  $w : \delta^*(p, w) \in F \Leftrightarrow \delta^*(q, w) \in F$ .

Pokud dva stavy nejsou ekvivalentní, říkáme že jsou rozlišitelné.

**Příklad:** Ten example je na slide 36, nejlepší s tím obrázkem

**Definice** (Algoritmus hledání rozpoznatelných stavů v DFA): Následující algoritmus nalezne rozlišitelné stavy:

1. Základ: Pokud  $p \in F$  (přijímající) a  $q \notin F$ , pak je dvojice  $\{p, q\}$  rozlišitelná.
2. Indukce: Necht'  $p, q \in Q, a \in \Sigma$  a o dvojici  $r, s : r = \delta(p, a), s = \delta(q, a)$  víme, že jsou rozlišitelné. Pak i  $\{p, q\}$  jsou rozlišitelné.
3. opakuj dokud  $\exists$  nová trojice  $p, q \in Q, a \in \Sigma$ .

**Doplň obrázky zo slidov, 37/38**

**Věta:** Pokud dva stavy nejsou odlišeny předchodícím algoritmem, pak jsou tyto stavy ekvivalentní.

**Důkaz:** Korektnost algoritmu

1. Uvažujeme špatné páry stavů, které jsou rozlišitelné a algoritmus je nerozlišil.
2. Vezměme z nich pár  $p, q$  rozlišitelný nejkratším slovem  $w = a_1 \dots a_n$ .
3. Stavy  $r = \delta(p, a_1), s = \delta(q, a_1)$  jsou rozlišitelné kratším slovem  $a_2 \dots a_n$ , takže pár není mezi špatnými.
4. Tedy jsou „vyškrtnuté“ algoritmem.
5. Tedy v příštím kroku algoritmus rozliší i  $p, q$ .

□

**Poznámka:** Čas výpočtu je polynomiální vzhledem k počtu stavů.

1. V jednom kole uvažujeme všechny páry, t.j.  $O(n^2)$ .
2. Kol je maximálně  $O(n^2)$ , protože pokud nepřidáme křížek, končíme.
3. Dohromady  $O(n^4)$ .

Algoritmus lze zrychlit na  $O(n^2)$  pamatováním stavů, které závisí na páru  $\{r, s\}$  a sledováním těchto seznamů „zpátky“.

**Definice (Redukovaný DFA):** Deterministický konečný automat je redukovaný, pokud

1. nemá dosažitelné stavy,
2. žádné dva stavy nejsou ekvivalentní.

**Definice (Redukt):** Konečný automat B je redukem automatu A, jestliže:

1. B je redukovaný,
2. A a B jsou ekvivalentní

**ADD PICS PLS** (*don't shout pls*)

**Věta** (Algoritmus na nalezení reduktu DFA A):

1. Ze vstupního DFA A eliminujeme stavy nedosažitelné z počátečního stavu.
2. Najdeme rozklad zbylých stavů na třídy ekvivalence.
3. Konstruujeme DFA B na třídách ekvivalence jakožto stavech.  
Přechodová funkce B  $\gamma$ , mějme  $S \in Q_B$ . Pro libovolné  $q \in S$  označíme  $T$  třídu ekvivalence  $\delta(q, a)$  a definujeme  $\gamma(S, a) = T$ . Tato třída musí být stejná pro všechna  $q \in S$ .
4. Počáteční stav B je třída obsahující počáteční stav A.
5. Množina přijímajících stavů B jsou bloky odpovídající přijímacím stavům A.



### 3 Třetí přednáška

**Definice** (Algoritmus na testování ekvivalence regulárních jazyků): Ekvivalenci regulárních jazyků  $L, M$  testujeme následovně:

1. Najdeme  $DFAA_L, A_M$  rozpoznávající  $L(A_L) = L, L(A_M) = M, Q_L \cap Q_M = \emptyset$ .
2. Vytvoříme  $DFA$  sjednocením stavů a přechodů  $(Q_L \cup Q_M, \Sigma, \delta_L \cap \delta_M, q_L, F_L \cap F_M)$ ; zvolíme jeden z počátečních stavů.
3. Jazyky jsou ekvivalentní právě když počáteční stavy původních DFA jsou ekvivalentní.

[Nedeterministické konečné automaty (NFA)] Nedeterministický automat může být ve více stavech paralelně. Má schopnost 'uhodnout' něco o vstupu.

**přidat obrázok zo slidov (61. slide)**

**Definice** (NFA): Nedeterministický konečný automat (NFA)  $A = (Q, \Sigma, \delta, S_0, F)$  sestává z:

1. konečné množiny stavů, zpravidla značíme  $Q$ ,
2. konečné množiny vstupních symbolů, značíme  $\Sigma$
3. přechodové funkce, zobrazení  $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$  vracející podmnožinu  $Q$ .
4. množiny počátečních stavů  $S_0 \subseteq Q$ ,
5. množiny koncových (přijímajících) stavů  $F \subseteq Q$ .

**Definice** (Rozšířená přechodová funkce): Pro přechodovou funkci  $\delta$  NFA je rozšířená přechodová funkce  $\delta^*$

$\delta^* : Q \times \Sigma^* \rightarrow \mathcal{P}(Q)$  definovaná indukcí:

start:  $\delta^*(q, \lambda) = q$ . ind. indukční krok:

$$\delta^*(q, wx) = \bigcup_{p \in \delta^*(q, w)} \delta(p, x)$$

t. j. množina stavů, do kterých se mohou dostat posloupností 'správně označených'

**Definice** (Jazyk přijímaný nedeterministickým konečným automatem): Mějme NFA  $A = (Q, \Sigma, \delta, S_0, F)$ , Pak

$$L(A) = \{w : (\exists q_0 \in S_0) \delta^*(q_0, w) \cap F \neq \emptyset\}$$

je jazyk přijímaný automatem  $A$ .

Tedy  $L(A)$  je množina slov  $w \in \Sigma^*$  takových, že  $\delta^*(q_0, w)$  obsahuje alespoň jeden přijímající stav.

**Algoritmus** (Podmnožinová konstrukce): Podmnožinová konstrukce začíná s NFA  $N = (Q_N, \Sigma, \delta_N, S_0, F_N)$ . Cílem je popis deterministického DFA  $D = (Q_D, \Sigma, \delta_D, S_0, F_D)$ , pro který  $L(N) = L(D)$ .

1.  $Q_D$  je množina podmnožin  $Q_N, Q_D = \mathcal{P}(Q_N)$  (potenční množina).

**Poznámka:** Nedosažitelné stavy můžeme vynechat

2. Počáteční stav DFA je stav označený  $S_0$ , t.j. prvek  $Q_D$ .
3.  $F_D = \{S : S \in \mathcal{P}(Q_N) \& S \cap F_N \neq \emptyset\}$ , tedy  $S$  obsahuje alespoň jeden přijímající stav  $N$ .

4. Pro každé  $S \subseteq Q_N$  a každý vstupní symbol  $a \in \Sigma$ ,

$$\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a).$$

**Věta** (Převod NFA na DFA): Pro DFA  $D = (Q_D, \Sigma, \delta_D, S_0, F_D)$  vytvořený podmnožinovou konstrukcí z NFA  $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$  platí  $L(N) = L(D)$ .

**Důkaz:** Indukcí dokážeme, že  $\delta_D^*(S_0, w) = \delta_N^*(q_0, w)$ . □

Můžeme přidat ještě tzv.  $\lambda$ -přechod.

**Definice** ( $\lambda$ -přechod): Dovolíme přechody na  $\lambda$ , prázdné slovo, t.j. bez přečtení vstupního symbolu.  
**doplnit obrázok, 68. slide**

**Definice** ( $\lambda$ -uzávěr): Pro  $q \in Q$  definujeme  $\lambda$ -uzávěr  $\lambda CLOSE(q)$  rekurzivně:

1. Stav  $q$  je  $\lambda CLOSE(q)$ .
2. Je-li  $p \in \lambda CLOSE(q)$  a  $r \in \delta(p, \lambda)$ , pak i  $r \in \lambda CLOSE(q)$ .

Pro  $S \subseteq Q$  definujeme  $\lambda CLOSE(S) = \bigcup_{q \in S} \lambda CLOSE(q)$ .

**Definice** (Rozšířená přechodová funkce a jazyk přijímaný  $\lambda$ -NFA): Nechť  $E = (Q, \Sigma, \delta, S_0, F)$  je  $\lambda$ -NFA. Rozšířenou přechodovou funkci  $\delta^*$  definujeme následovně:

1.  $\delta^*(q, \lambda) = \lambda CLOSE(q)$ .
2. indukční krok:  $v = wa$ , kde  $w \in \Sigma^*$ ,  $a \in \Sigma$ .

$$\delta^*(q, wa) = \lambda CLOSE \left( \bigcup_{p \in \delta^*(q, w)} \delta(p, a) \right)$$

**Věta** (Eliminace  $\lambda$ -přechodů): Jazyk  $L$  je rozpoznatelný  $\lambda$ -NFA právě když je  $L$  regulární.

**Důkaz:** Pro libovolný  $\lambda$ NFA  $E = (Q_E, \Sigma, \delta_E, S_0, F_E)$  zkonstruujeme DFA  $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$  přijímající stejný jazyk jako  $E$ .

1.  $Q_D \subseteq \mathcal{P}(Q_E), \forall S \subseteq Q_E : \lambda CLOSE(S) \in Q_D$ . V  $Q_D$  může být i  $\emptyset$ .
2.  $q_D = \lambda CLOSE(S_0)$ .
3.  $F_D = \{S : S \in Q_D \& S \cap F_E \neq \emptyset\}$ .
4. Pro  $S \in Q_D, a \in \Sigma$  definujeme  $\delta_D(S, a) = \lambda CLOSE(\bigcup_{p \in S} \delta(p, a))$ .

□

**Definice** (Množinové operace nad jazyky): Mějme dva jazyky  $L, M$ . Definujeme následující operace:

1. binární sjednocení  $L \cup M = \{w : w \in L \vee w \in M\}$

**Poznámka:** Příklad: jazyk obsahuje slova začínající  $a^i$  nebo tvaru  $b^j c^j$ .

2. průnik  $L \cap M = \{w : w \in L \& w \in M\}$

**Poznámka:** Příklad: jazyk obsahuje slova sudé délky končící na 'baa'.

3. rozdíl  $L - M = \{w : w \in L \& w \notin M\}$
4. doplněk (komplement)  $\bar{L} = -L = \{w : w \in L\} = \sigma^* - L$

**Poznámka:** Příklad: jazyk obsahuje slova nekončící na 'a'.

**Věta** (de Morganova pravidla): 1.  $L \cap M = \overline{\bar{L} \cup \bar{M}}$

$$2. L \cup M = \overline{\bar{L} \cap \bar{M}}$$

$$3. L - M = L \cap \bar{M}$$

**Věta** (Uzavřenost na množinové operace): *Mějme regulární jazyky  $L, M$ . Pak jsou následující jazyky také regulární:*

1. sjednocení  $L \cup M$
2. průnik  $L \cap M$
3. rozdíl  $L - M$
4. doplněk  $\bar{L} = \Sigma^* - L$ .

**Důkaz:**

1. Pokud  $\delta$  není pro některé dvojice  $q, a$  definovaná, přidáme nový nepřijímající stav  $q_n$  a do něj přechod pro vše dříve nedefinované plus  $\forall a \in \Sigma \cup \lambda : \delta(q_n, a) = q_n$ .
2. Pak stačí prohodit koncové a nekoncové stavy přijímajícího deterministického FA  $F = Q_A - F_A$ .
3. pro rozdíl doplníme funkci  $\delta$  na totální.
4. Zkonstruuujeme součinný automat,

$$Q = (Q_1 \times Q_2, \Sigma, \delta((p_1, p_2), x) = (\delta_1(p_1, x), \delta_2(x)), (q_{01}, q_{02}, F))$$

5. průnik:  $F = F_1 \times F_2$
6. sjednocení:  $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$
7. rozdíl:  $F = F_1 \times (Q_2 - F_2)$ .

□

**Definice** (Řetězcové operace nad jazyky):

1. zřetězení jazyků ...  $L.M = \{uv : u \in L \& v \in M\}$ ,  $L.x = L.x$  a  $x.L = x.L$  pro  $x \in \Sigma$
2. mocniny jazyka ...  $L^0 = \lambda$ ,  $L^{i+1} = L^i.L$
3. pozitivní iterace ...  $L^+ = L^1 \cup L^2 \dots \bigcup_{i \geq 1} L^i$
4. obecná iterace ...  $L^* = L^0 \cup L^1 \cup \dots = \bigcup_{i \geq 0} L^i$ , tedy  $L^* = L^+ \cup \lambda$
5. otočení jazyka ...  $L^R = \{u^R : u \in L\}$
6. levý kvocient  $L$  podle  $M$  ...  $M$   
 $L = \{u : uv \in L \& u \in M\}$

7. levá derivace  $L$  podle  $w \dots \partial_w L = \{w\}$   
 $L$

8. pravý kvocient  $L$  podle  $M \dots L/M = \{u : uv \in L \& v \in M\}$

9. pravá derivace  $L$  podle  $w \dots \partial_w^R L = L/\{w\}$ .

**Věta:** Jsou-li  $L, M$  regulární jazyky, je regulární i  $L.M, L^*, L^+, L^R, M$   
 $LaL/M$ .

**Věta:** Jsou-li  $L, M$  regulární jazyky, je regulární i  $L.M$ .

**Důkaz:** Vezmeme DFA  $A_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ , pak  $A_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$  tak, že  $L = L(A_1)$  a  $M = L(A_2)$ .

Definujeme Nedeterministický automat  $B = (Q \cup q_0, \Sigma, \delta, q_0, F_2)$  kde:

$Q = Q_1 \cup Q_2$ předpokládáme různá jména stavů, jinak přejmenujeme, končíme až po přečtení slova z $L_2$			
$\delta(q_0, \lambda)$	$= q_1, q_2$	pro $q_1 \in F_1$	$t.j. \lambda \in L(A_1)$
$\delta(q_0, \lambda)$	$= q_1$	pro $q_1 \notin F_1$	$t.j. \lambda \notin L(A_1)$
$\delta(q_0, x)$	$= \emptyset$	pro $x \in \Sigma$	
$\delta(q, x)$	$= \delta_1(q, x)$	pro $q \in Q_1 \& \delta_1(q, x) \notin F_1$	počítáme v $A_1$
	$= \delta_1(q, x), q_2$	pro $q \in Q_1 \& \delta_1(q, x) \in F_1$	nedet. přechod z $A_1$
	$= \delta_2(q, x)$	pro $q \in Q_2$	počítáme v $A_2$

Pak  $L(B) = L(A_1).L(A_2)$ .

I have zero idea how this should be formatted properly, pls fix later. - 3O11

□

## 4 Čtvrtá přednáška

**Věta** (Lemma( $L^*, L^+$ )): Je-li  $L$  regulární jazyk, je regulární i  $L^*, L^+$ .

1. Idea: Opakovaný výpočet automatu  $A = (Q, \Sigma, \delta, q_0, F)$
2. Realizace: nedeterministické rozhodnutí, zda pokračovat nebo restart
3. speciální stav pro příjem  $\lambda \in L^0$  (pro  $L^+$  vynecháme či  $\notin F$ ).

**Důkaz:** Vezmeme DFA  $A = (Q, \Sigma, \delta, q_0, F)$  tak, že  $L = L(A)$ . Definujeme NFA automat  $B = (Q \cup q_B, \Sigma, \delta_B, q_B, F \cup q_B)$ , kde:

$\delta_B(q_B, \lambda) = q_0$  nový stav  $q_B$  pro příjem  $\lambda$  přejdeme do  $q_0$

$\delta_B(q_B, x) = \emptyset$  pro  $x \in \Sigma$

$\delta_B(q, x) = \delta(q, x)$  pokud  $q \in Q \& \delta(q, x) \notin F$  uvnitř  $A$

$= \delta(q, x), q_0$  pokud  $q \in Q \& \delta(q, x) \in F$  možný restart

Pak  $L(B) = L(A)^*(q_B \in F_B), L(B) = L(A^+)(q_B \notin F_B)$ .

□

**Věta** (Lemma( $L^R$ )): Je-li  $L$  regulární jazyk, je regulární i  $L^R$ .

1. Zřejmě  $(L^R)^R = L$  a tedy stačí ukázat jeden směr.
2. idea: obrátíme šipky ve stavovém diagramu; nedeterministický FA

**Důkaz:** TO DOOT, 67. slide

□

**Věta** (Lemma ( $M \setminus LaL/M$ )): Jsou-li  $L, M$  regulární jazyky, je regulární i  $M \setminus L$  a  $M/L$ .

1. idea:  $A_L$  budeme startovat ve stavech, do kterých se lze dostat slovem  $M$ .

**Důkaz:**

1.  $v \in M \setminus L$
2.  $\Leftrightarrow (\exists u \in M) uv \in L$
3.  $\Leftrightarrow (\exists u \in M, \exists q \in Q) \delta(q_0, u) \& \delta(q, v) \in F$
4.  $\Leftrightarrow \exists q \in S_0 \& \delta(q, v) \in F$
5.  $\Leftrightarrow v \in L(B)$

Vezmeme DFA  $A = (Q, \Sigma, \delta, q_0, F)$  tak, že  $L = L(A)$ .

**TO DOOT, nestihol som**

□

**Definice** (Regulární výrazy): Regulární výrazy (RV) jsou:

1. algebraický popis jazyků
2. deklarativním způsobem, jak vyjádřit slova, která chceme přijímat.
3. Schopné definovat všechny a pouze regulární jazyky.
4. Můžeme je brát jako programovací jazyk, uživatelsky přívětivý popis konečného automatu
5. Syntaktická analýza potřebuje silnější nástroj, bezkontextové gramatiky, budou následovat

Regulární výrazy  $\alpha, \beta \in \text{RegE}(\Sigma)$  nad konečnou neprázdnou abecedou  $\Sigma = \{x_1, x_2, \dots, x_n\}$  a jejich hodnota  $L(\alpha)$  jsou definovány induktivně:

**TO DOOT, je tam tabuľka na ktorú pri tejto rýchlosti nemám čas prepísať správne, slide 70.**

**Definice** (Priorita): Nejvyšší prioritu má iterace  $*$ , nižší konkatenace (zřetězení), nejnižší sjednocení  $+$ .

**Věta** (Kleeneho věta (varianta)): Každý jazyk reprezentovaný konečným automatem lze zapsat jako regulární výraz.

*Každý jazyk popsany regulárním výrazem můžeme zapsat jako  $\lambda$ -NFA (a tedy i DFA).*

**Důkaz:** Převod RegE výrazu na  $\lambda$ -NFA automat.

Důkaz indukcí dle struktury R. Základ:

V každém kroku zkonstruujeme  $\lambda$ -NFA rozpoznávající stejný jazyk  $L(R) = L(E)$  se třemi dalšími vlastnostmi:

1. Právě jeden přijímající stav,
2. Žádné hrany do počátečního stavu,
3. Žádné hrany z koncového stavu.

**Asi je nutné doplniť obrázok, keďže aj na prednáške to bolo popísané obrázkom (slide 72)**

□

**Definice** (Regulární výraz z DFA): Mějme DFA  $AQ_A = \{1, \dots, n\}$  o  $n$  stavech.

Nechť  $R^{(k)}_{ij}$  je regulární výraz,  $L(R^{(k)}_{ij}) = \{w : \delta_{\leq k}^*(i, w) = j\}$  množina slov převádějících stav  $i$  do stavu  $j$  a  $A$  cestou, která neobsahuje stav s vyšším indexem než  $k$ .

Budeme rekurzivně konstruovat  $R^{(k)}_{ij}$  pro  $k = 0, \dots, n$ .

$k = 0, i \neq j : R^{(0)}_{ij} = \mathbf{a_1} + \mathbf{a_2} + \dots + \mathbf{a_m}$ , kde  $a_1, \dots, a_m$  jsou symboly označující hrany  $i$  do  $j$  (nebo  $R^{(0)}_{ij} = \emptyset$  nebo  $R^{(0)}_{ij} = \mathbf{a}$  pro  $m = 0, 1$ ).

$k = 0, i = j : \text{smyčky}, R^{(0)}_{ij} = \lambda + \mathbf{a_1} + \mathbf{a_2} + \dots + \mathbf{a_m}$ , kde  $a_1, a_2, \dots, a_m$  jsou symboly na smyčkách v  $i$ .

**Důkaz: TODOOT, celý slide je závislý na obrázku ... 75slide** □

**Poznámka** (Shrnutí převodu mezi reprezentacemi regulárních jazyků): **TODOO, takisto obrázok**  
Převod NFA na DFA

1.  $\lambda$  uzávěr v  $O(n^3)$  - prohledává  $n$  stavů násobeno  $n^2$  hran pro  $\lambda$  přechody.
2. Podmnožinová konstrukce, DFA s až  $2^n$  stavy. Pro každý stav,  $O(n^3)$  času na výpočet přechodové funkce.

Převod DFA na NFA

1. Přidat množinové závorky k přechodové funkci a přechody pro  $\lambda$  u  $\lambda$ -NFA.

Převod automatu DFA na RegE regulární výraz

1.  $O(n^3 4^n)$

RegE výraz na automat

1. V čase  $O(n)$  vytvoříme  $\lambda$ -NFA.

**Definice** (Substituce jazyků): Mějme konečnou abecedu  $\Sigma$ . Pro každé  $x \in \Sigma$  budiž  $\sigma(x)$  jazyk v nějaké abecedě  $Y_X$ . Dále položíme:

1.  $\sigma(\lambda) = \lambda$
2.  $\sigma(u.v) = \sigma(u).\sigma(v)$

Zobrazení  $\sigma : \Sigma \rightarrow P(Y^*)$ , kde  $Y = \bigcup_{x \in \Sigma} Y_X$  se nazývá substituce.

$\sigma(L) = \bigcup_{w \in L} \sigma(w)$

nevypouštějící substituce je substituce, kde žádné  $\sigma(x)$  neobsahuje  $\lambda$ .

**Definice** (Homomorfismus (jazyků), inverzní homomorfismus): homomorfismus  $h$  je speciální případ substituce, kde obraz je vždy jen jednoslovný jazyk (vynecháváme u něj závorky), t.j.  $(\forall x \in \Sigma) h(x) = w_x$ .

Pokud  $\forall x : w_x \neq \lambda$ , jde o nevypouštějící homomorfismus.

Inverzní homomorfismus  $h^{-1}(L) = \{w : h(w) \in L\}$ .

**Věta** (Uzavřenost na homomorfismus): *Je-li jazyk  $L$   $\forall x \in \Sigma$  jazyk  $\sigma(x), h(x)$  regulární, pak je regulární i  $\sigma(L), h(L)$ .*

**Důkaz:** Strukturální indukci „probubláváním“ algebraickým popisem jazyka o základních, sjednocení, zřetězení a iterace. Tvrzení:  $\sigma(L(E)) = L(\sigma(E))$ .  $\sigma(\lambda) = \lambda$ ,  $\sigma(\emptyset) = \emptyset$ ,  $\sigma(x) = \sigma(x)$ ,  $\sigma(L(\alpha + \beta)) = L(\sigma(\alpha) + \sigma(\beta))$  atd.

**TODOOT, 84 slide** □

**Definice** (Inverzní homomorfismus): Necht  $h$  je homomorfismus abecedy  $\Sigma$  do slov nad abecedou  $T$ . Pak  $h^{-1}(L)$ , 'h inverze L' je množina řetězců

$$h^{-1}(L) = \{w : w \in \Sigma^*; h(w) \in L\}.$$

**Věta:** *Je-li  $h$  homomorfismus abecedy  $\Sigma$  do abecedy  $T$  a  $L$  je regulární jazyk abecedy  $T$ , pak  $h^{-1}(L)$  je také regulární jazyk.*

**Důkaz:** Mějme DFA  $A = (Q, T, \delta, q_0, F)$  pro  $L$ .

Konstruujeme DFA pro  $h^{-1}(L)$ .

1. Definujeme  $B(Q, \Sigma, \gamma, q_0, F)$  kde  $\gamma(qq, a) = \delta^*(q, h(a))$  ( $\delta^*$  operace na řetězcích).
2. Indukcí dle  $|w|$ ,  $\gamma^*(q_0, w) = \delta^*(q_0, h(w))$ .
3. Proto  $B$  přijíma právě řetězce  $w \in h^{-1}(L)$ .

□

## 5 Pátá přednáška

**Příklad:** Necht'  $A = (Q, \Sigma, \delta, q_0, F)$  je DFA. Definujeme jazyk  $L = \{w \in \Sigma^*; \delta^*(q_0, w) \in F\}$  a pro každý stav  $q \in Q$  existuje prefix  $x_q$  slova  $w$  tak, že  $\delta^*(q_0, x_q) = q$ .

Tento jazyk  $L$  je regulární.

1.  $M$  označme  $M = L(A)$ .
2.  $T$  definujeme novou abecedu  $T$  trojic  $\{[pq]; p, q \in Q, a \in \Sigma, \delta(p, a) = q\}$ .
3.  $h$  definujeme homomorfizmus  $(\forall p, q, a)h([pq]) = a$ .
4.  $L_1$  Jazyk  $L = h^{-1}(M)$  je regulární, protože  $M$  je regulární (DFA inverzní homomorfizmus).
5.  $h^{-1}(101)$  obsahuje  $2^3 = 8$  řetězců, např.

$$[p1p][q0q][p1p] \in [p1p], [q1q][p0q], [q0q][p1p], [q1q].$$

$L_2$  Vynutíme začátek  $q_0$ . Definujeme:

$$E_1 = \bigcup_{a \in \Sigma, q \in Q} [q_0 a q] =$$

$$E_1 = [q_0 a_1 q_0], [q_0 a_2 q_1], \dots, [q_0 a_m q_n]$$

Pak  $L_2 = L_1 \cap L(E_1, T^*)$ .

$L_3$  Vynutíme stejné sousedící stavy. Definujeme ne-odpovídající dvojice:

$$E_2 = \bigcup_{q \neq r, p, q, r, s \in Q, a, b \in \Sigma} [paq][rbs].$$

Definujeme  $L_3 = L_2 - L(T^*.E_2.T^*)$ .

Končí v přijímajícím stavu, protože jsme začali v jazyku  $M$  přijímaném DFA  $A$ .

$L_4$  Všechny stavy.  $\forall q \in Q$  definujeme  $E_q$  jako regulární výraz sjednocení všech symbolů  $T$  takových, že  $q$  není ani na první, ani na poslední pozici. Odečteme  $L(E_q^*)$  od  $L_3$ .  $L_4 = L_3 - \bigcup_{q \in Q} E_q^*$ .

$L$  Odstráníme stavy, necháme symboly.  $L = h(L_4)$ , tedy  $L$  je regulární.

**Věta:** Lze algoritmicky rozhodnout, zda jazyk přijímaný DFA, NFA,  $\lambda$  – NFA je prázdný.

Jazyk je prázdný právě když žádný z koncových stavů není dosažitelný. Dosažitelnost lze testovat  $O(n^2)$ .

**Věta:** Pro daný řetězec  $w : |w| = n$  a regulární jazyk  $L$  lze algoritmicky rozhodnout, zda je  $w \in L$ .

**Důkaz:**

1. DFA: Spust' automat, pokud  $|w| = n$ , při dobré reprezentaci a konstantním čase přechodu  $O(n)$ .
2. NFA o  $s$  stavech: čas  $O(ns^2)$ . Každý vstupní symbol aplikujeme na všechny
3. DOPLNIT, NESTIHAM

□

**Definice** (Algebraický popis jazyků): Pro konečnou neprázdnou abecedu  $\Sigma$  označme  $RJ(\Sigma)$  nejmenší třídu jazyků, která:

1. obsahuje prázdný jazyk  $\emptyset$ ,

2. pro každé ...

### 3. NESTIHAM

**Věta (Kleene):** *NESTIHAM* Aight, please fill this in later.

Dvousměrné (dvoucestné) konečné automaty  
Konečný automat provádí následující činnosti:

1. přečte písmeno
  2. změní stav vnitřní jednotky
  3. posune čtecí hlavu doprava
- Čtecí hlava se nesmí vracet.

**Definice (Dvousměrný konečný automat):** Dvousměrným (dvoucestným) konečným automatem nazýváme pěticí  $A = (Q, \Sigma, \delta, q_0, F)$ , kde

1.  $Q$  je konečná množina stavů,
2.  $\Sigma$  je konečná množina vstupních symbolů přechodové funkce  $\delta$  je zobrazení  $Q \times \Sigma \rightarrow Q \times -1, 1$  rozšířené o pohyb hlavy
3.  $q_0 \in Q$  počáteční stav,
4. množina přijímajících stavů  $F \subseteq Q$ .

**Poznámka:** Je deterministický, nedeterministický zavádět nebudeme.

**Poznámka:** Nulový pohyb hlavy lze, jen trochu zkomplikuje důkaz dále.

**Definice:** Slovo  $w$  je přijato dvousměrným konečným automatem, pokud:

1. výpočet začal na prvním písmenu slova  $w$  vlevo v počátečním stavu,
2. čtecí hlava poprvé opustila slovo  $w$  vpravo v některém přijímajícím stavu,
3. mimo čtené slovo není výpočet definován (výpočet zde končí a slovo není přijato).

**Poznámka:** 1. Ke slovům si můžeme přidat speciální koncové znaky  $\# \notin \Sigma$

2. funkce  $\partial_{\#}$  odstraní  $\#$  zleva,  $\partial_{\#}^R$  zprava.
3. Je-li  $L(A) = \#w\# | w \in L \subseteq \Sigma^*$  regulární, potom i  $L$  je regulární
4.  $L = \partial_{\#} \dots$  **DOPLNIT**

**Věta:** Jazyky přijímané dvousměrným konečným automatem jsou právě regulární jazyky.

**Důkaz:** konečný automat  $\rightarrow$  dvousměrný automat  
**NESTIHOL SOM**

□

**Věta:** Algoritmus: Funkce  $f_u$  popisující výpočet 2DFA nad slovem  $u$

Definujeme funkci  $f_u : Q \cup q_0^{\downarrow} \rightarrow Q \cup 0$

1.  $f_u(q_0^{\downarrow})$  popisuje v jakém stavu poprvé odejdeme vpravo, pokud začneme výpočet vlevo v počátečním stavu  $q_0$ ,



2.  $f_u(p) : p \in Q$  v jakém stavu opět odejdeme vpravo, pokud začneme výpočet vpravo v  $p$ ,
3. symbol 0 značí, že daná situace nenastane (odejdeme vlevo nebo cyklus)
4. Definujeme ekvivalenci slov následovně:  $u \sim w \Leftrightarrow_d eff_u = f_w$

**Poznámka:** t.j. slova jsou ekvivalentní, pokud mají stejné 'výpočtové' funkce

*Regulárnost 2DFA*

### NESTIHOLOU SOM

#### Důkaz:

1. Potřebujeme převést návraty na lineární výpočet
2. Zajímají nás jen přijímající výpočty
3. Díváme se na řezy mezi symboly (v jakém stavu přechází na další políčko)

Pozorování:

1. stavy se v přechodu řezu střídají (doprava, doleva)
2. první stav jde doprava, poslední také doprava
3. v deterministických přijímajících výpočtech nejsou cykly
4. první a poslední řez obsahují jediný stav

Formální převod 2DFA na NFA

Nechť  $A = (Q, \Sigma, \delta, q_0, F)$  je dvousměrný (deterministický) konečný automat. Definujeme ekvivalentní nedeterministický automat  $B = (Q^l, \Sigma, \delta^l, (q_0), F^l)$ , kde:

1.  $Q^l$  jsou všechny korektní přechodové posloupnosti
  - (a) posloupnosti stavů  $(q^1, \dots, q^k) : q^i \in Q$
  - (b) délka posloupnosti je lichá ( $k = 2m + 1$ )
  - (c) žádný stav se neopakuje na liché ani sudé pozici ( $\forall i \neq j)(q^{2i} \neq q^{2j}) \& (\forall i \neq j)(q^{2i+1} \neq q^{2j+1})$ )
2.  $F^l = \{q \mid q \in F \text{ posloupnosti délky } 1\}$
3.  $\delta^l(c, a) = \{d \mid d \in Q^l \& c \xrightarrow{a} d \text{ je lokálně konzistentní přechod pro } a \}$ 
  - (a) existuje bijekce :  $h : c_{\text{odd}} \cup d_{\text{even}} \rightarrow c_{\text{even}} \cup d_{\text{odd}}$ , tak, že:
  - (b) pro  $h(q) \in c_{\text{even}}$  je  $(h(q), -1) = \delta(q, a)$
  - (c) pro  $h(q) \in d_{\text{odd}}$  je  $(h(q), +1) = \delta(q, a)$

□

Automaty s výstupem (motivace)

1. Dosud jediná zpráva z automatu: 'Jsme v přijímajícím stavu'.
2. Můžeme z FA získat více informací? Můžeme ... **NESTIHAM**

**Definice** (Mooreův stroj): **NESTIHOLOU SOM**

**Příklad** (Mooreův stroj pro tenis): Mooreův stroj pro počítání tenisového skóre.

1. Vstupní abeceda: ID hráče, který uhrál bod
2. Výstupní abeceda & stavy: skóre (t. j.  $Q = Y, \mu(q) = q$ )

**Doplň** obrázok z prednášok (98. slide)

**Definice** (Mealyho stroj): Mealyho (sekvenčním) strojem nazýváme šestici  $A = (Q, \Sigma, Y, \delta, \lambda_M, q_0)$ , resp. pěťici  $A = (Q, \Sigma, Y, \delta, \lambda_M)$ , kde

1.  $Q$  je konečná neprázdná množina stavů
2.  $\Sigma$  je konečná neprázdná množina symbolů (vstupní abeceda)
3.  $Y$  je konečná neprázdná množina symbolů (výstupní abeceda)
4. **NESTIHAM**

**TU BOL TIEZ NEJAKY TEXT ESTE**

**Příklad** (Mealyho stroj): Mealyho stroj - automat, který dělí vstupní slovo v binárním tvaru číslem 8 (celočíslně).

1. Posun o tři bity doprava
2. potřebujeme si pamatovat poslední trojici bitů
3. vlastně tříbitová dynamická paměť

I když nevíme, kde automat startuje, po třech symbolech začne počítat správně.

**Věta** (Převod Mealyho stroje na Mooreův): *Pro každý Mealyho stroj existuje Mooreův stroj převádějící každé vstupní slovo na stejné výstupní slovo.*

**Důkaz:** Sestrojíme Mooreův stroj  $B$  tak, aby  $\forall q, w$  **NESTIHOLO SOM** :) □

## 6 Šestá přednáška

**Definice** (Palindromy): Palindrom je řetězec  $w$  stejný při čtení zepředu i zezadu, tj.  $w = w^R$ .

**Věta:** *Jazyk  $L_{pal} = \{w : w = w^R, w \in \Sigma^*\}$  není regulární.*

**Důkaz:** Sporem. Předpokládejme  $L_{pal}$  je regulární, nechť  $n$  je konstanta z pumping lemma, uvažujme slovo  $w = 0^n 1 0^n$ .

Z pumping lemmatu lze rozložit  $w = xyz$ ,  $y$  obsahuje jednu nebo více z prvních  $n$  nul. Tedy  $xz$  má být v  $L_{pal}$  ale není, t.j. není regulární. □

### 6.1 Formální (generativní) gramatiky, bezkontextové gramatiky

**Definice:** Formální (generativní) gramatika je  $G = (V, T, P, S)$  složena z

1. konečné množiny **neterminálů** (variables)  $V$ ,
2. neprázdné konečné množiny **terminálních symbolů** (terminálů)  $T$ ,
3. **počáteční symbol**  $S \in V$ ,

4. konečné množiny **pravidel (produkcí)**  $P$  reprezentující rekurzivní definici jazyka. Každé pravidlo má tvar

$$\beta A_\gamma \rightarrow A \in V, \beta, \gamma, \omega \in (V \cup B)^*$$

t. j. levá strana obsahuje aspoň jeden neterminální symbol.

**Definice** (Bezkontextová gramatika (CFG)): je  $G = (V, T, P, S)$  gramatika, obsahující pouze pravidla tvaru

$$A \rightarrow \omega, A \in V, \omega \in (V \cup T)^*$$

**Definice** (Chomského hierarchie): **CHOMSKÉHO HIERARCHIE, DOPLNIT, 112/113**

**Definice** (Derivace  $\Rightarrow^*$ ): Mějme gramatiku  $G = (V, T, P, S)$ .

1. Říkáme, že  $\alpha$  se **přímo přepíše** na  $\omega$  (píšeme  $\alpha \Rightarrow_G \omega$  nebo  $\alpha \Rightarrow \omega$ ) jestliže

$$\exists \beta, \gamma, \eta, \nu \in (V \cup T)^* : \alpha = \eta \beta \nu, \omega = \eta \gamma \nu \& (\beta \rightarrow \gamma) \in P$$

2. Říkáme, že  $\alpha$  se přepíše na  $\omega$  (píšeme  $\alpha \Rightarrow^* \omega$ ) jestliže

$$\exists \beta_1, \dots, \beta_n \in (V \cup T)^* : \alpha = \beta_1 \Rightarrow \beta_2 \Rightarrow \dots \Rightarrow \beta_n = \omega$$

t. j.  $\alpha \Rightarrow^* \alpha$ .

3. posloupnost  $\beta_1, \dots, \beta_n$  nazýváme derivací (odvození)

4. pokud  $\forall i \neq j : \beta_i \neq \beta_j$ , hovoříme o minimálním odvození

**Definice** (Jazyk generovaný gramatikou  $G$ ): Jazyk  $L(G)$  gramatiky  $G = (V, T, P, S)$  je množina terminálních řetězců, pro které existuje derivace ze startovního symbolu

$$L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$$

Jazyk neterminálu  $A \in V$  definujeme  $L(A) = \{w \in T^* \mid A \Rightarrow_G^* w\}$ .

**Definice** (Gramatika typu 3, pravá lineární): Gramatika  $G$  je **pravá lineární, t.j. Typu 3** pokud obsahuje pouze pravidla tvaru

$$A \rightarrow wB, A \rightarrow w, A, B \in V, w \in T^*.$$

**Příklad:**

1.  $P = \{S \rightarrow 0S, 1A \mid \lambda, A \rightarrow 0A \mid 1B, B \rightarrow 0B \mid 1S\}$
2.  $S \Rightarrow 0S \Rightarrow 01A \Rightarrow 011B \Rightarrow 0110B \Rightarrow 01101S \Rightarrow 01101$

**Věta:**  $L \in RE \Rightarrow L \in \mathcal{L}_3$

*Pro každý jazyk rozpínavý konečným automatem existuje gramatika typu 3, která ho generuje.*

**Důkaz:**

1.  $L = L(A)$  pro deterministický konečný automat  $A$ .
2. definujeme gramatiku  $G = (Q, \Sigma, P, q_0)$ , kde pravidla  $P$  mají tvar

$$p \rightarrow aq \text{ když } \delta(p, a) = q$$

$$p \rightarrow \lambda \text{ když } p \in F$$

3. je  $L(A) = L(G)$ ?

$$(a) \lambda \in L(A) \Leftrightarrow q_0 \in F \Leftrightarrow (q_0 \rightarrow \lambda) \in P \Leftrightarrow \lambda \in L(G)$$

$$(b) a_1 \dots a_n \in L(A) \Leftrightarrow \exists q_0, \dots, q_n \in Q : \delta(q_i, a_{i+1}) = q_{i+1}, q_n \in F \Leftrightarrow (q_0 \xRightarrow{a_1} q_1 \xRightarrow{a_2} \dots \xRightarrow{a_n} q_n) \text{ je derivace pro } a_1, \dots, a_n \Leftrightarrow a_1, \dots, a_n \in L(G)$$

□

Příprava převodu gramatiky typu 3 na forall

1. Opačný směr

- (a) pravidla  $A \rightarrow aB$  kódujeme do přechodové funkce
- (b) pravidla  $A \rightarrow \lambda$  určují koncové stavy
- (c) pravidla  $A \rightarrow a_1, \dots, a_n B, A \rightarrow a_1, \dots, a_n$  s více neterminály rozepíšeme
  - i. zavedeme nové neterminály  $Y_2, \dots, Y_n, Z_1, \dots, Z_N$
  - ii. vytvoříme pravidla  $A \rightarrow a_1 Y_2, Y_2 \rightarrow a_2 Y_3, \dots, Y_n \rightarrow a_n B$
  - iii. resp.  $Z \rightarrow a_1 Z_1, Z_1 \rightarrow a_2 Z_2, \dots, Z_{n-1} \rightarrow a_n Z_n, Z_n \rightarrow \lambda$
- (d) pravidla  $A \rightarrow B$  obpovídají  $\lambda$  přechodům
  - i. zbavíme se jich tranzitivním uzávěrem
  - ii. nebo musíme tranzitivně uzavřít  $S \rightarrow B$  pro hledání  $S \rightarrow \lambda$ .

**Věta:** Ke každé gramatice typu 3 existuje gramatika typu 3, která generuje stejný jazyk a obsahuje pouze pravidla ve tvaru:  $A \rightarrow aB, A \rightarrow \lambda, A, B \in V, a \in T$ .

**Důkaz:** Pro gramatiku  $G = (V, T, S, P)$  definujeme  $G^| = (V^|, T, S, P^|)$ , kde pro každé pravidlo zavedeme dostatečný počet nových neterminálů  $Y_2, \dots, Y_n, Z_1, \dots, Z_n$  a definujeme

**DOPLNIT TABULKU**

□

**Věta** ( $\lambda - NFA$  pro gramatiku typu 3 rozpoznávající stejný jazyk): Pro každý jazyk  $L$  generovaný gramatikou typu 3 existuje  $\lambda - NFA$  rozpoznávající  $L$ .

**Důkaz:**

1. Vezmeme  $G = (V, T, P, S)$  obsaující jen pravidla tvaru  $A \rightarrow aB, A \rightarrow \lambda, A, B \in V, a \in T$  generující  $L$  (předchozí lemma)
2. definujeme nedeterministický  $\lambda - NFA$   $A = (V, T, \delta, S, F)$  kde :
  - (a)  $F = \{A | (A \rightarrow \lambda) \in P\}$
  - (b)  $\delta(A, a) = \{B | (A \rightarrow aB) \in P\}$

3.  $L(G) = L(A)$

(a) **DOPLNIT, 121**

□

**Definice** (Levé (a pravé) lineární gramatiky): Gramatiky typu 3 nazýváme také **pravé lineární** (neterminál je vždy vpravo).

Gramatika  $G$  je levá lineární, jestliže má pouze pravidla tvaru  $A \rightarrow Bw, A \rightarrow w, A, B \in V, w \in T^*$ .

**Věta:** Jazyky generované levou lineární gramatikou jsou právě regulární jazyky.

**Důkaz:** **DOPLNIT, 122**

□

**Definice** (Lineární gramatika, jazyk): Gramatika je lineární, jestliže má pouze pravidla tvaru  $A \rightarrow uBw$ ,  $A \rightarrow w$ ,  $A, B \in V$ ,  $u, w \in T^*$  (na pravé straně vždy maximálně jeden neterminál).

Lineární jazyky jsou právě jazyky generované lineárními gramatikami.

1. Zřejmě platí: regulární jazyky  $\subseteq$  lineární jazyky,
2. Jde o vlastní podmnožinu  $\subsetneq$ .

**Příklad** (Lineární, neregulární jazyk): Jazyk  $L = \{0^n 1^n | n \geq 1\}$  není regulární jazyk, ale je lineární, generovaný gramatikou s pravidly  $S \rightarrow 0S1 | 01$ .

**Definice** (Bezkontextová gramatika): Bezkontextová gramatika je gramatika, kde všechna pravidla jsou tvaru  $A \rightarrow \omega$ ,  $\omega \in (V \cup T)^*$

CFG pro jednoduché výrazy

1.  $E \rightarrow I$
2.  $E \rightarrow E + E$
3.  $E \rightarrow E * E$
4.  $E \rightarrow (E)$
5.  $I \rightarrow a$
6.  $I \rightarrow b$
7.  $I \rightarrow Ia$
8.  $I \rightarrow Ib$
9.  $I \rightarrow I0$
10.  $I \rightarrow I1$

Pravidla 1 až 4 definují výraz.

Pravidla 5 až 10 definují identifikátor  $I$  odpovídající regulárnímu výrazu  $(a + b)(a + b + 0 + 1)^*$ .

**Definice** (Derivační strom): Mějme gramatiku  $G = (V, T, P, S)$ . Derivační strom pro  $G$  je strom, kde

1. Kořen (kreslíme nahoře) je označen startovním symbolem  $S$ ,
2. každý vnitřní uzel je ohodnocen neterminálem  $V$ .
3. Každý uzel je ohodnocen prvkem  $\in V \cup T \cup \lambda$ .
4. Je-li uzel ohodnocen  $\lambda$ , je jediným dítětem svého rodiče.
5. Je-li  $A$  ohodnocení vrcholu a jeho děti *zleva pořadě* jsou ohodnoceny  $X_1, \dots, X_k$ , pak  $(A \rightarrow X_1, \dots, X_k) \in P$  je pravidlo gramatiky.

**Definice** (Strom dává slovo (yield)): Říkáme, že derivační strom dává slovo  $w$  (yield), jestliže  $w$  je slovo složené z ohodnocení listů bráno zleva doprava.

**Definice** (Levá a pravá derivace): 1. Levá derivace (leftmost)  $\Rightarrow_{lm}, \Rightarrow_{lm}^*$  v každém kroku přepisuje nejlevější neterminál.

2. Pravá derivace (rightmost)  $\Rightarrow_{rm}, \Rightarrow_{rm}^*$  v každém kroku přepisuje nejpravější neterminál.

**Věta:** Pro danou gramatiku  $G = (V, T, P, S)$  a  $w \in T^*$  jsou následující tvrzení ekvivalentní:

1.  $A \Rightarrow^* w$ .
2.  $A \Rightarrow_{lm}^* w$ .
3.  $A \Rightarrow_{rm}^* w$ .
4. Existuje derivační strom s kořenem  $A$  dávající slovo  $w$ .

**Věta:** Mějme  $CFG G = (V, T, P, S)$  a derivační strom s kořenem  $A$  dávající slovo  $w \in T^*$ . Pak existuje levá derivace  $A \Rightarrow_{lm}^* w$  v  $G$ .

**Tvrzení** (Příprava důkazu, „obalení derivace“): Mějme následující derivaci:

$$E \Rightarrow I \Rightarrow Ib \rightarrow ab.$$

**Důkaz:** Pro libovolná slova  $\alpha, \beta \in (V \cup T)^*$  je také derivace:

$$\alpha E \beta \Rightarrow \alpha I \beta \Rightarrow \alpha Ib \beta \Rightarrow \alpha ab \beta.$$

□

**Důkaz:**  $\exists$  derivační strom, pak  $\exists$  levá derivace  $\Rightarrow_{lm}$

Indukcí podle výšky stromu:

1. Základ: výška 1: Kořen  $A$  s dětmi dávajícími  $w$ . Je to derivační strom, proto  $A \rightarrow w$  je pravidlo  $\in P$ , tedy  $A \Rightarrow_{lm} w$  v jednom kroku.
2. Indukce: výška  $n > 1$ . Kořen  $A$  s dětmi  $X_1, X_2, \dots, X_k$ .
  - (a) Je-li  $X_i \in T$ , definujeme  $w_i \equiv X_i$ .
  - (b) Je-li  $X_i \in V$ , z indukčního předpokladu  $X_i \Rightarrow_{lm}^* w_i$ .

Levou derivaci konstruujeme induktivně pro  $i = 1, \dots, k$  složíme  $A \Rightarrow_{lm}^* w_1 w_2 \dots w_i X_{i+1} X_{i+2} \dots X_k$ .

- (a) Pro  $X_i \in T$  jen zavedeme čítač  $i++$ .
- (b) Pro  $X_i \in V$  prepíšeme derivaci:  $X_i \Rightarrow_{lm} \alpha_1 \Rightarrow_{lm} \alpha_2 \dots \Rightarrow_{lm} w_i n a$

$$w_1 w_2 \dots w_{i-1} X_i X_{i+1} X_{i+2} \dots X_k \Rightarrow_{lm}$$

$$w_1 w_2 \dots w_{i-1} \alpha_1 X_{i+1} X_{i+2} \dots X_k \Rightarrow_{lm}$$

$$\Rightarrow_{lm} w_1 w_2 \dots w_{i-1} w_i X_{i+1} X_{i+2} \dots X_k$$

Pro  $i = k$  dostaneme levou derivaci  $w \in A$ .

□

## 7 Sedmá přednáška

### 7.1 Zásobníkový automat

Zásobníkové automaty jsou rozšířením  $\lambda$ - $NFA$  nedeterministických konečných automatů s  $\lambda$  přechody.

Přidanou věcí je zásobník. Ze zásobníku můžeme číst, přidávat na vrch a odebírat z vrchu zásobníku znak  $\in \Gamma$ .

Může si pamatovat neomezené množství informace.

Zásobníkové automaty definují bezkontextové jazyky.

Deterministické zásobníkové automaty přijímají jen vlastní podmnožinu bezkontextových jazyků.

**Definice** (Zásobníkový automat): (PDA) je  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ , kde

1.  $Q$  konečná množina stavů,
2.  $\Sigma$  neprázdná konečná množina vstupních symbolů,
3.  $\Gamma$  neprázdná konečná zásobníková abeceda,
4.  $\delta$  přechodová funkce :  $Q \times (\Sigma \cup \lambda) \times \Gamma \rightarrow P_{FIN}(Q \times \Gamma^*)$ ,  $\delta(p, a, X) \ni (q, \gamma)$  kde  $q$  je nový stav a  $\gamma$  je řetězec zásobníkových symbolů, který nahradí  $X$  na vrcholu zásobníku,
5.  $q_0 \in Q$  počáteční stav
6.  $Z_0 \in \Gamma$  počáteční zásobníkový symbol. Víc na začátku zásobníku není.
7.  $F$  množina přijímajících (koncových) stavů; může být nedefinovaná.

V jednom časovém kroku zásobníkový automat:

1. Přečte na vstupu žádný nebo jeden symbol ( $\lambda$  přechody pro prázdný vstup.)
2. Přejde do nového stavu.
3. Nahradí symbol na vrchu zásobníku libovolným řetězcem ( $\lambda$  odpovídá samotnému *pop*, jinak následuje *push* jednoho nebo více symbolů).

**Příklad** (Zásobníkový automat pro jazyk:  $L_{ww^R} = ww^R : w \in (0 + 1)^*$ ): PDA přijímající  $L_{ww^R}$ :

1. Start  $q_0$  reprezentuje odhad, že ještě nejsme uprostřed.
2. V každém kroku nedeterministicky hádáme:
  - (a) Zůstat  $q_0$  (ještě nejsme uprostřed)
  - (b) Přejít  $\lambda$  přechodem do  $q_1$  (už jsme viděli střed).
3. V  $q_0$  přečte vstupní symbol a dá (push) ho na zásobník
4. V  $q_1$  srovná vstupní symbol s vrcholem zásobníku. Pokud se shodují, přečte vstupní symbol a umaže (pop) vrchol zásobníku
5. Když vyprázdníme zásobník, přijmeme vstup, který jsme doteď přečetli.

**Příklad** (PDA pro  $L_{ww^R}$ ): **TO DO, (slide 143, example 7.2)**

**Definice** (Přechodový diagram pro zásobníkový automat): Přechodový diagram pro zásobníkový automat obsahuje:

1. Uzly, které odpovídají stavům  $PDA$ .

2. Šipka 'odnikud' ukazuje počáteční stav, dvojité kruhy označují přijímající stavy.
3. Hrana odpovídá přechodu PDA. Hrana označená  $a, X \rightarrow \alpha$  ze stavu  $p$  do  $q$  znamená  $\delta(p, a, X) \ni (q, \alpha)$
4. Konvence je, že počáteční symbol zásobníku značíme  $Z_0$ .

**Definice** (Notace zásobníkových automatů):

$a, b, c, *, +, 1, (, )$	symboly vstupní abecedy
$p, q, r$	stavy
$u, v, w, x, y, z$	řetězce vstupní abecedy
$X, Y, E, I, S$	zásobníkové symboly
$\alpha, \beta, \gamma$	řetězce zásobníkových symbolů

**Definice** (Situace zásobníkového automatu): Situaci zásobníkového automatu reprezentujeme trojicí  $(q, w, \gamma)$ , kde

1.  $q$  je stav,
2.  $w$  je zbývajících vstup a
3.  $\gamma$  je obsah zásobníku (vrch zásobníku je vlevo).

Situaci značíme zkratkou  $(ID)$  z anglického *instantaneous description*.

**Definice** ( $\vdash, \vdash^*$  posloupnosti situací): Mějme PDA  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ . Definujeme  $\vdash_P$  nebo  $\vdash$  následovně:

1. Nechť  $\delta(p, a, X) \ni (q, \alpha), p, q \in Q, a \in (\Sigma \cup \lambda), X \in \Gamma, \alpha \in \Gamma^*$ .

$$\forall w \in \Sigma^*, \beta \in \Gamma^* : (p, aw, X\beta) \vdash (q, w, \alpha\beta).$$

2. Symboly  $\vdash_P^*$  a  $\vdash^*$  používáme na označení nuly a více kroků zásobníkového automatu, t. j.

- (a)  $I \vdash^* I$  pro každou situaci  $I$
- (b)  $I \vdash^* J$  pokud existuje situace  $K : I \vdash K \& K \vdash^* J$ .

3. Čteme  $I \vdash^* J$  situace  $I$  vede na situaci  $J$ ,  $I \vdash J$  situace  $I$  bezprostředně vede na situaci  $J$ .

**Definice** (Jazyk přijímaný koncovým stavem, prázdným zásobníkem): Mějme zásobníkový automat  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ . Pak  $L(P)$  je:

$$L(P) = \{w : (q_0, w, Z_0) \vdash_P^* (q, \lambda, \alpha) \text{ pro nějaké } q \in F \text{ a libovolný řetězec } \alpha \in \Gamma^*; w \in \Sigma^*\}$$

jazyk akceptovaný prázdným zásobníkem  $N(P)$  definujeme:

$$L(P) = \{w : (q_0, w, Z_0) \vdash_P^* (q, \lambda, \lambda) \text{ pro nějaké } q \in Q ; w \in \Sigma^*\}$$

**Příklad:** Zásobníkový automat z předchozího příkladu akceptuje  $L_{wwr}$  koncovým stavem.

**Příklad:**  $P' \equiv P$  z předchozího příkladu, jen změníme instrukci, aby umazala poslední symbol

$$\delta(q_1, \lambda, Z_0) = (q_2, Z_0) \text{ nahradíme}$$

$$\delta(q_1, \lambda, Z_0) = (q_2, \lambda)$$

$$\text{ nyní } L(P') = N(P') = L_{wwr}.$$

**Příklad** (If-else přijímané prázdným zásobníkem): **TO DO, 7.5, 149 slide**



**Příklad (Přijímání koncovým stavem): TO DO, 7.6, 149 slide**

**Věta** (Nečtený vstup a dno zásobníku  $P$  neovlivní výpočet): *Mějme  $PDAP = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  a  $(p, x, \alpha) \vdash_P^* (q, y, \beta)$ . Potom pro libovolné slovo  $w \in \Sigma^*$  a  $\gamma \in \Gamma^*$  platí:  $(q, xw, \alpha\gamma) \vdash_P^* (q, yw, \beta\gamma)$ . Speciálně pro  $\gamma = \lambda$  a/nebo  $w = \lambda$ .*

**Důkaz:** Indukcí podle počtu situací mezi  $(p, xw, \alpha\gamma)$  a  $(q, yw, \beta\gamma)$ . Každý krok  $(p, x, \alpha) \vdash_P^* (q, y, \beta)$  je určen bez  $w$  a/nebo  $\gamma$ . Proto je možný i se symboly na konci vstupu / dně zásobníku.  $\square$

**Poznámka:**

$$\text{Pro } PDAP = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F), (p, xw, \alpha) \vdash_P^* (q, yw, \beta) : (p, x, \alpha) \vdash_P^* (q, y, \beta).$$

Pro zásobník ale obdoba neplatí. PDA může zásobníkové symboly  $\gamma$  použít a zase je tam naskládat (push).  $L = 0^i 1^i 0^j 1^j$ , situace  $(p, 0^{i-j} 1^i 0^j 1^j, 0^j Z_0) \vdash^* (q, 1^j, 0^j Z_0)$ , mezitím vyčistíme zásobník k  $Z_0$

**Věta** (Od přijímajícího stavu k prázdnému zásobníku): *Mějme  $L = L(P_F)$  pro nějaký PDA.*

$P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$ . *Pak existuje  $PDAP_N$  takový, že  $L = L(P_N)$ .*

**Důkaz:** *Nechť  $P_N = (Q \cup p_0, p, \Sigma, \Gamma \cup X_0, \delta_N, p_0, X_0)$ , kde:*

1.  $\delta_N(p_0, \lambda, X_0) = (q, Z_0 X_0)$  start,
2.  $\forall (q \in Q, a \in \Sigma \cup \lambda, Y \in \Gamma) : \delta_N(q, a, Y) = \delta_F(q, a, Y)$  simulujeme
3.  $\forall (q \in F, Y \in \Gamma \cup X_0), \delta_N(q, \lambda, Y) \ni (p, \lambda)$  přijmout pokud  $P_F$  přijíma,
4.  $\forall (Y \in \Gamma \cup X_0), \delta_N(p, \lambda, Y) = (p, \lambda)$  vyprázdníme zásobník.

*Pak  $w \in N(P_N) \Leftrightarrow w \in L(P_F)$ .*  $\square$

**Věta** (Od prázdného zásobníku ke koncovému stavu): *Pokud  $L = N(P_N)$  pro nějaký PDA  $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$ , pak existuje PDA  $P_F : L = L(P_F)$ .*

**Důkaz:**

$$P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, p_f)$$

kde  $\delta_F$  je:

1.  $\delta_F(p_0, \lambda, X_0) = (q_0, Z_0 X_0)$  start
2.  $\forall (q \in Q, a \in \Sigma \cup \lambda, Y \in \Gamma), \delta_F(q, a, Y) = \delta_N(q, a, Y)$ .
3. Navíc,  $\delta_F(q, \lambda, X_0) \ni (p_f, \lambda) \forall q \in Q$ .

*Chceme ukázat, že  $w \in L(P_N) \Leftrightarrow w \in L(P_F)$ .*

1. (if)  $P_F$  přijíma následovně:  $(p_0, w, X_0) \vdash_{P_F} (q_0, w, Z_0 X_0) \vdash_{P_F=N_F}^* (q, \lambda, X_0) \vdash_{P_F} (p_f, \lambda, \lambda)$ .
2. (only if) Do  $p_f$  nelze dojít jinak než předchozím bodem.

$\square$

**Věta** ( $L(CFG), L(PDA), N(PDA)$ ): *Následující tvrzení jsou ekvivalentní:*

1. Jazyk  $L$  je bezkontextový, t.j. generovaný CFG,
2. Jazyk  $L$  je přijímaný nějakým zásobníkovým automatem koncovým stavem.
3. Jazyk  $L$  je přijímaný nějakým zásobníkovým automatem prázdným zásobníkem.

**Důkaz:** TO DO, doplnit graf z 153 (theorem 7.1)  $\square$

**Algoritmus** (Konstrukce  $PDA$  z  $CFG$ ): Mějme  $CFG$  gramatiku  $G = (V, T, P, S)$ . Konstruujeme  $PDA P = (q, T, V \cup T, \delta, q, S)$ .

1. Pro neterminály  $A \in V, \delta(q, \lambda, A) = (q, \beta) : A \rightarrow \beta$  je pravidlo  $G$ .
2. Pro každý terminál  $a \in T, \delta(q, a, a) = (q, \lambda)$ .

**Příklad** (Konverze gramatiky): **TODO, strana 154, example 7.7**

**Věta** (Přijímání prázdným zásobníkem ze  $CFG$ ): Pro  $PDA P$  konstruovaný z  $CFG G$  algoritmem výše je  $N(P) = L(G)$ .

**Důkaz:**

1. Levá derivace:  $E \Rightarrow E * E \Rightarrow I * E \Rightarrow a * E \Rightarrow a * I \Rightarrow a * b$
2. Posloupnost situací:  $(q, a * b, E) \vdash (q, a * b, E * E) \vdash (q, a * b, I * E) \vdash (q, a * b, a * E) \vdash (q, *b, *E) \vdash (q, b, E) \vdash (q, b, I) \vdash (q, b, b) \vdash (q, \lambda, \lambda)$ .

Pozorování:

1. Kroky derivace simulují  $PDA$  přepisy zásobníku
2. odmazávaný vstup u  $PDA$  v derivaci zůstává až do konce
3. až  $PDA$  vymaže terminály, pokračuje v přepisech.

$$w \in N(P) \Leftarrow w \in L(G)$$

Nechť  $w \in L(G)$ ,  $w$  má levou derivaci  $S = \gamma_1 \Rightarrow_{Im} \gamma_2 \Rightarrow \dots \Rightarrow \gamma_n = w$ .

Indukcí podle  $i$  dokážeme  $(q, w, S) \vdash_P^* (q, v_i, \alpha_i) : \gamma_i = u_i \alpha_i$  je levá sentenciální forma a  $u_i v_i = w$ . □

## 8 Osmá přednáška

**Příklad:** Gramatika  $G = (S, L, R, (, ), P, S)$ :

1.  $S \rightarrow LR|SS|LA$
2.  $A \rightarrow SR$
3.  $L \rightarrow ($
4.  $R \rightarrow )$

$$S \Rightarrow LR \Rightarrow (R \Rightarrow ()$$

$$S \Rightarrow SS \Rightarrow LRS \Rightarrow (RS \Rightarrow ()S \Rightarrow ()LR \Rightarrow ()(R \Rightarrow ())$$

$$S \Rightarrow LA \Rightarrow (A \Rightarrow (SR \Rightarrow (LRR \Rightarrow ((RR \Rightarrow ((()R \Rightarrow (())$$

**Příklad:** Gramatika pro zjednodušené aritmetické výrazy  $I \rightarrow a|b|Ia|Ib|I0|I1$

1.  $F \rightarrow I|(E)$
2.  $T \rightarrow F|T * F$
3.  $E \rightarrow T|E + T$

$$E \Rightarrow T \Rightarrow T * F \Rightarrow F * F \Rightarrow I * F \Rightarrow a * F \Rightarrow a * I \Rightarrow a * Ib \Rightarrow a * bb$$

**Příklad:** CYK Algoritmus

Gramatika  $G = (S, A, L, R, (, ), P, S)$

1.  $S \rightarrow LR|SS|LA$
2.  $A \rightarrow SR$
3.  $L \rightarrow ($
4.  $R \rightarrow )$

Tabulku vyplňujeme odspodu

**DOPLNIT TABULKU (pripnuta fotka na discordu)**

**Věta:** Algoritmus CYK (Cocke-Younger-Kasami) v čase  $O(n^3)$

1. Mějme gramatiku v ChNF  $G = (V, T, P, S)$  pro jazyk  $L$  a slovo  $w = a_1a_2 \dots a_n \in T^*$
2. Vytvořme trojúhelníkovou tabulku:
  - (a) horizontální osa  $w$ ,
  - (b)  $X_{ij}$  jsou množiny neterminálů  $A$  takových, že  $A \Rightarrow^* a_i a_{i+1} \dots a_j$ .
  - (c) Základ:  $X_{ii} = A; A \rightarrow a_i \in P$
  - (d) Indukce:  $X_{ij} = A \rightarrow BC; B \in X_{ik}, C \in X_{k+1,j}$
3. Vyplňujeme tabulku zdola nahoru
4. Pokud  $S \in X_{1,n}$ , potom  $w \in L(G)$ .

**Definice:** Chomského normální forma

Chomského normální forma: všechna pravidla tvaru  $A \rightarrow BC$  nebo  $A \rightarrow a$ ,  $A, B, C$  jsou neterminály,  $a$  terminál

Každý bezkontextový jazyk (kromě slova  $\lambda$ ) je generovaný gramatikou v Chomského normálním tvaru. Postupně provedeme zjednodušení gramatiky, nejdřív:

1. **nestihol som (+-163 slide)**

**Definice:** zbytečný, užitečný, generující, dosažitelný symbol

1. Symbol  $X$  je užitečný v gramatice  $G = (V, T, P, S)$  pokud existuje derivace tvaru  $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$ , kde  $w \in T^*, X \in (V \cup T), \alpha, \beta \in (V \cup T)^*$ .
2. Pokud  $X$  není užitečný, říkáme, že je zbytečný.
3.  $X$  je generující pokud  $X \Rightarrow^* w$  pro nějaké slovo  $w \in T^*$ . Vždy  $w \Rightarrow^* w$  v nula krocích.
4.  $X$  je dosažitelný, pokud  $S \Rightarrow^* \alpha X \beta$  pro nějaká  $\alpha, \beta \in (V \cup T)^*$

**Věta:** Nech  $G = (V, T, P, S)$  je CFG, předpokládejme  $L(G) = \emptyset$  Zkonstruujeme  $G_1 = (V_1, T_1, P_1, S)$  následovně:

1. Eliminujeme ne-generující symboly a pravidla je obsahující

2. poté eliminujeme všechny symboly které jsou nedosažitelné.

Pak  $G_1$  nemá zbytečné symboly a  $L(G_1) = L(G)$ .

**Věta:** Algoritmus : Generující symboly

*Základ:* Každý  $a \in T$  je generující.

*Indukce:*  $\forall$  pravidlo  $A \rightarrow \alpha, : \forall$  symbol  $\in \alpha$  je generující. Pak i  $A$  je generující. Včetně  $A \rightarrow \lambda$ .

**Věta:** Algoritmus : Dosažitelné symboly

*Základ:*  $S$  je dosažitelný.

*Indukce:* Je-li  $A$  dosažitelný, pro všechna pravidla  $A \rightarrow \alpha$  jsou všechny symboly v  $\alpha$  dosažitelné.

**Věta:** Výše uvedené algoritmy najdou právě všechny generující / dosažitelné symboly.

**Definice:** Nulovatelný neterminál

Neterminál  $A$  je nulovatelný, pokud  $A \Rightarrow^* \lambda$ .

Pro nulovatelné neterminály na pravé straně pravidla  $B \rightarrow CAD$ , vytvoříme dvě verze pravidla: s a bez nulovatelného terminálu.

**Definice:** Algoritmus: Nalezení nulovatelných symbolů v  $G$

*Základ:* Pokud  $A \rightarrow \lambda$  je pravidlo  $G$ , pak  $A$  je nulovatelné.

*Indukce:* Pokud  $B \rightarrow C_1 \dots C_k$ , kde jsou všechna  $C_i$  nulovatelná, je i  $B$  nulovatelné. (terminály nejsou nulovatelné nikdy).

**Definice:** Algoritmus: Konstrukce gramatiky bez  $\lambda$ -pravidel z  $G$

1. Najdi nulovatelné symboly

2. Pro každé ... **Nestihol som pretože borka nahodila rychlost jak ježek na principech**

**Definice:** Jednotkové pravidlo

Jednotkové pravidlo je  $A \rightarrow B \in P$ , kde  $A, B$  jsou oba neterminály.

**Definice:** Jednotkový pár

**nestihol som :)**

**Definice:** Algoritmus: Nalezení jednotkových párů

*Základ:*  $(A, A)$  pro každý  $A \in V$  je jednotkový pár.

*Indukce:* Je-li  $(A, B)$  jednotkový pár a  $(B \rightarrow C) \in P$ , pak  $(A, C)$  je jednotkový pár.

**Definice:** Nejaký ďalší algoritmus ktorý som nestihol

**Věta:** Gramatika v normálním tvaru, redukovaná

Mějme bezkontextovou gramatiku  $G, L(G) - \lambda \neq \emptyset$ . Pak existuje  $CFG G_1 : L(G_1) = L(G) - \lambda$  a  $G_1$  neobsahuje  $\lambda$ -pravidla ani zbytečné symboly. Gramatika  $G_1$  se nazývá redukovaná.

**Důkaz:** Idea důkazu:

1. Začneme eliminací  $\lambda$ -pravidel
2. Eliminujeme ... :))))))))))))))))))))))))))))))

□

## 9 Devátá přednáška

**Definice:** Chomského normální tvar

O bezkontextové gramatice  $G = (V, T, P, S)$  bez zbytečných symbolů kde jsou všechna pravidla v jedné ze dvou tvarů

1.  $A \rightarrow BC, A, B, C \in V,$
2.  $A \rightarrow a, A \in V, a \in T,$

říkáme, že je v Chomského normálním tvaru (ChNF).

Potřebujeme dva další kroky:

1. pravé strany délky 2 a více předělat na samé neterminály
2. rozdělit pravé strany délky 3 a více neterminálů na více pravidel

**Poznámka:** neterminály:

1. pro každý terminál  $a$  vytvoříme nový neterminál, řekněme  $A$ .
2. přidáme pravidlo  $A \rightarrow a$ ,
3. použijeme  $A$  místo  $a$  na pravé straně pravidel délky 2 a více.

**Poznámka:**

**Věta:** Gramatika v normálním tvaru, redukovaná

Mějme bezkontextovou gramatiku  $G, L(G) - \lambda \neq \emptyset$ . Pak existuje  $CFG G_1 : L(G_1) = L(G) - \lambda$  a  $G_1$  neobsahuje  $\lambda$ -pravidla, jednotková pravidla ani zbytečné symboly. Gramatika  $G_1$  se nazývá redukovaná.

**Důkaz:**

1. Začneme eliminací  $\lambda$ -pravidel
2. Eliminujeme jednotková pravidla. Tím nepřidáme  $\lambda$ -pravidla.
3. Eliminujeme zbytečné symboly. Tím nepřidáme žádná pravidla.

□

**Věta:** ChNF

Mějme bezkontextovou gramatiku  $G, L(G) - \lambda \neq \emptyset$ . Pak existuje  $CFG G_1$  v ChNF taková, že  $L(G_1) = L(G) - \lambda$

**Příklad:** DOPLNIT Z DISCORDU 8.10

**Věta:** Velikost derivačního stromu gramatiky v CNF

Mějme derivační strom podle gramatiky  $G = (V, T, P, S)$  v ChNF, který dává slovo  $w$ . Je-li délka nejdelší cesty  $n$ , pak  $|w| \leq 2^{n-1}$ . **Důkaz:**

Indukcí dle  $n$ ,

1.  $|a| = 1 = 2^0$
2.  $2^{n-2} + 2^{n-2} = 2^{n-1}$ .

□

**Poznámka:** Mějme derivační strom podle gramatiky  $G = (V, T, P, S)$  v ChNF, který dává slovo  $w$ ,  $|w| > p = 2^{n-1}$ . Pak ve stromě existuje cesta delší než  $n$ .

**Věta:** Lemma o vkládání (pumping) pro bezkontextové jazyky

Mějme bezkontextový jazyk  $L$ . Pak  $\exists n \in \mathbb{N} : \forall z \in L, |z| > n : z = uvwxy :$

1.  $|vwx| \leq n$
2.  $vx \neq \lambda$
3.  $\forall i \geq 0, uv^iwx^iy \in L$ .

**OBRAZOK 8.2 Důkaz:** náznak důkazu:

1. vezmeme derivační strom pro  $z$
2. najdeme nejdelší cestu
3. na ní dva stejné neterminály
4. tyto neterminály určí dva podstromy.
5. podstromy definují rozklad slova
6. nyní můžeme větší podstrom posunout ( $i > 1$ )
7. nebo nahradit menším podstromem ( $i = 0$ )

**Důkaz:**

1. vezmeme gramatiku v ChNF ( pro  $L = \lambda$  a  $\emptyset$  zvolíme  $n = 1$ )
2. Nechť  $|V| = k; n = 2^k$ .
3. Pro  $z \in L, |z| > 2^k$  má v derivačním stromu  $z$  cestu delší než  $k$ .
4. vezmeme nejdelší cestu; terminál kam vede označíme  $t$ .
5. ASpoň dva z posledních  $(k + 1)$  neterminálů na cestě do  $t$  jsou stejné
6. vezmeme dvojici  $A^1, A^2$  nejbližší  $k$   $t$  (určuje podstromy  $T^1, T^2$ )
7. cesta z  $A^1$  do  $t$  je nejdelší v podstromu  $T^1$  a má délku maximálně  $(k + 1)$  (tedy slovo dané stromem  $T^1$  není delší než  $2^k$  (tedy  $|vwx| \leq n$ ))
8. z  $A^1$  vedou dvě cesty (ChNF), jedna do  $T^2$ , druhá do zbytku  $wx$  (ChNF je nevyoustějící, tedy  $vx \neq \lambda$ )

9. *derivace slova*  $(A^1 \Rightarrow^* vA^2x, A^2 \Rightarrow^* w); S \Rightarrow^* uA^1y \Rightarrow^* uvA^2xy \Rightarrow^* uvwxy$
10. *posuneme-li*  $A^2$  *do*  $A^1 \rightarrow S \Rightarrow^* uA^2y \Rightarrow^* uwy$
11. *posuneme-li*  $A^1$  *do*  $A^2$  ( $i = 2, 3, \dots$ ) *rightarrow*  $S \Rightarrow^* uA^1y \Rightarrow^* uvA^1xy \Rightarrow^* uvvA^2xy \Rightarrow^* uvvwxxy$ .

□

**Definice:** Deterministický zásobníkový automat (DPDFA)

Zásobníkový automat  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  je deterministický PDA právě když platí:

1.  $\delta(q, a, X)$  je nejvýše jednoprvková  $\forall (q, a, X) \in Q \times (\Sigma \cup \lambda) \times \Gamma$ .
2. Je-li  $\delta(q, a, X)$  neprázdná pro nějaké  $a \in \Sigma$ , pak  $\delta(q, \lambda, X)$  musí být prázdná.

**Věta:**

$$RL \subsetneq L(P_{DPDA}) \subsetneq L(P_{PDA}) = CFL = N(P_{PDA}) \supsetneq N(P_{DPDA})$$

*Nechť  $L$  je regulární jazyk, pak  $L = L(P)$  pro nějaký DPDA. **Důkaz:** DPDA může simulovat deterministický konečný automat a ignorovat zásobník (nechat tam  $Z_0$ ).* □

**Poznámka:** Jazyk  $L_{w_{cwr}}$  je přijímaný DPDA ale není regulární.

Důkaz neregularity z pumping lemmatu na slovo  $0^n c 0^n$ .

## 10 Desátá přednáška

### 10.1 Uzávěrová vlastnosti

**Věta:** CFL uzavřené na sjednocení, konkatenaci, uzávěr, reverzi

CFL jsou uzavřené na sjednocení, konkatenaci, iteraci ( $*$ ), pozitivní iteraci ( $+$ ), homomorfismus, zrcadlový obraz  $w^R$ . **Důkaz:**

- *Sjednocení:*
  - pokud  $V_1 \cap V_2 \neq \emptyset$  přejmenujeme neterminály,
  - přidáme nový symbol  $S_{new}$  a pravidlo  $S_{new} \rightarrow S_1 | S_2$ .
- *Zřetězení  $L_1.L_2$* 
  - $S_{new} \rightarrow S_1 S_2$  (pro  $V_1 \cap V_2 = \emptyset$ , jinak přejmenujeme)
- *iterace  $L^* = \bigcup_{i \geq 0} L^i$* 
  - $S_{new} \rightarrow S S_{new} | \lambda$
- *pozitivní iterace  $L^+ = \bigcup_{i \geq 1} L^i$* 
  - $S_{new} \rightarrow S S_{new} | S$
- *zrcadlový obraz  $L^R = \{w^R | w \in L\}$* 
  - $X \rightarrow \omega^R$  obrátíme pravou stranu pravidel.

□

**Příklad:** • Jazyk  $L = \{0^n 1^n 2^n | n \geq 1\} = \{0^n 1^n 2^i | n, i \geq 1\} \cap \{0^i 1^n 2^n | n, i \geq 1\}$  není CFL, i když oba členy průniku jsou bezkontextové, dokonce deterministické bezkontextové.

$$\{0^n 1^n 2^i | n, i \geq 1\} \rightarrow \{S \rightarrow AC, A \rightarrow 0A1|01, C \rightarrow 2C|2\}$$

$$\{0^i 1^n 2^n | n, i \geq 1\} \rightarrow \{S \rightarrow AB, A \rightarrow 0A|0, B \rightarrow 1B2|12\}$$

- průnik není CFL z pumping lemmatu.
- paralelní běh dvou zásobníkových automatů
- řídicí jednotky umíme spojit (viz konečné automaty)
- čtení umíme spojit (jeden automat může čekat)
- bohužel dva zásobníky nelze obecně spojit do jednoho

dva neomezené zásobníky  $\rightarrow$  Turingův stroj, rekurzivně spočetné jazyky  $\mathcal{L}_0$

**Věta:** CFL i DCFL jsou uzavřené na průnik s regulárním jazykem

- Mějme  $L$  bezkontextový jazyk a  $R$  regulární jazyk.  
Pak  $L \cap R$  je bezkontextový jazyk.
- Mějme  $L$  deterministický CFL a  $R$  regulární jazyk.  
Pak  $L \cap R$  je deterministický CFL.

**Důkaz:**

- zásobníkový a konečný automat můžeme spojit
  - FA  $A_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
  - PDA přijímaný stavem  $M_1 = (Q_2, \Sigma, \Gamma, \delta_2, q_2, Z_0, F_2)$
- nový automat  $M = (Q_1 \times Q_2, \Sigma, \Gamma, \delta, (q_1, q_2), Z_0, F_1 \times F_2)$ 
  - $((r, s), \alpha) \in \delta((p, q), a, Z)$  právě když
  - $a \neq \lambda : r = \delta_1(p, a) \& (s, \alpha) \in \delta_2(q, a, Z) \dots$  automaty čtou vstup, PDA mění zásobník, FA stojí.
  - $a = \lambda : (s, \alpha) \in \delta_2(q, \lambda, Z)$
  - $r = p$
- zřejmě  $L(M) = L(A_1) \cap L(M_2)$ 
  - paralelní běh automatů.

□

**Příklad:** Substituce

Mějme gramatiku  $G = (E, a, +, (, ), E \rightarrow E + E | (E) | a, E)$ . Mějme substituci:

- $\sigma(a) = L(G_a), G_a = (I, a, b, 0, 1, I \rightarrow I0|I1|Ia|Ib|a|b, I)$ ,
- $\sigma(+) = -, \times, :, div, mod$ ,
- $\sigma(() = ($ ,
- $\sigma()) = )$ .



- $(a + a) + a \in L(G)$
- $(a001 - bba) * b_1 \in \sigma((a + a) + a) \subset \sigma(L(G))$ ,
- v  $\sigma(a)$  chybí  $+$  pro ukázkou, že  $(a + a) + a \notin \sigma(L(G))$ .

Co se stane, když změníme definici:

- $\sigma(( ) = (, [$ ,
- $\sigma()) = ), ]?$

**Příklad:** Homomorfismus

Mějme gramatiku  $G = (E, a, +, (, ), E \rightarrow E + E | (E)a, E)$ . Mějme homomorfismus:

- $h(a) = \lambda$
- $h(+) = \lambda$
- $h(( ) = left$
- $h()) = right$
- $h((a + a) + a) = leftright$ ,
- $h^{-1}(leftright) \ni (a + +)a$ .

**Věta:** CFL jsou uzavřené na substituci

Mějme CFL jazyk  $L$  nad  $\Sigma$  a substituci  $\sigma$  na  $\Sigma$  takovou, že  $\sigma(a)$  je CFL  $\forall a \in \Sigma$ . Pak je i  $\sigma(L)$  CFL (bezkontextový). **Důkaz:**

- *Idea:* listy v derivačním stromu generují další stromy,
- Přejmenujeme neterminály na jednoznačné všude v  $G = (V, \Sigma, P, S), G_a = (V_a, T_a, P_a, S_a), a \in \Sigma$ .
- Vytvoříme novou gramatiku  $G = (V', T', P', S)$  pro  $\sigma(L)$ :
  - $V' = V \cup \bigcup_{a \in \Sigma} V_a$ ,
  - $T'$

□

**Příklad:** Substitute

**Věta:** CFL jsou uzavřené na inverzní homomorfismus

Mějme CFL jazyk  $L$  a homomorfismus  $h$ . Pak  $h^{-1}(L)$  je bezkontextový jazyk. Je-li TODO **Důkaz:**

- pro  $L$  máme PDA  $M = (Q, \Sigma, \delta, q_0, Z_0, F)$  (koncovým stavem)
- $h : T \rightarrow \Sigma^*$
- definujeme PDA  $M' = (Q', T, \Gamma, \delta', [q_0, \lambda], Z_0, F \times \lambda)$ , kde

**DOPLNIT, slide 196**

Pro deterministický PDA  $M$  je i  $M'$  deterministický.

□

197

**Důkaz:** Důkaz sporem

- Necht  $L$  je bezkontextový jazyk
- $L_1 = \{01^j 2^k 3^l \mid j, k, l \geq 0\}$  je regulární Jazyk
  - $S \rightarrow 0B, B \rightarrow 1B|C, C \rightarrow 2C|D, D \rightarrow 3D|\lambda$
- $L \cap L_1 = 01^i 2^i 3^i \mid i \geq 0$  není bezkontextový  $\implies$  spor

□

**Věta:** Rozdíl s regulárním jazykem*Mějme bezkontextový jazyk  $L$  a regulární jazyk  $R$ . Pak:*

- $L - R$  je CFL.

**Důkaz:**  $L - R = L \cap R, \bar{R}$  je regulární.

□

**Věta:** CFL nejsou uzavřené na doplněk ani rozdíl*Mějme bezkontextové jazyky  $L, L_1, L_2$ , regulární jazyk  $R$ . Pak:*

- $\bar{L}$  nemusí být CFL. ...  $L_1 \cap L_2 = \overline{\bar{L}_1}$  **TODO**

## 10.2 Uzávěrové vlastnosti deterministických CFL

- Rozumné programovací jazyky jsou deterministické CFL.
- Deterministické bezkontextové jazyky
  - nejsou uzavřené na průnik
  - jsou uzavřené na průnik s regulárním jazykem
  - jsou uzavřené na inverzní homomorfismus

**Věta:** Doplněk deterministického CFL je opět deterministický CFL. **Důkaz:**

- *idea: prohodíme koncové a nekoncové stavy*
- *nedefinované kroky ošetříme 'podložkou' na zásobníku*
- *cyklus odhalíme pomocí čítače*
- *až po přečtení slova prochází koncové a nekoncové stavy (stačí si pamatovat, zda prošel koncovým stavem.)*

□

**Uzávěrové vlastnosti v kostce - doplnit tabulku (202/255-274)**

**Poznámka:** Ne-uzavřenost deterministických CFL **Důkaz:** Vzhledem k uzavřenosti DCFL na doplněk by byl DCFL i  $\bar{L} \cap a^* b^* c^* = a^i b^j c^k \mid i = j = k$ , o kterém víme, že není CFL (pumping lemma) □

**DCFL nejsou uzavřené na homomorfismus (201/255-274 )**

## 11 Jedenáctá přednáška

**Definice:** Dyckův jazyk

Dyckův jazyk  $D_n$  je definován nad abecedou  $Z_n = a_1, a_2, \dots, a_n, a_n^{\mid}, \dots, a_2^{\mid}, a_1^{\mid}$  následující gramatikou:  $S \rightarrow \lambda \mid SS^{\mid} \mid a_1 S a_1^{\mid} \mid \dots \mid a_n S a_n^{\mid}$ .

Úvodní pozorování:

- jedná se zřejmě o jazyk bezkontextový
- Dyckův jazyk  $D_n$  popisuje správně uzávorkované výrazy s  $n$  druhy závorek
- tímto jazykem lze popisovat výpočty libovolného zásobníkového automatu
- pomocí Dyckova jazyka lze popsat libovolný bezkontextový jazyk.

$$L = h(D \cap R)$$

- $h \dots$  homomorfismus
- $D \dots$  dyckův jazyk
- $R \dots$  regulární jazyk

### 11.1 Jak charakterizovat bezkontextové jazyky?

- Pokud do zásobníku pouze přidáváme, stačí si pamatovat poslední symbol; stačí konečná paměť  $\rightarrow$  konečný automat
- potřebujeme ze zásobníku také odebírat (čtení symbolu), takový proces nelze zaznamenat v konečné struktuře
- přidávání a odebírání není zcela libovolné, jedná se o zásobník, t. j. LIFO strukturu
- roztáhneme si výpočet se zásobníkem do lineární struktury:

–  $X$  symbol přidání

**Věta:** Pro každý bezkontextový jazyk  $L$  existuje regulární jazyk  $R$  tak, že  $L = h(D \cap R)$  pro vhodný Dyckův jazyk  $D$  a homomorfismus  $h$ . **Důkaz:**

- máme PDA přijímající přijímající  $L$  prázdným zásobníkem
- převedeme na instrukce tvaru  $\delta(q, a, Z) \in (p, w), |w| \leq 2$ ; delší psaní na zásobník rozdělíme vedením nových stavů
- nechť  $R^{\mid}$  obsahuje všechny výrazy,
  - $q^{-1}aa^{-1}Z^{-1}BAp$  pro instrukci  $\delta(q, a, Z) \ni (p, AB)$
  - podobně pro instrukce  $\delta(q, a, Z) \in (p, A), \delta(q, a, Z) \in (p, \lambda)$
  - je-li  $a = \lambda$ , potom dvojici  $aa^{-1}$  nezařazujeme
- definujeme  $R : Z_0 q_0 (R^{\mid})^* Q^{-1}$
- Dyckův jazyk je definován nad abecedou  $\Sigma \cup \Sigma^{-1} \cup Q \cup Q^{-1} \cup \Gamma \cup \Gamma^{-1}$
- $D \cap Z_0 q_0 (R^{\mid})^* Q^{-1}$  popisuje korektní výpočty

$$Z_0 q_0 q_0^{-1} a a^{-1} Z_0^{-1} B A p p^{-1} b b^{-1} A^{-1} q q^{-1} c c^{-1} B^{-1} r r^{-1}$$

- homomorfismus  $h$  vydělí přečtené slovo, t. j.
  - $h(a) = a$  pro vstupní (čtené) symboly,
  - $h(y) = \lambda$  pro ostatní

□

**Věta:** Algoritmus: Konstrukce PDA z CFG

Mějme CFG gramatiku  $G = (V, T, P, S)$ , konstruuujeme PDA  $P = (q, T, V \cup T, \delta, q, S)$ .

1. Pro neterminály  $A \in V$ ,  $\delta(q, \lambda), A = (q, \beta) | A \rightarrow \beta$  je pravidlo  $G$
2. pro každý terminál  $a \in T$ ,  $\delta(q, a, a) = (q, \lambda)$ .

## 11.2 Od zásobníkového automatu ke gramatice CFG

- Zásobní automat bere jeden symbol ze zásobníku. Stav před a pro kroku může být různý.
- Neterminály gramatiky budou složené symboly  $[qXr]$ , PDA vyšel z  $q$ , vzal  $X$  a přešel do  $r$ ; zavedeme nový počáteční symbol  $S$ .

**Věta:** Mějme PDA  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ . Pak existuje bezkontextová gramatika  $G$  taková, že  $L(G) = N(P)$ . **Důkaz:** Pravidla definujeme:

- $\forall p \in Q : S \rightarrow [q_0 Z_0 p]$ , t. j. uhodni koncový stav a spusť PDA na  $(q_0, w, Z_0) \vdash^* (p, \lambda, \lambda)$ .
- Pro všechny dvojice  $(r, Y_1 Y_2 \dots Y_k) \in \delta(q, s, X)$ ,  $s \in \Sigma \cup \lambda$ ,  $\forall r_1, \dots, r_{k-1} \in Q$  vytvoř pravidlo

$$[qXr_k] \rightarrow s[rY_1 r_1][r_1 Y_2 r_2] \dots [r_{k-1} Y_k r_k]$$

- spec. pro  $(r, \lambda) \in \delta(q, a, A)$  vytvoř  $[qAr] \rightarrow a$ .

Pro  $w \in \Sigma^*$  dokazujeme:

$[qXp] \Rightarrow^* w \Leftrightarrow (q, w, X) \vdash^* (p, \lambda, \lambda)$ ; indukcí v obou směrech (počet kroků PDA, počet kroků derivace).

□

## 11.3 Greibachové normální forma

- při analýze zhora (tvorbě levé derivace daného slova  $w$ ) potřebujeme vědet, které pravidlo vybrat.
- Speciálně vadí pravidla tvaru  $A \rightarrow A\alpha$  (levá rekurze).

**Definice:** Říkáme, že gramatika je v Greibachové normální formě, jestliže všechna pravidla mají tvar  $A \rightarrow a\beta$ , kde  $a \in T, \beta \in V^*$  (řetězec) **TODO**

**Definice:** Jednoznačnost a víceznačnost CFG

- Bezkontextová gramatika  $G = (V, T, P, S)$  je víceznačná, pokud existuje aspoň jeden řetězec  $w \in T^*$  pro který můžeme najít dva různé derivační stromy
- **TODO**

**Věta:**  $L = N(P_{DPDA}) \Rightarrow L$  má jednoznačnou CFG.

- Nechť  $L = N(P)$  pro nějaký DPDA  $P$ . Pak  $L$  má jednoznačnou CFG
- **TODO**

## 11.4 Turingovy stroje

- 1931 - 1936 pokusy o formalizaci pojmu algoritmu (Kleene, Church, Turing...)
- Turingův stroj:
  - zachycení práce matematika
    - \* nekonečná tabule, lze z ní číst a lze na ni psát
    - \* mozek (řídící jednotka)
  - Formalizace Turing Machine (TM):
    - \* místo tabule oboustranně nekonečná páska
    - \* místo křídý čtecí a zapisovací hlava **TODO** :)

**Definice:** Turingův stroj (TM) je sedmice  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  se složkami:

- $Q$  konečná množina stavů
- $\Sigma$  konečná neprázdná množina vstupních symbolů
- $\Gamma$  konečná množina všech symbolů pro pásku. Vždy  $\Gamma \supseteq \Sigma, Q \cap \Gamma = \emptyset$ .
- $\delta$  částečná přechodová funkce.  $(Q - F) \times \Gamma \rightarrow Q \times \Gamma \times L, R$
- $\delta(q, x) = (p, Y, D)$ :
  - $q \in (Q - F)$  aktuální stav
  - $X \in \Gamma$  aktuální symbol na pásce
  - $p$  nový stav,  $p \in Q$
  - $Y \in \Gamma$  symbol pro zasání do aktuální buňky, přepíše aktuální obsah
  - $D \in L, R$  je směr pohybu hlavy (doleva, doprava).
- $q_0 \in Q$  počáteční stav
- $B \in \Gamma - \Sigma$  Blank. Symbol pro prázdné buňky na začátku všude kromě konečného počtu buněk se vstupem
- $F \subseteq Q$  množina koncových neboli přijímajících stavů.

**Definice:** Konfigurace Turingova stroje

- je řetězec  $X_1X_2 \dots X_{i-1}qX_iX_{i+1} \dots X_n$  kde
- $q$  je stav Turingova stroje
- čtecí hlava je vlevo od  $i$ -tého symbolu
- $X_1, \dots, X_n$  je část pásky mezi nejlevějším a nejpravějším symbolem různým od prázdného (B). S výjimkou v případě, že je hlava na kraji - pak na tom kraji vkládáme jeden  $B$  navíc.

**Definice:** Krok Turingova stroje  $M$  značíme  $\vdash_M, \vdash_M^*, \vdash^*$  jako u zásobníkových automatů. Pro  $\delta(q, X_i) = (p, Y, L)$

$$X_1X_2 \dots X_{i-1}qX_iX_{i+1} \dots X_n \vdash_M X_1X_2 \dots X_{i-2}pX_{X_i-1} \dots \text{TODO}$$

**Definice:** Turingův stroj  $M$  přijímá jazyk  $L(M) = \{w \in \Sigma^* : q_0w \vdash_M^* \alpha p \beta, p \in F, \alpha, \beta \in \Gamma^*, \text{ t. j. množinu slov, po jejichž přečtení se dostane do koncového stavu. Pásku (u nás) nemusí uklízet. Jazyk nazveme rekurzivně spočetným, pokud je přijíman nějakým Turingovým strojem } T, \text{ t. j. } L = L(T)\}$ .

**Definice:** TM zastaví, pokud vstoupí do stavu  $q$ , s čteným symbolem  $X$  a není instrukce pro tuto situaci, t.j.  $\delta(q, X)$  není definováno.

- Předpokládáme, že v přijímajícím stavu  $q \in F$  TM zastaví
- dokud nezastaví, nevíme, jestli přijme nebo nepřijme slovo.

**Definice:** Říkáme, že TM  $M$  rozhoduje jazyk  $L$ , pokud  $L = L(M)$  a pro každé  $w \in \Sigma^*$  stroj nad  $w$  zastaví.

Jazyky rozhodnutelné TM nazýváme *rekurzivní jazyky*.

**Definice:** Přejchodový diagram pro TM sestává z uzlů odpovídajícím stavům TM. Hrany  $q \rightarrow p$  jsou označeny seznamem všech dvojic  $X/YD$ , kde  $\delta(q, X) = (p, Y, D)$ ,  $D \in \leftarrow, \rightarrow$ . Pokud neuvedeme jinak,  $B$  značí blank - prázdný symbol.

## 12 Dvanáctá přednáška

**Příklad:** Jazyk  $L = a^{2n} | n \geq 0$  přijíma Turingův stroj  $M = (q_0, q_1, q_F, a, a, B, \delta, q_0, B, q_F)$  s  $\delta$  v tabulce:

**TODO, 220/212-233 Example 11.2**

- Regulární jazyky:
  - simulujeme konečný automat, pohyb hlavy vždy vpravo
  - vidíme-li  $B$ , t. j. konec vstupu, přejdu do nového přijímacího stavu  $q_F$ .
  - (Z přijímajících stavů nemá TM instrukci.)
- Bezkontextové jazyky: nejsnáze s pomocnou páskou simulující zásobník, bude za chvíli.

**Věta:** Každý rekurzivně spočetný jazyk je typu 0 **Důkaz:** pro TM  $T$  najdeme gramatiku  $G$ ,  $L(T) = L(G)$

1.  $G = (S, C, D, E) \cup \underline{X}_{x \in \Sigma \cup \Gamma} \cup Q_{i_{q_i} \in Q}, \Sigma, P, S)$ ,  $P$  je:
2. gramatika nejdříve vygeneruje pásku stroje a kopii slova  $wB^n \underline{X}^R Q_0 B^m$ , kde  $B^i$  představují volný prostor pro výpočet
3. potom simuluje výpočet (stavy jsou součástí slova)
4. v koncovém stavu smažeme pásku, necháme pouze kopii slova

**TODO, 224/212-233 Proof of Theorem 11.1** Ještě  $L(T) = L(G)$  ?

- $w \in L(T)$ 
  - existuje konečný výpočet stroje  $T$  (konečný prostor)
  - gramatika vygeneruje dostatečně velký prostor pro výpočet
  - simuluje výpočet a smaže dvojníky
- $w \in L(G)$ 
  - pravidla v derivaci nemusí být v pořadí, jakým **TODO, 224**

□

**Věta:** Každý jazyk typu 0 je rekurzivně spočetný **Důkaz:** idea: TM postupně generuje všechny derivace

- derivaci  $S \Rightarrow \omega_1 \Rightarrow \dots \Rightarrow \omega_n = w$  kódujeme jako slovo  $\#S\#\omega_1\#\dots\#w\#$
- umíme udělat TM, který:
  - přijímá slova  $\#\alpha\#\beta\#$ , kde  $\alpha \Rightarrow \beta$
  - přijímá slova  $\#\omega_1\#\dots\#\omega_k\#$ , kde  $\omega_1 \Rightarrow^* \omega_k$
  - postupně generuje všechny slova

□

**Definice:** Počáteční pozice

- vstup na první pásce, ostatní zcela prázdné
- první hlava vlevo od vstupu, ostatní libovolně
- hlava v počátečním stavu

Jeden krok vícepáskového TM

- hlava přejde do nového stavu
- na každé pásce napíšeme nový symbol
- každá hlava se nezávisle posune vlevo, zůstane, vpravo

**Věta:** Každý jazyk přijímaný vícepáskovým TM je přijímaný i nějakým (jednopáskovým) Turingovým strojem TM. **Důkaz:**

- konstruujieme TM  $M$
- pásku si představíme, že má  $2k$  stop
  - liché stopy: pozice  $k$ -té hlavy
  - sudé stopy: znak na  $k$ -té pásce
- pro simulaci jednoho kroku navštívíme všechny hlavy
- ve stavu si pamatujeme
  - počet hlav vlevo
  - $\forall k$  symbol pod  $k$ -tou hlavou
- pak už umíme provést jeden krok (znovu běhat)

□ Simulaci výpočtu  $k$ -páskového stroje o  $n$  krocích lze provést v čase  $O(n^2)$  (simulace jednoho kroku z prvních  $n$  trvá  $4n + 2k$ , hlavy nejvýš  $2n$  daleko, přečíst, zapsat, posunout značky).

**Definice:** Nedeterministický TM

nazýváme sedmici  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , kde  $Q, \Sigma, \Gamma, q_0, B, F$  jsou jako u TM a  $\delta : (Q - F) \times \Gamma \rightarrow P(Q \times \Gamma \times L, R)$ .

Slovo  $w \in \Sigma^*$  je přijímáno nedeterministickým TM  $M$ , pokud existuje nějaký výpočet  $q_0 w \vdash^* \alpha p \beta, p \in F$ .

**Věta:** Pro každý  $M_N$  nedeterministický TM existuje deterministický TMM<sub>D</sub> takový, že  $L(M_N) = L(M_D)$ . **Důkaz:** prohledáváme do šířky možné výpočty  $M_N$

- odvozeno v  $k$  krocích
- maximálně  $m$  konfigurací (kde  $m = \max|\delta(q, x)|$  je max. počet voleb  $M_N$ )

□

## 13 Dvanáctá přednáška

**Definice:** Separovaná gramatika

Gramatika je separovaná, pokud obsahuje pouze pravidla tvaru  $\alpha \rightarrow \beta$ , kde:

- buď  $\alpha, \beta \in V^+$  (neprázdné posloupnosti neterminálů)
- nebo  $\alpha \in V; \beta \in T \cup \lambda$ .

**Věta:** *Tu bolo nejaké lemma ktoré som nestihol napísať* **Důkaz:** *a tu bola proof ktorú som nestihol napísať* □

**Definice:** monotónní (nevypouštějící) gramatika

Gramatika je monotónní, jestliže pro každé pravidlo  $(\alpha \rightarrow \beta) \in P : |\alpha| \leq |\beta|$ .

**Věta:** *Ke každé monotónní gramatice lze nalézt ekvivalentní gramatiku kontextovou* **Důkaz:**

- nejprve převedeme gramatiku na separovanou - tím se monotonie neporuší (a pravidla  $A' \rightarrow a$  jsou kontextová)
- zbývající pravidla  $A_1 \dots A_m \rightarrow B_1 \dots B_n, m \leq n$  převedeme na pravidla s novými terminály  $C$

*TODO, tabulka, pod definicí 12.2, 238/234-314* □

### 13.1 Lineárně omezené automaty

Ještě potřebujeme ekvivalent pro kontextové gramatiky. Kontextovou gramatiku dostaneme z libovolné monotónní gramatiky.

**Definice:** Lineárně omezený automat LBA je nedeterministický TM, kde na pásce je označen levý a pravý konec,  $l, r$ . Tyto symboly nelze při výpočtu přepsat a nesmí se jít nalevo od  $l$  a napravo od  $r$ . Slovo  $w$  je přijímáno lineárně omezeným automatem, pokud  $q_0 l w r \vdash^* \text{TODO}$

**Věta:** *Každý kontextový jazyk lze přijímat pomocí LBA.* **Důkaz:**

- derivaci gramatiky budeme simulovat pomocí LBA.
- použijeme pásku se dvěma stopami
- slovo  $w$  dáme nahoru, na začátek dolní stopy  $S$
- přepisujeme slovo ve druhé stopě podle pravidel  $G$ 
  - nedeterministicky vybereme část  $k$  přepsání
  - provedeme přepsání dle pravidla (pravá část se odsune)
- pokud jsou ve druhé stopě samé terminály, porovnáme ji s první stopou. Slovo přijmeme nebo ho zamítneme.

□

**Věta:** *LBA přijímají pouze kontextové jazyky* **Důkaz:**

- potřebujeme převést LBA na monotónní gramatiku, t.j. gramatika nesmí generovat nic navíc
- výpočet ukryjeme do 'dvoustupých' neterminálů
- generuj slovo ve tvaru  $(a_0, [q_0, l, a_0]), (a_1, a_1), \dots, (a_n, [a_n, r])$



- *simuluj práci LBA ve 'druhé' stopě (stejně jako u TM)*
  - *pro  $\delta(p, x) = (q, x', R) : Px \rightarrow x'Q$*
  - *pro  $\delta(p, x) = (q, x', L) : yPx \rightarrow Qyx'$*
- *pokud je stav koncový smaž 'druhou' stopu*
- *speciálně je třeba ošetřit přijímání prázdného slova (pokud LBA přijíma  $\lambda$ , přidáme speciální startovací pravidlo)*

□

## 13.2 Rekurzivní jazyky

**Definice:** TM zastaví, pokud vstoupí do stavu  $q$ , s čteným symbolem  $X$ , a není instrukce pro tuto situaci, t. j.  $\delta(q, X)$  není definováno.

- Předpokládáme, že v přijímacím stavu  $q \in F$  TM zastaví,
- dokud nezastaví, nevíme jestli přijme nebo nepřijme slovo.

**Definice:** Říkáme, že TM  $M$  rozhoduje jazyk  $L$ , pokud  $L = L(M)$  a pro každé  $w \in \Sigma^*$  stroj nad  $w$  zastaví.

Jazyky rozhodnutelné TM nazýváme rekurzivní jazyky.

**Věta:** Je-li  $L$  rekurzivní, je rekurzivní i  $\bar{L}$ .

**Věta:** Postova věta

*Jazyk  $L$  je rekurzivní právě když  $L$  i  $\bar{L}$  (doplňěk) jsou rekurzivně spočetné. **Důkaz:***

- Máme  $TM L = L(M_1) \& \bar{L} = L(M_2)$ .
- pro dané slovo  $w$  naraz simulujeme  $M_1$  i  $M_2$  (dvě pásky, stav se dvěma komponentami)
- Pokud jeden z  $M_i$  přijme,  $M$  zastaví a odpoví.
- Jazyky jsou komplementární, jeden z  $M_i$  vždy zastaví,  $L$  je rekurzivní.

□

## 13.3 Jazyk, který není rekurzivně spočetný

Směřujeme k důkazu nerozhodnutelnosti jazyka dvojic  $(M, w)$  takových, že:

- $M$  je binárně kódovaný Turingův stroj s abecedou  $0, 1$ ,
- $w \in 0, 1^*$  a
- $M$  nepřijímá vstup  $w$ .

Postup:

- Kódování TM binárním kódem pro libovolný počet stavů TM
- Kód TM vezmeme TM jako binární řetězec.
- Pokud kód nedává smysl, reprezentuje TM bez transakcí. Tedy každý kód reprezentuje nějaký TM.

- Diagonální jazyk  $L_d = \{w; \text{TM reprezentovaný jako } w \text{ takový, že nepřijímá } w\}$
- Neexistuje TM přijímající jazyk  $L_d$ . Spuštění takového stroje na vlastním kódu by vedlo k paradoxu.

Jazyk  $L_d$  není rekurzivně spočetný. Proto  $\overline{L_d}$  není rekurzivní. Lze dokázat.

### 13.4 Kódování

- Pro kódování  $TMM = (Q, 0, 1, \Gamma, \delta, q_1, B, q_2)$  očíslovíme stavy, symboly a směry  $L, R$ .
- Předpokládáme:
  - Počáteční stav je vždy  $q_1$ .
  - Stav  $q_2$  je vždy jediný koncový stav (nepotřebujeme víc, TM zastaví).
  - První symbol je vždy 0, druhý 1, třetí B, prázdný symbol. Ostatní symboly pásky očíslovíme libovolně.
  - Směr  $L$  je 1, směr  $R$  je 2.
- Jeden krok  $\delta(q_i, X_j) = (q_k, X_l, D_m)$  kódujeme.  $0^j 10^j 10^k 10^l 10^m$ . Všechna  $i, j, k, l, m \geq 1$ , takže se dvě jedničky za sebou nevyskytují.
- Celý TM se skládá z kódů všech přechodů v nějakém pořadí oddělených dvojicemi jedniček 11 -  $C_1 11 C_2 11 \dots C_{n-1} 11 C_n$ .

Budeme potřebovat uspořádat řetězce do posloupnosti:

- Řetězce budeme uspořádávat podle délky, stejně dlouhé uspořádáme lexikograficky.
- První je  $\lambda$ , druhý 0, třetí 1, čtvrtý 11
- **TODO, ešte jedna vec tu bola**

**Definice:** Diagonální jazyk  $L_d$  je definovaný jako  $L_d = \{w; \text{TM reprezentovaný jako } w \text{ takový, že nepřijímá } w\}$

**Věta:**  $L_d$  není rekurzivně spočetný jazyk, t. j. neexistuje TM přijímající  $L_d$  **Důkaz:**

- Předpokládáme  $L_d \in RE$ ,  $L_d = L(M_d)$  pro nějaký TM  $M_d$ .
- Jazyk je 0, 1. Přijímá TM  $M_d$  vstupní slovo  $w_j$ ?
- Alespoň jeden řetězec ho kóduje. řekněme, že  $code(M_d) = w_d$ .
- Je  $w_D \in L_d$ ?
  - Pokud 'ano', na diagonále má být 0, t. j.  $w_d \notin L(M_d) = L_d$ , spor.
  - Pokud 'ne', na diagonále má být 1, t. j.  $w_d \in L(M_d) = L_d$ , spor.

Proto takový  $M_d$  neexistuje. Tedy  $L_d$  není rekurzivně spočetný. □

**Definice:** Univerzální jazyk definujeme  $L_u$  jakožto množinu binárních řetězců, které kódují pár  $(M, w)$ , kde  $M$  je TM a  $w \in L(M)$ .

TM který spracuje  $L_u$  se nazývá Univerzální Turingův stroj.

**Věta:** Existuje Turingův stroj  $U$ , pro který  $L_u = L(U)$ . **Důkaz:** Popíšeme  $U$  jako vícepáskový Turingův stroj.

- Přechody  $M$  jsou napsány na první pásce spolu s řetězcem  $w$ .

- Na druhé pásce simulujeme výpočet  $M$ , používající formát

- **TODO**

Operace  $U$  jsou následující:

- Otestuj, zda je kód  $M$  legitimní; pokud ne,  $U$  zastav bez přijetí.
- Inicializuj druhou pásku kódovaným slovem  $w$ : 10 pro 0 ve  $w$ , 100 pro 1, blank jsou nechané prázdné a nahrazeny 1000 pouze v případě potřeby.
- Napiš 0, počáteční stav  $M$ , na třetí pásku. Posuň hlavu druhé pásky na první simulované políčko.
- Simuluj jednotlivé přechody  $M$ 
  - Najdi na první pásce správnou transakci  $0^i 10^j 10^k 10^l 10^m$ ,  $0^i$  na pásce 3,  $0^j$  na pásce 2.
  - Změň obsah pásky 3 na  $0^k$ .
  - Nahraď  $0^j$  na 2. pásce řetězcem  $0^l$ . Použij čtvrtou 'scratch tape' pro správné mezery.
  - Posuň hlavu 2. pásky na pozici vedle 1 vlevo nebo vpravo, podle pohybu  $m$ .
- pokud **TODO**

□

The End