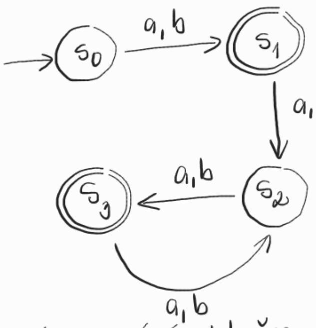
Teoretická informatika 2024/25 – Erika Do, xdoeri00

1. Uvažujme abecedu $\Sigma = \{a, b\}$ a jazyk $L_1 = \{w \in \Sigma^* : |w| \bmod 2 = 1\}$. Sestrojte relaci pravé kongruence $\sim$, která splňuje následující tři podmínky: (1) L_1 je sjednocením některých tříd rozkladu $\Sigma^*/_\sim$, (2) index $\sim$ je konečný a soudělný s indexem $\sim_{L_1}$ a (3) jedna ze tříd rozkladu $\Sigma^*/_\sim$ má právě dva prvky.

- ze 3 podmínek se nejdříve pokusíme splnit podmínku (2), ta nám říká, že $\sim$ je soudělná s indexem $\sim_{L_1}$, což znamená, že jejich největší společný dělitel musí být větší než 1
- jelikož počet stavů minimálního DKA přijímajícího L_1 je roven indexu $\sim_{L_1}$, tak ten samý DKA (avšak neminimální) přijímající L_1 určuje jistou pravou kongruenci s konečným indexem
- sestrojme takový DKA + definujme si jednotlivé jazyky přístupových řetězců pro dané stavy:



$L^{-1}(s_0) = \varepsilon$

$L^{-1}(s_1) = (a+b)$

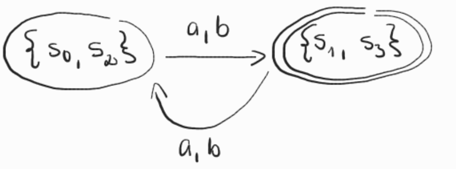
$L^{-1}(s_2) = (a+b)(a+b)((a+b)(a+b))^*$

$L^{-1}(s_3) = \Sigma^* \setminus \{\varepsilon, (a+b), (a+b)(a+b)((a+b)(a+b))^*\}$

$\Sigma^*/_\sim = \{L^{-1}(s_0), L^{-1}(s_1), L^{-1}(s_2), L^{-1}(s_3)\} \Rightarrow$ index 4

- není minimální, dokažme si zde:

$\overset{0}{\equiv}$	a	b
$\rightarrow s_0$	II	II
I s_2	II	II
$\leftarrow s_1$	I	I
$\leftarrow s_3$	I	I
II		



$L^{-1}(\{s_0, s_2\}) = \{w \in \Sigma^* \mid |w| \bmod 2 = 0\}$

$L^{-1}(\{s_1, s_3\}) = \{w \in \Sigma^* \mid |w| \bmod 2 = 1\}$

$\Sigma^*/_{\sim_{L_1}} = \{L^{-1}(\{s_0, s_2\}), L^{-1}(\{s_1, s_3\})\} \Rightarrow$ index 2

- podmínka (2) tedy splněna. Podmínka (1) je taktéž splněna, jelikož L_1 může být reprezentováno sjednocením $L^{-1}(s_1)$ a $L^{-1}(s_3)$, což jsou 2 třídy rozkladu $\Sigma^*/_\sim$.
- v neposlední řadě je podmínka (3) splněna tím, že právě 1 třída (přesněji $L^{-1}(s_1)$) má 2 prvky

3. Uvažujte následující jazyk nad abecedou $\Sigma = \{a, b\}$:

$$L_3 = \{a^k b^\ell \mid \ell = k^2\}$$

Dokažte, že jazyk L_3 není bezkontextový.

- L_3 není bezkontextový, to dokážeme pomocí Pumping lemmatu
- uvažujme o lib. $k > 0$ a pro každé takové k vyberme slovo $z = a^k b^{k^2}$
- pro každé takové slovo z uvažujme o všech rozděleních $z = uvwxy$, kde $|vx| > 0$ a $|vwx| \leq k$
- všechna taková rozdělení zařadíme do dvou tříd:
  1. část vwx obsahuje jediný typ symbolu
  2. část vwx obsahuje dva typy symbolů

(1) část vwx obsahuje jediný typ symbolu
- v případě, že vx obsahuje pouze symboly a (b), tak pumpováním změníme počet symbolů a (b) na $k + i \cdot |v| + |x|$, ale počet symbolů b zůstane k^2. Potom už zaručeně nebude platit, že počet b je 2. mocninou počtu a $\Rightarrow$ tedy slovo $uv^iwx^iy \notin L_3$ pro $i \neq 1$ (při pumpování pouze b bychom změnili jejich počet, bez toho abychom nijak pozměnili počet symbolů a. Což opět znamená, že počet symbolů b nebude roven 2. mocnině počtu symbolů a $\Rightarrow uv^iwx^iy \notin L_3$ pro $i \neq 1$)

(2) část vwx obsahuje dva typy symbolů
- v případě, že vx obsahuje symboly a i b, tak vwx obsahuje pouze menší část z a^k i b^{k^2}. Ať už budeme pumpovat jakkoliv, tak porušíme pořadí symbolů a i b ve slově z $\Rightarrow$ dostáváme slova, jež neodpovídají formátu $a^k b^{k^2}$ $\Rightarrow$ tedy $uv^iwx^iy \notin L_3$ pro $i \neq 1$

4. Navrhněte algoritmus, který pro bezkontextovou gramatiku $G = (N, \Sigma, P, S)$ spočítá množinu

$$N_{abc} = \{A \in N \mid \exists u, v \in \Sigma^* : A \Rightarrow_G^* uabcv\}.$$

V algoritmu můžete využít množiny $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$ a $N_t = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow_G^+ w\}$. Doporučujeme nadefinovat si další vhodné množiny neterminálů a algoritmicky popsat jejich výpočet (u N_ϵ a N_t popis výpočtu není potřeba).

Ilustrujte použití algoritmu na příkladě gramatiky nad abecedou $\Sigma = \{a, b, c, d\}$ s pravidly

$S \to V \mid caUca \mid RTcU \quad V \to Vabc \quad U \to b \mid \epsilon \quad R \to Uca \quad T \to W \mid aU \quad W \to UbU$

- nejdřív si vypočítáme N_ε a N_t
- dále si definujeme pomocné množiny pro pravidla

1) končící a:
- vstup: $G = (N, \Sigma, P, S)$
- výstup: $N_a = \{A \in N \mid A \Rightarrow_G^+ za;\ a \in \Sigma;\ z \in (N_t \cup \Sigma)^*\}$
- princip:
  1. $N_a^0 = \emptyset$
  2. for i in [0...]:
  3. $N_a^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in ((N_t \cup \Sigma)^*(\{a\} \cup N_a^i) N_\varepsilon^*)\} \cup N_a^i$
  4. if $N_a^{i+1} == N_a^i$
  5. return N_a^i

2) začínající c:
- vstup: $G = (N, \Sigma, P, S)$
- výstup: ${}_cN = \{A \in N \mid A \Rightarrow_G^+ cz;\ c \in \Sigma;\ z \in (N_t \cup \Sigma)^*\}$
- princip:
  1. ${}_cN^0 = \emptyset$
  2. for i in [0...]:
  3. ${}_cN^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\varepsilon^*(\{c\} \cup {}_cN^i)(N_t \cup \Sigma)^*)\} \cup {}_cN^i$
  4. if ${}_cN^{i+1} == {}_cN^i$:
  5. return ${}_cN^i$

3) končící ab:
- vstup: $G = (N, \Sigma, P, S)$
- výstup: $N_{ab} = \{A \in N \mid A \Rightarrow_G^+ zab;\ a \in \Sigma;\ b \in \Sigma;\ z \in (N_t \cup \Sigma)^*\}$
- princip:
  1. $N_{ab}^0 = \emptyset$

2. for i in [0...]:
3. $N_{ab}^{i+1} = \{A \in N \mid \exists (A,\alpha)\in P : \alpha \in ((N_t \cup \Sigma)^*((\{a\}N_\varepsilon^*\{b\}) \cup N_{ab}^i)N_\varepsilon^*)\} \cup N_{ab}^i$
4. if $N_{ab}^{i+1} == N_{ab}^i$
5. return N_{ab}^i

4) začínající bc:
- vstup: $G=(N,\Sigma,P,S)$
- výstup: ${}_{bc}N = \{A\in N \mid A \Rightarrow_G^+ bcz;\ c\in\Sigma;\ b\in\Sigma;\ z\in(N_t\cup\Sigma)^*\}$
- princip: 1. ${}_{bc}N^0=\emptyset$
2. for i in [0...]:
3. ${}_{bc}N^{i+1} = \{A\in N\mid \exists(A,\alpha)\in P: \alpha\in(N_\varepsilon^*((\{b\}N_\varepsilon^*\{c\})\cup {}_{bc}N^i)(N_t\cup\Sigma)^*)\}\cup {}_{bc}N^i$
4. if ${}_{bc}N^{i+1} == {}_{bc}N^i$:
5. return ${}_{bc}N^i$

5) generující pouze 1 b:
- vstup: $G=(N,\Sigma,P,S)$
- výstup: $N_b = \{A\in N\mid A\Rightarrow_G^+ b;\ b\in\Sigma\}$
- princip: 1. $N_b^0=\emptyset$
2. for i in [0...]:
3. $N_b^{i+1} = \{A\in N\mid\exists(A,\alpha)\in P: \alpha\in(N_\varepsilon^*(N_b^i\cup\{b\})N_\varepsilon^*)\}\cup N_b^i$
4. if $N_b^{i+1} == N_b^i$
5. return

6) obsahující podřetězec „abc":
- vstup: $G=(N,\Sigma,P,\ )$
- výstup: $N_{„abc"} = \{A\in N\mid A\Rightarrow_G^+ zabcz;\ a\in\Sigma;\ c\in\Sigma;\ b\in\Sigma;\ z\in(N_t\cup\Sigma)^*\}$
- princip: 1. $N^0_{„abc"}=\emptyset$
2. for i in [0...]:
3. $N^{i+1}_{„abc"} = \{A\in N\mid\exists(A,\alpha)\in P:$
$\alpha\in((N_t\cup\Sigma)^*(\{a\}N_\varepsilon^*\{b\}N_\varepsilon^*\{c\})\cup N^i_{„abc"})(N_t\cup\Sigma)^*)\}\cup N^i_{„abc"}$
4. if $N^{i+1}_{„abc"} == N^i_{„abc"}$
5. return $N^i_{„abc"}$

– nyní musíme vzít v potaz všechny možné kombinace, které mohou nastat při generování N_{abc}, nejdříve proběhne výpočet pomocných podmnožin:
1. N_ε
2. N_t
3. N_a = 1. algoritmus (G, N_t)
4. ${}_cN$ = 2. algoritmus (G, N_t)
5. N_{ab} = 3. algoritmus (G, N_t)
6. ${}_{bc}N$ = 4. algoritmus (G, N_t)
7. N_b = 5. algoritmus (G, N_t)

– rozdělení:

a) $N_a\{b\}{}_cN$
- vstup: $G=(N,\Sigma,P,S)$
- výstup: $N_{abc1} = \{A\in N\mid \exists w\in\Sigma^*: A\Rightarrow_G^+ w \wedge \exists u,v\in\Sigma^*: w=uabcv\}$
- princip: 1. $N^0_{abc1}=\emptyset$
2. for i in [0...]:
3. $N^{i+1}_{abc1} = \{A\in N\mid\exists(A,\alpha)\in P: \alpha\in((\{a\}\cup N_a)\{b\}(\{c\}\cup{}_cN))\}\cup N^i_{abc1}$
4. if $N^{i+1}_{abc1} == N^i_{abc1}$
5. return N^i_{abc1}

– další rozdělení budou fungovat obdobně, kde každý algoritmus bude využívat náležité pomocné množiny (jejich iterace vypadají následovně):

b) $N_a\ N_b\ {}_cN$
$N^{i+1}_{abc2} = \{A\in N\mid\exists (A,\alpha)\in P: \alpha\in((\{a\}\cup N_a)N_b(\{c\}\cup {}_cN))\}\cup N^i_{abc2}$

c) $N_{ab}\ {}_cN$
$N^{i+1}_{abc3} = \{A\in N\mid\exists (A,\alpha)\in P: \alpha\in(((\{a\}N_\varepsilon^*\{b\})\cup N_{ab})(\{c\}\cup{}_cN))\}\cup N^i_{abc3}$

d) $N_{ab}\{c\}$
$N^{i+1}_{abc4} = \{A\in N\mid\exists (A,\alpha)\in P: \alpha\in(((\{a\}N_\varepsilon^*\{b\})\cup N_{ab})N_\varepsilon^*\{c\}(N_t\cup\Sigma)^*)\}\cup N^i_{abc4}$

e) $N_a\ {}_{bc}N$
$N^{i+1}_{abc5} = \{A\in N\mid\exists (A,\alpha)\in P: \alpha\in((\{a\}\cup N_a)((\{b\}N_\varepsilon^*\{c\})\cup{}_{bc}N))\}\cup N^i_{abc5}$

f) $N_a\{bc\}$
$N^{i+1}_{abc6} = \{A\in N\mid\exists (A,\alpha)\in P: \alpha\in((\{a\}\cup N_a)N_\varepsilon^*(\{b\}N_\varepsilon^*\{c\})(N_t\cup\Sigma)^*)\}\cup N^i_{abc6}$

g) $\{ab\}{}_cN$
$N^{i+1}_{abc7} = \{A\in N\mid\exists (A,\alpha)\in P: \alpha\in((N_t\cup\Sigma)^*(\{a\}N_\varepsilon^*\{b\})N_\varepsilon^*(\{c\}\cup{}_cN))\}\cup N^i_{abc7}$

h) $\{a\}{}_{bc}N$
$N^{i+1}_{abc8} = \{A\in N\mid\exists (A,\alpha)\in P: \alpha\in((N_t\cup\Sigma)^*\{a\}N_\varepsilon^*((\{b\}N_\varepsilon^*\{c\})\cup{}_{bc}N))\}\cup N^i_{abc8}$

j) $\{a\}N_b\{c\}$
$N^{i+1}_{abc9} = \{A\in N\mid\exists (A,\alpha)\in P: \alpha\in((N_t\cup\Sigma)^*\{a\}N_b\{c\}(N_t\cup\Sigma)^*)\}\cup N^i_{abc9}$

– tyto spočítané algoritmy se využijí ve finální výpočtu N_{abc}:
- vstup: $G=(N,\Sigma,P,S)$
- výstup: $N_{abc} = \{A\in N\mid\exists u,v\in\Sigma^*: A\Rightarrow_G^* uabcv\}$
- princip: 1. $N^0_{abc}=\emptyset$
2. for i in [0...]:
3. $N^{i+1}_{abc} = \{N_{abc1}\cup N_{abc2}\cup N_{abc3}\cup N_{abc4}\cup N_{abc5}\cup N_{abc6}\cup N_{abc7}\cup N_{abc8}\cup N_{abc9}\}$
$\cup\ (N_t\cup\Sigma)^* N^i_{abc}(N_t\cup\Sigma)^*$
4. if $N^{i+1}_{abc} == N^i_{abc}$
5. return N^i_{abc}

- demonstrace:

$N_\varepsilon^0 = \emptyset$
$N_\varepsilon^1 = \{U\} = N_\varepsilon$

$N_f^0 = \emptyset$
$N_f^1 = \{U\}$
$N_f^2 = \{U, R, T, W\} = N_f$

$N_a^0 = \emptyset$
$N_a^1 = \{T, R, S\} = N_a$

${}_cN^0 = \emptyset$
${}_cN^1 = \{S, R\} = {}_cN$

$N_{ab}^0 = \emptyset = N_{ab}$

${}_{bc}N^0 = \emptyset = {}_{bc}N$

$N_b^0 = \emptyset$
$N_b^1 = \{U, W\} = N_b$

$N_{\text{"abc"}}^0 = \emptyset = N_{\text{"abc"}}$

$N_{abc1}^0 = \emptyset = N_{abc1}$

$N_{abc2}^0 = \emptyset = N_{abc2}$

$N_{abc3}^0 = \emptyset = N_{abc3}$

$N_{abc4}^0 = \emptyset = N_{abc4}$

$N_{abc5}^0 = \emptyset = N_{abc5}$

$N_{abc6}^0 = \emptyset = N_{abc6}$

$N_{abc7}^0 = \emptyset = N_{abc7}$

$N_{abc8}^0 = \emptyset = N_{abc8}$

$N_{abc9}^0 = \emptyset$
$N_{abc9}^1 = \{S\} = N_{abc9}$

$N_{abc}^0 = \emptyset$
$N_{abc}^1 = \{S\} = N_{abc}$

2. Uvažte následující operaci na jazycích nad abecedou Σ:

$$\square L = \{w \in L \mid \forall u, v \in \Sigma^* : w = uv \Rightarrow u \in L\},$$

Rozhodněte a dokažte, zda jsou následující třídy jazyků uzavřeny na operaci $\square$:

(a) třída regulárních jazyků a

(b) třída rekurzivně vyčíslitelných jazyků.

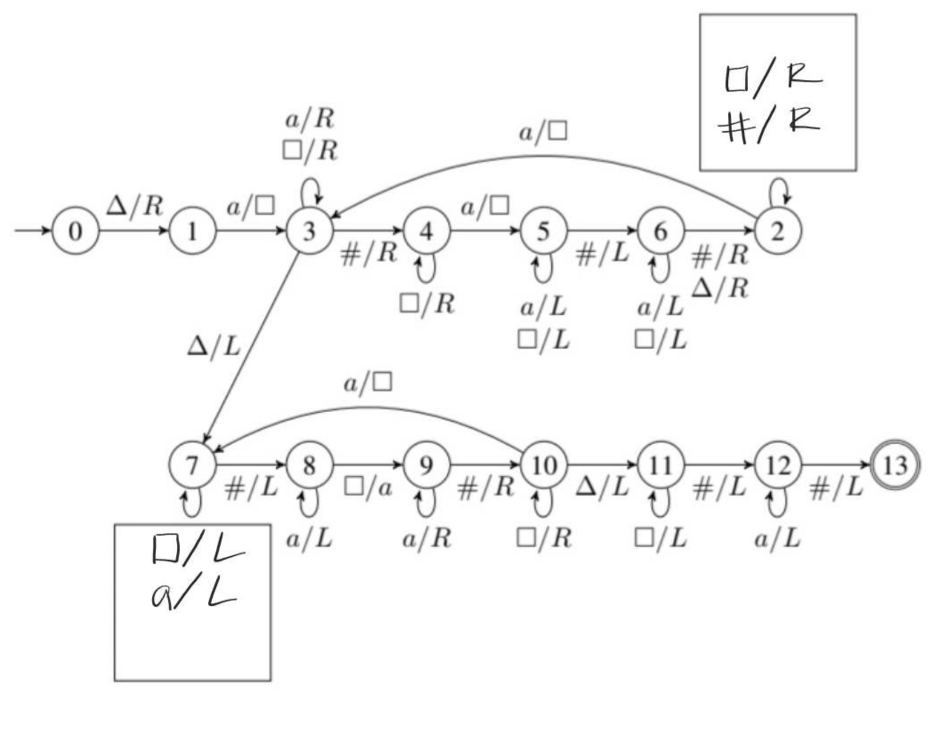
a)

- z definice regulárního jazyka víme, že jazyk je regulární, pokud existuje KA, jež akceptuje všechna slova z jazyka
- předpokládejme, že je jazyk L regulární a chceme dokázat, že jazyk $\square L$ je taktéž regulární
- konstruujme KA $A = (Q, \Sigma, \delta, q_0, F)$ takový, aby $L(A) = L$
- nyní musíme rozhodnout, zda $w \in \square L \Rightarrow$ je nutné ověřit, zda pro každou dekompozici $w = uv$ platí, že $u \in L$
- sestavíme nový automat $B = (Q_2, \Sigma_2, \delta_2, q_{2}, F_2)$ takový, že $L(B) = \square L$:
  - důležité je si uvědomit, že B bude mít stavy odpovídající stavům A, kde pro každý stav $q \in A$ platí, že pokud pro každou dekompozici $w = uv$ (včetně ε jako prefix) dojde do akceptujícího stavu F ze startujícího stavu q_0, pak B dané w přijme
  - toto můžeme provést pomocí simulace, která pro každý prefix u prochází automat A a kontroluje, zda u končí v akceptujícím stavu F
- konstrukcí nového automatu B jsme dokázali, že pro L existuje regulární jazyk $\square L \Rightarrow$ třída reg. jazyků je uzavřena na operaci $\square$

b)

- definice $\mathcal{L}_{REC}$ víme, že jazyk L je rekurzivně vyčíslitelný, pokud existuje TS, který akceptuje všechna slova z L a buď odmítá, nebo cyklí pro slova, která nejsou v L
- předpokládejme, že je jazyk L rek. vyčíslitelný a chceme dokázat, že jazyk $\square L$ je taktéž rek. vyčíslitelný
- každé slovo w má koneč. počet předpon $u \Rightarrow$ dokážeme zjistit všechny jeho předpony
- konstruujme TS T takový, kde pro každé slovo w v L zkontroluje, zda platí, že pro každou možnou dekompozici $w = uv$ (u prefix, v zbytek slova) platí, že $u \in L$
  - TS pro L může vždy rozhodnout, zda $u \in L$ pro každý prefix u slova $w \Rightarrow$ v případě, že pro všechny dekompozice w dostaneme, že $u \in L$, pak $w \in \square L$
- jelikož existuje koneč. počet předpon pro každé slovo a zároveň každé rozhodnutí o náležitosti dané předpony zabere koneč. čas $\Rightarrow$ jsme schopni v koneč. časem určit, zda $w \in \square L$
- tímto způsobem jsme dokázali, že pokud L je rek. vyčíslitelný, pak také $\square L$ je rek. vyčíslitelný $\Rightarrow$ třída rek. vyčíslitelných jazyků je uzavřena na operaci $\square$

5. Doplňte do rámečků v přechodovém diagramu Turingova stroje M_5 s páskovou abecedou $\Gamma = \{a, \#, \square, \Delta\}$ v Obrázku 1 chybějící popisky přechodů tak, aby platilo, že $L(M_5) = \{a^{\ell_1}\#a^{\ell_2}\#a^{\ell_3} \mid 1 \leq \ell_1 \leq \ell_2 \leq \ell_3 \wedge \ell_2 - \ell_1 = \ell_3 - \ell_2\}$. V jednom popisku může být i více operací. Nic jiného nepřidávejte.



Obrázek 1: Přechodový diagram Turingova stroje M_5