Teoretické základy informatických vied

Regulárne gramatiky

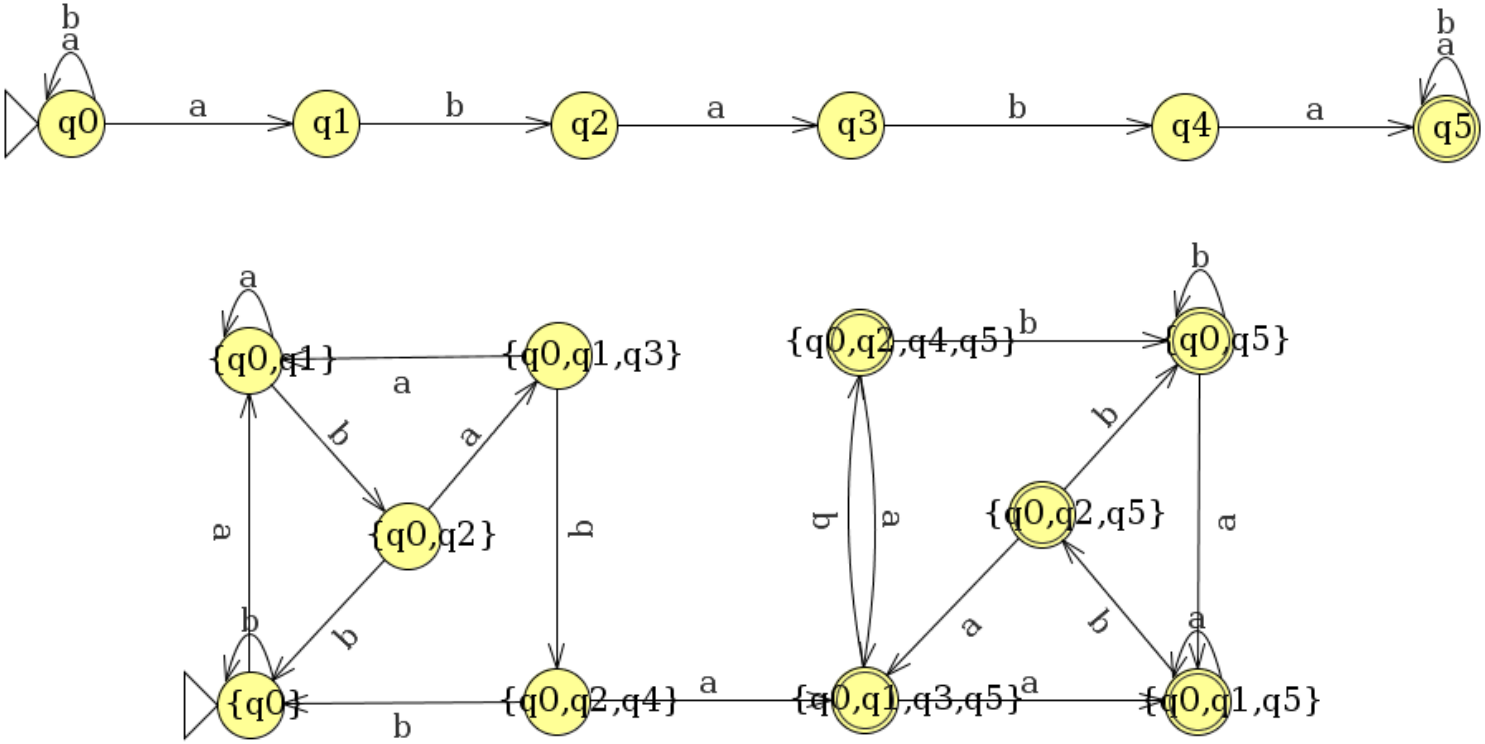
Mgr. Martin Bobák, PhD.


Ústav informatiky
Slovenská akadémia vied

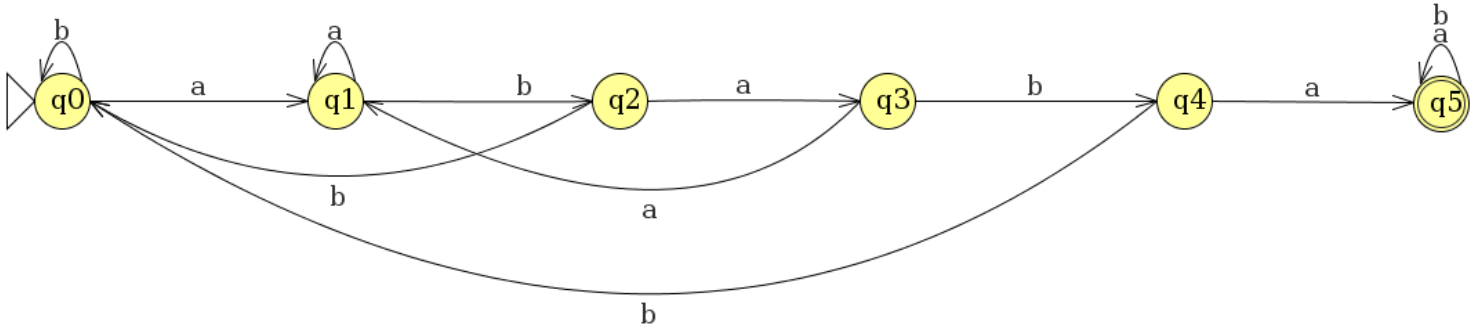
2022/2023

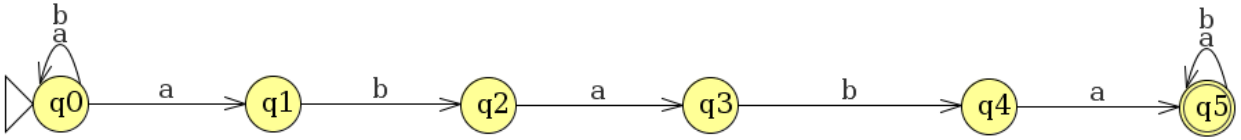
Pôvodný autor: doc. Mgr. Daniela Chudá, PhD. Teoretické základy informatických vied, FIIT STU, 2020.

$$L(A) = \{xababay | x, y \in \{a, b\}^*\}$$



$$L(A) = \{xababay | x, y \in \{a, b\}^*\}$$





Poznámka:

- Automat, ktorý rozpoznáva podreťazec s t.j. rozpoznáva jazyk $L = \{xsy | x, y \in \Sigma^*\}$, má počet stavov rovný $|s| + 1$[1], ak je dĺžka $s \geq 1$[2].
- V tomto prípade platí pre NFA a DFA, že sa nelíšia stavmi, ale iba prechodovými funkciami.

[1] +1 pre počiatočný stav

[2] pre $s = \varepsilon$ potrebujeme dva stavy t.j. počet stavov je $|s| + 2$

Oznamy

- prvý test na 4. cvičení:
  - čas: 20 minút
  - témy: 1. – 3. cvičenie (množiny, jazyky a konečné automaty)
  - bodové hodnotenie: 8 bodov
  - prevažne vyriešenie podobných úloh ako na cvičení, okrajovo teória

Teória formálnych jazykov

Reprezentácia jazyka:

- Matematický opis jazyka – **množina**.
- Systematické generovanie slov z jazyka (generovanie jazyka) – **gramatika**.
- Zostrojenie algoritmu, ktorý určí, či dané slovo patrí do jazyka (rozpoznanie jazyka) – **automat**

Teória jazykov:

- štúdium množín znakov, reťazcov, ich reprezentácií, štruktúr a vlastností

Frázová gramatika

Definícia (Definícia frázovej gramatiky)

Frázová gramatika *je štvorica* $G = (N, T, P, S)$, *kde* N *a* T *sú abecedy terminálnych resp. neterminálnych symbolov, pričom platí:*
$(N \cap T) = \emptyset$
$P \subseteq_{KON} (N \cup T)^* N (N \cup T)^* \times (N \cup T)^*$ [a]
$S \in N$ *je počiatočný (štartovací) neterminálny symbol.*

[a] množina pravidiel je konečná

Poznámky:

- ľavá strana pravidla musí obsahovať aspoň jeden neterminál
- $(u, v) \in P$ môžeme zapisovať $u \rightarrow_P v, u \rightarrow_G v$, prípadne len $u \rightarrow v$ (ak vieme, o akú množinu pravidiel P, resp. gramatiku G ide).
- Pravidlá $u \rightarrow v_1$, $u \rightarrow v_2$,... $u \rightarrow v_n$ s rovnakou ľavou stranou zapisujeme skrátene $u \rightarrow v_1 | v_2 | ... | v_n$.

Frázová gramatika

Príklad

G = (N, T, P, A), kde N = {A, B}, T = {a, b},

$$P = \{A \rightarrow aA|B$$
$$B \rightarrow bB|\varepsilon\}$$

$L(G) = \{a^i b^j | i, j \in \mathbb{N}\}$

A	použité pravidlo:
aA	$A \rightarrow aA$
aaA	$A \rightarrow aA$
aaaA	$A \rightarrow aA$
aaaB	$A \rightarrow B$
aaabB	$B \rightarrow bB$
aaabbB	$B \rightarrow bB$
aaabb	$B \rightarrow \varepsilon$

Definícia (Definícia odvodenia)

Odvodenie $S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_n$ *je postupnosť krokov odvodení.* ***Dĺžka odvodenia*** *je počet takýchto krokov.*

Definícia (Definícia vetnej formy)

Vetná forma *je slovo z* $(N \cup T)^*$*, ktoré môžeme získať odvodením zo začiatočného neterminálu.*

Definícia (Definícia kroku odvodenia)

Krok odvodenia *v gramatike G je binárna relácia $\Rightarrow_G$ na množine $(N \cup T)^* \times (N \cup T)^*$ definovaná nasledovne:*

$$x \Rightarrow_G y,$$

ak existujú $w_1, w_2 \in (N \cup T)^*$ a pravidlo $(u \to v) \in P$ také, že $x = w_1 u w_2$ a $y = w_1 v w_2$.

Poznámky:

- binárna relácia
- operácie: mocniny, tranzitívny ($\Rightarrow^+$), reflexívno-tranzitívny uzáver ($\Rightarrow^*$), atď.
  $u \Rightarrow^2 v$, ak $\exists w$ také, že $u \Rightarrow w$ a $w \Rightarrow v$. Znamená to, že z vetnej formy u vieme na 2 kroky odvodiť vetnú formu v.
- $u \Rightarrow^* v$ znamená, že z u vieme v vygenerovať konečným počtom krokov.

Definícia (Definícia jazyka generovaného gramatikou)

Jazyk generovaný gramatikou *G je množina $L(G)$ daná nasledovne:*

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

pričom $\Rightarrow_G^*$ je reflexívno-tranzitívny uzáver relácie $\Rightarrow_G^*$.

Treba ukázať obe inklúzie:

- Každé slovo, ktoré gramatika generuje, patrí do L (gramatika negeneruje nič navyše) $L(G) \subseteq L$.
- Každé slovo z množiny L má v gramatike odvodenie – $L \subseteq L(G)$.

Definícia (Definícia ekvivalencie gramatík)

Gramatiky G_1 a G_2 sú ekvivalentné, ak $L(G_1) = L(G_2)$.

Gramatika

- Gramatika – prostriedok na opis nekonečného jazyka konečným spôsobom
- Opis jazyka generatívnou paradigmou
- Široké uplatnenie pri definovaní umelých (najmä počítačových) jazykov

Gramatika

- Gramatika je konečný objekt (lebo všetky množiny/prvky tvoriace gramatiku, hlavne množina pravidiel, sú konečné). Gramatikám jedno-jednoznačne priradiť prirodzené čísla, a teda rôznych **gramatík je spočítateľne veľa**.
- množina $\{a, b\}^*$ je nekonečná spočítateľná, a teda množina jej podmnožín (jazykov nad abecedou {a,b}) je nespočítateľná. **Rôznych jazykov je nespočítateľne veľa**.
- **k väčšine jazykov neexistuje gramatika**, ktorá ich generuje.

Definícia (Definícia klasifikácie gramatík (Chomsky))

Nech $G = (N,T,P,S)$ je gramatika. Potom hovoríme, že gramatika G je:

- ***frázová*** *(typu 0), ak sa na tvar prepisovacích pravidiel nekladú žiadne obmedzenia*
- ***kontextová*** *(typu 1), ak každé prepisovacie pravidlo z P má tvar $u \to v$, kde $|u| \leq |v|$*
- ***bezkontextová*** *(typu 2), ak každé prepisovacie pravidlo z P má tvar $A \to w$, kde $A \in N$, $w \in (N \cup T)^*$*
- ***regulárna*** *(typu 3), ak každé prepisovacie pravidlo z P má jeden z tvarov $A \to wB$ alebo $A \to w$, kde $A, B \in N$,$w \in T^*$*

Klasifikácia gramatík

Poznámky:

- V pôvodnej Chomského definícii kontextových gramatík mali ich pravidlá tvar $uAv \to uwv$, kde $u, v \in (N \cup T)^*, w \in (N \cup T)^+, A \in N$. (neterminál A sa mohol prepísať na w len vtedy, ak sa vyskytoval v správnom kontexte – odtiaľ názov kontextová gramatika). Dá sa ukázať, že tieto dve definície sú ekvivalentné.
- Bezkontextová gramatika má pravidlá, ktorých ľavú stranu tvorí práve jeden neterminál. Názov bezkontextová pochádza teda zo skutočnosti, že príslušný neterminál môžeme prepísať bez ohľadu na kontext (okolie), v ktorom sa nachádza.
- Regulárna gramatika má navyše (oproti bezkontextovej) obmedzenú aj pravú stranu pravidiel – tá môže obsahovať najviac jeden neterminál a ten musí byť posledný. Vďaka tomu regulárna gramatika generuje slovo zľava doprava.

Veta

Každý konečný jazyk je regulárny.

Dôsledok: Umelé jazyky C, C++, Pascal, UML sú nekonečné.

Poznámka: Platnosť tohto dôsledku sa dá rozšíriť aj pre mnoho iných umelých jazykov.

Veta

Ku každej regulárnej gramatike G existuje regulárna gramatika G' taká, že platí:

- *dĺžka pravej strany každého pravidla v G' je nanajvýš 2*
- $L(G') = L(G)$

Silnejšie normálne tvary regulárnych gramatík:

- v každom kroku generuje regulárna gramatika najviac jeden terminál – $P \subseteq_{KON} N \times (T \cup \{\varepsilon\})(N \cup \{\varepsilon\})$
- $P \subseteq_{KON} N \times (T \cup T(N \setminus \{S\})) \cup \{S \to \varepsilon\}$

Neformálna idea dôkazu:

- ak pravidlo generuje viacej (ne)terminálov rozbijeme ho na viacej pravidiel. Pridaním nových neterminálov zabezpečíme, aby sme nové pravidlá museli použiť v správnom poradí a dosiahli rovnakú vetnú formu ako v pôvodnej gramatike.

$$L = \{a^{3n} | n \in \mathbb{N}\}$$

$G = (N, T, P, S)$
$N = \{S\}$,
$T = \{a\}$,
$P = \{\mathbf{S} \to aaa\mathbf{S} | \varepsilon\}$

$G = (N, T, P, S)$
$N = \{S, S_1, S_2\}$,
$T = \{a\}$,
$P = \{S \to aS_1 | \varepsilon$
$S_1 \to aS_2$
$S_2 \to aS\}$

$$L = \{b^{3k}c^2 | k \in \mathbb{N}\}$$

$G = (N, T, P, S)$
$N = \{S\}$,
$T = \{b, c\}$,
$P = \{\mathbf{S} \to cc | bbb\mathbf{S}\}$

$G = (N, T, P, S)$
$N = \{S, U, V, Z\}$,
$T = \{b, c\}$,
$P = \{S \to cU$
$U \to c$
$S \to bV$
$V \to bZ$
$Z \to bS\}$

Ekvivalencia konečných automatov a regulárnych gramatík

Regulárna gramatika:

- normálne tvary gramatík zabezpečujú, že vetná forma je buď terminálne slovo, alebo obsahuje práve jeden neterminál, ktorý sa nachádza na jej konci
- generuje slovo zľava doprava
- prenos informácie je zabezpečený zmenou neterminálu

Konečný automat:

- číta slovo zľava doprava
- pamätá si konečnú informáciu v stave

Ekvivalencia konečných automatov a regulárnych gramatík

Veta

K ľubovoľnému nedeterministickému konečnému automatu A existuje regulárna gramatika G taká, že L(G)=L(A).

- generovanie slova v gramatike G simuluje výpočet v automate A
- neterminál na konci vetnej formy zodpovedá stavu, v ktorom sa nachádza automat A po prečítaní zodpovedajúcej časti slova.
- odvodenie ukončíme, keď aktuálny neterminál zodpovedá akceptačnému stavu

Ekvivalencia konečných automatov a regulárnych gramatík

Veta

K ľubovoľnému nedeterministickému konečnému automatu A existuje regulárna gramatika G taká, že L(G)=L(A).

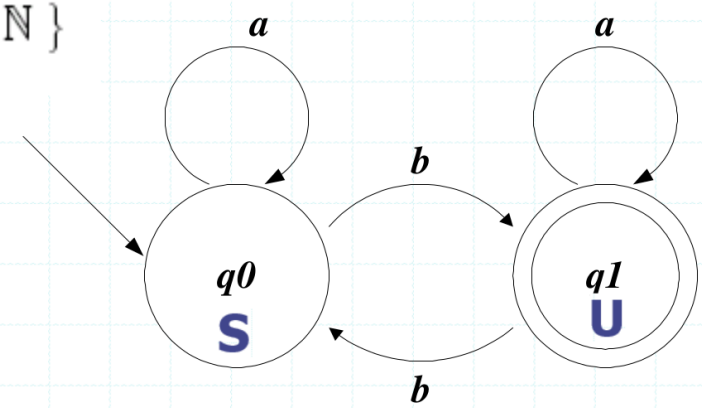
Nech $A = (K, \Sigma, \delta, q_0, F)$, potom $G = (K, \Sigma, P, q_0)$, kde

$$P = \{q \to ap | \delta(q, a) = p\} \cup \{q_f \to \varepsilon | q_f \in F\}$$

Veta

K ľubovoľnému nedeterministickému konečnému automatu A existuje regulárna gramatika G taká, že L(G)=L(A).

$$A(L_1) = \{\ w \in \{a,b\}^* \mid \#_b w = 2k+1,\ k \in \mathbb{N}\ \}$$



G = (N, T, P, S) N = {S, U} T = {a,b}

P: **S** -> a**S** | b**U**

U -> a**U** | b**S** | ε

Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

Veta

K ľubovoľnej regulárnej gramatike G existuje nedeterministický konečný automat A taký, že L(A)=L(G).

Nech G má pravidlá v normálovom tvare[3], potom zostrojme ekvivalentný NKA A nasledovne:
$A = (N \cup \{q_f\}, T, \delta, S, F = \{q_f\})$, pričom:

$\forall A \in N, \forall a \in T \cup \{\varepsilon\}$:

$$\delta(A, a) = \{B | (A \to aB) \in P\} \cup \{q_f | (A \to a) \in P\}$$

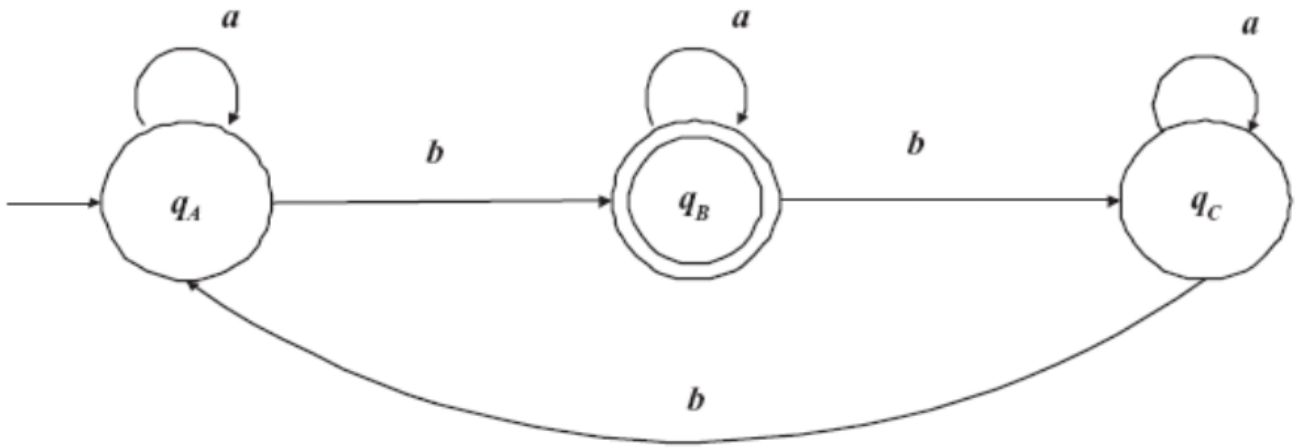
[3] Gramatika vie v jednom kroku vygenerovať veľa znakov, zatiaľ čo automat ich musí čítať po jednom.

Ekvivalencia konečných automatov a regulárnych gramatík

Príklad 3.3.2 *Vzťah konečných automatov a regulárnych gramatík.*
Majme regulárnu gramatiku G_8.
Nech $G_8 = (N, T, P, A)$, *kde* $N = \{A, B, C\}$, $T = \{a, b\}$,
$P = \{$

$$A \to aA \mid bB$$
$$B \to aB \mid bC$$
$$C \to aC \mid bA$$
$$B \to \varepsilon$$

$\}$.
Gramatika G_8 *generuje jazyk* $L(G_8) = \{\ w \in \{a,b\}^* \mid \#_b w = 3k+1,\ k \in \mathbb{N}\ \}$.
Skonštruujme konečný automat A_8, *taký že* $L(A_8) = L(G_8)$.



Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

Uzáverové vlastnosti regulárnych jazykov

Definícia

Trieda jazykov L je uzavretá vzhľadom na operáciu $\square : \mathcal{L} \times \mathcal{L} \to \mathcal{L}$ ak platí:

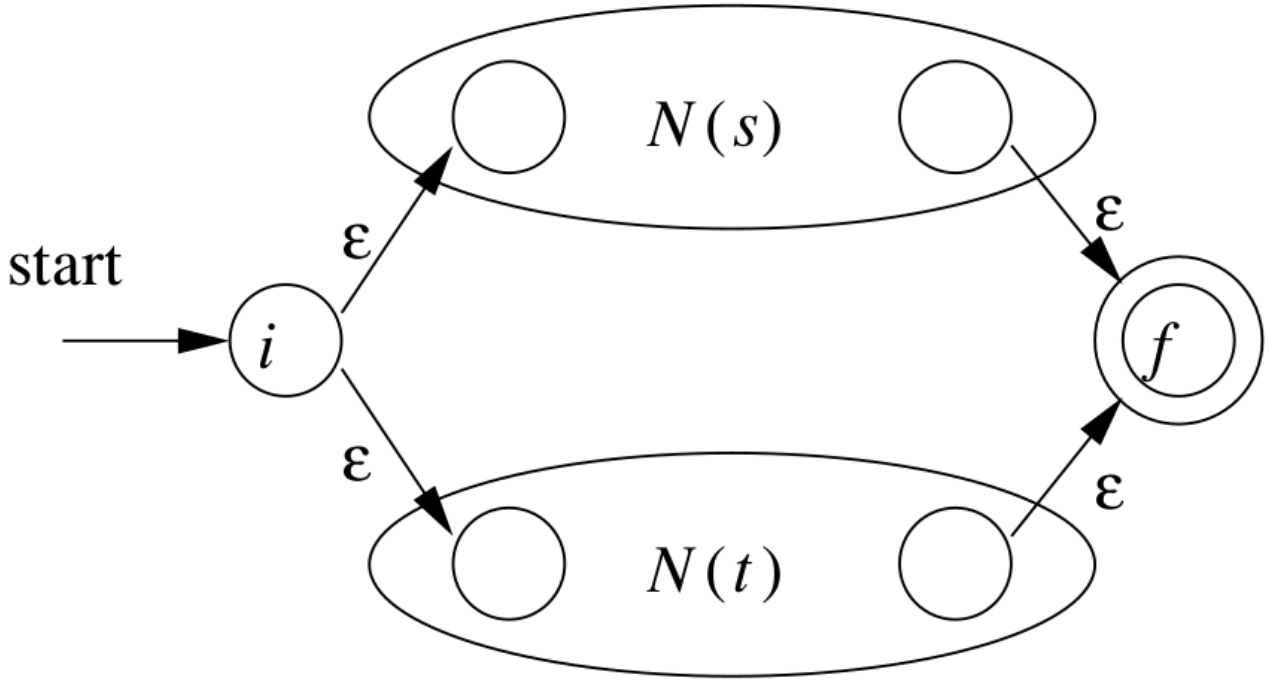
$$\forall L_1, L_2 \in \mathcal{L} : (L_1 \square L_2) \in \mathcal{L}$$

Trieda regulárnych jazykov je uzavretá na nasledujúce operácie:

- zjednotenie
- zreťazenie,
- prienik,
- Kleeneho (kladnú) iteráciu,
- doplnok,
- zrkadlový obraz
- (nevymazávací) homomorfizmus

Veta

$\mathcal{R}$ je uzavretá na zjednotenie.



Zdroj: Alfred V. Aho , Monica S. Lam , Ravi Sethi , Jeffrey D. Ullman. Compilers: Principles, Techniques, and Tools. Addison Wesley. 2nd Edition.2006.

Veta

$\mathcal{R}$ je uzavretá na zjednotenie.

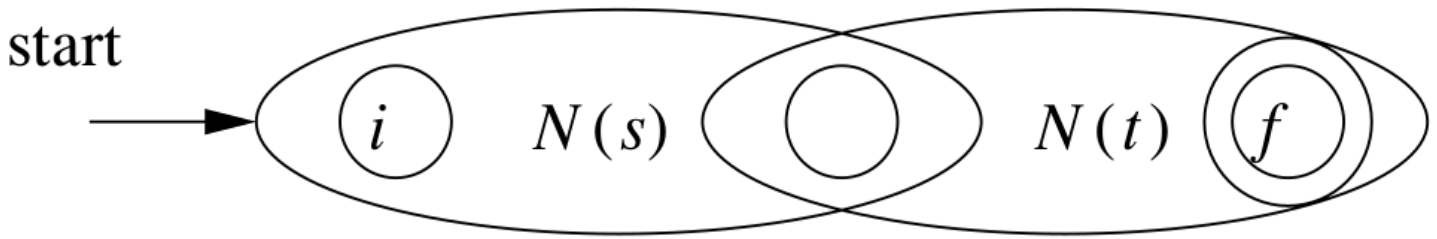
- z gramatík pre pôvodné jazyky zostrojíme regulárnu gramatiku pre ich zjednotenie.
- v prvom kroku odvodenia sa rozhodne, či vygeneruje slovo z L_1 alebo z L_2.
- Majme $L_1, L_2 \in \mathcal{R}$. K nim existujú regulárne gramatiky G_1, G_2, ktoré ich generujú. Nech $G_1 = (N_1, T_1, P_1, S_1)$, $G_2 = (N_2, T_2, P_2, S_2)$. Môžeme predpokladať, že množiny $N_1, N_2, T_1 \cup T_2$ sú po dvoch disjunktné[4].
- $G = (N_1 \cup N_2 \cup \{S\}, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}, S)$

[4] neterminály vieme vhodne preznačiť

Veta

$\mathcal{R}$ je uzavretá na zreťazenie.

- Majme $L_1, L_2 \in \mathcal{R}$. K nim existujú NKA v normálnom tvare $A_i = (K_i, \Sigma, \delta_i, q_{0i}, \{q_{fi}\})$. Nech majú disjunktné množiny stavov ($K_1 \cap K_2 = \emptyset$). Zostrojme potom automat $A = (K_1 \cup K_2, \Sigma, \delta, q_{01}, \{q_{f2}\})$, kde δ obsahuje obe δ_i a navyše definujeme $\delta(q_{f1}, \varepsilon) = \{q_{02}\}$.



Zdroj: Alfred V. Aho , Monica S. Lam , Ravi Sethi , Jeffrey D. Ullman. Compilers: Principles, Techniques, and Tools. Addison Wesley. 2nd Edition.2006.

Uzáverové vlastnosti regulárnych jazykov

Veta

$\mathcal{R}$ je uzavretá na Kleeneho iteráciu.

- Podobne ako pre zreťazenie – zoberieme automat pre L v normálnom tvare a pridáme mu ε-hranu z akceptačného stavu do počiatočného stavu.

Poznámka:

- Keďže $L^+ = L \cdot L^*$, potom $\mathcal{R}$ je uzavretá aj na kladnú iteráciu.

Uzáverové vlastnosti regulárnych jazykov

Veta

$\mathcal{R}$ je uzavretá na komplement.

- Ak $A = (K, \Sigma, \delta, q_0, F)$ je DKA akceptujúci jazyk L, tak $A' = (K, \Sigma, \delta, q_0, K \setminus F)$ je DKA akceptujúci jazyk $L^C = \Sigma^* \setminus L$.
- ak v A výpočet na w nebol akceptačný, v A' bude a naopak.

Uzáverové vlastnosti regulárnych jazykov

Veta

$\mathcal{R}$ je uzavretá na prienik.

- pomocou konštrukcie kartézskym súčinom.
- konečný automat bude naraz simulovať viacero iných konečných automatov – v svojom stave pamätá stavy všetkých simulovaných automatov (počet stavov automatu je len konečne veľa).
- Nech $L_1, L_2 \in \mathcal{R}$, nech $A_i = (K_i, \Sigma, \delta_i, q_{0i}, F_i)$ sú DKA také, že $L(A_i) = L_i$. Potom zostrojme automat $A = (K, \Sigma, \delta, q_0, F)$, kde $K = K_1 \times K_2, q_0 = [q_{01}, q_{02}], F = F_1 \times F_2$ a
$$\delta([q_x, q_y], a) = [\delta_1(q_x, a), \delta_2(q_y, a)]$$
.
- A je po prečítaní w v stave $[q_x, q_y] \iff A_1$ je po prečítaní w v stave q_x a A_2 v q_y
- vlastnosť vyplýva aj z de Morganovych zákonov – $L_1 \cap L_2 = (L_1^C \cup L_2^C)^C$.

Uzáverové vlastnosti regulárnych jazykov

Veta

$\mathcal{R}$ je uzavretá na homomorfizmus.

- Pravidlá regulárnej gramatiky pre daný jazyk upravíme tak, že každý terminál nahradíme jeho homomorfným obrazom.
- Výsledná gramatika bude generovať homomorfný obraz pôvodného jazyka
- Majme $L \in \mathcal{R}, h : T^* \to U^*$ nech je ľubovoľný homomorfizmus. Vieme, že existuje regulárna gramatika G=(N,T,P,S) taká, že L(G)=L.
- Definujme nový homomorfizmus h' nasledovne: $\forall A \in N : h'(A) = A$ a $\forall a \in T : h'(a) = h(a)$. (h' sa správa rovnako ako h, len navyše vie zobraziť aj neterminály gramatiky G a necháva ich na pokoji.)
- G'= (N, U, h'(P),S). Množina h'(P) sú všetky pravidlá z P zobrazené homomorfizmom h' –

$$h'(P) = \{A \to h'(w) | A \to w \in P\}.$$

Uzáverové vlastnosti regulárnych jazykov

Veta

$\mathcal{R}$ je uzavretá na zrkadlový obraz.

- zoberieme NKA v normálnom tvare, vymeníme akceptačný stav so začiatočným stavom a obrátime smer šípok.
- Nech $L \in \mathcal{R}, L = L(A)$, kde $A = (K, \Sigma, \delta, q_0, \{q_f\})$ je v normálnom tvare. Potom zostrojme $A' = (K, \Sigma, \delta', q_f, \{q_0\})$, kde $p \in \delta'(q, x) \iff q \in \delta(p, x)$.

Uzáverové vlastnosti regulárnych jazykov

Úplný algebraický systém:

- pomocou operácií: $\cup$, ., * zadefinujeme akýkoľvek regulárny jazyk.

Definícia

Viachlavý NKA *je 6-ica* $A = (K, \Sigma, \delta, q_0, F, k)$, *kde* K, Σ, q_0, F *sú ako pri NKA,* $k > 0$ *je počet hláv a* $\delta : K \times (\Sigma \cup \{\varepsilon\})^k \to 2^k$ *je prechodová funkcia.*

Definícia

Konfiguráciou viachlavého NKA *nazveme (k+1)-ticu* $(q, w_1, ..., w_k)$, *kde* $q \in K$ *je aktuálny stav a* $w_i \in \Sigma^*$ *je časť slova, ktorú ešte neprečítala i-ta hlava.*

Definícia

Krokom výpočtu *viachlavého NKA A nazveme binárnu reláciu* $\vdash_A$, *definovanú na množine konfigurácií nasledovne:*

$$(q, a_1 w_1, ..., a_k w_k) \vdash_A (p, w_1, ..., w_k) \iff p \in \delta(q, a_1, ..., a_k)$$

kde $p, q \in K, a_i \in (\Sigma \cup \{\varepsilon\}), w_i \in \Sigma^*$

Definícia

Jazyk *akceptovaný viachlavým NKA A je množina*

$$L(A) = \{w | \exists q_f \in F : (q_0, \underbrace{w, ..., w}_{k}) \vdash_A^* (q_f, \underbrace{\varepsilon, ..., \varepsilon}_{k})\}$$

Výpočtová sila viachalvých automatov:

- nie je menšia, ako sila obyčajných konečných automatov (s jednou hlavou).
- už dvojhlavý automat dokáže akceptovať napr. jazyk $L1 = \{a^n b^n | n > 0\}$, ktorý nie je regulárny – ostrá nadmnožina regulárnych jazykov
- vedia akceptovať aj jazyky, ktoré nie sú bezkontextové $L_2 = \{w\sharp w | w \in \{a, b\}^*\}$ – neporovnateľné s triedou bezkontextových jazykov
- ostrou podmnožinou (rozšírených) kontextových jazykov
- pre každé k je trieda jazykov, pre ktoré existuje k-hlavý konečný automat, je vlastnou podmnožinou triedy jazykov, pre ktoré existuje (k+ 1)-hlavý konečný automat.

Definícia

Dvojsmerným (nedeterministickým konečným) automatom *(2NKA) nazývame 5-icu* $A = (K, \Sigma, \delta, q_0, F)$, *kde* K, Σ, q_0, F *sú rovnaké ako pri DKA,* $¢, \$ \notin \Sigma, \delta : K \times (\Sigma \cup \{¢, \$\}) \to 2^{K \times \{-1,0,1\}}$ *je prechodová funkcia, pričom platí:*[a]

$$\forall q \in K : \delta(q, ¢) \subseteq K \times \{0, 1\}$$
$$\forall q \in K : \delta(q, \$) \subseteq K \times \{-1, 0\}$$

[a] Ak je automat na niektorom konci slova, nesmie ho prekročiť

Definícia

Konfiguráciou *2NKA A nazývame trojicu* $(q, ¢w\$, i)$, *kde* $q \in K$ *je aktuálny stav,* $w \in \Sigma^*$ *je vstupné slovo a* $i \in \{0, ..., |w| + 1\}$ *je pozícia hlavy.*

Poznámka: 2NKA nečíta slovo deštruktívne ako KA.

Definícia

*Nech w_i je i-ty znak slova w, pričom definujeme $w_0 = ¢$ a $w_{|w|+1} = \$$. Potom **krokom výpočtu** 2NKA A nazývame binárnu reláciu $\vdash_A$ na množine konfigurácií definovanú nasledovne:*

$$(q, ¢w\$, i) \vdash_A (p, ¢w\$, i + j) \iff (p, j) \in \delta(q, w_i)$$

Definícia

***Jazyk** akceptovaný 2NKA A je množina*

$$L(A) = \{w | \exists q_f \in F : (q_0, ¢w\$, 1) \vdash_A^* (q_f, ¢w\$, |w| + 1)\}$$

Poznámka: Analogicky vieme definovať aj deterministické dvojsmerné automaty (2DKA).

Akceptovanie slova:

- automat prekročí pravý koniec vstupného slova w a skončí v akceptačnom stave.

Zamietnutie slova:

- automat prekročí ľavý koniec vstupného slova w
- automat prekročí pravý koniec vstupného slova w a neskončí v akceptačnom stave.
- zacyklí sa
- zasekne sa

Veta

Ku každému DKA A existuje ekvivalentný 2NKA A'.

Vyzerá identicky, len δ-funkcia navyše vracia vždy 1, teda pohyb doprava.

Veta

Ku každému 2NKA A existuje ekvivalentný NKA A'.

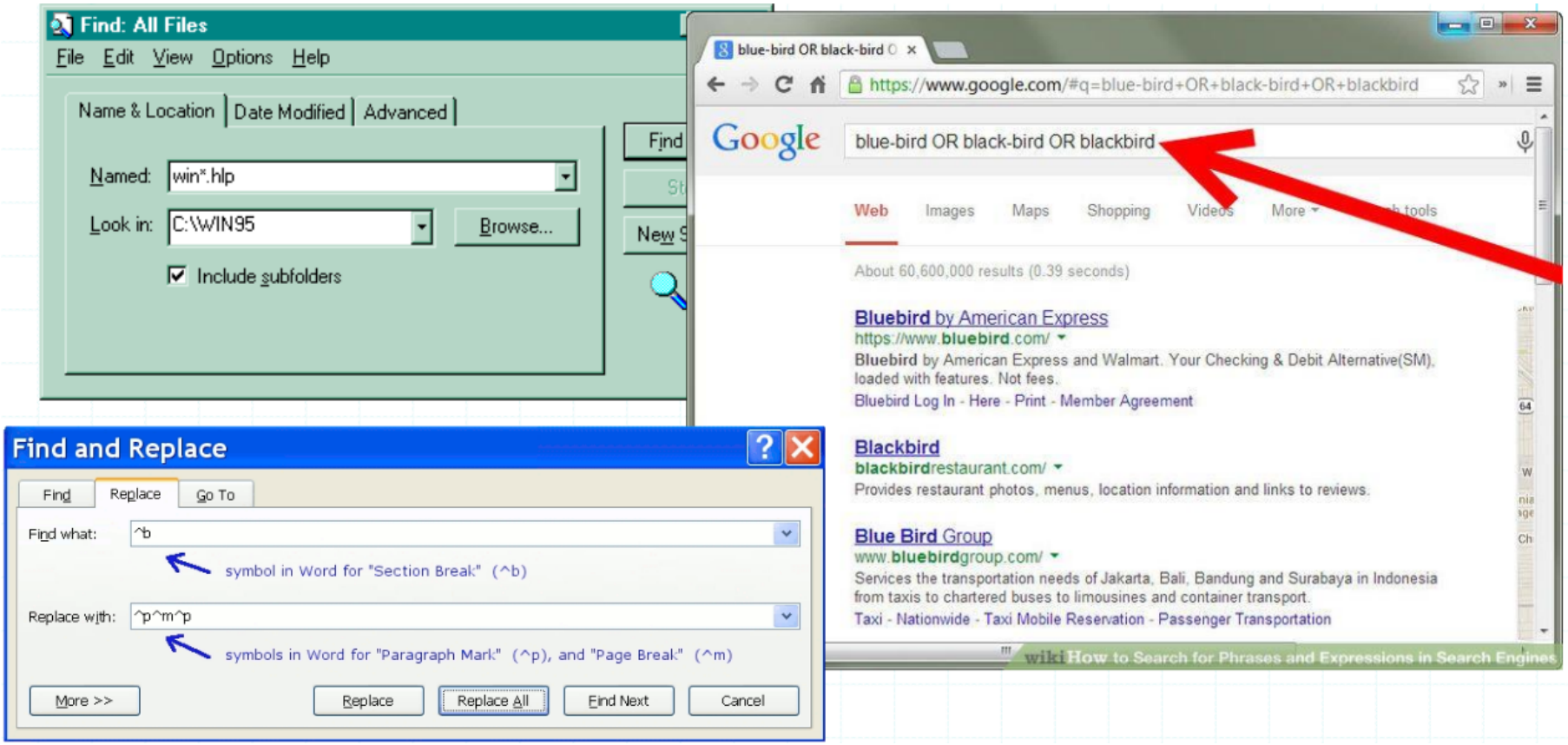
Dôsledok: Dvojsmerné konečné automaty nie sú výpočtovo silnejšie ako jednosmerné konečné automaty.
Ich prínos spočíva v zjednodušení dokazovania, že niektoré jazyky sú regulárne.

Regulárne výrazy

Regulárne výrazy (ang. Regular expression $\rightarrow$ Regex)

- postupnosť znakov definujúcich vzor
- vzor, ktorý používame: find / replace

Použitie regulárnych výrazov



Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

Definícia

Regulárny výraz *(syntakticky) môžeme rekurzívne definovať nasledovne:*

- $\overline{\emptyset}$ *je regulárny výraz, predstavujúci jazyk* $\emptyset$.
- *Nech* $x \in \Sigma \cup \{\varepsilon\}$, *potom* $\overline{x}$ *je regulárny výraz, predstavujúci jazyk* $\{x\}$.
- *Nech* $\overline{R}_1, \overline{R}_2$ *sú regulárne výrazy, predstavujúce jazyky* L_1, L_2, *potom* $\overline{(R_1|R_2)}$ *je regulárny výraz, predstavujúci jazyk* $L_1 \cup L_2$.
- *Nech* $\overline{R}_1, \overline{R}_2$ *sú regulárne výrazy, predstavujúce jazyky* L_1, L_2, *potom* $\overline{R_1 R_2}$ *je regulárny výraz, predstavujúci jazyk* $L_1 \cdot L_2$.
- *Nech* $\overline{R}$ *je regulárny výraz, predstavujúci jazyk* L, *potom* $\overline{(R)^*}$ *je regulárny výraz, predstavujúci jazyk* L^*.
- *Nič iné nie je regulárny výraz a žiaden regulárny výraz nepredstavuje žiaden iný jazyk.*

Regulárne výrazy

- Regulárny výraz slúži na opis regulárneho jazyka Notácia pozostáva z:
  - Reťazce a symboly z abecedy Σ
  - Zátvorky
  - Operátory $|$, $\cdot$, $*$

Príklad: $a^* \cdot (a \mid b)$

Opis jazyka regulárnych výrazov

- Jazyk L(r) opísaný reg. výrazom r je definovaný pravidlami (sémantika, význam):
  1. $\emptyset$ je reg. výraz opisujúci prázdnu množinu $L(\emptyset) = \emptyset$
  2. ε je reg. výraz opisujúci množinu slov $L(\varepsilon) = \{\varepsilon\}$
  3. Pre každé $a \in \Sigma$, a je reg. výraz opisujúci $\mathrm{L}(a) = \{a\}$
- Ak r_1 a r_2 sú regulárne výrazy, potom
  1. $\mathrm{L}(r_1 \mid r_2) = \mathrm{L}(r_1) \cup \mathrm{L}(r_2)$
  2. $\mathrm{L}(r_1 \cdot r_2) = \mathrm{L}(r_1)\ \mathrm{L}(r_2)$
  3. $\mathrm{L}((r_1)) = \mathrm{L}(r_1)$
  4. $\mathrm{L}(r_1^*) = (\ \mathrm{L}(r_1)\ )^*$

Regulárne výrazy

Veta

Každý regulárny výraz predstavuje regulárny jazyk.

Jazyky $\emptyset$ a $\{x\}$ (pre$x \in \Sigma$) sú regulárne. Regulárne jazyky sú uzavreté na zjednotenie, zreťazenie a iteráciu.

Veta

Ku každému regulárnemu jazyku existuje regulárny výraz, ktorý ho predstavuje.

Opis jazyka regulárnych výrazov

$r = a^* \cdot (a \mid b)$
$L(r) = ?$

$L(r) = L(a^*)L(a|b)$
$L(r) = (L(a))^*(L(a) \cup L(b))$
$L(r) = \{\varepsilon, a, aa...\}\{a, b\}$
$L(r) = \{a, aa, aaa, ..., b, ab, aab...\}$

Opis jazyka regulárnych výrazov

$r = (aa)^*(bb)^*b$
$L(r) = ?$

$L(r) = a^{2n}b^{2m+1}, n \geq 0, m \geq 0$

- **NIE:** $a^{2n}b^{2n+1}$ **!**
- Nedá sa, pretože regulárny výraz nedokáže opísať bezkontextový jazyk

Opis jazyka regulárnych výrazov

L = $\{w \in \{0,1\}^* |$ w obsahuje aspoň jednu dvojicu po sebe idúcich núl$\}$ r=?

r = $(0|1)^*00(0|1)^*$

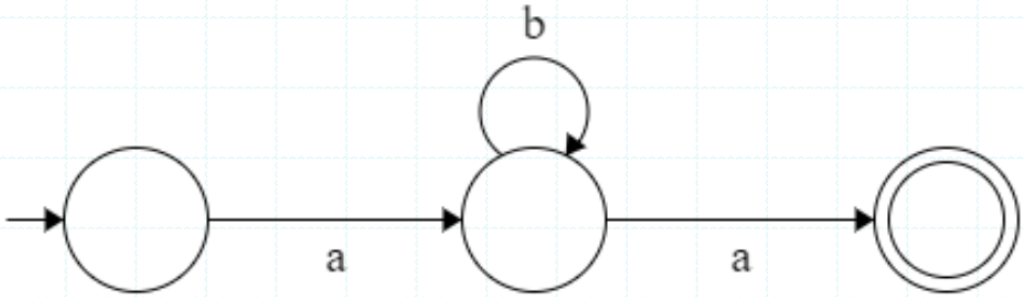
Spojitosť s regulárnymi jazykmi

- Pre každý reg. výraz r je možné zostrojiť konečný automat, ktorý akceptuje jazyk L(r) $\Rightarrow$ r opisuje regulárny jazyk.
- Pri konštrukcií automatu vieme každú operáciu v regulárnom výraze reprezentovať v stavovom diagrame konečného automatu.

Konštrukcia konečného automatu

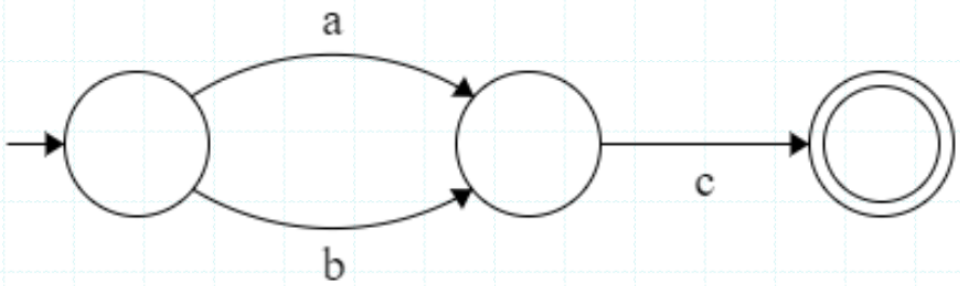
r = ab*a

- L(r) = {aa, aba, abba, abbba,...}



r = (a|b)c

- L(r) = {ac, bc}

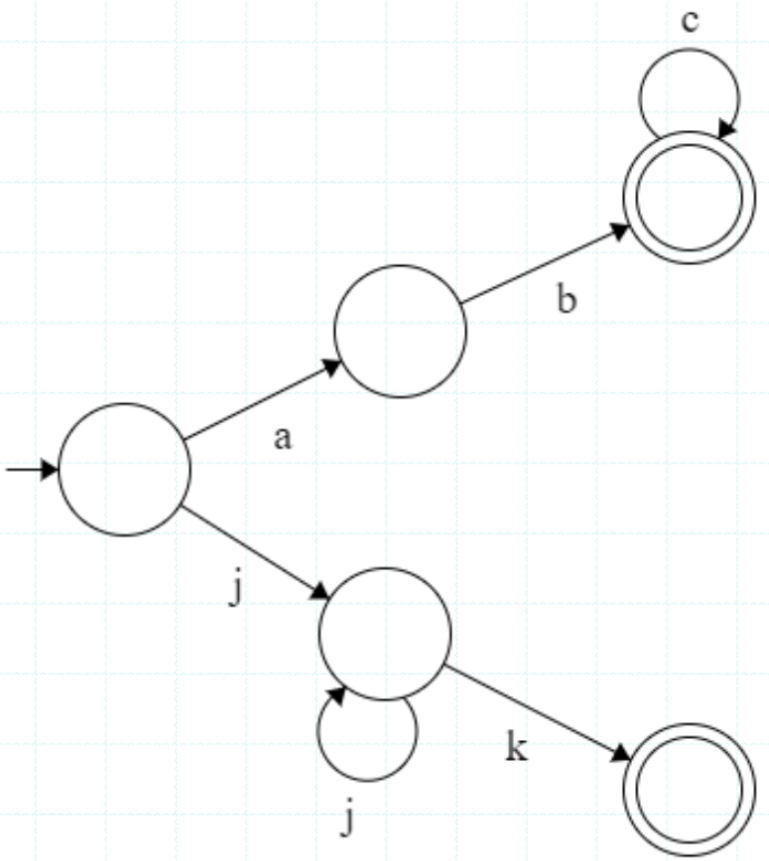


Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

Konštrukcia konečného automatu

$r = abc^*|j^+k$

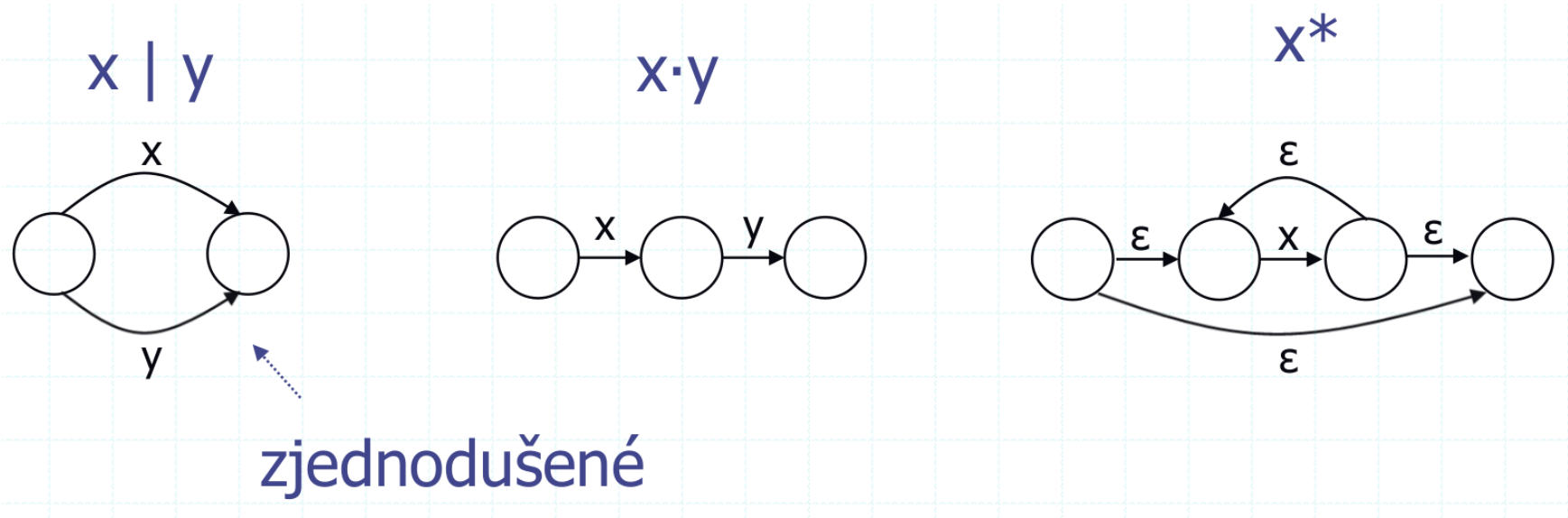
- L(r) = {ab, abc, abcc,..., jk, jjk,...}



Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

Konštrukcia konečného automatu – Algoritmicky

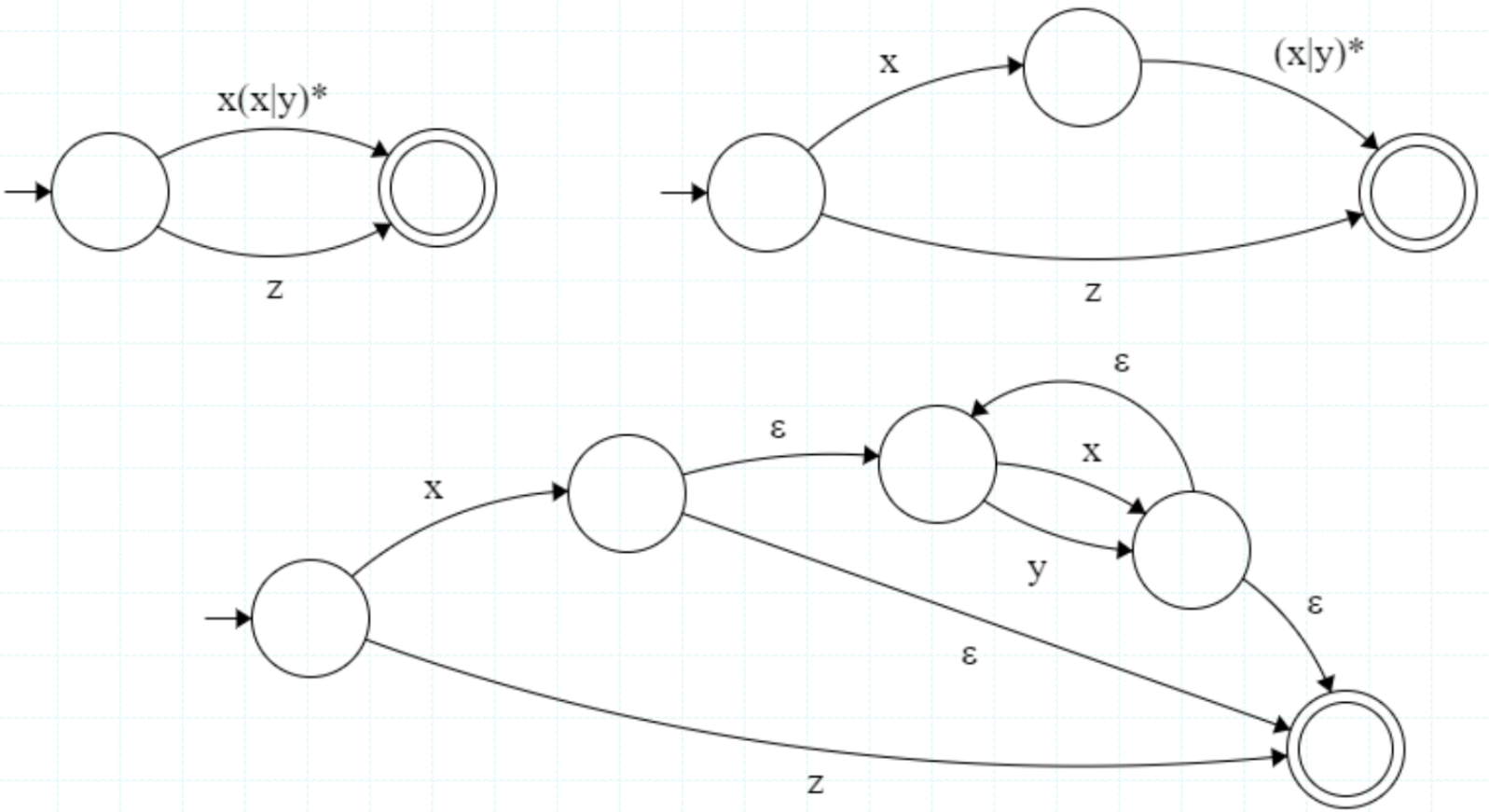
- najprv zostrojíme NKA
- ten sa prevedie na DKA



Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

Konštrukcia konečného automatu

- r = x(x|y)*|z
  - L(r) = {x, z, xx, xyx, xyy, ...}



Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

Regulárne výrazy v praxi

$\wedge$ -začiatok reťazca
$ -koniec reťazca
. -jeden ľubovoľný znak
* -opakovanie ľubovoľne krát
+ -opakovanie aspoň raz
? -opakovanie najviac raz
$\{min, max\}$ -opakovanie od min do max
$\{n\}$ -opakovanie práve n-krát
(podvýraz) -vytvorenie podvýrazu
[*abc*] -výpis znakov (znaky a,b,c)
$[a - zA - Z]$ -výpis znakov (všetky písmena)
$[\wedge 0 - 9]$ -negácia výpisu (všetko okrem číslic)
[[:<:]] -začiatok slova
[[:>:]] -koniec slova
[[: *trieda*]] - iba znaky danej triedy

Regulárne výrazy v praxi

alnum - trieda: písmená anglickej abecedy + číslice
alpha - trieda: písmená anglickej abecedy
blank - trieda: medzera + tabulátor
cntrl - trieda: riadiace znaky
digit - trieda: číslice
graph - trieda: znaky s grafickým znázornením
lower - trieda: malé písmená anglickej abecedy
print - trieda: tlačiteľné znaky + medzera
punct - trieda: interpunkčné a pomocné znaky (@...)
space - trieda: medzera (+ tabulátor, nový riadok,...)
upper - trieda: veľké písmená anglickej abecedy
xdigit - trieda: číslice + písmena a - f, A - F

Regulárne výrazy v praxi

- $[hc] + at$
- hat, cat, hhat, chat, hcat, cchchat,
- ~~at~~

- $[hc]?at$
- hat, cat, at
- ~~chat~~

- $[hc] * at$
- hat, cat, hhat, chat, hcat, cchchat, at

Regulárne výrazy v praxi

- `https://regex101.com/`
- `https://rubular.com/`

Regulárne výrazy v praxi

- cat|dog
- "cat"
- "dog"

- Pe(t|p)a
- "Peta"
- "Pepa"

- Ba*f
- "Bf", "Baf", "Baaf", "Baaaf"...

Regulárne výrazy v praxi

- Implementácie v rôznych programovacích jazykoch s rôznou syntaxou
- Existuje štandard podľa IEEE POSIX – používa sa napr. v unixových programoch grep, egrep
- Knižnica PCRE (Perl Compatible Regular Expressions) – založená na reg. výrazoch v jazyku Perl
- "Regex-y"majú podporu aj pre rozoznávanie vyšších jazykov ako regulárnych ($\Rightarrow$ nejde o reálne regulárne výrazy)

Egrep

- Program, ktorý hľadá v súbore vzor (vypíše riadky)
- Použitie: egrep $'[hc]+at'$ file.txt

Špeciálny znak	**Význam**
.	ľubovoľný znak
[ ]	Jeden zo znakov obsiahnutý v zátvorkách Napr. [abc] alebo [a-c] zodpovedá *a*, *b* alebo *c*
[^]	Znak iný ako uvedený v zátvorkách Napr. [^abc] znamená iný znak ako *a*, *b*, *c*
^	Začiatok riadku
$	Koniec riadku
*	Opakovanie bloku 0 a viac krát Napr. (ab)* značí "", "ab", "abab",...
+	Opakovanie bloku 1 a viac krát
?	Opakovanie bloku 0 alebo 1 krát
|	Výber

Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

Ďakujem vám za pozornosť.

Dotazník k prednáške:
https://forms.gle/
gPpbqYUYSnGfbG9X9