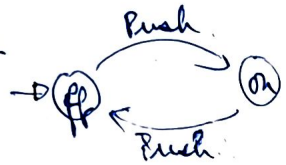
Def: Deterministický konečný automat (DFA) je $A=(Q, \Sigma, \delta, q_0, F)$, kde

- Q ... konečná množina stavů
- Σ ... konečná neprázdná abeceda
- δ ... přechodová funkce $\delta: Q \times \Sigma \to Q$
- q_0 ... počáteční stav $q_0 \in Q$ ... vede do něj šipka "odnikud" →
- F ... neprázdná množina koncových stavů $F \subseteq Q$

bývá jen 1, jinak uděláme nový p.s. a z něj hrany do všech ostatních

Ukázka



neformální

↳ kreslíme 

👁 Nedává smysl nemít koncový stav

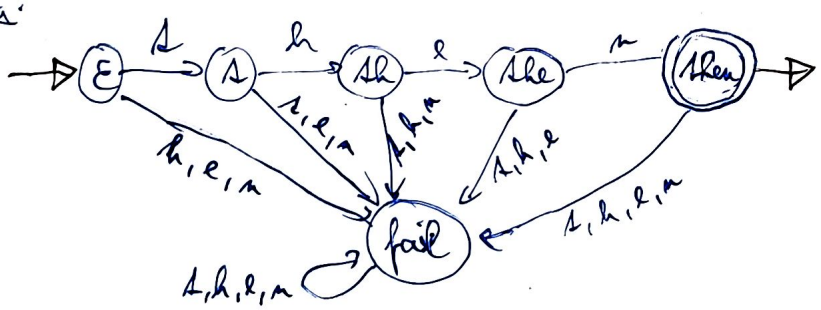
👁 přechodová funkce není úplná (def. obor není celá $Q \times \Sigma$)

Úmluva:

1, Pokud pro $q \in Q$ a $s \in \Sigma$ není definovaný přechod, tak přidáme nový stav 'fail' a dodefinujeme $\delta(q, s) := fail$.

2, Pokud je F prázdná, tak přidáme nový finální stav 'final' a přechody $\forall s \in \Sigma: \delta(final, s) := final$.

Ukázka:



Kdyby 'then' nebyl koncový:

Def: Pro neprázdnou množinu symbolů Σ definujeme

- slovo je libovolná posloupnost symbolů z Σ ... slovo je $x \in \Sigma^*$
  ↳ ε ... prázdné slovo
- Σ^* := množina všech slov ... $\Sigma^* := \bigcup_{n \in \mathbb{N}} \Sigma^n$
- Σ^+ := množina všech neprázdných slov $\Sigma^+ := \Sigma^* \setminus \{\varepsilon\}$
- jazyk je libovolná $L \subseteq \Sigma^*$

👁 jazyk je vždy spočetný

👁 jazyků je nespočetně moc

Def: Nad slovy Σ^* definujeme operace:

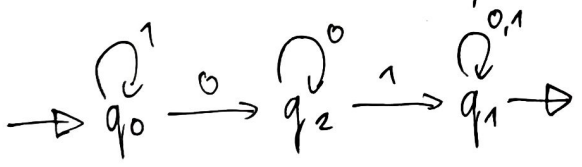
1) zřetězení slov: $u.v$ nebo uv
2) mocnina: u^n ($u^0 := \varepsilon$, $u^1 := u$, $u^{n+1} := u^n . u$)
3) délka slova: $|u|$ ($|\varepsilon| = 0$, $|\text{auto}| = 4$)
4) # výskytů znaku $s \in \Sigma$ ve slově u: $|u|_s$ ($|\text{kmvrlina}|_z = 2$)

• Regulární jazyky

Def: Mějme přechodovou fci $\delta : Q \times \Sigma \to Q$. Rozšířenou přechodovou fci $\delta^* : Q \times \Sigma^* \to Q$ definujeme induktivně:

1) $\delta^*(q, \varepsilon) := q$
2) $\delta^*(q, wx) := \delta(\delta^*(w), x)$, kde $w \in \Sigma^*$ a $x \in \Sigma$

→ prostě tranzitivní uzávěr δ
→ když budeme psát δ aplikované na slova, tak myslíme δ^*



$\delta^*(q_0, 1100) = q_2$

Def: Jazyk rozpoznávaný (přijímaný) DFA $A = (Q, \Sigma, \delta, q_0, F)$ je jazyk

$$L(A) := \{ w \in \Sigma^* \mid \delta^*(q_0, w) \in F \}.$$

Slovo w je přijímáno automatem $A \equiv w \in L(A)$.

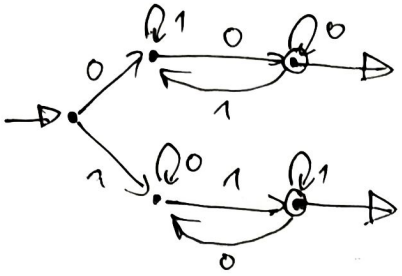
Def: Jazyk L je rozpoznatelný $\equiv \exists$ det. konečný automat A t.ř. $L(A) = L$.

Třídu jazyků rozpoznatelných konečnými automaty označíme $\mathcal{F}$.
Těmto jazykům říkáme regulární jazyky.

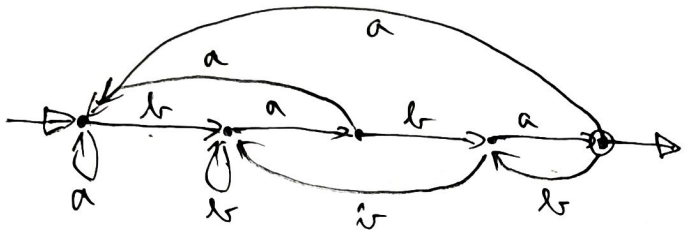
Automaty jsou konečné ⇒ je jich spočetně mnoho } většina jazyků nemá
Jazyků je nespočetně mnoho } žádný automat

Příklady regulárních jazyků

1) $L = \{w \in \{0,1\}^* \mid w = xux,\ x \in \{0,1\},\ u \in \{0,1\}^*\}$



2) $L = \{w \in \{a,b\}^* \mid w = ubaba,\ u \in \{a,b\}^*\}$

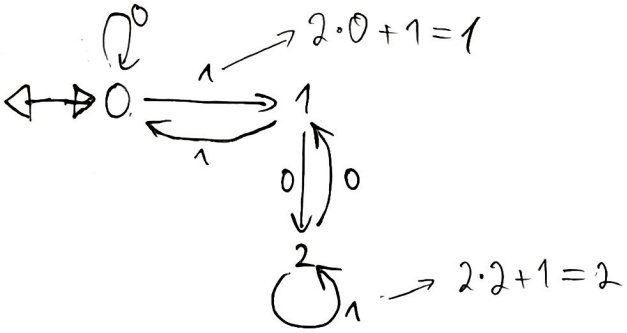


3) $L = \{w \in \{0,1\}^* \mid w$ je binární zápis čísla dělitelného 3$\}$

1010 ... slova čtena →

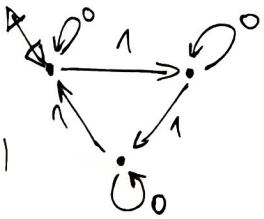
1 ~ shift left + 1

0 ~ shift left + 0



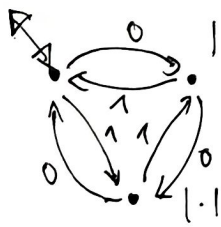
Jaký jazyk přijímá tento automat?

1)



přijímá slova z 0 a 1, kde #1 mod 3 = 0

2)

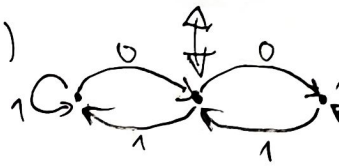


#0 = #1 (mod 3)

Když je za sebou { 00 → dostanu se doprava ; 11 → dostanu se doleva

↳ jedno, co předtím

3)



→ kontrola, jestli je slovo v L(A): najdu poslední výskyt 00/11, potom se opakování 1010101/01010

→ musí to končit a začínat na stejný znak

• Pumping lemma pro regulární jazyky

Motivace: Když je v automatu cyklus, tak ho lze opětovat

→ automat má 4 stavy

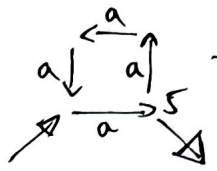
→ přijímá $a^5 = aaaaa$

? patří tyto slova do jazyka?

a^{13} a^{14} a^{15} a^{16} a^{17} a^{18} a^{19}

· □ □ □ (·) □ □

△ · △ △ (·) △ △



□: → přijímá: 1, 5, 9, 13, 17, ...

△: 5, 8, 11, 14, 17, ...

—: 5, 7, ... 17, ...

→ určitě přijímá a^{17}, respektive $5 + k \cdot 12$

Lemma: Nechť L je regulární jazyk. Pak $\exists n \in \mathbb{N}$ t.ž. $\forall w \in L$, $|w| \geq n$ lze rozdělit na tři části, $w = xyz$, že

i) $y \neq \varepsilon$

ii) $|xy| \leq n$

iii) $xy^k z \in L$ ... pro $\forall k \in \mathbb{N}_0$

Dk: L je regulární, tedy existuje DFA $A = (Q, \Sigma, \delta, q_0, F)$ t.ž. $L(A) = L$.

Označme n #stavů A. Nyní nechť $w = a_1 a_2 \dots a_m \in L$, $m \geq n$.

→ definujme $p_i := \delta^*(q_0, a_1 a_2 \dots a_i)$ a $p_0 := q_0$.

→ ∵ $m \geq n$, tak máme alespoň $n+1$ p_i ale jenom n stavů

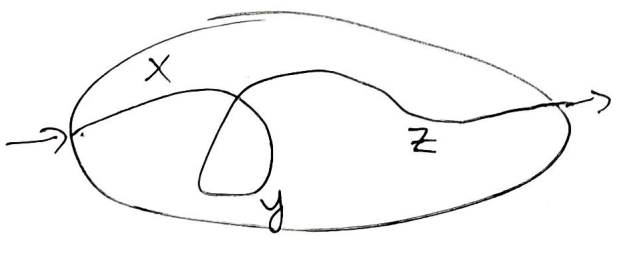
⇒ z principu holubníku $\exists i, j : 0 \leq i < j \leq n : p_i = p_j$

→ definujme $x := a_1 a_2 \dots a_i$

$y := a_{i+1} \dots a_j$ 👁 $|xy| \leq n$ $|y| \geq 1$

$z := a_{j+1} \dots a_m$ 👁 $xyz = w$

→ $p_0 \xrightarrow{x} p_i \xrightarrow{z} p_m$ → 👁 smyčka nad p_i se může opětovat libovolně krát ▨



• Využití pumping lemmatu

→ dokazuj že daný jazyk není regulární

1) $L = \{w \mid |w|_0 = |w|_1\}$ ... slova se stejným počtem 0 a 1

→ kdyby byl regulární, tak máme nějaké n z PL

$\Rightarrow w = 0^n 1^n \in L$, $w = xyz$, $|xy| \leq n$

$\Rightarrow xy$ = samé nuly, $|y| > 1$

→ z PL je $xy^0z = xz \in L$, ale sám je o 1 nulu míň

2) $L = \{0^n 1^n \mid n \geq 0\}$ není reg.

3) $L = \{1^p \mid p$ je prvočíslo $\}$ není reg.

→ zvolme n z PL a prvočíslo $p \geq n+2$, $w := 1^p$

→ rozložíme $w = xyz$, $m := |y| \Rightarrow |xz| = p - m$

→ $xy^{p-m}z \in L$, ale

$|xy^{p-m}z| = |xz| + (p-m)|y| = p - m + (p-m) \cdot m = (p-m)(m+1)$

→ což není prvočíslo ∵ $p - m > 1$ & $m + 1 > 1$

• Podtrhávací pumping lemma

→ v řetězu slov podtrhneme nějaká písmenka

Lemma: Nechť L je regulární, pak $\exists n \in \mathbb{N}$ t.ž. $\forall w \in L$, $|w|_{\_} \geq n$

$\Rightarrow w = xyz$, $|y|_{\_} > 0$ & $|xy|_{\_} \leq n$

Př: dlouhý = slabý

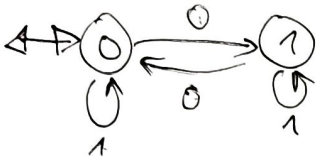
holubové = stavy do kterých jsem se dostal podtržením písmenka

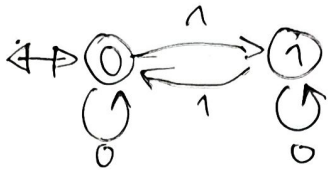
• Součin automatů

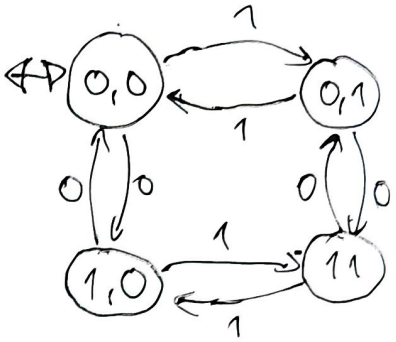
$\varphi_1 = |w|_0 = 2k$, $k \in \mathbb{N}_0$

$\varphi_2 = |w|_1 = 2l$, $l \in \mathbb{N}_0$

$L_1 = \{w \in 2^* \mid \varphi_1(w)\}$ $L_2 = \{w \in 2^* \mid \varphi_2(w)\}$ $L = \{w \in 2^* \mid \varphi_1(w) \& \varphi_2(w)\}$







Def: Automaty $A_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$
$A_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$

Součin automatů A_1, A_2 je automat

$$A = (Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}), F_1 \times F_2)$$

$$\delta((q_1, q_2), x) := (\delta_1(q_1, x), \delta_2(q_2, x))$$

Def (dosažitelné stavy). Mějme DFA $A = (Q, \Sigma, \delta, q_0, F)$. Řekneme, že

$q \in Q$ je dosažitelný $\equiv \exists w \in \Sigma^* : \delta^*(q_0, w) = q$

Algoritmus: hledání dosažitelných stavů

→ iterativně

1, $M_0 = \{q_0\}$

2, $M_{i+1} = M_i \cup \{q \in Q \mid \exists p \in M_i \ \& \ \exists x \in \Sigma : \delta(p, x) = q\}$

3, opakuj dokud $M_{i+1} \neq M_i$

Dk:

korektnost: $M_0 \subseteq M_1 \subseteq \dots \subseteq Q$ a sudé M_i obsahuje jen dos. stavy

úplnost: Nechť q je dosažitelný a $w = x_1 x_2 \dots x_n$ je nejkratší w t.ž. $\delta^*(q_0, w) = q$.

→ zřejmě $\delta^*(q_0, x_1, \dots x_i) \in M_i$, speciálně $\delta^*(q_0, w) = q \in M_n$ ∎

• Myhill-Nerodova věta

Def (kongruence). Nechť Σ je konečná abeceda a $\sim$ je ekvivalence na Σ^*.

1, $\sim$ je pravá kongruence $\equiv \forall u,v,w \in \Sigma^*: u \sim v \Rightarrow uw \sim vw$

2, $\sim$ je konečného indexu $\equiv$ rozklad $\Sigma^*/\sim$ má konečný počet tříd

↳ třídu kongruence (ekvivalence) slova u značíme $[u]_\sim$.

Příklad:

1, $\sim_{END}$ "končí stejným písmenem" je pravá kongruence

2, $\sim$ "končí stejně jako začíná" není ∵ $aa \sim bb$, ale $aaa \not\sim bba$

Věta (Myhill-Nerodova): Jazyk L nad konečnou abecedou Σ je regulární

$\Leftrightarrow \exists$ pravá kongruence $\sim$ konečného indexu nad Σ^* t.ž.

L je sjednocením jistých tříd rozkladu $\Sigma^*/\sim$.

Důk: $\Rightarrow$: máme automat, chceme kongruenci

→ $u \sim v \equiv \delta^*(q_0,u) = \delta^*(q_0,v)$

↳ přirozená kongruence, konečný index ∵ konečně mnoho stavů

→ L je sjednocení nějakých tříd $\sim$

$\Rightarrow L = \{w \mid \delta^*(q_0,w) \in F\} = \bigcup_{q \in F} \{w \mid \delta^*(q_0,w) = q\}$

↳ z ∀ neprázdné takové množiny vybereme reprezentanta a sjednotíme $[w]_\sim$

$\Leftarrow$: kongruence $\Rightarrow$ automat.

→ abeceda Σ

→ stavy $Q := \Sigma^*/\sim$

→ $q_0 := [\lambda]$ ... prázdné slovo

→ $F = \{C_1, \dots C_n\}$ t.ž. $L = UF = \bigcup_i C_i$

→ $\delta([u],x) := [ux]$ $\quad \delta^*([\lambda],w) = [w]$

$\Rightarrow$ nějaký akceptor $L(A) = L$ .... $w \in L \Leftrightarrow [w] \in F$

$w \in L \Leftrightarrow w \in \bigcup_i C_i \Leftrightarrow w \in C_1 \vee \dots \vee w \in C_n \Leftrightarrow [w] = C_1 \vee \dots \vee [w] = C_n$

$\Leftrightarrow [w] \in F \Leftrightarrow w \in L(A)$

• Myhill-Nerodova věta

Def (kongruence). Nechť Σ je konečná abeceda a $\sim$ je ekvivalence na Σ^*.

1, $\sim$ je pravá kongruence $\equiv \forall u,v,w \in \Sigma^*: u \sim v \Rightarrow uw \sim vw$

2, $\sim$ je konečného indexu $\equiv$ rozklad $\Sigma^*/\sim$ má konečný počet tříd

↳ třídu kongruence (ekvivalence) slova u značíme $[u]_\sim$.

Příklad:

1, $\sim_{END}$ "končí stejným písmenem" je pravá kongruence

2, $\sim$ "končí stejně jako začíná" není ∵ $aa \sim bb$, ale $aaa \not\sim bba$

Věta (Myhill-Nerodova): Jazyk L nad konečnou abecedou Σ je regulární

$\Leftrightarrow \exists$ pravá kongruence $\sim$ konečného indexu nad Σ^* t.ž.

L je sjednocením jistých tříd rozkladu $\Sigma^*/\sim$.

Důk: $\Rightarrow$: máme automat, chceme kongruenci

$\rightarrow u \sim v \equiv \delta^*(q_0,u) = \delta^*(q_0,v)$

↳ přirozená kongruence, konečný index ∵ konečně mnoho stavů

$\rightarrow L$ je sjednocení nějakých tříd $\sim$

$\Rightarrow L = \{w \mid \delta^*(q_0,w) \in F\} = \bigcup_{q \in F} \{w \mid \delta^*(q_0,w) = q\}$

↳ z ∀ neprázdné takové množiny vyberu reprezentanta a sjednotím $[w]_\sim$

$\Leftarrow$: kongruence $\Rightarrow$ automat.

$\rightarrow$ abeceda Σ

$\rightarrow$ stavy $Q := \Sigma^*/\sim$

$\rightarrow q_0 := [\lambda]$ ... prázdné slovo

$\rightarrow F = \{C_1, \dots C_n\}$ t.ž. $L = \bigcup F = \bigcup_i C_i$

$\rightarrow \delta([u],x) := [ux]$

$\Rightarrow$ nějaký akceptor $L(A) = L$ .... $w \in L \Leftrightarrow [w] \in F$

$\delta^*([\lambda],w) = [w]$

$w \in L \Leftrightarrow w \in \bigcup_i C_i \Leftrightarrow w \in C_1 \vee \dots \vee w \in C_n \Leftrightarrow [w] = C_1 \vee \dots \vee [w] = C_n$

$\Leftrightarrow [w] \in F \Leftrightarrow w \in L(A)$

Příklad: Neregulární pumpovatelný jazyk

$L = \{ w \mid w = a^+ b^i c^i \vee w = b^i c^j, +\in N^+, i,j \in N \}$

→ vždy lze pumpovat první písmena

→ kdyby byl regulární, tak ∃ pravá kongruence ~ konečného indexu m, t.ž. L je sjednocením některých tříd $\Sigma^*/\sim$

⇒ vezmu množinu reprezentů $\{ab, abb, abbb, \dots ab^{m+1}\}$

↳ je jich $m+1$, ale tříd je m

⇒ $\exists i \neq j$: $ab^i \sim ab^j$

$ab^i c^i \sim ab^j c^i$ ... pravá kongruence

→ $ab^i c^i \in L \Rightarrow [ab^i c^i] \subseteq L \Rightarrow ab^j c^i \in L$

– ale $i \neq j$, takže $ab^j c^i \notin L$ ▨

• Ekvivalence automatů

☼ stejný jazyk přijímá více automatů

Def (homomorfismus): Nechť A_1, A_2 jsou DFA.

Zobrazení $h: Q_1 \to Q_2$, Q_1 na Q_2 je homomorfismem ≡

i) $h(q_{01}) = q_{02}$ ... stejné počáteční stavy

ii) $h(\delta_1(q,x)) = \delta_2(h(q), x)$ ... přech. fce

iii) $q \in F_1 \Leftrightarrow h(q) \in F_2$ ... koncové stavy

Pokud je h navíc i prosté (tedy bijekce), tak to je isomorfismus.

Def: Automaty A, B nad stejnou abecedou Σ jsou ekvivalentní ≡ $L(A) = L(B)$.

Věta: Existuje homomorfismus $A_1, A_2 \Rightarrow A_1, A_2$ jsou ekvivalentní.

Dk: ☼ $h(\delta_1^*(q_1, w)) = \delta_2^*(h(q), w)$

$w \in L(A_1) \Leftrightarrow \delta_1^*(q_{01}, w) \in F_1 \Leftrightarrow h(\dots) \in F_2 \Leftrightarrow \delta_2^*(h(q_{01}), w) \in F_2$

$\Leftrightarrow \delta_2^*(q_{02}, w) \in F_2 \Leftrightarrow w \in L(A_2)$ ▨

Def: Stavy $p, q \in Q$ jsou **ekvivalentní** $\equiv$

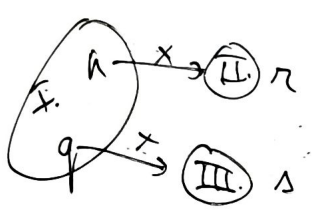
$$\forall w \in \Sigma^*: \delta^*(p, w) \in F \iff \delta^*(q, w) \in F$$

Pokud nejsou ekviv. jsou **rozlišitelné**.

(0) ekvivalence na stavech je kongruencí

Algoritmus: hledání rozlišitelných stavů

1. pokud $p \in F$ a $q \notin F$, tak jsou p, q rozlišitelné
2. Pokud $p, q \in Q$, $x \in \Sigma$, $\delta(p, x) = r$, $\delta(q, x) = s$,



r, s jsou rozlišitelné
$\Rightarrow p, q$ jsou také rozlišitelné

3. opakuj dokud existuje nová dvojice p, q, x

Korektnost: Nechť existují nějaké rozlišitelné páry, co alg. nerozlišil
→ vezměme pár p, q rozlišitelný nejkratším slovem $w = a_1 \dots a_n$
→ stavy $r := \delta(p, a_1)$, $s := \delta(q, a_1)$ jsou rozlišitelné slovem $a_2, \dots a_n$,
což je kratší než $a_1, \dots a_n$, takže je alg. rozlišil.
$\Rightarrow$ takže v dalším kroku rozliší i p, q.

Složitost:

- v 1 kole uvažujeme všechny páry $\Rightarrow n^2 \cdot |\Sigma|$
- kol nejvýše n^2 ... v jednom kole rozlišíme jen jeden stav

$$\Rightarrow O(n^2 |\Sigma|)$$

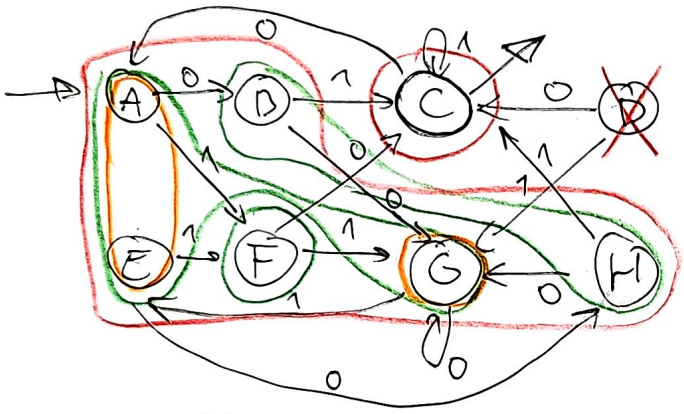
Def: DFA je redukovaný $\equiv$

1) nemá nedosažitelné stavy
2) žádné dva stavy nejsou ekvivalentní
3) δ je totální

Def: Automat B je reduktem automatu $A \equiv B$ je redukt. a jsou ekviv.

Algoritmus: nalezení reduktu

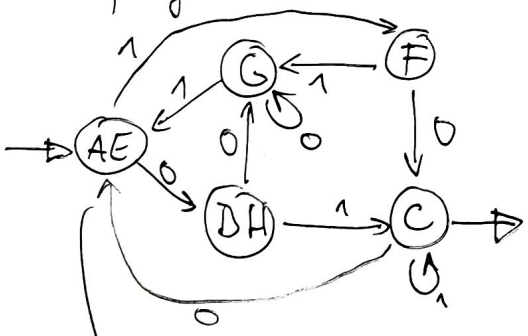
1) odstraníme nedosažitelné stavy



I.
II. F(0), BH(1), AE G
III. BH nerozlišíme
$G \xrightarrow{0} G$
$AE \xrightarrow{0} BH$

2) udělám rozklad stavů na třídy ekvivalence: AE, BH, C, F, G

3) Vyrobíme redukt B na třídách ekvivalence



přechodová fce B ... γ

$\gamma(S,x)$, $S \in Q_B \subseteq Q$ $x \in \Sigma$

$\rightarrow q \in S \Rightarrow p := \delta(q,x)$

$\Rightarrow \gamma(S,x) := [p]_\equiv$ ... třída ekvivalence

$\rightarrow$ počáteční stav = třída obsahující $q_0 \in Q$

množina koncových stavů = třídy jejichž sjednocením dostanu F

Lemma: Každé dva redukované ekviv. automaty jsou izomorfní

Dk: Sestrojíme prostý homomorfismus $h: Q_1 \rightarrow Q_2$.

- $\forall q \in Q_1$ je dosažitelný $\Rightarrow \exists$ slovo w t.ž. $\delta_1^*(q_{01}, w) = q$

$\Rightarrow$ definuji $h(q) := \delta_2^*(q_{02}, w)$

$\rightarrow$ není sporné dokázat: $h(q_{01}) = q_{02}$, $h(\delta_1(q,x)) = \delta_2(h(q),x))$, $q \in F_1 \Leftrightarrow h(q) \in F_2$

$\rightarrow$ je to bijekce z redukovanosti těch automatů

Důsledek: Pro každý DFA existuje jednoznačně určený redukt, až na izomorfismus.

Algoritmus: testování ekvivalence dvou DFA (resp. regulárních jazyků)

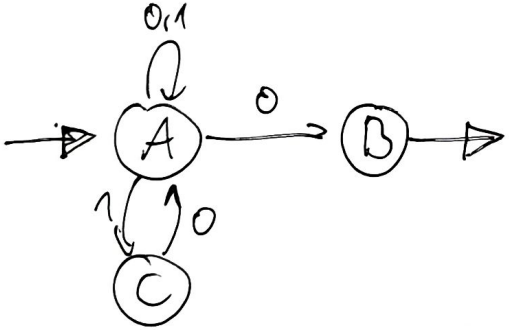
1. oba automaty zredukujeme → ~~minimální~~
2. Vytvoříme nový automat sjednocením stavů těch dvou reduktů, předpokládáme $Q_1 \cap Q_2 = \emptyset$

$$A = (Q_1 \cup Q_2, \Sigma, \delta_1 \cup \delta_2, q_{01}, F_1 \cup F_2)$$

↳ zvolíme libovolně q_{01} nebo q_{02}

3. původní automaty jsou ekvivalentní
   $\Leftrightarrow$ q_{01}, q_{02} jsou ekvivalentní v A

• **Redukce nedet-FA.**



přijímá slova končící na 0

→ C lze vypustit

→ ale alg. A, C rozliší vstupem 0

Nedeterministické FA s λ přechody

, λ, nebo ε používá se obojí

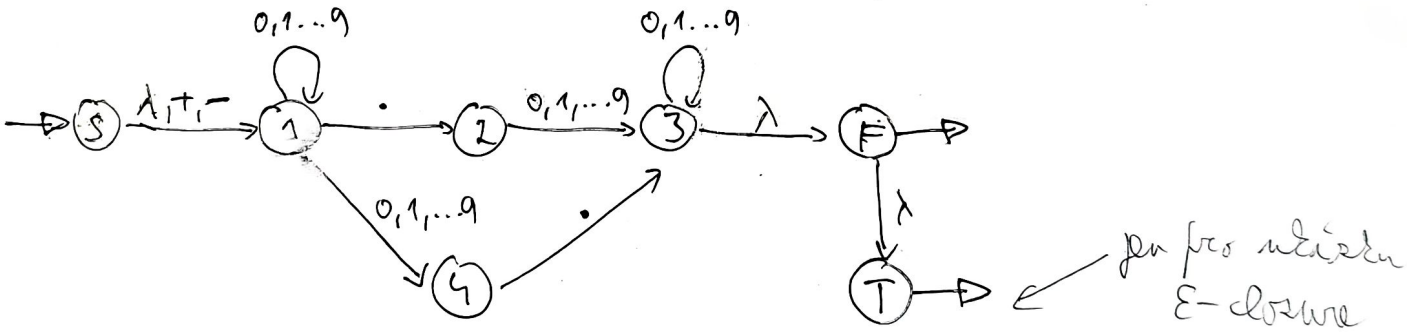
Def: λ-NFA je $A=(Q,\Sigma,\delta,q_0,F)$ t.ř.

1) Q ... konečná množina stavů
2) Σ ... konečná množina vstupních symbolů
3) $\delta: Q\times(\Sigma\cup\{\lambda\}) \to \mathcal{P}(Q)$ ... přechodová fce
4) $q_0 \in Q$ ... počáteční stav
5) $F \subseteq Q$ ... koncové stavy

Pozn.: Může mít množina počátečních stavů $S \subseteq Q$
$\Rightarrow$ udělám nový stav q_0 a přechody $\delta(q_0,\lambda)=S$.

Příklad: Automat generující desetinná čísla

- volitelně znaménko + nebo − nebo nic
- 123.456, .456, 456. , ale ne .



Def: Pro $q\in Q$ definujeme ε-uzávěr, ε-closure(q) induktivně:

1) $q \in \varepsilon\text{-closure}(q)$
2) $p \in \varepsilon\text{-closure}(q)$ & $r\in\delta(p,\varepsilon) \Rightarrow r \in \varepsilon\text{-closure}(q)$

Pro $S\subseteq Q$ definujeme $\varepsilon\text{-closure}(S) := \bigcup_{q\in S} \varepsilon\text{-closure}(q)$

Příklady: ε-closure

S ... $\{S,1\}$
1 ... $\{1\}$
3 ... $\{3,F,T\}$
$\{3,4\}$ ... $\{4,3,F,T\}$

$O(n^3)$ → n stavů, pro každý až n^2 hran s ε-přechody

Def (δ^* pro ε-NFA): Pro ε-NFA $(Q, \Sigma, \delta, q_0, F)$ definujeme δ^* jako

1. $\delta^*(q, \varepsilon) := \varepsilon\text{-closure}(q)$ → pro DFA: $\delta(\delta^*(q, w), x)$
2. $\delta^*(q, wx) := \varepsilon\text{-closure}\left(\bigcup \{\delta(p, x) \mid p \in \delta^*(q, w)\}\right)$

Příklad:
$\delta^*(S, \varepsilon) = \{S, q_1\}$
$\delta^*(S, 5) = \varepsilon\text{-cl}(\delta(S, 5) \cup \delta(q_1, 5)) = \{q_1, q_4\}$
$\delta^*(S, 5.) = \varepsilon\text{-cl}(\delta(q_1, .) \cup \delta(q_4, .)) = \{q_2, q_3, F, T\}$
$\delta^*(S, 5.6) = \varepsilon\text{-cl}(\delta(q_2, 6) \cup \delta(q_3, 6) \cup \emptyset \cup \emptyset) = \{q_3, F, T\}$

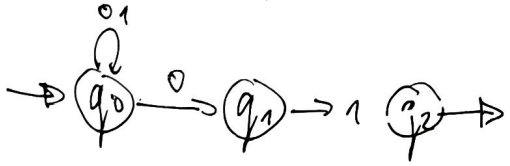
Def (Jazyk ε-NFA): Pro ε-NFA $(A, \Sigma, \delta, q_0, F)$ je

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

• Převod NFA na DFA

Def: Konfigurace ε-NFA $(Q, \Sigma, \delta, q_0, F)$ nacházejícího se ve stavu q s nepřečteným vstupem $x \in \Sigma^*$ je dvojice (q, x).

Def: $A = (Q, \Sigma, \delta, q_0, F)$, vstupní slovo $w \in \Sigma^*$. Vrcholy výpočetního grafu jsou konfigurace A nad slovem w. $(p, xu) \to (q, u) \equiv q \in \delta(p, x)$



$(q_0, 00101) \xrightarrow{0} (q_0, 0101) \xrightarrow{0} (q_0, 101) \xrightarrow{1} (q_0, 01) \xrightarrow{0} (q_0, 1) \xrightarrow{1} (q_2, \varepsilon)$
$\downarrow 0$ $(q_1, 0101)$
$\downarrow 0$ $(q_1, 101)$
$\downarrow 1$ $(q_2, 01)$ STUCK
$\downarrow 0$ $(q_1, 1) \xrightarrow{1} (q_2, \varepsilon)$

Věta: L je rozpoznatelný ε-NFA $\Leftrightarrow$ je reg.

Alg: Pro ε-NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ sestrojujeme DFA $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$

→ nové stavy = ε-uzavřené podmnožiny těch starých
$\Rightarrow \forall S \subseteq Q_N$: $\varepsilon\text{-closure}(S) \in Q_D \Rightarrow Q_D \subseteq \mathcal{P}(Q_N)$, může být $\emptyset \in Q_D$.

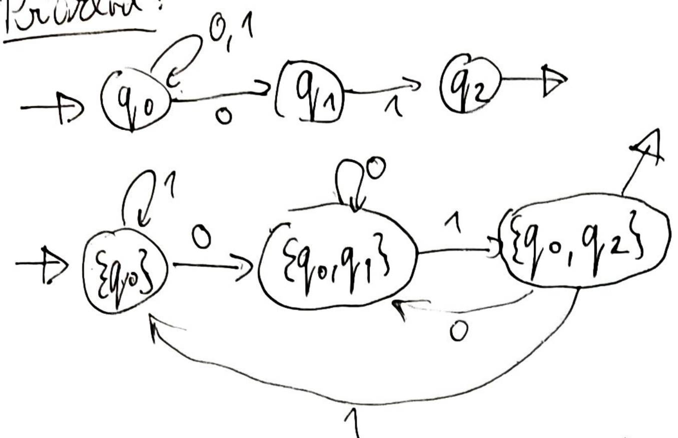
- počáteční stav $q_D := \varepsilon\text{-closure}(q_0)$
- přijímající = množiny obsahující nějaký přijímací stav: $F_D := \{S \in Q_D \mid S \cap F_N \neq \emptyset\}$
- přechod. fce mě musí dostat do množiny všech stavů, kde bych mohl být

$$\delta_D(S, x) := \varepsilon\text{-closure}\left(\bigcup \{\delta_N(q, x) \mid q \in S\}\right)$$

Korektnost: Indukcí $\delta_N^*(q_0, w) = \delta_D^*(q_{D_0}, w)$

$O(2^n \cdot n^2)$ (2^n stavy, n^2 ε-closure)

Příklad:



	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$\leftarrow \{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\leftarrow \{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\leftarrow \{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$\leftarrow \{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

(nekreslím nedosažitelné stavy)

Množinové operace nad jazyky

Def: Pro jazyky L, M definujeme
- sjednocení, průnik, rozdíl tak jako by člověk čekal
- absolutní doplněk: $\overline{L} = -L := \{w \in \Sigma^* \mid w \notin L\} = \Sigma^* - L$

Lemma (de Morgan):

$$L \cap M = \overline{\overline{L} \cup \overline{M}}$$
$$L \cup M = \overline{\overline{L} \cap \overline{M}}$$
$$L - M = L \cap \overline{M}$$

Věta: Regulární jazyky jsou uzavřené na množinové operace. $\cup, \cap, -, \overline{}$

Dk: 1) doplněk. Pro L máme DFA $A = (Q, \Sigma, \delta, q_0, F)$

→ prohlásíme $F_{complement} := Q - F$

→ ale pokud δ není totální, tak nejprve přidáme nový stav q_{FAIL} a do něj přechod pro vše dříve nedef.

↳ navíc $\delta(q_{FAIL}, x) = q_{FAIL}$, $\forall x \in \Sigma$

2) $A_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ $\quad A_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$

→ $A := (Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}), F)$

↳ $\delta((p_1, p_2), x) := (\delta_1(p_1, x), q_2(p_2, x))$ → tomuto jsme předtím říkali součin autom.

- průnik $L(A_1) \cap L(A_2)$ → $F = F_1 \times F_2$
- sjednocení $L(A_1) \cup L(A_2)$ → $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$
- rozdíl $L(A_1) - L(A_2)$ → $F = F_1 \times (Q_2 - F_2)$

! nejprve opět doplníme přechodovou fci na totální pomocí q_{FAIL} ■

Příklady

① $L = \{w \in 2^* \mid |w|_1 = 3k+2 \ \& \ w \text{ neobsahuje } '11' \text{ jako podslovo}\}$

$\Rightarrow L_1 := \{w \in 2^* \mid |w|_1 = 3k+2\}$

$L_2 := \{w \in 2^* \mid w = u11v,\ u,v \in \Sigma^*\}$ $\Rightarrow L = L_1 - L_2$

② $L = \{w \mid |w|_0 \neq |w|_1\}$ není regulární

↳ z pumping lemma víme $\overline{L} = \{w \mid |w|_0 = |w|_1\}$ není reg. ▨

③ $L_{0\neq1} = \{0^i 1^j \mid i \neq j\}$ není reg

- jazyk $L_{01} := \{0^i 1^j\}$ je reg.

→ ale $L_{01} - L_{0\neq1} = \{0^i 1^i\}$ není reg.

↳ kdyby byl $L_{0\neq1}$ reg. Tak máme spor

Def (řetězcové operace): Nad jazyky L, M definujeme

- zřetězení $L.M := \{uv \mid u \in L \ \& \ v \in M\}$
- mocnina $L^0 := \{\lambda\}$, $L^{i+1} := L^i.L$
- pozitivní iterace $L^+ := L^1 \cup L^2 \cup \ldots = \bigcup_{i \in \mathbb{N}^+} L^i$
- obecná iterace $L^* := L^+ \cup \{\lambda\} = \bigcup_{i \in \mathbb{N}} L^i$
- otočení jazyka $L^R := \{u^R \mid u \in L\}$, $(x_1 x_2 \ldots x_m)^R := x_m \ldots x_2 x_1$
- levý kvocient L podle M: $M \setminus L := \{v \mid uv \in L \ \& \ u \in M\}$

↳ ze slov L odstraním zleva slova z M

- pravý kvocient L podle M: $L / M := \{u \mid uv \in L \ \& \ v \in M\}$
- levá derivace L podle w: $\partial_w L := \{w\} \setminus L$
- pravá derivace L podle w: $\partial^R_w L := L / \{w\}$

Věta Jsou-li L, M reg., jsou regulární i $L.M, L^*, L^+, L^R, M \setminus L, L / M$.

→ důkaz postupně ↳ uzavřenost na operace

Lemma: $L.\Pi$ je reg.

Dk: $A_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ $\quad A_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

$\Rightarrow$ nadělíme NFA $B = (Q_1 \cup Q_2 \cup \{q_0\}, \Sigma, \delta, q_0, F_2)$

- q_0 je nový stav

↳ $\delta(q_0, \varepsilon) := \begin{cases} \{q_1, q_2\}, & q_1 \in F_1 \quad \Leftrightarrow \varepsilon \in L(A_1) \\ \{q_1\}, & q_1 \notin F_1 \quad \Leftrightarrow \varepsilon \notin L(A_1) \end{cases}$

$\delta(q_0, x) := \emptyset, \quad x \in \Sigma$

$\rightarrow \delta(q, x) := \begin{cases} \{\delta_1(q,x)\} & \dots q \in Q_1 \ \& \ \delta_1(q,x) \notin F_1 \\ \{\delta_1(q,x), q_2\} & \dots q \in Q_1 \ \& \ \delta_1(q,x) \in F_1 \\ \{\delta_2(q,x)\} & \dots q \in Q_2 \end{cases}$ ▨

Lemma: L^*, L^+ jsou reg.

Dk: idea: opakovaný výpočet automatu $A = (Q, \Sigma, \delta, q_0, F)$

↳ nedet. rozhodnem zda chceme pokračovat nebo restart

pro L^* navíc potřebujeme speciální stav pro přijetí $\lambda \in L^*$

důkaz:

definujeme NFA $D := (Q \cup \{q_D\}, \Sigma, \delta_D, q_D, F \cup \{q_D\})$

$\delta_D(q_D, \lambda) := \{q_0\}$ ... q_D je pro přijetí ε

$\delta_D(q_D, x) := \emptyset, \quad x \in \Sigma$

$\delta_D(q, x) := \begin{cases} \{\delta(q,x)\} \dots q \in Q \ \& \ \delta(q,x) \notin F \\ \{\delta(q,x), q_0\} \dots \text{jinak} \end{cases}$

$\rightarrow$ pak 1) $q_D \in F_D \Rightarrow L(D) = L(A)^*$

2) $q_D \notin F_D \Rightarrow L(D) = L(A)^+$ ▨

Lemma: L reg $\Rightarrow L^R$ reg.

Dk: idea: nedeterministický backward-search

$\rightarrow$ def. nedet $R = (Q \cup \{q_D\}, \Sigma, \delta_D, q_D, \{q_0\})$

- $\delta_D(q, x) := \{p \mid \delta(p, x) = q\}, \quad q \in Q$
- $\delta_D(q_D, \varepsilon) := F$
- $\delta_D(q_D, x) := \emptyset$

▨

Lemma: L, M reg $\Rightarrow M \setminus L$, L / M reg.

Dk: Nejprve $M \setminus L$: Sledy $w \in M \setminus L \iff \exists u \in M : uw \in L$

→ vezmeme DFA $A = (Q, \Sigma, \delta, q_0, F)$

↳ definujeme NFA $B := (Q \cup \{q_\varepsilon\}, \Sigma, \delta_B, q_\varepsilon, F)$

idea: Automat pro L budeme číst u a stavech, kam lze dojít slovy z L

→ $\forall q \in Q: \delta_B(q, x) := \{\delta(q, x)\}$ ... stále jednoduše def.

→ $\delta_B(q_\varepsilon, \varepsilon) := \{q \in Q \mid \exists u \in M : q = \delta^*(q_0, u)\}$

$\Rightarrow w \in M \setminus L \iff \exists u \in M : uw \in L$

$\iff (\exists u \in M)(\exists q \in Q): \delta^*(q_0, u) = q \ \& \ \delta^*(q, w) \in F$

$\iff \exists q \in \delta_B(q_\varepsilon, \varepsilon) \ \& \ \delta^*(q, w) \in F$

$\iff w \in L(B)$

Nyní L / M: $L / M = (M^R \setminus L^R)^R$

$x_1 x_2 \dots x_m . y_1 y_2 \dots y_n \longrightarrow x_1 x_2 \dots x_m$

↳ $y_n \dots y_2 y_1 x_m \dots x_2 x_1 \longrightarrow x_m \dots x_2 x_1 \xrightarrow{R} x_1 x_2 \dots x_m$

Regulární výrazy

Def: $Reg\,E(\Sigma)$ bude nejmenší třída uzavřená na operace, které definují reg. výrazy se symboly z $\Sigma = \{a_1, a_2, \dots a_n\} \neq \emptyset$.

→ hodnotou (jazyk) reg. výrazu α značíme $L(\alpha)$.

Základ indukce:

$\alpha = \lambda \Rightarrow L(\alpha) = \{\lambda\}$ … prázdný řetězec

$\alpha = \cdot \Rightarrow L(\alpha) = \emptyset$ … prázdný reg. výraz

$\alpha = a \Rightarrow L(\alpha) = \{a\}$

Indukce:

$L(\alpha + \beta) := L(\alpha) \cup L(\beta)$ — graf 1

$L(\alpha\beta) := L(\alpha) . L(\beta)$

$L(\alpha^*) := L(\alpha)^*$

Priorita: $* > . > +$

Příklad: obtékající se nuly a jedničky: 0101, 10, 101, …

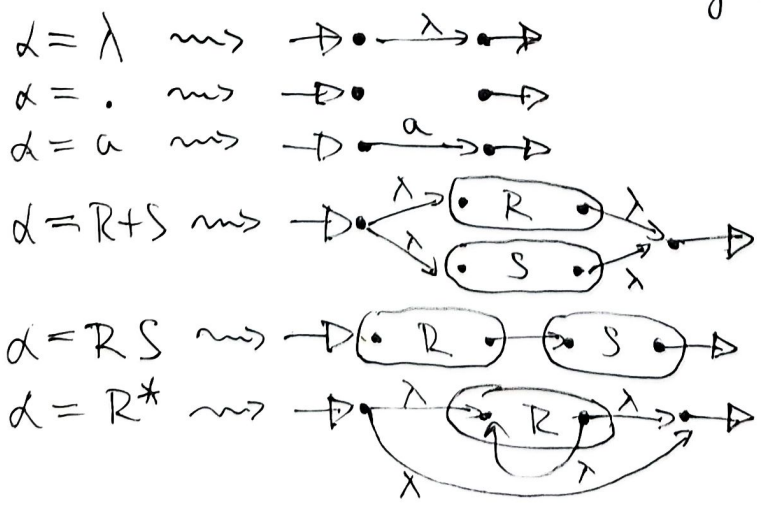
$(01)^* + (10)^* + 1(01)^* + 0(10)^*$

$(\lambda+1)(01)^*(\lambda+0)$ > ekviv.

Věta (Kleeneho): Jazyk L je regulární $\Leftrightarrow$ lze popsat regulárním výrazem.

Dk: $RegE \Rightarrow \exists NFA \Rightarrow \exists DFA$

- důkaz indukcí podle struktury R



Tyto automaty splňují
1) právě 1 koncový stav
2) 0 hrany do poč. stavu
3) 0 hrany z konc. stavu

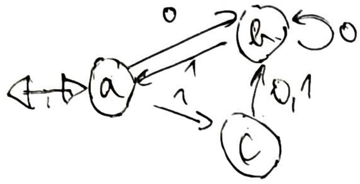
Převod DFA ⇝ Reg E – opačný směr

→ Floyd-Warshall – hledá sledy mezi všemi dvojicemi vrcholů

→ chceme najít sledy z počátečního stavu do koncových, co využívají všechny stavy

⇒ stavy očíslujeme, postupně budeme přidávat povolené stavy

$Q = \{1, 2, \dots n\}$ → vyrobíme n matic $n \times n$



$R^{\emptyset}$:

	a	b	c
a	λ	0	1
b	1	$\lambda+1$	·
c	··	$0+1$	λ

$i \neq j$: (i) $\xrightarrow{a_1, a_2, \dots a_m}$ (j) $R^{\emptyset}_{ij} = a_1 + a_2 + \dots + a_m$

$i = j$: (i) smyčka $a_1, \dots a_m$ $R^{\emptyset}_{ii} = \lambda + a_1 + a_2 + \dots + a_m$

indukce:

$$R^{N \cup \{x\}}_{ij} = R^{N}_{ij} + R^{N}_{i,x} (R^{N}_{x,x})^{*} R^{N}_{x,j}$$

R^N_{ij} ↙ to jsem to uměl předtím

$i \longrightarrow x \longrightarrow j$ (se smyčkou * v x)

N = povolené vrcholy
x = nový vrchol

→ nakonec: $\text{Reg } E := \bigcup_{p \in F} R^{Q}_{q_0, p}$

→ q_0 = počáteční
$p \in F$ = koncové
R^{Q} = mám všechny vrcholy

👁 záleží (pro rychlost) na pořadí přidávání vrcholů

→ na konci mě zajímají jen $R_{q_0, p}$

→ pro jejich vyhodnocení potřebuju nějaké vrcholy, ale ne všechny

→ když vrchol není ani koncový, ani počáteční, tak můžu jeho řádek po přidání do matice zapomenout

⇒ koncové + počáteční přidám až jako poslední

$R^{\{b\}}$:

	a	c
a	$\lambda + 0(\lambda+1)^{*}1$	$1 + 0(\lambda+1)^{*}\_$
c	$(0+1)(\lambda+1)^{*}1$	$\lambda + (0+1)(\lambda+1)^{*}\_$

≈

$R^{\{b\}}$:

	a	c
a	$\lambda + 01^{+}$	1
c	$(0+1)1^{+}$	λ

$R^{\{b,c\}}$:

	a
a	$\lambda + 01^{+} + 1(0+1)1^{+}$

(λ^{*})

$$R^{\{a,b,c\}}_{a,a} = \left(R^{\{a,b\}}_{a,a}\right)^{+} = \left(\lambda + (0 + 10 + 11)1^{+}\right)^{+}$$

Homomorfismus

→ substituce ~ vložení podprogramu do slova
→ homomorfismus ~ přeředění autobusu na jízdy

Def: Mějme jazyk L nad abecedou Σ.

- **Substituce** σ: přiřadí slova na jazyky

  $\forall a \in \Sigma: \sigma(a) = L_a$, jazyk abecedy Σ_a

  $\sigma(\lambda) = \{\lambda\}$

  $\sigma(a_1 \dots a_n) = \sigma(a_1) \dots \sigma(a_n)$

  $\sigma(L) := \bigcup \{\sigma(w) \mid w \in L\}$

  $\sigma: \Sigma^* \to \mathcal{P}\left(\left(\bigcup_{a \in \Sigma} \Sigma_a\right)^*\right)$

  → **nevypouštějící substituce** $\equiv \forall a: \lambda \notin \sigma(a)$

- **Homomorfismus** h: přiřadí slova na slova

  $\forall a \in \Sigma: h(a) = \{w_a\}$, píšeme $h(a) = w_a$, $w_a \in \Sigma_a^*$

  $h(\lambda) = \lambda$

  $h(a_1 \dots a_n) = h(a_1) \dots h(a_n)$

  $h(L) := \{h(w) \mid w \in L\}$

  $h: \Sigma^* \to \left(\bigcup_{a \in \Sigma} \Sigma_a\right)^*$

  → **nevypouštějící homomorfismus** $\equiv \forall a: h(a) \neq \lambda$

- **Inverzní homomorfismus**: přiřadí slova zpět

  $h^{-1}(L) := \{w \mid h(w) \in L\}$

Příklad ① Znaky nahradíme TEX zápisem: $h(\mu) = \backslash mu$

② $h(0) := ab$, $h(1) := \lambda$ $\Rightarrow h(0011) = abab$

$L = 10^*1 \Rightarrow h(L) = (ab)^*$

Věta (uzavřenost na homomorfismus): Je-li L reg. a pro $\forall x \in \Sigma$ je $\sigma(x)$ reg. ↳ nevypouštějící subst. $\Rightarrow$ hom. pak je $\sigma(L)$ také reg.

Dk: L je reg $\Rightarrow$ L lze popsat reg. výrazem → indukcí podle struktury Regé

$\sigma_R(\alpha+\beta) = \sigma(L(\alpha)) \cup \sigma(L(\beta))$ KLEENE

$\sigma_R(\alpha\beta) = \{\sigma(u)\sigma(v) \mid u \in L(\alpha) \,\&\, v \in L(\beta)\}$

$\sigma(L(\alpha)^*) = \sigma(L(\alpha)^0) \cup \sigma(L(\alpha)^1) \cup \dots =$

$= \sigma_R(\alpha)^0 \cup \sigma_R(\alpha)^1 \cup \dots = (\sigma_R(\alpha))^*$

→ prvočinitelé jsou reg (jistě.)
↳ zachová se to díky uzavřenosti na sjednocení a zřetězení

Věta: Nechť L je reg. jazyk abecedy Σ a $h: T \to \Sigma^*$ je homomorfismus abecedy T do abecedy Σ. Pak je $h^{-1}(L)$ také reg.

Dk: Pro L máme DFA $A = (Q, \Sigma, \delta, q_0, F)$

→ definujeme ε-NFA $B = (Q', T, \delta', [q_0, \lambda], F \times \{\lambda\})$

$$Q' = \{[q, u] \mid q \in Q,\ u \in \Sigma^*,\ (\exists x \in T)(\exists v \in \Sigma^*): h(x) = vu\}$$

→ u je buffer, začíná prázdný

↳ v koncovém stavu musí být také prázdný

$$\delta'([q, \lambda], x) = [q, h(x)]$$

$$\delta'([q, xw], \lambda) = [p, w], \quad \delta(q, x) = p$$

→ $\delta'([q_0, \lambda], x)$ → do bufferu uložím $h(x)$

→ pak přes λ-přechody odebírám postupně písmenka a simuluji běh automatu A. → nakonec se dostanu do nějakého $[q, \lambda]$

⇒ začnu dalším písmenkem ze zkonstruovaného slova w atd.

→ pokud mi projde celé slovo w, tak podle běhu automatu $w(w) \in L$

$$\Rightarrow L(B) = L(h^{-1}(L)) \Rightarrow h^{-1}(L) \text{ je reg.} \qquad \blacksquare$$

• Rozhodovací problémy pro reg. jazyky

① je jazyk přijímaný DFA / NFA / ε-NFA prázdný?

→ je prázdný $\Leftrightarrow$ žádný z koncových stavů není dosažitelný

↳ iterativně $O(|Q|^2)$

② náleží řetězec w do reg. jazyka L?

DFA: spustím automat, pokud $|w| = n$, tak při dobré reprezentaci $O(n)$

NFA: $O(n|Q^2|)$ ... k vstupní symbol aplikujeme na všech stavu předchozího kroku → těch nejvýše s

ε-NFA: nejprve si předpočítám ε-closure

↳ pak na výsledek předchozí fce vždy aplikovat ε-closure

Palindromy

Def: Palindrom je třeba "Madam, I'm Adam".

Lemma: Jazyk $L = \{w \in \Sigma^* \mid w = w^R\}$ není reg.

Dk: kdyby ano, pak nechť n je konstanta z pumping lemmatu

$\hookrightarrow w := \sigma^n 1 \sigma^n$

$xy \rightarrow$ pumpna dolu $\rightarrow$ ale vlevo mění $\sigma \Rightarrow$ není pal. ☠

GRAMATIKY

$\rightarrow L_i \rightarrow P_i$ pravidla

Def: Gramatika je $G = (V_N, V_T, P, S, \lambda)$

- V_N ... neterminály
- V_T ... terminály
- P ... pravidla $\alpha X \beta \rightarrow \omega$, $X \in V_N$, $\alpha, \beta, \omega \in (V_N \cup V_T)^*$ → obsahují net.
- S ... počáteční symbol $S \in V_N$
- λ ... příznak $\lambda \in L(G)$ → je potřeba aby fungovala kompatibilita Chomského hierarchie

Def: Klasifikace gramatik

$\mathcal{L}_0$ = množina jazyků generovaných gram. typu 0 = všechny gramatiky

pravidla: $\alpha X \beta \rightarrow \omega$

normální f: $\alpha X \beta \rightarrow \alpha \gamma \beta$

$X \in V_N$, $\alpha, \beta, \gamma, \omega \in (V_N \cup V_T)^*$

$\mathcal{L}_1$: Monotónní (kontextové) gramatiky $|L_i| \leq |P_i|$

$|L_i| \leq |P_i| \rightarrow$ nezkracující gramatiky

normální f: $\alpha X \beta \rightarrow \alpha \gamma \beta$, $X \in V_N$, $\alpha, \beta \in (V_N \cup V_T)^*$, $\gamma \in (V_N \cup T_V)^+$

! bez příznaku λ by nešlo vygenerovat λ

$\mathcal{L}_2$: Bezkontextové gram. $|L_i| = 1$

normální f: $X \rightarrow YZ \mid a$, $X, Y, Z \in V_N$, $a \in V_T$

$\mathcal{L}_3$: Regulární gram.

pravidla: $X \rightarrow wY \mid w$

norm. f: $X \rightarrow aY \mid \lambda$

$X, Y \in V_N$, $w \in V_T^*$

$a \in V_T$

Chomského hierarchie: → definice $\subsetneq$

Věta: $\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$

Dk: $\mathcal{L}_1 \subseteq \mathcal{L}_0$ ... $\mathcal{L}_0$ jsou všechny gram.

$\mathcal{L}_3 \subseteq \mathcal{L}_2$ ... protože gram typu $\mathcal{L}_3$ má $|L_i| = 1$

$\mathcal{L}_2 \subseteq \mathcal{L}_1$ ... $|L_i| = 1$, chci $|L_i| \leq |P_i|$

→ problém jsou pravidla $X \to \lambda$

$S \to \dots \to Y \to \alpha X \beta \to \alpha\beta \to \dots$

→ zbavím se jich: $\forall X, Y$: $Y \to \alpha X \beta$, $X \to \lambda$ $\Rightarrow$ $Y \to \alpha X \beta \mid \alpha\beta$

→ pokud $S \to \lambda$, tak nastavím příznak $\lambda = \text{True}$. ▨

Def $G = (V_N, V_T, P, S)$, $\alpha, \omega \in (V_N \cup V_T)^*$

1) α se **přímo přepíše** na ω: $\alpha \Rightarrow \omega$ $\equiv$

$\alpha = \beta\varphi\gamma$, $\varphi \to \psi$, $\beta, \gamma, \varphi, \psi \in (V_N \cup V_T)^*$

$\omega = \beta\psi\gamma$

2) α se **přepíše** na ω: $\alpha \Rightarrow^* \omega$ $\equiv$

$\alpha = \beta_1 \Rightarrow \beta_2 \Rightarrow \dots \Rightarrow \beta_n = \omega$

derivace

→ pokud $\forall i \neq j : \beta_i \neq \beta_j$: **minimální odvození**

→ pokud $S \Rightarrow^* \omega$, pak ω nazýváme **sentenciální forma**

Def: $G = (V_N, V_T, P, S)$

- **jazyk gramatiky** $L(G) := \{w \in V_T^* \mid S \Rightarrow^* w\}$
- **jazyk neterminálu** $A \in V$: $L(A) := \{w \in V_T^* \mid A \Rightarrow^* w\}$

Regulární gramatiky – typu 3

$A, B \in V_N$, $w \in V_T^*$

Def: G je *pravá lineární / regulární* $\equiv$ pravidla: $A \to wB \mid w$.

- # sentenciální forma 1 neterminál, vždy vpravo
- pravidlo $A \Rightarrow w$ se nazývá

idea: neterminál = stav DFA
pravidla = přechodová fce

Lemma (normální forma pro reg. g.): Ekvivalentně $A \to xB \mid \lambda$.

Dk:

① $A \to a_1 a_2 \dots a_n B$

$\Rightarrow A \to a_1 Y_2,\ Y_2 \to a_2 Y_3,\ \dots\ Y_n \to a_n B$

② $A \to a_1 a_2 \dots a_n$

$\Rightarrow A \to a_1 Z_2,\ Z_2 \to a_2 Z_3, \dots\ Z_n \to a_n X,\ X \to \lambda$

③ $A \Rightarrow^* B,\ B \to xC \mid \lambda$ ↑ DÚNO

- $B \to xC \rightsquigarrow A \to xC$
- $B \to \lambda \rightsquigarrow A \to \lambda$ ∎

Věta: Jazyk L je regulární $\Leftrightarrow$ ho generuje nějaká G typu 3.

Dk: Automat $\Rightarrow$ gramatika

$A = (Q, \Sigma, \delta, q_0, F) \Rightarrow V_N = Q,\ V_T = \Sigma$ $\quad$ Start $= q_0$

$\bar{\delta}(q, X) = p \Rightarrow q \to Xp$

$q \in F \Rightarrow q \to \lambda$

- Gramatika $\Rightarrow$ automat

$G = (V_N, V_T, P, S)$ → stavy = neterminály, abeceda = terminály $\quad q_0 = S$, $F = \{A \mid A \to \lambda\}$

→ $\delta(A, x) = \{B \mid A \to xB\}$

→ měla by se ukázat korektnost $\lambda \in L(G) \Leftrightarrow \lambda \in L(A)$

$a_1 \dots a_n \in L(G) \Leftrightarrow a_1 \dots a_n \in L(A)$ ∎

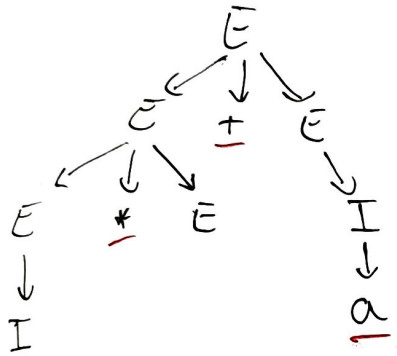
Bezkontextové – CFG, typ 2. $|L_i| = 1$

Lemma: $\mathcal{L}_3 \subset \mathcal{L}_2$

Dk: Jazyk $L = \{0^n 1^n \mid n \in \mathbb{N}\}$ není reg., ale generuje ho bezkontextová gramatika $S \to 0S1 \mid 01$.

Def: Derivační strom gramatiky G

$E \to E + E \to (E * E) + E \to (I * E) + E \to (I * E) + I \to (I * E) + a$



→ strom dává sentenciální formu w jestliže je w zřetězení listů

$I * E + a$

Def: Levá derivace $\Rightarrow_{LM}$ přepisuje nejlevější neterminál

Pravá derivace $\Rightarrow_{RM}$ přepisuje nejpravější — " —.

Věta: Následující tvrz. jsou ekviv.

① $A \Rightarrow^*_{LM} w$

② $A \Rightarrow^* w$

③ $\exists$ der. strom s kořenem A dávající w

Dk: 1 ⇒ 2, 2 ⇒ 3 easy

3 ⇒ 1 : je třeba pro libovolný strom najít levou derivaci

→ indukce podle výšky stromu

• Chomského norm. forma bezkontextové g.

Věta: Každou bezkontext. g. $|L_i| = 1$ lze zapsat jako $A \to BC \mid a$.

Algoritmus:

0 $S \to 01A \mid CD$
$A \to a \mid D \mid BE$
$B \to \lambda \mid b \mid E$
$C \to DaB$
$D \to dC$
$E \to A \mid e$

1 $S \to 01A$
$A \to a \mid B \mid BE$
$B \to \lambda \mid b \mid E$
$E \to A \mid e$

2 ↗

---

3 $D \to \lambda$
$A \to B \to \lambda$
$E \to A \to \lambda$

• $D \to \lambda$ odstraním

$S \to 01A \mid 01$
$A \to a \mid B \mid E \mid BE$
$B \to b \mid E$
$E \to A \mid e$

4 $(A,B), (A,E), (B,E)$
$(E,A), (E,B), (B,A)$

$S \to 01A \mid 01$
$A \to a \mid b \mid e \mid BE$
$B \to b \mid e \mid a \mid BE$ } merge
$E \to e \mid a \mid b \mid BE$

5 ⇓

$S \to 01A \mid 01$
$A \to a \mid b \mid e \mid AA$

---

① Eliminace negenerujících symbolů

X je generující $\equiv X \Rightarrow^* w \in T^*$

1. $\forall a \in T$ je generující
2. $A \to \alpha$, $\forall X \in \alpha$ je gen. $\Rightarrow A$ je gen

gen: $a, b, d, e, 0, 1$ ← terminály

$A \to a$, $B \to \lambda$, $S \to 01A$, $E \to e$

→ $C \to \underline{D}aB$ $D \to d\underline{C}$ ⇒ C, D nejsou

② Eliminace nedosažitelných symb.

X je dosažitelný $\equiv S \Rightarrow^* \alpha X \beta$

dos: $S \to 0, 1, A$ / $B \to b$
$A \to a, B, E$ / $E \to e$

| α je nedos.

③ Odstranění λ-pravidel

A je nulovatelný $\equiv A \Rightarrow^* \lambda$

1. $A \to \lambda$ ... A je nul.
2. $A \to C_1 \dots C_k$, $\forall i$: C_i je nul $\Rightarrow A$ je nul

→ $\forall$ pravidla $A \to X_1 \dots X_k$, $k \geq 1$

↳ pokud $m \leq k$ X_i nul $\Rightarrow 2^m$ nových pravidla

→ pokud $S \Rightarrow^* \lambda$, tak přidám λ = True

④ Jednotková pravidla $A \to B$

(A,A) je jednotkový pár

(A,B) j.p. & $B \to C$ $\Rightarrow (A,C)$ je j.p.

→ (A,B), $B \to \alpha$ není jednotkové $\Rightarrow A \to \alpha$

→ Teď máme redukovanou gramatiku

(5) odstranění dlouhých pravidel

$X \to abYcZ$

1. $X \to \bar{a}\bar{b}Y\bar{c}Z$
2. $X \to \bar{a}\tilde{b}$, $\tilde{b} \to \bar{b}\tilde{Y}$, $\tilde{Y} \to Y\tilde{c}$, $\tilde{c} \to \bar{c}Z$

1. pro term. $a \in V_T \Rightarrow \bar{a}$ nový net, $\bar{a} \to a$
2. pro $X \to B_1 \dots B_k$ nové net. $\tilde{B}_2, \dots \tilde{B}_{k-1}$, $X \to B_1\tilde{B}_2$, $\tilde{B}_i \to B_i\tilde{B}_{i+1}$, $\tilde{B}_{k-1} \to B_{k-1}B_k$

5

$S \to \bar{0}\bar{1}A \mid \bar{0}\bar{1}$, $\bar{0} \to 0$, $\bar{1} \to 1$

$A \to a \mid b \mid c \mid AA$ ✓

→ je potřeba opravit jen $S \to \bar{0}\bar{1}A \rightsquigarrow S \to \bar{0}\tilde{1}$, $\tilde{1} \to \bar{1}A$

Pumping lemma pro CFG

→ derivační strom CFG v Ch. n. f. je binární (# listů)

⇒ pokud odvozuje slovo w a má hloubku n, pak $|w| \leq 2^{n-1}$

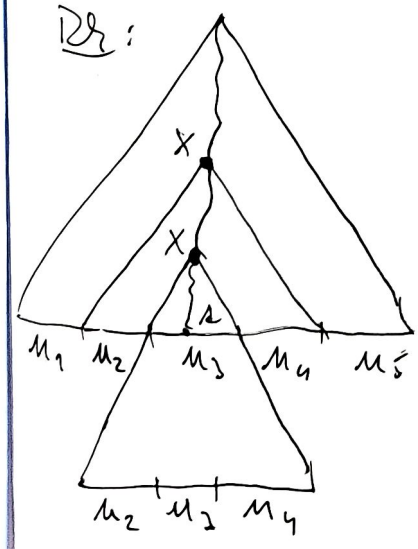
Důsl: pokud strom dává w & $|w| > 2^{n-1} \Rightarrow \exists$ cesta delší než n.

Věta: Pro $\forall$ bezkontextový jazyk L: $\exists n \in \mathbb{N}$: $\forall w \in L$, $|w| > n$ lze rozložit na $w = u_1u_2u_3u_4u_5$

$\forall i$: $u_1u_2^iu_3u_4^iu_5 \in L$

$|u_2u_4| > 0$ & $|u_2u_3u_4| < n$.

Dk:

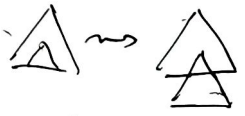


#net. = k, $n := 2^k$

→ pro $|w| > n$ je v der. stromě nejdelší cesta délky $> k$ (list = A)

⇒ z principu holubníku se zde nějaké X opakuje

→ nerozeme dvojici v posledních $k+1$ uzlech

→ $i = 0$:  , $i > 0$: 

- $|u_2u_4| > 0$ ∵ to horní $X \to AB \Rightarrow$ rozvětvilo se
- $|u_2u_3u_4| < n = 2^k$ ∵ cesta z horního X do A má délku max. $k+1$

Příklad:

① $\{0^n 1^n 2^n \mid n \in \mathbb{N}\}$ není bezkontextový

→ z lemma máme $n \Rightarrow w := 0^n 1^n 2^n$

→ žádné dělení nesplňuje PL

↳ pumpované slovo $|u_2 u_3 u_4| \leq n$

$\Rightarrow$ vždy pumpujeme max. 2 různé symboly

② $\{0^i 1^j 2^k \mid 0 \leq i \leq j \leq k\}$ není bezkontextový

→ $w := 0^n 1^n 2^n$ → opět pumpujeme max. 2 různé symboly

- pokud 0 (nebo 1) $\Rightarrow$ pumpujeme nahoru $\not\in$ $i \leq j$ $(j \leq k)$
- pokud 2 (nebo 1) $\Rightarrow$ pumpujeme dolů $\not\in$ $j \leq k$ $(i \leq j)$

③ $\{0^i 1^j 2^i 3^j \mid i, j \in \mathbb{N}\}$

④ $\{ww \mid w \in \{0,1\}^*\}$

- Cocke-Younger-Kasami alg. – CYK

problém: náleží slovo w do jazyka dané CFG?

→ exp. složitost: vygenerovat všechny derivační stromy do hloubky $|w| = n$

Algoritmus: CYK $O(n^3)$

gram. generující uzávorkování → $(()()) \in L$?

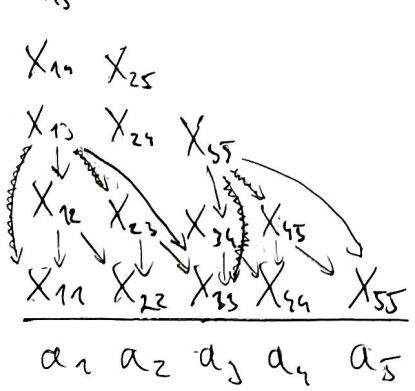
$S \to LR \mid SS \mid LA$

$A \to SR \quad S \to LSR$

$L \to (\quad , \quad R \to)$

→ vyplňujeme trojúh. tabulku zezdola

$w = a_1 a_2 \dots a_n$



→ základ $w = a_1 a_2 \dots a_n$

$X_{i,j} := \{A \in V_N \mid A \Rightarrow^* a_i \dots a_j\}$

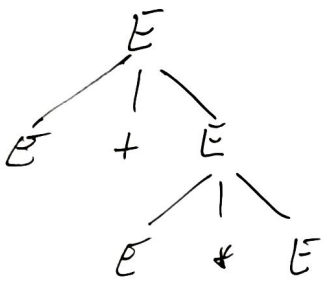
→ základ: $X_{ii} = \{A \mid A \to a_i\}$

→ indukce: $X_{ij} = \{A \mid A \to BC, \; \substack{B \in X_{ik} \\ C \in X_{k+1,j}}, \; k = i, \dots, j-1\}$

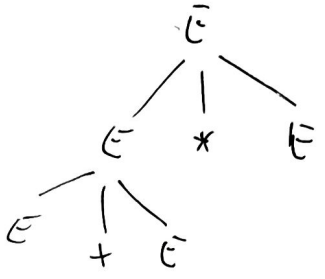
$\Rightarrow w \in L(G) \iff S \in X_{1,n}$

Víceznačnost gramatik

$E \Rightarrow E+E \Rightarrow E+E*E$, $E \Rightarrow E*E \Rightarrow E+E*E$



$1+(2*3)=7$



$(1+2)*3=9$

→ chceme to zjednoznačnit

→ je OK, když různé derivace mají stejný der. strom

Def: FCG $G=(V_N, V_T, P, S)$ je <u>víceznačná</u> ≡

$\exists w \in V_T^*$ t.ž. $\exists$ dva různé derivační stromy dávající w.

→ jinak je G <u>jednoznačná</u>

→ <u>jazyk je jednoznačný</u> ≡ $\exists$ jednoznačná CFG co to generuje

→ <u>jazyk je víceznačný</u> ≡ $\forall$ G co to generuje je víceznačná

<u>Fakt</u>: nelze sestrojit alg. co rozhodne víceznačnost gramatiky

existují 2 a jazyky, pro které neexistuje jednoznačná CFG

→ problém je většinou kvůli nerespektování priority operátorů

→ tohle musí řešit kompilátory prog. jazyků

→ zavede se více neterminálů, každý pro jednu úroveň priority

$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$ → čísla nejvyšší

$F \to I \mid (E)$ → závorky

$T \to F \mid T*F$ → násobení

$S = E \to T \mid E+T$ → sčítání

↳ vždy možnost přejít na [redacted] priorita

Výstup

Zásobníkové automaty = Push Down Automata PDA

→ rozšíření ε-NFA o zásobník, co je nad abecedou Γ

→ v ∀ kroku vidíme horní písmeno X a můžeme show dát libovolné $\gamma \in \Gamma^*$

↳ det. PDA přijímají jen vlastní podmnožinu bezkontext. jazyků

Def (PDA) $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to \mathcal{P}_{FIN}(Q \times \Gamma^*)$$

→ jsem v p, přečtu a, navrchu je X

→ transition do q, X nahradím za $\gamma \in \Gamma^*$

$\Rightarrow \delta(p, a, X) \ni (q, \gamma)$

Grafická notace: hrana: vstupní_znak, zás_znak → push_řetězec

Def: Konfigurace PDA je trojice (q, w, γ) → také ID

q je aktuální stav

w je zbývající vstup

γ je obsah zásobníku, vrchol zás. je vlevo

$$(p, aw, X\beta) \vdash (q, w, \alpha\beta) \equiv \delta(p, a, X) \ni (q, \alpha)$$

↳ konf. vede bezprostředně na tu vpravo

Def: Slova můžeme přijímat

1) koncovým stavem: $w \in L(P) \equiv (q_0, w, Z_0) \vdash^* (f, \lambda, \alpha)$, $f \in F$

2) prázdným zás.: $w \in N(P) \equiv (q_0, w, Z_0) \vdash^* (q, \lambda, \lambda)$

Lemma:

$(p, u, \alpha) \vdash^* (q, v, \beta) \Rightarrow (p, uw, \alpha\gamma) \vdash^* (q, vw, \beta\gamma)$

$(p, uw, \alpha) \vdash^* (q, vw, \beta) \Rightarrow (p, u, \alpha) \vdash^* (q, v, \beta)$

→ nečtené dno zásobníku, resp. vstup neovlivní výpočet

Lemma: Přijímání koncovým stavem a prázdným zás. jsou stejně silné.

DŮ: $L = L(P_F) \Rightarrow \exists P_N : L(P_N) = L$

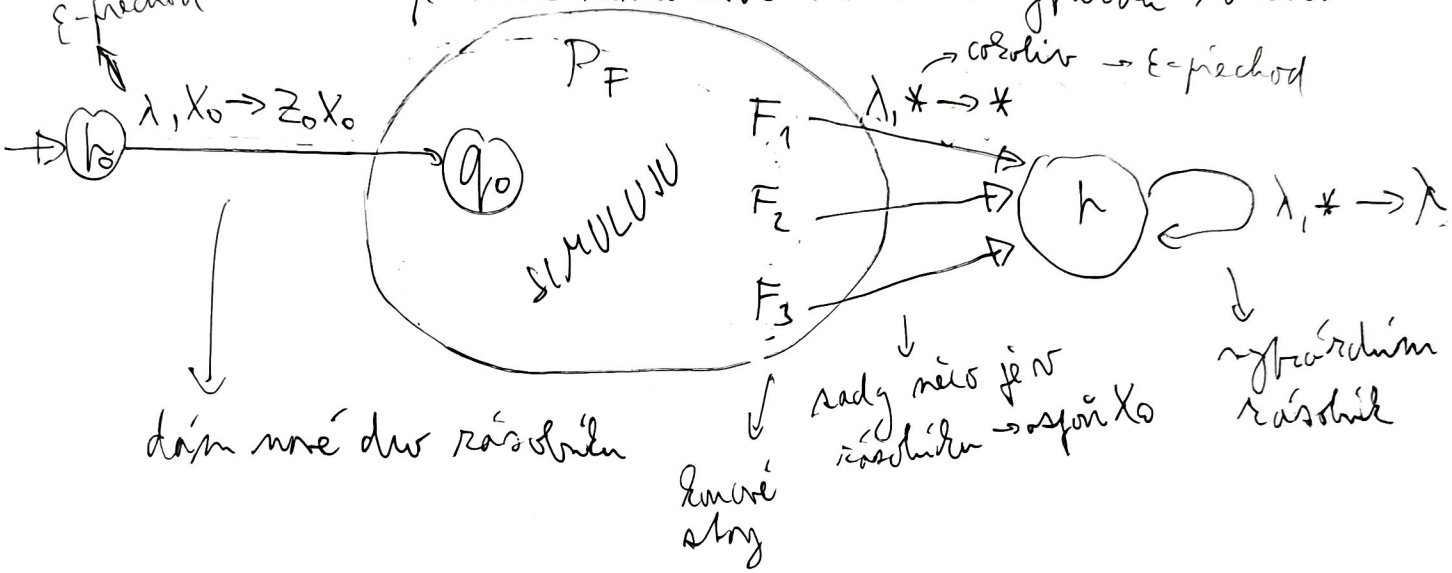
koncový stav — zásobník

$P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$

$P_N := (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$

↳ p_0 nový počáteční stav

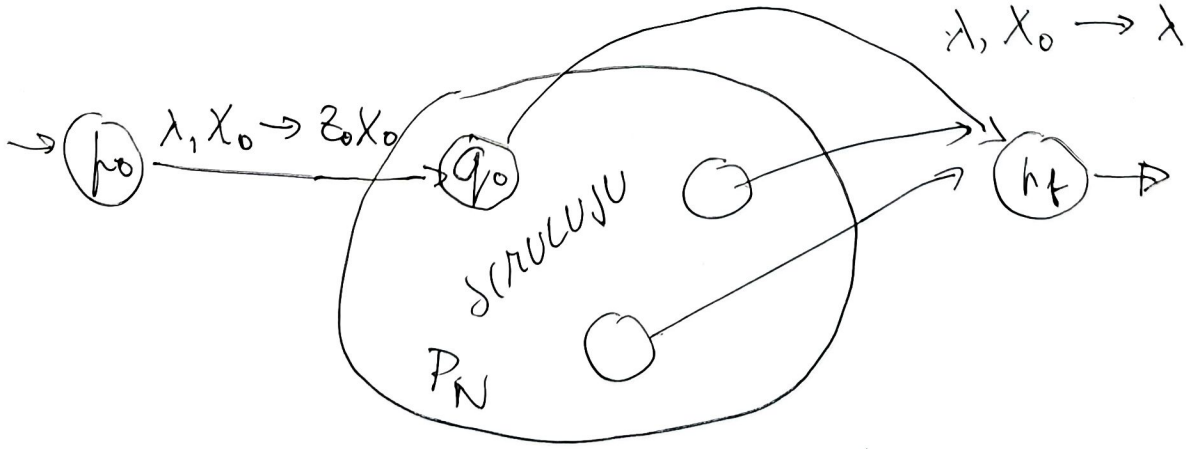
↳ p bude simulovat konec → vyprázdní zásobník



- **zásobník ⇒ koncový stav**

$P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$ — počáteční, konec

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



→ z ∀ stavu umožním ε-přechod do něj, pokud je zás. prázdný → to poznám počátečním X_0

Věta: $\mathcal{L}_2 = L(PDA) = N(PDA)$

↳ bezkontextové gramatiky ~ zás. automaty

Dk: $L(PDA) = N(PDA)$ minulé lemma

→ zbývá $\mathcal{L}_2 = N(PDA)$ ... prázdný zás

Algoritmus: konstrukce PDA z CFG

$G = (V_N, V_T, P, S)$

→ PDA: jediný stav q ... počáteční

$\Sigma = V_T$

$\Gamma = V_N \cup V_T$

$Z_0 = S$

→ protože stav se neměním, budu psát jen zás.

1) $A \in V_N$ s přidělem $A \to \beta_1 | \beta_2 | \beta_3 \dots$

$A \xrightarrow{\lambda} \beta_i$ ... nedet. výstup

$\delta(q, \lambda, A) = \{(q, \beta) \mid A \to \beta\}$

2) $a \in V_T : \delta(q, a, a) = \{(q, \lambda)\}$

→ pís kanda postupky generuje cokoliv s neterminály

→ pak mám vygenerované terminály, zase generuje atd.

$E \Rightarrow E*E \Rightarrow I*E \Rightarrow a*E \Rightarrow a*I \Rightarrow a+b$

$(a+b, E) \vdash (a+b, E*E) \vdash (a+b, I*E) \vdash (a+b, a*I) \vdash$

$\vdash (+b, +I) \vdash (b, I) \vdash (b, b) \vdash (\lambda, \lambda)$

↳2. ↳2. ↳2.

↳ stav v konfiguraci nepíšu

→ indukcí $w \in N(PDA) \Leftrightarrow w \in L(G)$

Opačně: Máme PDA, chceme CFG

$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$

→ vždy berem 1 symbol ze zás, ale stav jiný a to se musíme lišit

⇒ neterminály jsou symboly $[qXp]$ $q \xrightarrow{\text{zás}=X} p$

→ počáteční net. S

$\forall p \in Q$: $S \to [q_0, Z_0, p]$ → nějaký net.

$(r, \lambda) \in \delta(q, a, A)$: $[qAr] \to a$ → terminál

$(r, Y) \in \delta(q, a, A)$: $[qAp_k] \to a\,[rYp_k]$

$(r, Y_1 Y_2 \dots Y_k) \in \delta(q, a, A)$: $\forall p_1 \dots p_{k-1} \in Q$: → $\forall p_k \in Q$

$[qXp_k] \to a\,[rY_1p_1][p_1Y_2p_2]\dots[p_{k-1}Y_kp_k]$

tohle jsou jakoby věci co jsou na zásobníku → n

Deterministické PDA

Def: PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je det. $\equiv$

1. $\delta(q, a, X)$ je nejvýše jednoprvková vždy
2. $\delta(q, a, X) \neq \emptyset \Rightarrow \delta(q, \lambda, X) = \emptyset$

↓ $a \in \Sigma$

↳ aby nebyla volba mezi ε-přechodem a čtením vstupního symbolu

Věta: $RL \subset L_{DPDA} \subset L_{PDA} = CFL = N_{PDA} \supset N_{DPDA}$

(RL ↓ reg. jazyky; CFL ↓ bezk. jazyky)

Lemma: $RL \subseteq L_{DPDA}$

Dk: prostě budu ignorovat zásobník

Lemma $RL \subset L_{DPDA}$

Dk: Jazyk $L_{wcwr} = \{wcw^R \mid w \in (0+1)^*\}$ je přijímaný DPDA, ale není reg

neregularita:

pumping L. na $0^n c 0^n$

Lemma: $L_{DPDA} \subset CFL$

Dk: $L_{abb} = \{a^i b^i \mid i \in \mathbb{N}\} \cup \{a^i b^{2i} \mid i \in \mathbb{N}\}$

→ nějakým trikem vyrobíme automat přijímající $\{a^i b^i c^i \mid i \in \mathbb{N}\}$

↳ ale to není bezkontext (pumping lemma)

Bezprefixové jazyky

Def: Jazyk $L \subset \Sigma^*$ je bezprefixový $\equiv$ pro žádná slova $u, v \in L$ není v prefixem u

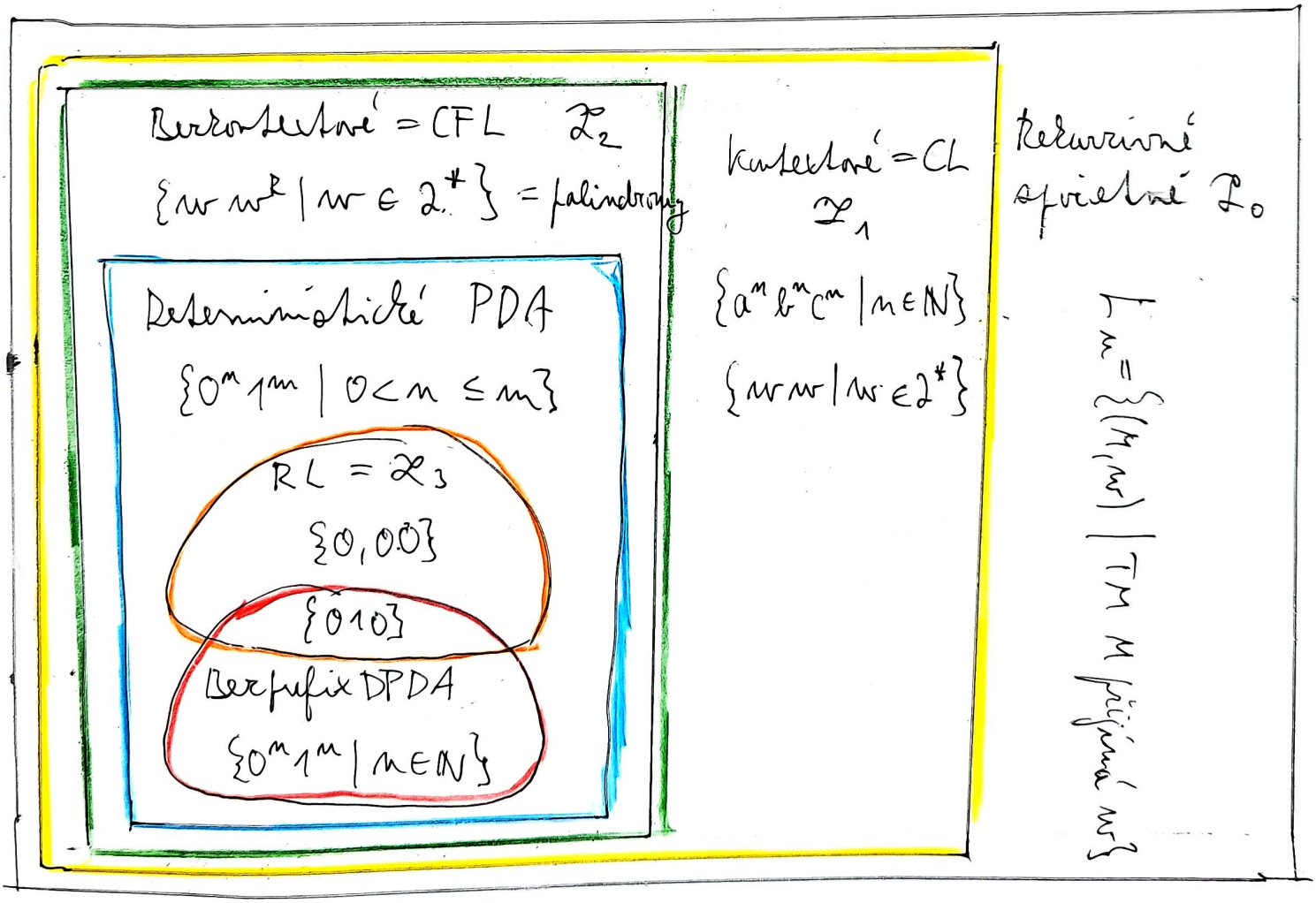
$$(\nexists u, v \in L): u = vz, \; z \in \Sigma^+$$

$\rightarrow$ jazyk L_{wcw^R} je bezprefix

Věta: $L \in N(P_{DPDA}) \iff L$ bezprefix & $L \in L(P_{DPDA})$

$\rightarrow$ pro det. PDA je přijímání prázdným zásobníkem slabší než koncovým st.

Dk: Vyplývá z převodů $N(P) \leftrightarrow L(P)$



Uzávěrové vlastnosti CFL

	RL	CFL	det. CFL
sjednocení	✓	✓	×
průnik	✓	×	×
$\cap$ s RL	✓	✓	✓
doplněk	✓	×	✓
homomorf.	✓	✓	×
inverzní hom.	✓	✓	✓

Věta: CFL uzavřené na $\cup$, konkatenaci, iteraci $(*, +)$, obraz w^R.

Dk:

- sjednocení: $V_{N_1} \cap V_{N_2} = \emptyset$ BÚNO, jinak přejmenuju
  → nový symbol $S \to S_1 \mid S_2$
- zřetězení $S \to S_1 S_2$
- iterace $L^* \Rightarrow S \to S_{OLD} S \mid \lambda$
  $L^+ \Rightarrow S \to S_{OLD} S \mid S_{OLD}$
- reverze $L^R = \{w^R \mid w \in L\}$
  $X \to w^R$ ... obrátím pravou stranu pravidel

Lema: CFL nejsou uzavřené na průnik. (dokonce DCFL ⇒ platí i pro DCFL)

Dk: $L = \{0^n 1^n 2^n \mid n \in \mathbb{N}\} = \{0^n 1^n 2^i \mid n, i \in \mathbb{N}\} \cap \{0^i 1^n 2^n \mid n, i \in \mathbb{N}\}$

↓ není CLF (pumping lemma)

① ②

① : $S \to AC$, $A \to 0A1 \mid 01$, $C \to 2C \mid 2$

② : $S \to AB$, $B \to 1B2 \mid 12$, $A \to 0A \mid 0$

Důvod: paralelní běh dvou PDA: nemůžeme sjednotit zásobníky

Lemma: CFL i DCFL jsou uzavřené na průnik s RL

Př: zás. a koneč. automat spojíme

DFA $A_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

PDA $M_1 = (Q_2, \Sigma, \Gamma, \delta_2, q_2, z_0, F_2)$

$\Rightarrow$ nový $M = (Q_1 \times Q_2, \Sigma, \Gamma, \delta, (q_1, q_2), z_0, F_1 \times F_2)$

$\delta((h,q), a, z) \ni ((r,s),\alpha) \equiv r = \delta_1(h,a) \wedge (s,\alpha) \in \delta_2(q,a,z)$

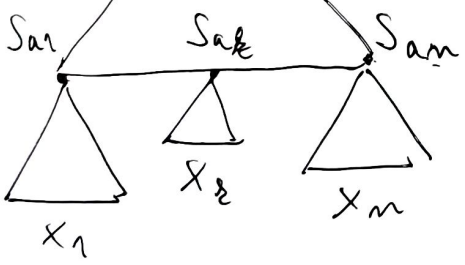
$\delta((h,q), \lambda, z) \ni ((h,s),\alpha) \equiv (s,\alpha) \in \delta_2(q,\lambda,z)$

Věta: CFL jsou uzavřené na substituci $\Rightarrow$ taktéž i homomorfismus.

$\rightarrow$ L je CFL nad Σ a σ je substituce v Σ t.ž. $\sigma(a)$ je CFL $\forall a \in \Sigma$.

$\Rightarrow$ pak je $\sigma(L)$ taktéž CFL

Př:



idea: listy v derivačním stromě generují další stromy

$\rightarrow$ sjednotím všechny V_N a V_T z těch $\sigma(a)$

$\rightarrow$ pravidla převezmu taktéž z gramatik těch $\sigma(a)$

$\rightarrow$ navíc pro $\forall a \in V_T$ nahradím v pravidlech z původní gramatiky a za S_a

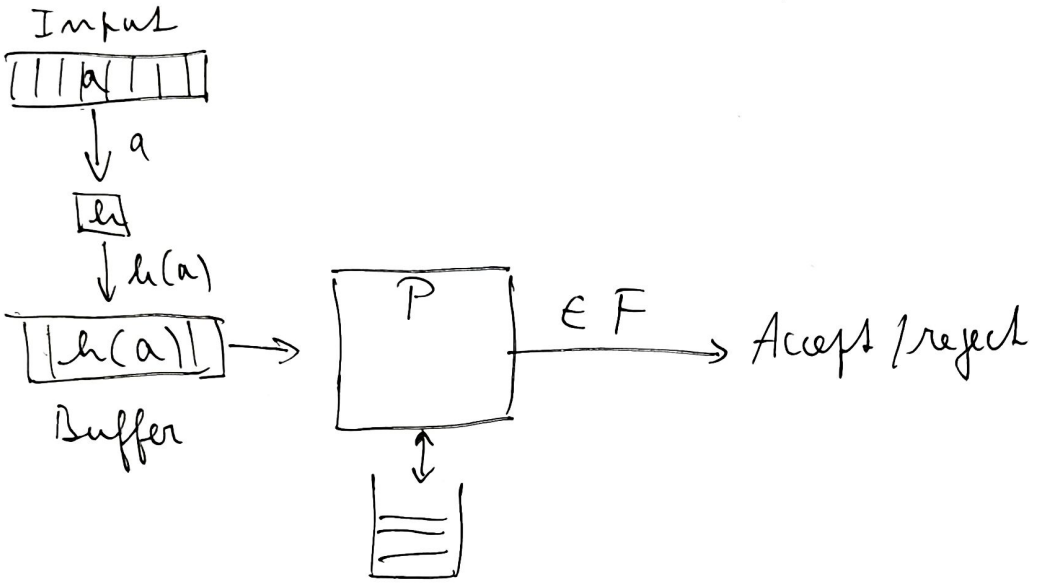
$X \rightarrow aY$ $\rightsquigarrow$ $X \rightarrow S_a Y S_b$

$\rightarrow$ S_a, S_b jsou počátky pro $\sigma(a), \sigma(b)$

Věta: CFL jsou uzavřené na inverzní homomorfismy.

→ Mějme CFL jazyk L, homomorfismus h. Pak $h^{-1}(L)$ je CFL
→ navíc $L \in DCFL \Rightarrow h^{-1}(L) \in DCFL$

Dk: Pro L existuje PDA P, co přijímá konc. stavem



→ z inputu postupně čtu písmenka
→ vždy udělám transformaci na nějaké slovo formálně
→ simuluju běh P na tomto "vstupním" slově
→ po vyčerpání buffer načtu nové slovo
→ ~~slovo~~ je přijato $\Leftrightarrow$ buffer je prázdný & P je v koncovém stavu

? Jak simulovat buffer ?
→ je konečný $\Rightarrow$ zapomenu se ho ve stavu
→ stavy jsou uspořádané dvojice (stav P, buffer)

Věta: CFL nejsou uzavřené na doplněk ani rozdíl.

Dk: víme, že nejsou na průnik

- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ → kdyby doplněk $\Rightarrow$ i průnik ↯
- $L_1 - L_2$ ... kdyby to vždy byl CFL, tak $\overline{L} = \Sigma^* - L$ je taky CFL

Uzavřenost vlastnosti DCFL

- nejsou uzavřené na průnik
- jsou na průnik s RL
- jsou na inverzní homom.

Lemma: Jsou uzavřené na doplněk:

Dk: Podobně jako u DFA můžeme prohodit koncové a nekoncové stavy
→ nedefinované kroky ošetříme zahozením na zásobníku
→ cykly odhalíme pomocí čítače ∎

Lemma: DCFL nejsou uzavřené na $\cup$

Dk: $L = \{a^i b^j c^k \mid i \neq j \vee j \neq k \vee i \neq k\} = L_{i \neq j} \cup L_{j \neq k} \cup L_{i \neq k}$

↳ L je CFL, ale není DCFL

→ kdyby byl DCFL, tak díky uzavřenosti DCFL na doplněk by byl DCFL i $\overline{L} = \{a^i b^j c^k \mid i = j = k\}$, který není CFL ∎

Lemma: DCFL nejsou uzavřené na homomorfismus

→ jazyky $L_{i \neq j}$ ($= L_1$), $L_{j \neq k}$ ($= L_2$), $L_{i \neq k}$ ($= L_3$) jsou DCFL

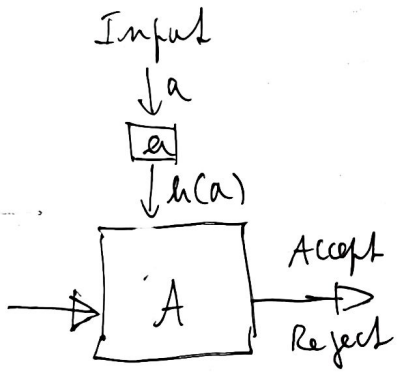
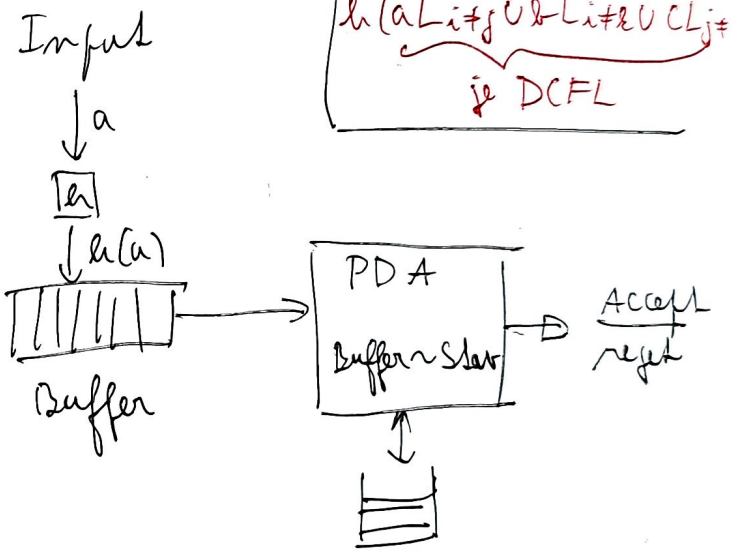
→ jazyk $L_1 \cup L_2 \cup L_3$ není DCFL

→ ale $0L_1 \cup 1L_2 \cup 2L_3$ je DCFL

↳ položíme $h(0) = h(1) = h(2) = \lambda$
$h(x) = x$ pro ostatní symboly

$h(0L_1 \cup 1L_2 \cup 2L_3) = L_1 \cup L_2 \cup L_3$, což není DCFL ∎

Uzávěrové vlastnosti v kostce

	RL	CFL	DCFL
sjednocení	$F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$	$S \to S_1 \mid S_2$	$A \cap B = \overline{\overline{A} \cup \overline{B}}$
průnik	$F = F_1 \times F_2$	$L = \{0^n 1^n 2^m \mid n \in \mathbb{N}\} = \{0^n 1^n 2^i\} \cap \{0^i 1^n 2^n\}$	
$\cap$ s RL	$F = F_1 \times F_2$	$F = F_1 \times F_2$	$F = F_1 \times F_2$
doplněk	$F_{COMPL} = Q - F$, δ totál.	$A \cap B = \overline{\overline{A} \cup \overline{B}}$	$F_{COMP} = Q - F_1$, Z_0, cyklí, δ totál.
homom.	Kleene + Reg $\mathcal{E}$ + uz.	$a \mapsto S_a \approx$ strom $\sigma(a)$	$L = \{0^i 1^j 2^k \mid i \neq j \vee i \neq k \vee j \neq k\}$ ↪ není CFL
			$L = L_{i \neq j} \cup L_{i \neq k} \cup L_{j \neq k}$
			$h(a) = h(b) = h(c) = \lambda$
			$h(aL_{i \neq j} \cup bL_{i \neq k} \cup cL_{j \neq k}) = L$ je DCFL
inverzní homom.			

Alternativně zůstává konové stavy, ale TM musí po skončení vyprázdnit pásku

⇒ je plná Blanků

Turingovy stroje – nekonečná páska

Def: TM je $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

- Q ... stavy
- Σ ... vstupní symboly
- Γ ... abeceda symbolů na pásku, $\Sigma \subseteq \Gamma$
- $\delta : (Q - F) \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$

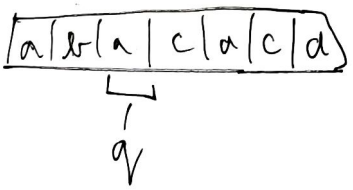
  $\delta(q, X) = (p, Y, D)$

  q ... aktuální stav $\notin F$; X ... symbol; p ... nový stav; Y ... před posunutím hlavy tiskne X na Y; D ... směr $\leftarrow$ nebo $\rightarrow$ kam posunout hlavu
- q_0 ... počáteční stav
- $B \in \Gamma - \Sigma$ ... Blank = symbol páskové buňky

  ↳ na počátku výpočtu kromě konečného počtu buněk se vstupem
- $F \subseteq Q$ ... koncové stavy

Def: Konfigurace TM je $X_1 X_2 \dots X_{i-1} (q, X_i) X_{i+1} \dots X_n$

↳ obsah pásky + poloha hlavy + vnitřní stav



$\rightsquigarrow$ a b (q, a) c a c d

Def: Kroky výpočtu + jsou v PDA

- $\delta(q, X_i) = (p, Y, \rightarrow)$

  $X_1 \dots X_{i-1} (q, X_i) X_{i+1} \dots X_n \vdash X_1 \dots X_{i-1} Y (q, X_{i+1}) \dots X_n$
- $\delta(q, X_i) = (p, Y, \leftarrow)$ podobně

Def (jazyk TM) $w \in L(M) \iff (q_0, X) w \vdash^* \alpha (f, y) \beta$, $f \in F$

↳ říkáme jim rekurzivně spočetné

Def: **Vícepáskový TM**

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

Jeden krok:

- hlava přejde do nového stavu
- na ∀ pásce napíšu nový symbol
- ∀ hlava se nezávisle posune ←, •, →

Věta: TM jsou stejně silné jako vícepáskové TM

idea: na jedné pásce simulujeme 2k stop
↳ abeceda (symboly) jsou 2k-tice
→ pro liché stopy = pozice k-té hlavy
sudé stopy = znak na k-té pásce

→ simulace 1 kroku navštíví všechny hlavy

→ ve stavu si pamatujeme
\# hlav vlevo
\# k: symbol pod k-tou hlavou

Def **Nedeterministický** TM je $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$.

$$\delta : (Q - F) \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{\leftarrow, \rightarrow\})$$

Věta: Pro ∀ NTM M_N ∃ DTM M_D t.ž. $L(M_N) = L(M_D)$.

Idea: M_D bude simulovat prohledávání do šířky výpočtu M_N

→ M_D má 2 pásky → na 1. si pamatuje již rozšířené konfigurace
↘ na 2. pomocný výpočet

→ zpracování konfigurace: přečti ID z 1. pásky
↳ je-li stav ∈ F ⇒ KONEC, jinak postupně zapiš na 1. pásku všechny konf. kam lze z této přejít — BFS

Def: TM **zastaví** ≡ vstoupí do stavu q, na pásce je X a pro tuto konfiguraci neexistuje instrukce.

↳ dále předpokládáme, že v $q \in F$ zastaví vždy

Def: **Správně rekurzivně** jazyk L $\Leftrightarrow$ $L = L(TM)$

$w \in L \Leftrightarrow$ s w se dá odvodit koncový stav.

Rekurzivní jazyk $L \equiv L = L(TM)$ & pro $\forall w \in \Sigma^*$ stroj zastaví.

↳ někdy se nerozlišují dvě nekonzistence

↳ říkáme, že TM **rozhoduje** jazyk L

Monotónní a kontextové gramatiky $\mathcal{L}_1$

$|L_i| \in |P_i|$

Lemma: normální tvar: $\alpha A \beta \to \alpha \omega \beta$
$\hookrightarrow \omega \in (V_N \cup V_T)^+$

① převedeme na sešpiralovou gramatiku.

Def: G je sešpiralovaná $\equiv$ pravidla jsou tvaru
$\alpha \to \beta$, $\alpha, \beta \in V_N^+$
$X \to a \mid \lambda$

Alg: Ke každé G $\exists$ ekviv. sešpiralovaná
1. Ke každému $a \in V_T$ zavedeme nový $\overline{a} \in V_N$
→ v pravidlech nahradíme a za $\overline{a}$, $\overline{a} \to a$

② pravidla $A_1 \dots A_m \to B_1 \dots B_n$, $m \le n$
převedeme na pravidla s nejvíc neterminály C

kontext $A_1 A_2 \dots A_m \to C_1 A_2 \dots A_m \to C_1 C_2 \dots A_m \to \dots \to C_1 \dots C_{m-1} C_m$

$C_1 C_2 \dots C_m \to B_1 C_2 \dots C_m \to B_1 B_2 \dots C_m \to \dots \to B_1 \dots B_{m-1} C_m$
$\to B_1 \dots B_{m-1} B_m \dots B_n$ ▨

Lineárně omezené automaty

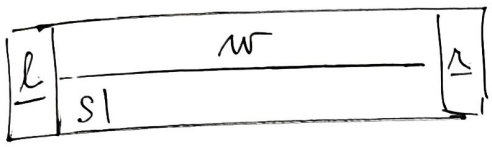
Def: LBA je nedet. TM, kde na pásce je levá a pravá závorka $\underline{\ell}$ a $\underline{r}$. Tyto symboly nelze přepsat ani je překročit.

$w \in L(LBA) \equiv (q_0, \underline{\ell}) w \underline{r} \overset{*}{\vdash} \underline{\ell} \alpha (q, X) \beta \underline{r}$, $q \in F$

$\Rightarrow$ páska je „omezená"

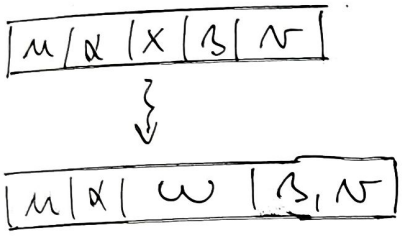
Věta: Každý kontextový jazyk lze přijmout formou LBA.

Dk: prázdné slovo vyřešíme zvlášť:



→ přijímáme pásku se dvěma stopami
→ na horní pásku dáme w
→ na začátek dolní S
→ přijímané slovo dole podle pravidel G
1. nedeterm. vyber část k přepsání
2. aplikuj pravidlo, βw odsuň doprava

$\alpha X \beta \to \alpha w \beta$



↳ pokud už zbudí terminály
⇒ horní a dolní stopa → slovo přijímáme (zamítáme) □

Věta: Každý LBA odpovídá nějaké kontextové g.

Dk: Máme LBA, chceme gram, co nebude generovat nic navíc

→ pro stavy $p, q \ldots$ → neterm. $P, Q \ldots$

$\delta(p, x) = (q, x', \to)$: $Px \to x'Q$

$\delta(p, x) = (q, x', \leftarrow)$: $yPx \to Qyx'$

→ začátek q_0, $q_0 \leftarrow$ první znak vstupu

→ nějaké se jistě musí objevit konec

Věta Rekurzivně spočetné $\subseteq \mathcal{L}_0$.

Dk: Pro TM M najdeme gramatiku G, $L(M) = L(G)$

→ gramatika nejprve vygeneruje pásku stroje a kopii slova

→ potom simuluje výpočet (stroj pomocí slova)

→ v koncovém stavu smaže pásku, nechá jen kopii slova

Věta: $\mathcal{L}_0 \subseteq$ rekurzivně spočetné

Dk: idea: TM postupně najde všechny derivace

→ derivaci $S \Rightarrow w_1 \Rightarrow \dots \Rightarrow w_n = w$ kódujeme jako

$$\#S\#w_1\#\dots\#w\#$$

• Diagonální jazyk

Def: Kódování TM pomocí nul a 1.

→ očíslujeme stavy a symboly na pásce

$\delta(q_i, X_j) = (q_k, X_l, D_m) \rightsquigarrow 0^i 1 0^j 1 0^k 1 0^l 1 0^m$

left = 0
right = 00

→ celý TM se skládá z kódů všech přechodů oddělených dvojicemi 11

$$C_1 11 C_2 11 C_3 11 \dots 11 C_{n-1} 11 C_n$$

→ $C_1, C_2, \dots C_n$ musíme nějak uspořádat

→ podle délky, stejně dlouhé lexikograficky

Def: Diagonální jazyk $L_D := \{w \in 2^* \mid$ TM reprezentovaný jako w nepřijímá $w\}$

Věta: L_D není rekurzivně spočetný ... žádný TM přijímající L_D.

Def: Univerzální jazyk $L_U := \{(M, w) \in 2^* \mid M$ je TM a $w \in L(M)\}$

↳ kódování dvojice pomocí 0 a 1

Věta: Existuje univerzální TM přijímající L_U.

Halting problem

Def: Problém = matematicky definovaná množina dvojic kódovaná řetězci nad abecedou Σ s odpověďmi True / False

Problém je rozhodnutelný $\equiv \exists$ TM $\mathcal{M}$ t.ž. pro $\forall$ vstup z problému zastaví a navíc přijme pouze True

Příklady:

- obsahuje slovo 5 nul?
- je vstupní slovo korektně definováno kódem TM podle definice výše?
- Zastaví TM $\mathcal{M}$ kódu M nad slovem w?

↳ halting problem není rozhodnutelný

Věta (Postova): Jazyk L je rekurzivní $\Leftrightarrow$ L i $\bar{L}$ jsou rek. spočetné.

Dk: Máme Turingovy stroje $L = L(\mathcal{M}_1)$, $\bar{L} = L(\mathcal{M}_2)$

→ pro dané slovo w navíc simulujeme oba stroje ve TM M

↳ 2 pásky ⇒ stroje jsou nezávislé

→ pokud jeden z M_1 nebo M_2 přijme, M zastaví a odpoví

→ protože jsou L a $\bar{L}$ komplementární, tak jeden z M_i vždy zastaví

⇒ $L(\mathcal{M}) = L$ je rekurzivní

↳ pokud M_1 přijme ⇒ True
↳ pokud M_2 přijme ⇒ False

Def: Redukce problému P_1 na P_2 je algoritmus R, který pro každou instanci $w \in P_1$ zastaví a vydá $R(w) \in P_2$ t.ž.

$$P_1(w) = \text{True} \quad \Leftrightarrow \quad P_2(R(w)) = \text{True}$$

Časová složitost

Def: Časová složitost TM M je fce $f: \mathbb{N} \to \mathbb{N}$, kde
$f(n)$ je max # kroků výpočtu M nad vstupy délky n

$$f(n) \in O(g(n)) \equiv \exists c, n_0 \in \mathbb{N}^+ : \forall n \geq n_0 : f(n) \leq c \cdot g(n)$$

Def. Pro fci $t: \mathbb{N} \to \mathbb{R}^+$ definujeme třídu časové složitosti
TIME($t(n)$) jako množinu všech jazyků, které jsou
rozhodnutelné jednopáskovým TM v čase $O(t(n))$
↳ tedy pro vstup délky n roste nejpřísněji jako $O(t(n))$ kroků.

Příklad: Jazyk $\{0^i 1^i \mid i \in \mathbb{N}\}$ je $O(n^2)$

1) zkontroluj vstup $0^* 1^*$, pokud na 1 je 0 → False $O(n)$
2) vrať se na začátek ... $O(n)$
3) dokud jsou nuly v $O(n^2)$
   a) přepiš 0 na X
   b) najdi 1 a přepiš na X
   c) vrať se na začátek
4) když už není 0, ověř že není 1 a přijmi nebo odmítni → $O(n)$

Def (PTIME): Třída P je třída jazyků rozhodnutelných polynomiálně

$$P = \bigcup_k TIME(n^k)$$

Věta: Každý bezkontextový jazyk patří do P

Dk: Gramatika převedena do CNF (nezávisí na n)
↳ CYK algoritmus je $O(n^3)$

Def: Verifikátor jazyka L je algoritmus V t.ž.

$L = \{w \mid V$ pro nějaký řetězec c přijímá $\langle w, c\rangle\}$

↓

certifikát

→ časová složitost V se měří pouze vzhledem k délce $|w|$

→ rozhoduje v poly. čase

(10) Platí $\exists$ poly. verifikátor, právě když $\exists$ poly. certifikát

(Co dělal by verifikátor, aniž nestihl přečíst)

Def: Třída NP := jazyky ~~rozhodnutelné~~ ověřitelné v polynomiálním čase

= jazyky s poly. verifikátorem.

Třída $NTIME(\Delta(n))$ pro fci $\Delta: \mathbb{N} \to \mathbb{R}^+$

$NTIME(\Delta(n)) := \{L \mid$ je rozhodnutelný nedet. TM v čase $O(\Delta(n))\}$

Věta: $NP = \bigcup_k NTIME(n^k)$

Def (Převoditelnost v poly. čase) Funkce $f: \Sigma^* \to \Sigma^*$ je

vyčíslitelná v poly. čase $\equiv \exists$ TM který pro každý vstup w roste v poly. čase a $f(w)$ na pásce

Jazyk A je převoditelný v poly. čase na jazyk B, $A \leq_P B \equiv$

$\exists f: \Sigma^* \to \Sigma^*$ vyčísl. poly. a $w \in A \Leftrightarrow f(w) \in B$

↳ polynomiální redukce A do B

Def: Jazyk B je NP-úplný $\equiv B \in NP$ & $\forall A \in NP: A \leq_P B$

Věta: Pokud je B NP úplný & $B \in P \Rightarrow P = NP$

Věta: ——"—— & $B \leq_P C \Rightarrow C$ je NP-úplný

Věta (Cook-Levinova): SAT a 3-SAT jsou NP-úplné.

Def: Jazyk $L \subseteq \Sigma^*$ patří do třídy co-NP $\equiv \overline{L} \in NP$.

↳ $P \subseteq NP$ & $P \subseteq$ co-NP

↳ pokud NP-úplný problém v co-NP $\Rightarrow P = NP$

Věta: Problém, zda je daná výroková formule tautologie je co-NP.

• Prostorová složitost

Def: Pro det. TM, který roste pro každý vstup je prostorová složitá funkce $f: \mathbb{N} \to \mathbb{N}$, kde $f(n) :=$ max # buněk pásky použitých při vstupu délky n

→ pro nedet TM → max. přes všechny vstupy & větve výpočtu

Def: $f: \mathbb{N} \to \mathbb{R}^+$, definujeme třídy prostorové složitosti

$SPACE(f(n)) = \{L \mid L$ je rozhodnutelný v prostoru $O(f(n))$ det. TM$\}$

$NSPACE(f(n)) = \{L \mid L$ je rozhodnutelný v prostoru $O(f(n))$ nedet. TM$\}$

$PSPACE = \bigcup_{k \in \mathbb{N}} SPACE(n^k)$

Věta (Savitch): $f(n) \geq n \Rightarrow NSPACE(f(n)) \subseteq SPACE(f^2(n))$

Důsledek: není nám definovat NSPACE $\because$ (polynom)^2 = polynom

Věta: $P \subseteq NP \subseteq PSPACE = NSPACE \subseteq EXPTIME := \bigcup_k TIME(2^{n^k})$

→ víme jenom $P \neq EXPTIME$ ostře