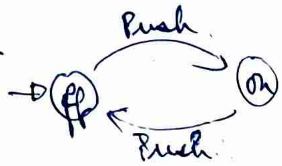
Def: Deterministický konečný automat (DFA) je $A=(Q, \Sigma, \delta, q_0, F)$, kde
- Q ... konečná množina stavů
- Σ ... konečná neprázdná abeceda
- δ ... přechodová funkce $\delta : Q \times \Sigma \to Q$
- q_0 ... počáteční stav $q_0 \in Q$ ... vede do něj šipka "odnikud" →
- F ... neprázdná množina koncových stavů $F \subseteq Q$

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

↳ kreslíme ◎→

Ukázka:



neformální

👁 tady dává smysl nemít koncový stav

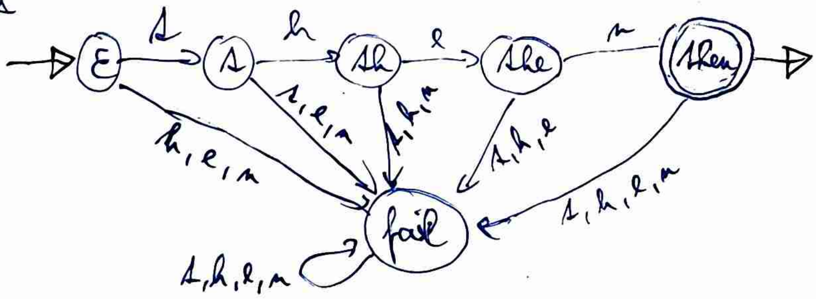
👁 přechodová funkce není úplná (def. obor není celá $Q \times \Sigma$)

Úmluva:

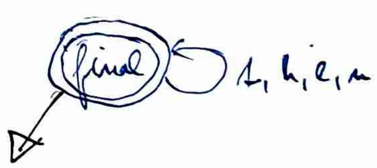
1) Pokud pro $q \in Q$ a $s \in \Sigma$ není definovaný přechod, tak přidáme nový stav 'fail' a dodefinujeme $\delta(q, s) := fail$.

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
- $\Sigma^* :=$ množina všech slov ... $\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$
- $\Sigma^+ :=$ množina všech neprázdných slov $\Sigma^+ := \Sigma^* \setminus \{\varepsilon\}$
- Jazyk je libovolná $L \subseteq \Sigma^*$

👁 jazyk je vždy spočetný

👁 jazyků je nespočetně mnoho

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: $|u|$ ($|\varepsilon| = 0$, $|auto| = 4$)

4, # výskytů znaku $s \in \Sigma$ ve slově u: $|u|_s$ ($|kuvrlina|_z = 2$)

- 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.ž. $L(A) = L$.

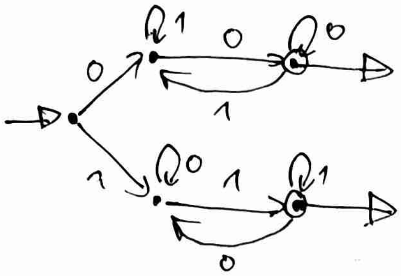
Třídu jazyků rozpoznatelných konečnými automaty označíme $\mathcal{F}$.

Těmto jazykům říkáme regulární jazyky.

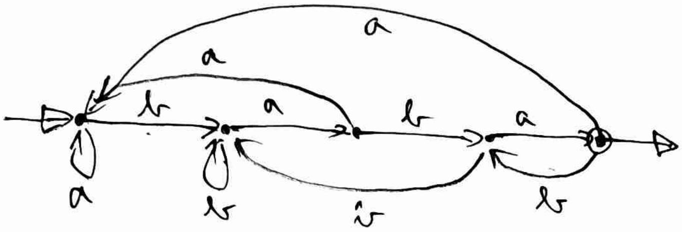
👁 Automaty jsou konečné ⇒ je jich spočetně mnoho } většina jazyků nemá
Jazyků je nespočetně mnoho } žádný automat

Ukázky regulárních jazyků

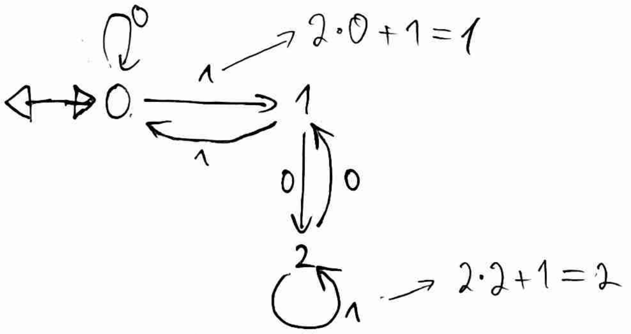
1) $L = \{w \in \{0,1\}^* \mid w = xux,\ x \in \{0,1\},\ u \in \{0,1\}^*\}$



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$\}$



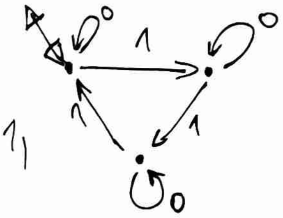
1010 ... slova čteme →

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)

$1 \cdot 1_0 - 1 \cdot 1_1 = 1$ (mod 3)

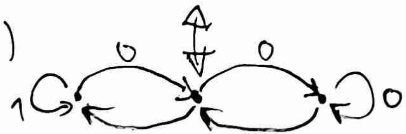
$1 \cdot 1_0 - 1 \cdot 1_1 = 2$ (mod 3)

#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 opakovat

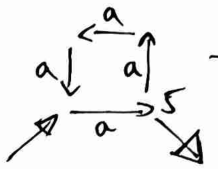
→ automat má 4 stavy

→ 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 $x := a_1 a_2 \dots a_i$

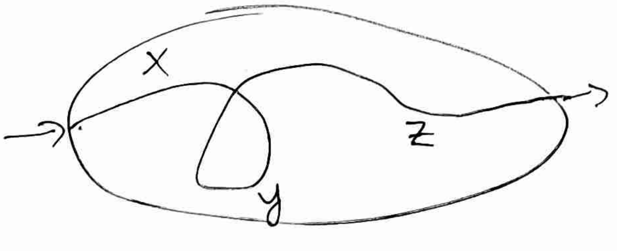
$y := a_{i+1} \dots a_j$

$z := a_{j+1} \dots a_m$

👁 $|xy| \leq n$ $|y| \geq 1$

👁 $xyz = w$

👁 smyčka nad p_i se může opakovat libovolně krát ∎



• Využití pumping lemmatu

→ důkazy, 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

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

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

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

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

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

→ zvolme n z PL a prvočíslo $\underline{p \ge n+2}$, $w := 1^p$

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

→ $xy^{p-m}z \in L$, ale

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

→ což není prvočíslo $\because$ $\underline{p-m > 1}$ & $m+1 > 1$

• Podtrhávací pumping lemma

→ v každém slově podtrhneme nějaká písmenka

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

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

Pz: díry = stavy

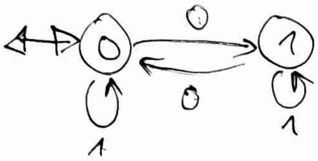
holubové = stavy do kterých jsem se dostal podtržením písmenka

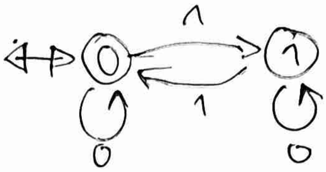
- Součin automatů

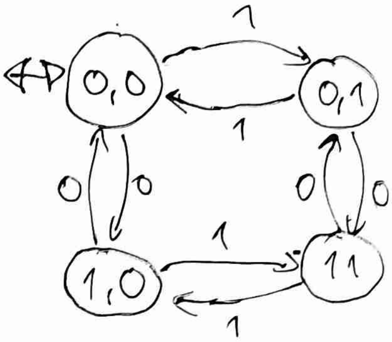
$\varphi_1 = |w|_0 = 2k$, $k \in \mathbb{N}_0$

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

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







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: $M_0 \subseteq M_1 \subseteq \ldots \subseteq Q$ a každé M_i obsahuje jen dos. stavy

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

→ zřejmě $\delta^*(q_0 x_1 \ldots x_i) \in M_i$, 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 $\because$ $aa \sim bb$, ale $aaa \nsim bba$

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

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

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

Dk: $\Rightarrow$: máme automat, cheme kongruenci

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

↳ příjmé kongruence, konečný index $\because$ 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 $\forall$ neprázdné takové množiny vyberu reprezentanta a sjednotím $[w]_\sim$

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

→ abeceda Σ

→ stavy $Q := \Sigma^*/\sim$

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

→ $F = \{c_1, \dots c_n\}$ t.ž. $L = \bigcup F = \bigcup_i c_i$

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

→ $\delta([u],x) := [ux]$

$\Rightarrow$ zbývá ukázat $L(A) = L$ .... $w \in L \Longleftrightarrow [w] \in F$

$w \in L \Leftrightarrow w \in \bigcup_i c_i \Leftrightarrow w \in c_1 \vee \dots \vee w \in c_n \Leftrightarrow [w] = c_1 \vee \dots \vee [w] = c_n$

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

• 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 $\because$ $aa \sim bb$, ale $aaa \nsim bba$

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 $\Sigma^*/\sim$.

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

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

↳ přijmě kongruence, konečný index $\because$ 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 $\forall$ neprázdné takové množiny vyberu reprezentanta a sjednotím $[w]_\sim$

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

→ abeceda Σ

→ stavy $Q := \Sigma^*/\sim$

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

→ $F = \{c_1, \dots c_n\}$ t.ž. $L = \bigcup F = \bigcup_i c_i$

→ $\delta([u],x) := [ux]$

$\Rightarrow$ zbývá ukázat $L(A) = L$ .... $w \in L \Leftrightarrow [w] \in F$

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

$w \in L \Leftrightarrow w \in \bigcup_i c_i \Leftrightarrow w \in c_1 \vee \dots \vee w \in c_n \Leftrightarrow [w] = c_1 \vee \dots \vee [w] = c_n$

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

Příklad: Neregulární pumpovatelný jazyk

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

→ vždy lze pumpovat první písmeno

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

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

↳ 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 $\equiv$

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 $\equiv L(A)=L(B)$.

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

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

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

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

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

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

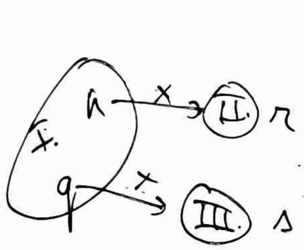
Pokud nejsou ekviv. jsou **rozlišitelné**.

☼ ekvivalence na stavech je tranzitivní

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 taky rozlišitelné

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

Korektnost: Nechť existují nějaké rozlišitelné páry, co alg. nerozlišil

→ vezmeme 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:

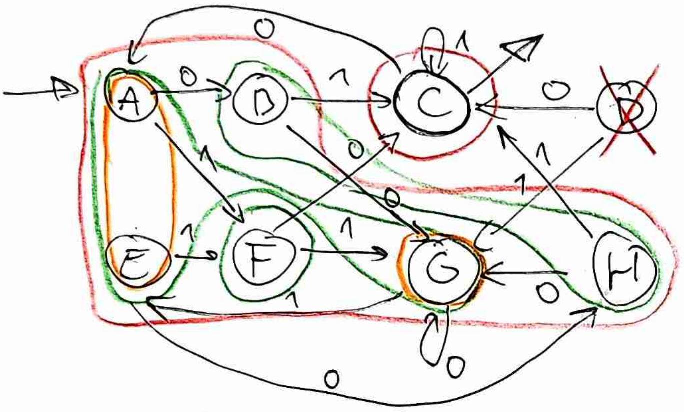
- v 1 kole uvažujeme všechny páry $\Rightarrow n^2 \cdot |\Sigma|$
- kol nejvýše n^2 … v jednom kole rozliším jen jeden stav

$$\Rightarrow O(n^4 |\Sigma|)$$

Def: DFA je redukovaný ≡
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.

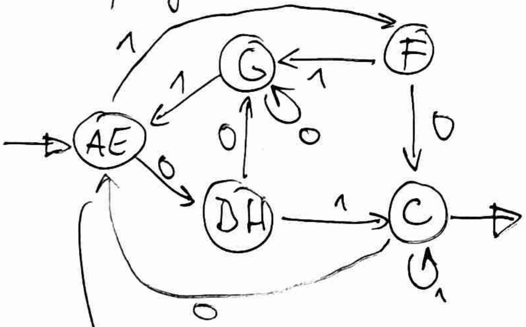
Algoritmus: nalezení reduktu
1, odstraníme nedosažitelné stavy



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

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

3, Vyrobíme redukt B na třídách ekvivalence



přechodová fce B ... γ
$\gamma(S, x)$, $S \in Q_B \subseteq Q$ $x \in \Sigma$

$\to q \in S \Rightarrow p := \delta(q, x)$
$\Rightarrow \gamma(S, x) := [p]_\sim$ ... třída ekvivalence

$\to$ 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 $h : Q_1 \to Q_2$.
- $\forall q \in Q_1$ je dosažitelný $\Rightarrow \exists$ slovo w t.ž. $\delta_1^*(q_0, w) = q$.
  $\Rightarrow$ definuji $h(q) := \delta_2^*(q_{0_2}, w)$

$\to$ není těžké dokázat: $h(q_{01}) = \cancel{X}(q_{02}$, $h(\delta_1(q, x)) = \delta_2(h(q), x))$, $q \in F_1 \Leftrightarrow h(q) \in F_2$

$\to$ 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)

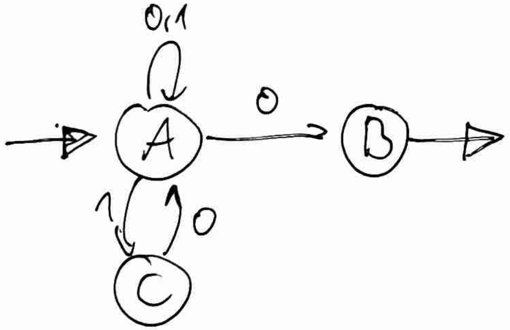
1. oba automaty zredukuju → [illegible]
2. Vytvoříme nový automat sjednocením stavů těch dvou reduktů, předpokládáme $Q_1 \cap Q_2 = \emptyset$

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

↳ zvolím libovolně q_{01} nebo q_{02}

3. původní automaty jsou ekvivalentní
   $\iff q_{01}, q_{02}$ 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

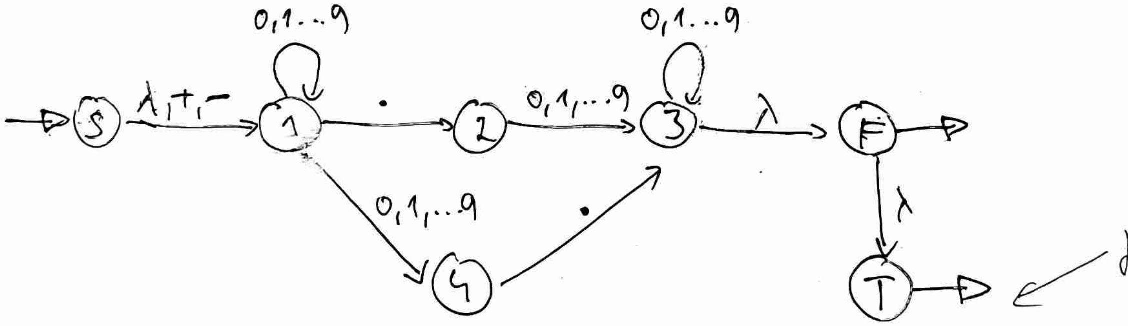
Def.: λ-NFA je $A=(Q,\Sigma,\delta,q_0,F)$ t.ž.

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

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

Příklad: Automat generující desetinná čísla

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



jen pro ukázku
ε-closure

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

Příklad: 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,Σ,δ,q_0,F) definujeme δ^* jako

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

Příklad:

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

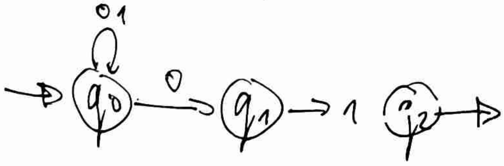
Def (Jazyk ε-NFA): Pro ε-NFA (Q,Σ,δ,q_0,F) je

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

- Převod NFA na DFA

Def: Konfigurace ε-NFA (Q,Σ,δ,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. $(p,xu) \to (q,u) \equiv q \in \delta(p,x)$.



$$\begin{tikzcd}
(q_0,00101) \arrow[r, "0"] \arrow[d, "0"] & (q_0,0101) \arrow[r, "0"] \arrow[d, "0"] & (q_0,101) \arrow[r, "1"] & (q_0,01) \arrow[r, "0"] \arrow[d, "0"] & (q_0,1) \arrow[r, "1"] & (q_0,\varepsilon \\
(q_1,0101) & (q_1,101) \arrow[d, "1"] & & (q_1,1) \arrow[r, "1"] & (q_2,\varepsilon) & \\
& (q_2,01) & \text{STUCK} & & &
\end{tikzcd}$$

Věta: L je rozpoznatelný ε-NFA ⟺ je reg.

Alg: Pro ε-NFA $N=(Q_N,\Sigma,\delta_N,q_0,F_N)$ zkonstruujeme 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 $\emptyset \in Q_D$.

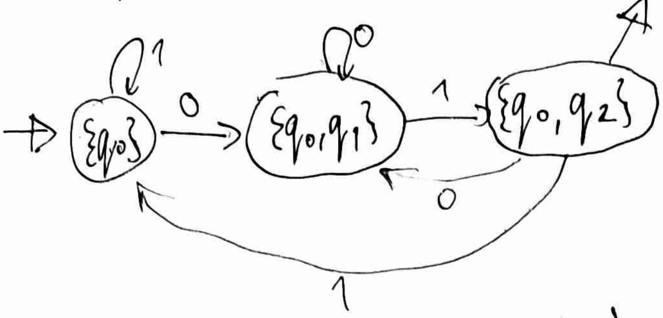
- počáteční stav $q_D := \varepsilon\text{-closure}(q_0)$
- přijímající = množiny obsahující nějaký přijímací stav: $F_D := \{S \in Q_D \mid S \cap F_N \neq \emptyset\}$
- přechod. fce mě musí dostat do množiny všech stavů, kde bych mohl být

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

$$O(\underbrace{2^n}_{\text{stavy}} \cdot \underbrace{n^3}_{\varepsilon\text{-closure}})$$

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

Příklad:



(nekreslím nedosažitelné stavy)

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
→ $\{q_0\}$	$\{q_0,q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
← $\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$
← $\{q_0,q_2\}$	$\{q_0,q_1\}$	$\{q_0\}$
← $\{q_1,q_2\}$	$\emptyset$	$\{q_2\}$
← $\{q_0,q_1,q_2\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$

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. $\cup, \cap, -, \overline{L}$

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

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

↳ $\delta((p_1,p_2),x) := (\delta_1(p_1,x), q_2(p_2,x))$

- průnik $L(A_1) \cap L(A_2) \rightarrow F = F_1 \times F_2$ → tomuto jsme předtím říkali součin automatů
- sjednocení $L(A_1) \cup L(A_2) \rightarrow F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$
- rozdíl $L(A_1) - L(A_2) \rightarrow F = F_1 \times (Q_2 - F_2)$

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

Příklady

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

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

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

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 $L_{0\neq1}=\{0^i1^j\mid i\neq j\}$ není reg

- jazyk $L_{01}:=\{0^i1^j\}$ je reg.

→ ale $L_{01}-L_{0\neq1}=\{0^i1^i\}$ 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 $L^+:=L^1\cup L^2\cup\dots=\bigcup_{i\in\mathbb{N}^+}L^i$
- obecná iterace $L^*:=L^+\cup\{\lambda\}=\bigcup_{i\in\mathbb{N}}L^i$
- otočení jazyka $L^R:=\{u^R\mid u\in L\}$, $(x_1x_2\dots x_n)^R:=x_n\dots x_2x_1$
- 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: $L.M$ je reg.

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

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

- q_0 je nový stav

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

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

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

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

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

$\hookrightarrow$ nedet. rozhodneme zda chceme pokračovat nebo restartovat

pro L^* navíc potřebujeme speciální stav pro přijem $\lambda\in L^0$

důkaz:

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

$\delta_B(q_B,\lambda):=\{q_0\}$ ... q_B je pro přijem ε

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

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

$\rightarrow$ pak 1, $q_B\in F_B \Rightarrow L(B)=L(A)^*$

2, $q_B\notin F_B \Rightarrow L(B)=L(A)^+$ ▨

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

Dk: idea: nedeterministický backward-search

$\rightarrow$ def. nedet $B=(Q\cup\{q_B\},\Sigma,\delta_B,q_B,\{q_0\})$

- $\delta_B(q,x):=\{p\mid\delta(p,x)=q\}$, $q\in Q$
- $\delta_B(q_B,\varepsilon)=F$
- $\delta_B(q_B,x):=\emptyset$ ▨

Lemma: L, M reg $=$ $M \backslash L$, L / M reg.

Dk: Nejprve $M \backslash L$: tedy $v \in M \backslash L \iff \exists u \in M : uv \in L$

→ 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 [illegible] ve stavech, kam lze dojít slovy z L

→ $\forall q \in Q : \delta_B(q, x) := \{\delta(q, x)\}$ … tohle jakoby det.

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

$\Rightarrow v \in M \backslash L \iff \exists u \in M : uv \in L$

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

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

$\iff v \in L(B)$

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

$x_1 x_2 \dots x_n y_1 y_2 \dots y_n \rightarrow x_1 x_2 \dots x_n$

↳ $y_n \dots y_2 y_1 x_n \dots x_2 x_1 \rightarrow x_n \dots x_2 x_1 \xrightarrow{R} x_1 x_2 \dots x_n$

Regulární výrazy

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

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

Základ indukce:

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

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

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

Indukce:

$L(\alpha+\beta) := L(\alpha) \cup L(\beta)$ ~ grep |

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

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

Priorita: $* > . > +$

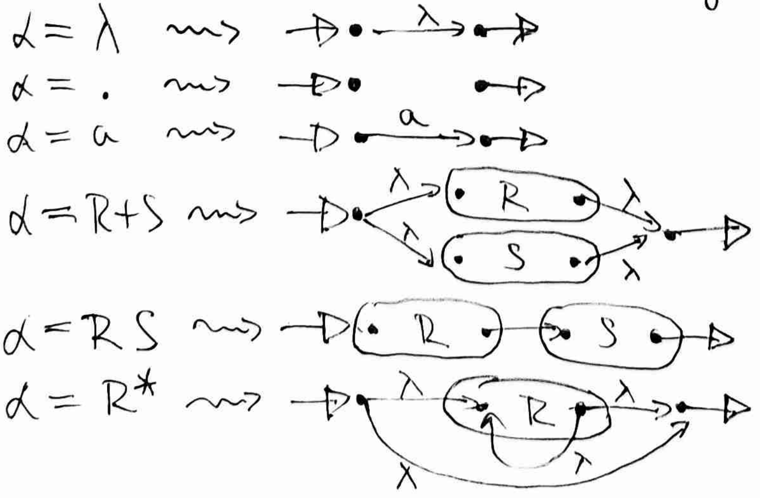
Příklad: střídající se nuly a jedničky: 0101, 10, 101, ...

$$\left.\begin{array}{l}(01)^* + (10)^* + 1(01)^* + 0(10)^* \\ (\lambda+1)(01)^*(\lambda+0)\end{array}\right\} \text{ekviv.}$$

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) ∄ hrany do poč. stavu

3) ∄ hrany z konc. stavu

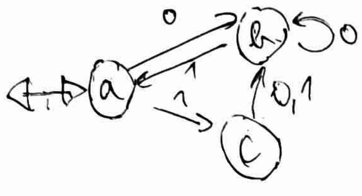
Převod DFA ⇝ Reg E – opačný směr

→ 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

$Q = \{1, 2, \ldots n\}$ → vyrobíme n matic n×n



$R^{\emptyset}$:

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

$i \neq j$: $(i) \xrightarrow{a_1, a_2, \ldots a_m} (j)$

$R^{\emptyset}_{ij} = a_1 + a_2 + \ldots + a_m$

$i = j$: $(i) \circlearrowleft a_1, \ldots a_m$

$R^{\emptyset}_{ii} = \lambda + a_1 + a_2 + \ldots + a_m$

indukce:

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

↙ tož jsem to uměl předtím

$$i \longrightarrow \overset{\circlearrowleft^*}{x} \longrightarrow j$$

N = povolené vrcholy
x = nový vrchol

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

→ 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	$\lambda + 01^+ + \underbrace{1(0+1)}_{\lambda^*}1^+$

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

Homomorfismus

→ substituce ~ vložení podprogramu do stavu

→ homomorfismus ~ převedení automatu na jiný

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

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

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

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

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

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

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

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

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

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

  → 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. tak je $\sigma(L)$ také reg.

↳ uvážíme subst. ⇒ hom.

Dk: L je reg ⇒ L lze popsat reg. výrazem → indukcí podle struktury RegE

↳ KLEENE

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

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

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

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

→ prvočinitele jsou reg (předp.)

↳ zachová se to díky uzavřenosti na sjednocení a zřetězování

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, začne prázdný

↳ v koncovém stavu musí být také prázdný

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

$$\delta'([q, xv], \lambda) = [p, v] \quad , \ \delta(q,x) = p$$

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

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

⇒ zadám další písmenko ze zkoumaného slova w atd.

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

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

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

NFA: $O(n|Q^2|)$ … ∀ vstupní symbol aplikujeme na všechny stavy předchozího kroku → těch nejvýše s

ε-NFA: nejprve si předpočítám ε-closure

↳ pak na výsledek přechodové fce vždy aplikovat ε-closure

- Polindromy

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

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

Dk: kdyby ano, tak nechť n je konstanta z pumping lemma

$\hookrightarrow w := \underbrace{0}_{xy}{}^{n} 1\, 0^{n}$

xy → pumpuj dolu → ale vlevo méně 0 ⇒ není pal. ■

GRAMATIKY

→ $L_i \to P_i$ pravidla

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

- V_N … neterminály
- V_T … terminály
- P … pravidla $\overbrace{\alpha X \beta} \to \omega$, $X \in V_N$, $\alpha, \beta, \omega \in (V_N \cup V_T)^*$ → obsahuje 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: $\alpha X \beta \to \omega$ $\quad X \in V_N$, $\alpha, \beta, \gamma, \omega \in (V_N \cup V_T)^*$

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

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

$|L_i| \le |P_i|$ → nezkracující gramatiky

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

! bez příznaku λ by nešlo vygenerovat λ

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

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

$\mathcal{L}_3$: Regulární gram.

pravidla: $X \to wY \mid w$ $\quad X, Y \in V_N$, $w \in V_T^*$

norm. f: $X \to aY \mid \lambda$ $\quad a \in V_T$

Chomského hierarchie:

→ dokonce $\subsetneq$

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: $\forall X, Y$: $\begin{matrix} Y \to \alpha X \beta \\ X \to \lambda \end{matrix}$ ⇝ $Y \to \alpha X \beta \mid \alpha\beta$

→ 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 : $\alpha \Rightarrow^* \omega$ $\equiv$

$$\alpha = \underbrace{\beta_1 \Rightarrow \beta_2 \Rightarrow \dots \Rightarrow \beta_n}_{\text{derivace}} = \omega$$

→ 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 ≡ pravidla: $A \to wB \mid w$.

👁 ∀ sentenciální forma 1 neterminál, zcela vpravo
pravidlo $A \to w$ to uzavře

idea: neterminál = stav DFA
pravidla = přechodová fce

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

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 $A \Rightarrow^* B,\ B \to xC \mid \lambda$ $\uparrow$ BÚNO

- $B \to xC \leadsto A \to xC$
- $B \to \lambda \leadsto A \to \lambda$

▨

Věta: Jazyk L je regulární ⟺ ho generuje nějaká G typu 3.

Dk: Automat ⇒ gramatika

$A = (Q, \Sigma, \delta, q_0, F) \Rightarrow \begin{matrix} V_N = Q \\ V_T = \Sigma \end{matrix}$ Start $= q_0$

$\delta(q, x) = p \Rightarrow q \to xp$

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

- Gramatika ⇒ automat

$G = (V_N, V_T, P, S)$ → stavy = neterminály
abeceda = terminály

$q_0 = S$
$F = \{A \mid A \to \lambda\}$

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

⇝ měla by se ukázat korektnost $\lambda \in L(G) \iff \lambda \in L(A)$
$a_1 \dots a_n \in L(G) \iff a_1 \dots a_n \in L(A)$ ▨

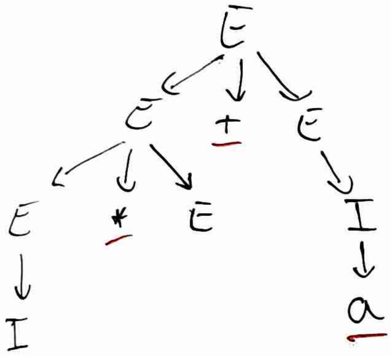
Bezkontextové – CFG, typ 2.. $|L_i| = 1$

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

$$E \to \underline{E}+E \to (\underline{E}*E)+E \to (I*E)+\underline{E} \to (I*E)+\underline{I} \to (I*E)+a$$



→ strom dává senteciální 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 ∃ 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 bezkontextová g.

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

Algoritmus:

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

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. $B \to \lambda$
$A \to B \to \lambda$
$E \to A \to \lambda$

- $B \to \lambda$ 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 $(A,B), (A,E), (B,E)$
$(E,A), (E,B), (B,A)$

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

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

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

② Eliminace nedosažitelných symb.

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

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

→ ∀ pravidlo $A \to X_1 \dots X_k$, $k \geq 1$

↳ pokud m z X_i nul ⇒ 2^m verzí pravidla

→ pokud $S \Rightarrow^* \lambda$, tak příznak λ-True

④ Jednotková pravidla $A \to B$

(A, A) je jednotkový pár
(A, B) j.p. & $B \to C$ ⇒ (A, C) je j.p.
→ (A, B), $B \to \alpha$ není jednotkové ⇒ $A \to \alpha$

→ teď máme redukovanou gramatiku

⑤ odstranění dlouhých pravidel

$X \to abYcZ$ 1. pro term. $a \in V_T \Rightarrow \bar{a}$ nový net.

$\bar{a} \to a$

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

2. pro $X \to B_1 \dots B_k$

nové net. $\tilde{B}_2, \dots \tilde{B}_{k-1}$

$X \to B_1\tilde{B}_2$, $\tilde{B}_i \to B_i\tilde{B}_{i+1}$, $\tilde{B}_{k-1} \to B_{k-1}B$

2. $X \to \bar{a}\tilde{b}$

$\tilde{b} \to \bar{b}\tilde{Y}$

$\tilde{Y} \to Y\tilde{c}$

$\tilde{c} \to \bar{c}Z$

5

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

$A \to a \mid b \mid e \mid AA$ ✓

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

Pumping lemma pro CFG

\# listů

→ derivační strom CFG v Ch.n.f. je binární

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

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

Věta: Pro ∀ bezkontextový jazyk L: $\exists m \in \mathbb{N}: \forall w \in L, |w| > m$

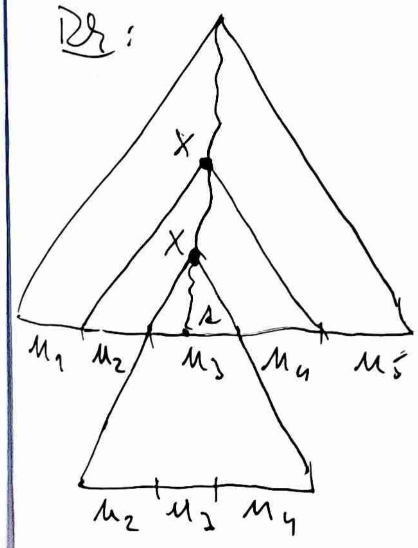
lze rozložit na $w = u_1u_2u_3u_4u_5$

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

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

list = a

Dk:



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

→ pro $|w| > m$ je v der. stromě nejdelší cesta délky $> k$

⇒ z principu holubníku se tam nějaké X opakuje

→ vezmeme dvojici v posledních $k+1$ vrcholech

→ $i = 0$:  ⇝ 

, $i > 0$:  ⇝ 

- $|u_2u_4| > 0$ ∵ to horní $X \to AB$ ⇒ rozvětvilo se
- $|u_2u_3u_4| < m = 2^k$ ∵ 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 ⇒ $w := 0^n 1^n 2^n$

→ žádné dělení nesplňuje PL

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

⇒ vždy pumpuju 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 pumpuju max. 2 různé symboly

- pokud 0 (nebo 1) ⇒ pumpuju nahoru ↯ $i \leq j$ $(j \leq k)$
- pokud 2 (nebo 1) ⇒ pumpuju dolů ↯ $j \leq k$ $(i \leq j)$

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: vyzkoušet všechny derivační stromy do hloubky $|w| = n$

Algoritmus: CYK $O(n^3)$

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

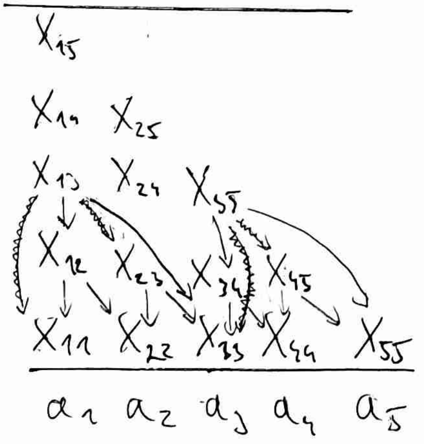
$S \to LR \mid SS \mid LA$

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

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

→ vyplňujeme trojúh. tabulku zezdola

$w = a_1 a_2 \dots a_n$



→ základ $w = a_1 \ a_2 \ \dots \ a_n$

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

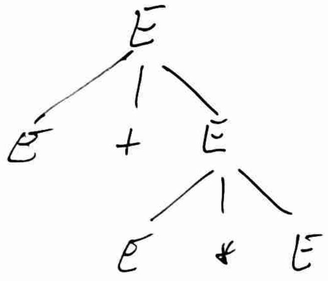
→ základ: $X_{ii} = \{A \mid A \to a_i\}$

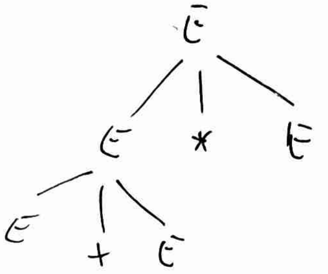
→ indukce: $X_{ij} = \{A \mid A \to BC, \ \begin{smallmatrix} B \in X_{ik} \\ C \in X_{k+1,j} \end{smallmatrix}, \ k = i, \dots, j-1\}$

⇒ $w \in L(G) \iff S \in X_{1,n}$

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 víceznačná ≡

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

→ jinak je G jednoznačná

→ jazyk je jednoznačný ≡ ∃ jednoznačná CFG co ho generuje.

→ jazyk je víceznačný ≡ ∀G co ho generuje je víceznačná

Fakt: neexistuje alg. co rozhodne víceznačnost gramatiky
existují $\mathcal{L}_2$ 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 vyšší prioritu

Zásobníkové automaty = Push Down Automata PDA

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

→ v $\forall$ kroku vidíme horní písmeno X a můžeme ho 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, na vrchu 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: $w \in L(P) \equiv (q_0, w, Z_0) \vdash^* (\overset{f \in F}{f}, \lambda, \alpha)$

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.

DK: $L = L(P_F)$ ⇒ $\exists P_N : L(P_N) = L$

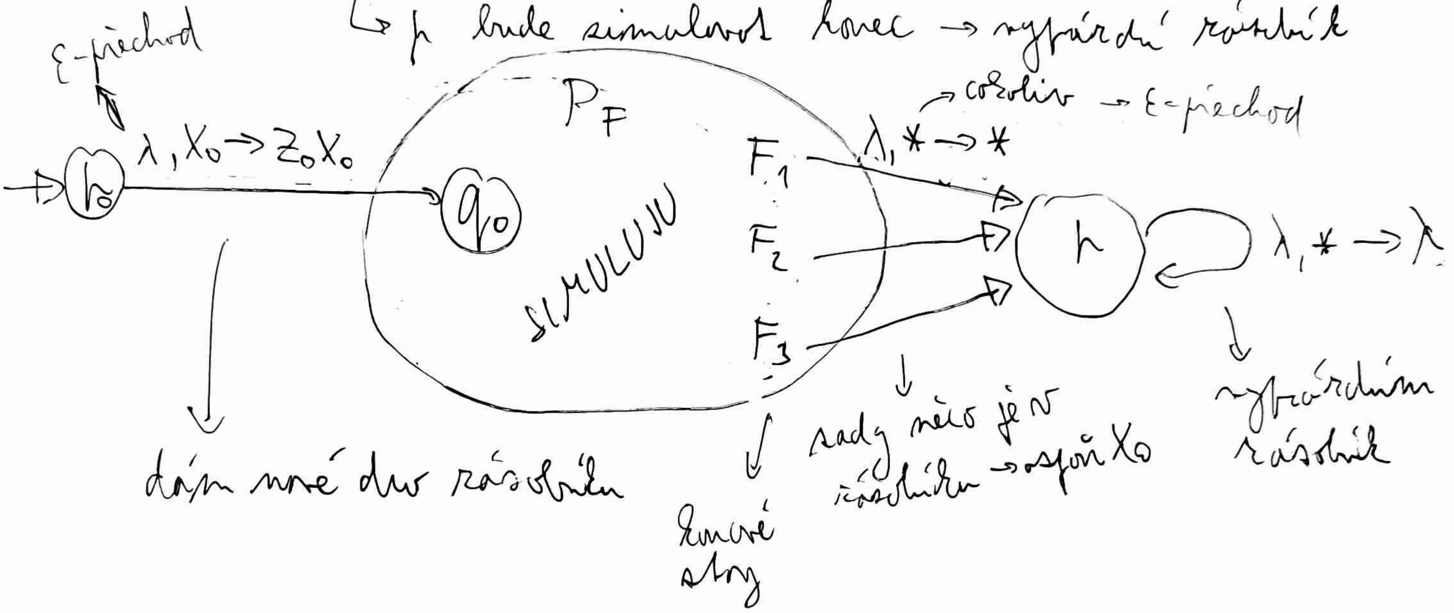
koncový stav — zásobník

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

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

↳ p_0 nový počáteční stav

↳ p bude simulovat konec → vyprázdní zásobník

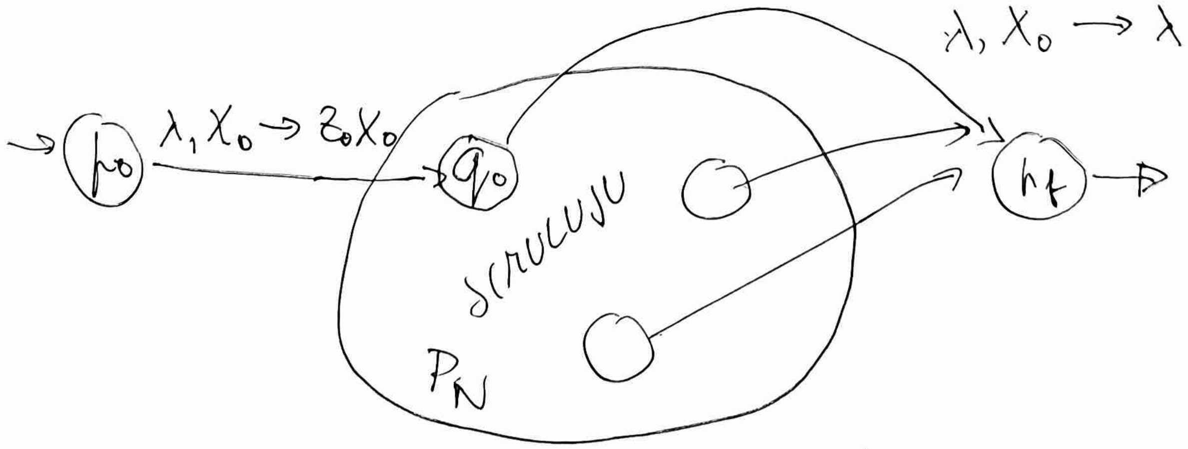


- zásobník ⇒ koncový stav

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

rozšířen — 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 cíle, pokud je zás. prázdný → to poznám zarážkou X_0

▨

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

↳ bezkontextové gram ~ zás. automaty

Dk: $L(PDA) = N(PDA)$ minulé lemma

→ ukážeme $\mathcal{L}_2 = N(PDA)$ ... 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 nemění, budu psát jen zás.

1) $A \in V_N$ s pravidlem $A \to B_1 | B_2 | B_3 \dots$

$A \xrightarrow{\lambda} B_i$ ... nedet. vyberu

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

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

→ přes landa přechody generuje cokoliv s neterminály

→ pak maže vygenerované terminály, zase generuje atd.

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

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

$\underset{\hookrightarrow 2.}{\underline{\vdash}} (*b, *I) \underset{\hookrightarrow 2.}{\underline{\vdash}} (b, I) \vdash (b, b) \underset{\hookrightarrow 2.}{\underline{\vdash}} (\lambda, \lambda)$

↳ 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 bereme 1 symbol ze zás, ale stav před a po se můžou lišit

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

→ počáteční net: S

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

$(p, \lambda) \in \delta(q, a, A): \quad [qAp] \to \underline{a}$ → terminál

$(p, Y) \in \delta(q, a, A): \quad [qAp_k] \to a[pYp_k]$ → $\forall p_k \in Q$

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

$$[qXp_k] \to a \underbrace{[pY_1p_1][p_1Y_2p_2] \dots [p_{k-1}Y_k, p_k]}$$

tohle jsou pokaždy věci co jsou
na zásobníku → n

Deterministické PDA

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

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

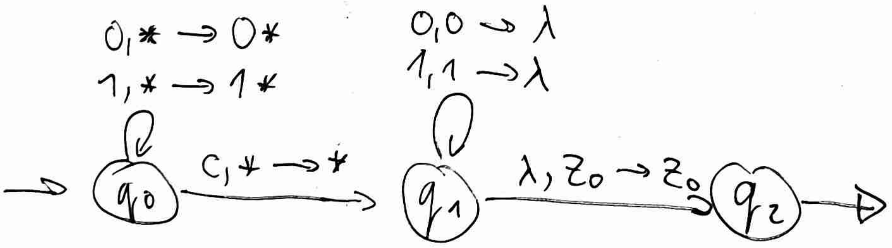
↓ reg. jazyky ↓ bezk. jazyky

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

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

↳ ale to není bezkontextový (pumping lemma)

Bezprefixové jazyky

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

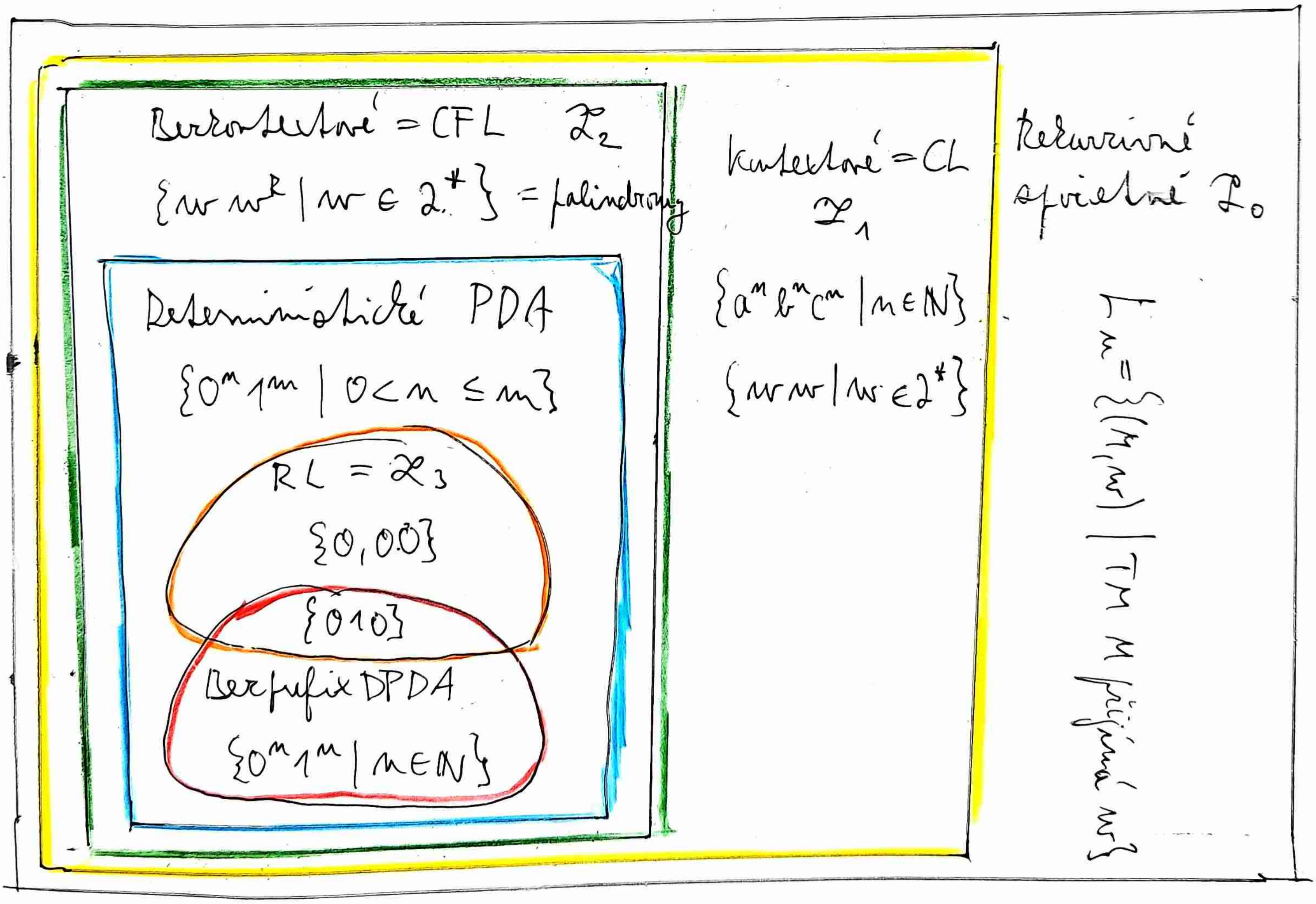
$$(\nexists u, v \in L): u = vz, \ z \in \Sigma^+$$

→ jazyk L_{wcw^R} je bezprefix

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

→ pro det. PDA je přijímání prázdným zásobníkem slabší než koncovým st.

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



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
  $\rightarrow$ nový symbol $S \rightarrow S_1 | S_2$
- zřetězení $S \rightarrow S_1 S_2$
- iterace $L^* \Rightarrow S \rightarrow S_{OLD} S | \lambda$
  $L^+ \Rightarrow S \rightarrow S_{OLD} S | S_{OLD}$
- reverze $L^R = \{w^R | w \in L\}$
  $X \rightarrow \omega^R$ ... obrátím pravou stranu pravidel

Lemma: CFL nejsou uzavřené na průnik. ← důkaz ⇒ věta i pro DCFL

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

↓ není CLF (pumping lemma)

① : $S \rightarrow AC$, $A \rightarrow 0A1 | 01$, $C \rightarrow 2C | 2$

② : $S \rightarrow AB$, $B \rightarrow 1B2 | 12$, $A \rightarrow 0A | 0$

Důvod: paralelní běh dvou PDA: neumíme spojit zásobníky

Lemma: CFL i DCFL jsou uzavřené na průnik s RL

Dk: zás. a konečný 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 $M = (Q_1 \times Q_2, \Sigma, \Gamma, \delta, (q_1, q_2), Z_0, F_1 \times F_2)$

$$\delta((p,q), a, Z) \ni ((r,s),\alpha) \equiv r = \delta_1(p,a) \ \& \ (s,\alpha) \in \delta_2(q,a,Z)$$

$$\delta((p,q), \lambda, Z) \ni ((p,s),\alpha) \equiv (s,\alpha) \in \delta_2(q,\lambda,Z)$$

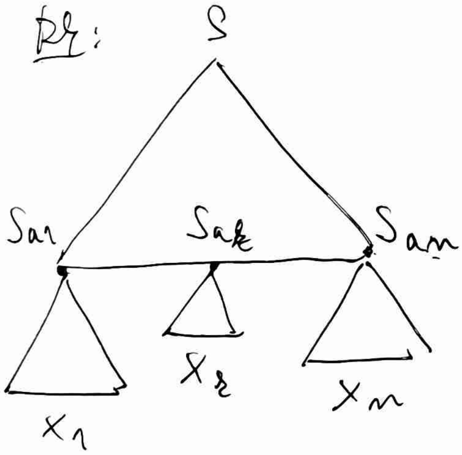
∎

Věta: CFL jsou uzavřené na substituci ⇒ tedy i homomorfismus.

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

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

Dk:



idea: listy v derivačním stromě generují další stromy

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

→ pravidla převezmu také z gramatik těch $\sigma(a)$

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

$X \to aYb \leadsto X \to S_a Y S_b$

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

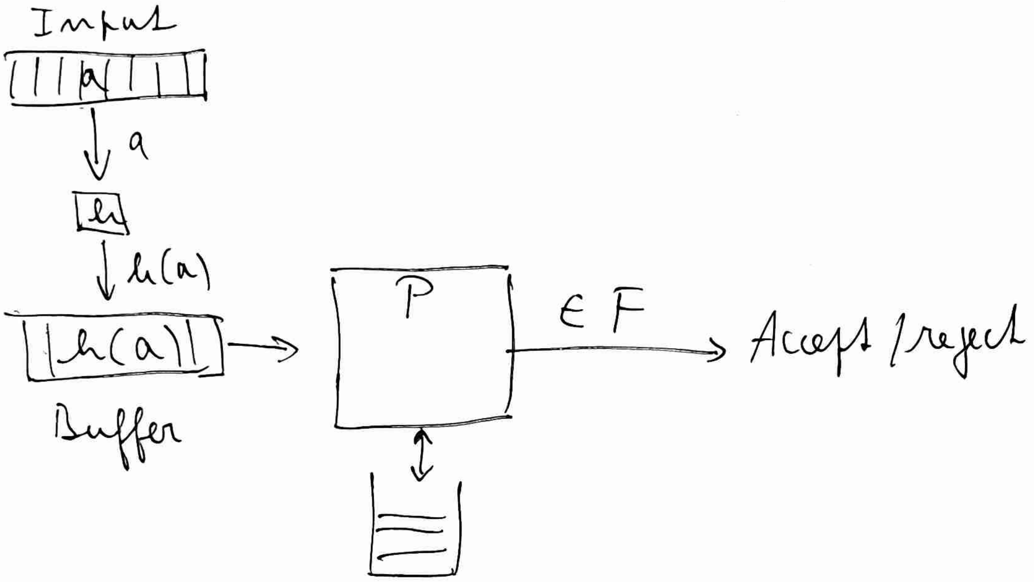
∎

Věta: CFL jsou uzavřené na inverzní homomorfismus.

→ 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 koncovým stavem



→ z inputu postupně čtu písmenka

→ vždy udělám transformaci na nějaké slovo pomocí h

→ simuluji běh P na tomto "vstupním" slově

→ po vyprázdnění bufferu načtu nové slovo

→ ~~slovo~~ je přijato ⟺ buffer je prázdný & M je v koncovém stav

? jak simulovat buffer ?

→ je konečný ⇒ pamatujeme si 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 ⇒ pak průnik ↯.
- $L_1 - L_2$ … kdyby to vždy byl CFL, tak $\overline{L} = \Sigma^* - L$ je taky CFL

Uzávěrové 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 zarážkou na zásobníku

→ cyklus 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 $\underset{\underset{L_1}{\|}}{L_{i\neq j}}$, $\underset{\underset{L_2}{\|}}{L_{j\neq k}}$, $\underset{\underset{L_3}{\|}}{L_{i\neq k}}$ jsou DCFL

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

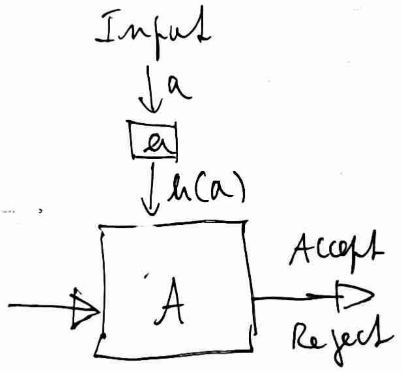
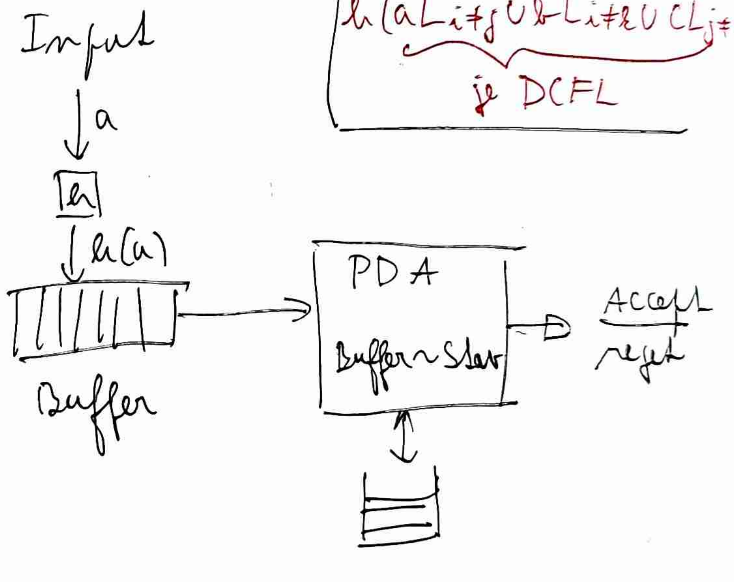
→ ale $0L_1 \cup 1L_2 \cup 2L_3$ je DCFL

↳ 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	$F = F_1 \times F_2$	$L = \{0^n 1^n 2^n \mid n \in \mathbb{N}\} = \{0^n 1^n 2^i\} \cap \{0^i 1^n 2^n\}$	
∩ 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 tot.	$A \cap B = \overline{\overline{A} \cup \overline{B}}$	$F_{COMP} = Q - F$, Z_0, cykly, δ tot.
homom.	Kleene + RegE + uz.	$a \leadsto S_a \approx$ strom $\sigma(a)$	$L = \{0^i 1^j 2^k \mid i \neq j \vee i \neq k \vee j \neq k\}$ ↳ není CFL $L = L_{i \neq j} \cup L_{i \neq k} \cup L_{j \neq k}$ $h(a) = h(b) = h(c) = \lambda$ $h(\underbrace{a L_{i \neq j} \cup b L_{i \neq k} \cup c L_{j \neq k}}_{\text{je DCFL}}) = L$
inverzní homom.			

Alternativně vnímám koncové stavy, ale TM musí po ukončení výpočtu uklidit 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 pásce, $\Sigma \subseteq \Gamma$
- $\delta : (Q - F) \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$

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

q → aktuální stav $\notin F$

X → symbol

p → nový stav

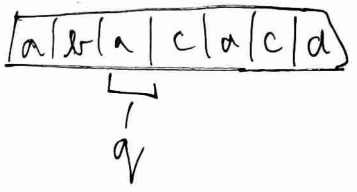
Y → před posunutím hlavy přepíšu X na Y

D ↪ směr ← nebo → kam posunu hlavu

- q_0 ... počáteční stav
- $B \in \Gamma - \Sigma$ ... Blank = symbol prázdné buňky
  ↪ na počátku všude kromě konečného počtu buněk se vstupem
- $F \subseteq Q$ ... koncové stavy

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

↳ obsah pásky + poloha hlavy + vnitřní stav



⇝ $ab(q,a)cacd$

Def: Kroky značíme $\vdash$ jako 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) ~~x~~ $w \in L(M) \iff (q_0, X) w \vdash^* \alpha (\overset{\in F}{f}, \gamma) \beta$

↳ 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 k liché stopy = pozice k-té hlavy
sudé stopy = znak na k-té pásce

→ simulace 1 kroku zjištění všechny hlavy

→ ve stavu si pamatujeme

\# hlav vlevo

\# k: symbol pod k-tou hlavou

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

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

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 pamatuju ještě neotevřené 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.

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

Def: Spočetně rekurzivní jazyk $L \iff L = L(TM)$

$w \in L \iff$ z w se dá odvodit koncový stav.

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

↳ nikdy se nezacyklí do nekonečna

↳ říkáme, že TM rozhoduje jazyk L

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

$|L_i| \leq |P_i|$

Lemma: normální tvar: $\alpha A B \to \alpha \omega B$

$\hookrightarrow \omega \in (V_N \cup V_T)^+$

① převedeme na separovanou gramatiku.

Def: G je separovaná ≡ pravidla jsou tvaru

$\alpha \to \beta \quad , \alpha, \beta \in V_N^+$

$X \to a \mid \lambda$

Alg: ke každé G ∃ ekviv. separovaná

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 $A_1 \dots A_m \to B_1 \dots B_n \quad , m \leq n$

převedeme na pravidla s novými neterminály C

kontext

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

$C_1 \underline{C_2 \dots C_m} \to \underline{B_1} \, \underline{C_2 \dots C_m} \to B_1 B_2 \dots C_m \to \dots \to \underline{B_1 \dots B_{m-1} C_m}$

$\to B_1 \dots B_{m-1} B_m \dots B_n$ ▨

Lineárně omezené automaty

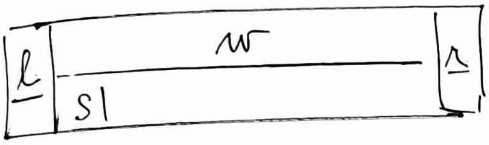
Def: LBA je nedet. TM, kde na pásce je levá a pravá zarážka $\underline{l}$ a $\underline{r}$. Tyto symboly nelze přepsat ani je překročit.

$w \in L(LBA) \equiv (q_0, \underline{l}) w \underline{r} \vdash^* \underline{l} \, \alpha (\overset{\in F}{p}, X) \beta \, \underline{r} \quad , p \in F$

$\Longrightarrow$ páska je ohraničená

Věta: Každý kontextový jazyk lze přijímat pomocí LBA.

Dk: prázdné slovo vyřešíme zvlášť:



→ použijeme pásku se dvěma stopami

→ na horní pásku dáme w

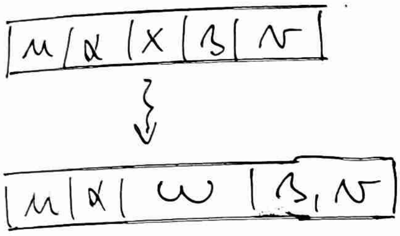
→ na začátek dolní S

→ přepisujeme slovo dole podle pravidel G

1. nedeter. vyber část k přepsání

2. aplikuj pravidlo, βv odsuň doprava

$\alpha X \beta \rightarrow \alpha \omega \beta$



↳ pokud už samé terminály

→ porovnej s horní stopou → slovo přijmeme / zamítneme ▨

Věta: Každý LBA odpovídá nějaké kontextové g.

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

→ pro stavy p, q... → netermin. P, Q...

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

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

→ začátek q_0, a_0 ← první znak vstupu

→ nějak se ještě musí ošetřit konec

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

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

→ gramatika nejprve vygeneruje pásku, stavy a kopii slova

→ potom simuluje výpočet (stavy jsou součástí 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 $S \Rightarrow \omega_1 \Rightarrow \dots \Rightarrow \omega_n = w$ kódujeme jako

$$\#S\#\omega_1\#\dots\#w\#$$

• Diagonální jazyk

Def: Kódování TM pomocí řetězců 0 a 1.

→ očíslujeme stavy a symboly na pásce

left = 0
right = 00

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

→ 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 ... ∄ 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 otázek kódovaná řetězci nad abecedou Σ s odpověďmi True / False

Problém je rozhodnutelný ≡ ∃ TM t.ž. pro ∀ vstup z problému zastaví a navíc přijíme pouze True

Příklady:

- obsahuje slovo 5 nul?
- je vstupní slovo korektní definujícím kódem TM podle definice výše?
- Zastaví TM kódu M nad slovem w?

↳ halting problem není rozhodnutelný

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

Dk: Máme Turingovy stroje $L = L(M_1)$, $\bar{L} = L(M_2)$

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

↳ 2 pásky ⇒ stav jsou uspořádané dvojice

→ 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

⇒ $L(M) = L$ 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.

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

Časová složitost

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

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

Def. Pro fci $t: N \to R^+$ 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 rozhodne nejpozději po $O(t(n))$ krocích.

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

1) zkontroluj vstup $0^i 1^j$, pokud za 1 je 0 → False $O(n)$
2) vrať se na začátek ... $O(n)$
3) procházej postupně 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 ani 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: Gramatiku převedeme do ChNF (nezávisí na n)

↳ CYK algoritmus je $O(n^3)$

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

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

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

👁 : pokud ∃ polyg. verifikátor (→ rozhodne v poly. čase), pak vždy ∃ poly. certifikát

↳ delší by verifikátor ani nestihl přečíst

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

= jazyky s poly. verifikátorem.

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

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

Věta: $NP = \bigcup_k \text{NTIME}(n^k)$

Def (Převoditelnost v poly. čase) Funkce $f: \Sigma^* \to \Sigma^*$ je vyčíslitelná v poly. čase ≡ ∃ TM který pro každý vstup w zastaví v poly. čase s $f(w)$ na pásce

Jazyk A je převoditelný v poly. čase na jazyk B, $A \leq_p B$ ≡

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

↳ polynomiální redukce A do B

Def: Jazyk B je NP-úplný ≡ $B \in NP$ & $\forall A \in NP: A \leq_p B$

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

Věta: ———— " ———— & $B \leq_p C$ ⇒ 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 zastaví pro každý vstup je prostorová slož. 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: nemá ani 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