Faculty of Informatics
Masaryk University Brno

Cvičení k předmětu IA006
Vybrané kapitoly z teorie automatů

Jiří Barnat
Ivana Černá

poslední modifikace 22. ledna 2009

Funkce FIRST a FOLLOW

Opakování a motivace

1.1 Je dána následující gramatika G. Navrhněte PDA (zásobníkový automat), který analyzuje slova nad abecedou a,b,c metodou shora dolů.

$G = (\{S, A\},\{a, b, c\},P, S)$, kde
$P = \{\ S \ \rightarrow\ aSa \ \mid\ bSc \ \mid\ aA,$
$\quad A \ \rightarrow\ bbAb \ \mid\ \varepsilon\ \}$

1.2 Navrhněte $LL(1)$ jednoduchou gramatiku pro jazyk zapsaný následující množinou

a) $\{1^n\,2\,0^n\,1^m\,2\,0^m \mid n > 0, m \geq 0\}$

b) $\{1^n\,2\,0^n\,1^m\,2\,0^m \mid n \geq 0, m \geq 0\}$

Poznámka: V jednoduché $LL(1)$ gramatice začínají všechny pravé strany pravidel terminálem, pravidla se stejnou levou stranou začínají různým terminálem.

1.3 Rozmyslete si jak probíhá analýza jednoduchých $LL(1)$ gramatik.

1.4 Najděte jazyk, který se nedá generovat žádnou jednoduchou $LL(1)$ gramatikou.

FIRST a FOLLOW

Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika. Funkce $FIRST^G$ a $FOLLOW^G$ jsou definovány následovně:

$FIRST_k^G : (\Sigma \cup N)^* \longrightarrow 2^{\Sigma^*}$
$FIRST_k^G(\alpha) = \{w \in \Sigma^* \mid (\alpha \Rightarrow^* w \ \wedge\ |w| \leq k) \ \vee\ (\alpha \Rightarrow^* wu \ \wedge\ |w| = k\,;\ u \in \Sigma^*)\}$

$FOLLOW_k^G : N \longrightarrow 2^{\Sigma^*}$
$FOLLOW_k^G(A) = \{w \in \Sigma^* \mid S \Rightarrow^* uA\alpha,\ w \in FIRST_k(\alpha)\ ;\ u \in \Sigma^*,\ \alpha \in (\Sigma \cup N)^*\}$

Poznámka: Pozor na typ argumentu u jednotlivých funkcí. Funkce $FIRST_k^G(\alpha)$ bere jako argument řetězec terminálů a neterminálů ($\alpha \in (\Sigma \cup N)^*$), narozdíl od funkce $FOLLOW_k^G(A)$, jejíž argumentem je vždy právě jeden neterminál ($A \in N$). Běžně se používají zkrácené zápisy funkcí, $FI_k(\alpha)$ pro $FIRST_k^G(\alpha)$ a $FO_k(A)$ pro $FOLLOW_k^G(A)$ (G je zřejmé z kontextu a typicky se neuvádí).

Poznámka: Definice funkcí $FIRST_k^G$ a $FOLLOW_k^G$ lze přirozeně rozšířit také na množiny odpovídajících argumentů, což je užitečné zejména pro funkci $FIRST$.

$FIRST_k^G : 2^{(\Sigma\cup N)^*} \longrightarrow 2^{\Sigma^*}$
$FIRST_k^G(M) = \bigcup_{\alpha\in M} FIRST_k^G(\alpha)$

$FOLLOW_k^G : 2^N \longrightarrow 2^{\Sigma^*}$
$FOLLOW_k^G(M) = \bigcup_{A\in M} FOLLOW_k^G(A)$

Operátor $\oplus_k$

$\oplus_k : 2^{\Sigma^*} \times 2^{\Sigma^*} \to 2^{\Sigma^*}$
$A \oplus_k B = \{w \in \Sigma^* \mid (w \in A \cdot B \wedge |w| \leq k) \vee (wu \in A \cdot B \wedge |w| = k \wedge u \in \Sigma^*)\}$

Poznámka: Neformálně řečeno, operátor $\oplus_k$ provádí zřetězení množin terminálních řětězců a ořezání výsledných řetězců na řetězce délky maximálně k. Tuto operaci je také možné realizovat v různých algoritmech jako $FIRST_k^G(A \cdot B)$. Všimněme si ale, že striktně formálně lze toto použít pouze je-li z kontextu dána gramatika G.

Algoritmus pro výpočet funkce FIRST

Je dána gramatika $G = (N, \Sigma, P, S)$ a řetězec $\alpha = Y_1 \cdot Y_2 \cdot \ldots \cdot Y_l$, kde $Y_x \in N \cup \Sigma$.

1) $FI_k(x) = \{x\}$ pro $x \in \Sigma \cup \{\varepsilon\}$, $1 \leq x \leq l$

2) Výpočet $FI_k(x)$ pro $x \in N$:

   Nechť $N = \{X_1, X_2, \ldots, X_n\}$. Budeme počítat hodnotu $FI_k(X_i)$ současně pro všechny neterminály $(i = 1, \ldots, n)$. Nechť všechna pravidla pro neterminál X_i jsou tato:

   $$X_i \to Y_1^1 \ldots Y_{k_1}^1 \mid Y_1^2 \ldots Y_{k_2}^2 \mid \ldots \mid Y_1^j \ldots Y_{k_j}^j$$

   Potom

   $$\begin{aligned} FI_k(X_i) = {} & [\, FI_k(Y_1^1) \oplus_k FI_k(Y_2^1) \oplus_k \ldots \oplus_k FI_k(Y_{k_1}^1) \,] \\ & \cup \ldots \cup \\ & [\, FI_k(Y_1^1) \oplus_k FI_k(Y_2^1) \oplus_k \ldots \oplus_k FI_k(Y_{k_1}^1) \,]. \end{aligned}$$

   Hodnoty $FI_k(X_i)$ jsou pevnými body uvedené soustavy rekurzivních rovnic. Počáteční hodnoty jsou $FI_k(X_i) = \emptyset$.

3) $FIRST_k(\alpha) = FI_k(Y_1) \oplus_k FI_k(Y_2) \oplus_k \cdots \oplus_k FI_k(Y_l)$

Algoritmus pro výpočet funkce FOLLOW

Je dána redukovná gramatika $G = (N, \Sigma, P, S)$. Funkce FO je definována pouze pro neterminály.

Postupně počítáme hodnoty: $FO_1(A)$ pro všechny $A \in N$,
$FO_2(A)$ pro všechny $A \in N$
$\vdots$
$FO_k(A)$ pro všechny $A \in N$

Při výpočtu $FO_i(A)$ postupujeme následovně:

1) $FO_0(A) := \{\varepsilon\}$ pro $A \in N$.
   $FO_i(S) := \{\varepsilon\}$
   $FO_i(A) := \emptyset$ pro $A \in N \setminus \{S\}$.

2) Pro každé pravidlo tvaru: $B \to \alpha A \beta \in P$, kde $\beta \neq \varepsilon$

   $$FO_i(A) := FO_i(A) \cup [(FI_i(\beta) - \{\varepsilon\}) \oplus_i FO_{i-1}(B)]$$

3) OPAKUJ
   Pro každé pravidlo tvaru: $B \to \alpha A \beta \in P$, kde $\beta = \varepsilon$ nebo $\varepsilon \in FI_1(\beta)$

   $$FO_i(A) := FO_i(A) \cup FO_i(B)$$

   Tak dlouho, dokud se nedosáhne pevného bodu.

Poznámka: Hodnotu výrazu $FIRST_k^G(\alpha)$ lze **intuitivně** vypočítat alternativním způsobem. Přesně řečeno lze prozkoumat všechny možné derivace vycházející z řetězce α a vzít v potaz terminální prefixy délky k z obdržených větných forem (větné formy musí být normované, tj. vyderivovatelné do terminálního řetězce).

Poznámka: Intuitivně (tj. ne přesně podle algoritmu) lze postupovat i při výpočtu hodnoty funkce $FOLLOW_k^G(A)$. Sestaví se rovnice tak, jak uvedeno v algoritmu a tyto se řeší "líným vyhodnocováním", tj. jednotlivé členy rovnic se počítají až v okamžiku, kdy jsou potřebuja. Avšak i při tomto postupu může dojít k definici rekurzivní rovnice, například:

$$FO_2(A) = \{bb\} \cup \{a, bc\} \cdot FO_1(B) \cup FO_2(A).$$

Tuto situaci lze na **intuitivní** úrovni řešit vypuštěním členu $FO_2(A)$, neboť $FO_2(A)$ nepřináší nic nového do $FO_2(A)$, tj namísto předchozí rekurzivní rovnice se použije:

$$FO_2(A) = \{bb\} \cup \{a, bc\} \cdot FO_1(B).$$

V obecné rovině takto rekurzivní rovnice samozřejmě řešit nelze, ale v tomto kontextu je to možné.

1.5 Podle algoritmu řešte $FI_2(A)$ a $FI_3(Ae)$ pro gramatiku

$G = (\{A, B, C\},\{c, a, d\},P, A)$, kde
$P = \{\ A \rightarrow Bc \mid a,$
$\quad B \rightarrow A \mid C \mid d,$
$\quad C \rightarrow B \mid d\ \}$

1.6 Podle algoritmu řešte $FI_2(A)$ a $FI_3(A)$ pro gramatiku

$G = (\{A\},\{a, b\},P, A)$, kde
$P = \{\ A \rightarrow Aa,$
$\quad A \rightarrow b\ \}$

1.7 Vypočítejte $FI_1(BBb)$, $FI_2(BBb)$, $FO_1(A)$, $FO_1(S)$, $FO_1(B)$, $FO_1(C)$, $FO_3(A)$, $FO_3(S)$, $FO_3(C)$, $FI_1(SAcB)$, $FI_4(SAcB)$ pro následující gramatiku:

$G = (\{S, A, B, C\},\{a, c, b, e, d\},P, S)$, kde
$P = \{\ S \rightarrow aAc \mid B,$
$\quad A \rightarrow aA \mid bSCe \mid \varepsilon,$
$\quad B \rightarrow aC \mid \varepsilon,$
$\quad C \rightarrow d \mid \varepsilon\ \}$

1.8 Podle algoritmu řešte $FO_k(X)$, kde $k = 1, 2, 3, 4$ a $X \in \{S, A, B, C, D\}$ pro následující gramatiku:

$G = (\{S, A, B, C, D\},\{a, b, d, c, x, y, z\},P, S)$, kde
$P = \{\ S \rightarrow aABbCD \mid \varepsilon,$
$\quad A \rightarrow ASd \mid \varepsilon,$
$\quad B \rightarrow SAc \mid xC \mid \varepsilon,$
$\quad C \rightarrow Sy \mid Cz \mid \varepsilon,$
$\quad D \rightarrow aBD \mid \varepsilon\ \}$

1.9 Vypočítejte $FO_k(X)$, kde $k = 1, 2, 3, 4$ a $X \in \{S, A, B, C, D\}$ pro následující gramatiku:

$G = (\{S, B, A, D, C\},\{a, c, b, d\},P, S)$, kde
$P = \{\ S \rightarrow aBcB,$
$\quad A \rightarrow aA \mid Aa,$
$\quad B \rightarrow DAc \mid bA,$
$\quad C \rightarrow cBc \mid aaB,$
$\quad D \rightarrow d \mid dC\ \}$

SLL(k) gramatiky a analyzátory

Gramatika je $SLL(k)$, právě když pro všechny neterminály $A \in N$, a pro každá dvě různá pravidla $A \to \beta, A \to \gamma$ platí: $FI_k(\beta \cdot FO_k(A)) \cap FI_k(\gamma \cdot FO_k(A)) = \emptyset$

2.1 Ověřte, zda následující gramatika je $SLL(2)$:

$G = (\{S, X, Y\}, \{b, a\}, P, S)$, kde
$P = \{\ S \to X,$
$\quad X \to Y \mid bYa,$
$\quad Y \to a \mid \varepsilon\ \}$

2.2 Ověřte, zda gramatika je $SLL(3)$

$G = (\{S, A, B\}, \{a, b\}, P, S)$, kde
$P = \{\ S \to aAaB \mid bAbB,$
$\quad A \to a \mid ab,$
$\quad B \to aB \mid a\ \}$

2.3 Navrhněte $SLL(2)$ analyzátor pro následující gramatiku a analyzujte slovo `acaac` a slovo `abaac`.

$G = (\{S, A, B, D\}, \{a, c, b\}, P, S)$, kde
$P = \{\ S \to aAaA,$
$\quad S \to aBaB,$
$\quad A \to aA,$
$\quad A \to c,$
$\quad B \to bD,$
$\quad D \to bD,$
$\quad D \to \varepsilon\ \}$

2.4 Navrhněte $SLL(3)$ analyzátor pro gramatiku a analyzujte slovo `bababab`.

$G = (\{S, A, B\}, \{b, a\}, P, S)$, kde
$P = \{\ S \to baAa,$
$\quad S \to baBb,$
$\quad A \to aA,$
$\quad A \to aB,$
$\quad B \to bA,$
$\quad B \to \varepsilon\ \}$

LL(k) gramatiky a analyzátory

Gramatika je $LL(k)$, právě když pro dvě libovolná různá pravidla gramatiky $A \to \beta, A \to \gamma$ a pro všechny nejlevější větné formy tvaru $wA\alpha$ platí: $FI_k(\beta \cdot \alpha) \cap FI_k(\gamma \cdot \alpha) = \emptyset$

Gramatika je $LL(1)$ právě když je $SLL(1)$. Je-li gramatika G $SLL(k)$, pak je také $LL(k)$.

3.1 Ověřte, zda je následující gramatika $LL(2)$:

$G = (\{S, X, Y\}, \{b, a\}, P, S)$, kde
$P = \{\ S \to X,$
$\quad X \to Y \mid bYa,$
$\quad Y \to a \mid \varepsilon\ \}$

3.2 Ověřte, zda je následující gramatika $LL(3)$:

$G = (\{S, A, B\}, \{a, b\}, P, S)$, kde
$P = \{\ S \to aAaB \mid bAbB,$
$\quad A \to a \mid ab,$
$\quad B \to aB \mid a\ \}$

3.3 Ukažte, že gramatika není $LL(k)$ pro žádné k:

$G = (\{S, A, B\}, \{a, b, 0, 1\}, P, S)$, kde
$P = \{\ S \to A \mid B,$
$\quad A \to aAb \mid 0,$
$\quad B \to aBbb \mid 1\ \}$

3.4 Ukažte, že gramatika je $LL(k)$:

$G = (\{S, T, A, B\}, \{a, b, c\}, P, S)$, kde
$P = \{\ S \to aT$
$\quad T \to SA \mid A$
$\quad A \to bB \mid c$
$\quad B \to b^{k-1}d \mid \varepsilon$

3.5 Zkonstruujte LL(2) analyzátor pro gramatiku G:

$G = (\{S, A\}, \{a, b\}, P, S)$, kde
$P = \{\ S \to \varepsilon,$
$\quad S \to abA,$
$\quad A \to Saa,$
$\quad A \to b\ \}$

3.6 Zkonstruujte LL(3) analyzátor pro gramatiku G:

$G = (\{S, A, B\}, \{a, b\}, P, S)$, kde
$P = \{\ S \to aAaB,$
$\quad S \to bAbB,$
$\quad A \to a,$
$\quad A \to ba,$
$\quad B \to aB,$
$\quad B \to a\ \}$

3.7 Najděte LL(1) analyzátor pro jazyk generovaný následující gramatikou:

$G = (\{STAT, VAR, IDLIST\}, \{if, id, then, else, fi, while, do, od, (,), :=\}, P, STAT)$

$$
\begin{aligned}
P = \{\ & STAT \rightarrow if\ id\ then\ STAT\ else\ STAT\ fi \\
& STAT \rightarrow while\ id\ do\ STAT\ od \\
& STAT \rightarrow VAR := VAR \\
& STAT \rightarrow id \mid (IDLIST) \\
& VAR \rightarrow id \mid id\ (IDLIST) \\
& IDLIST \rightarrow id \mid id\ (IDLIST)
\end{aligned}
$$

Transformace LL(k) gramatik

Motivace:

- Gramatika je $LL(1)$, právě když je $SLL(1)$.
- Pro gramatiky, které jsou $SLL(1)$, se snadno konstruuje analyzátor.

Platí:

- Pro danou bezkontextovou gramatiku G je nerozhodnutelné, zda je $LL(k)$ pro nějaké $k \geq 0$.
- Je-li dána bezkontextová gramatika G a pevné k takové, že G není $LL(k)$, pak je nerozhodnutelné určit, zda G má ekvivaletní gramatiku, která je $LL(k)$.

Navzdory výše uvedeným faktům existuje několik transformací, které zachovávají jazykovou ekvivalenci a které *někdy* vedou k $LL(1)$ gramatice.

Gramatika není $LL(1)$ principielně ze dvou důvodů. Těmi jsou konflikt „first-first“ a konflikt „first-follow“. Správný postup řešení příkladu „převod na $LL(1)$ gramatiku“ obsahuje tyto tři kroky: nalezení konfliktů v transformované gramatice (pomocí ověření na $LL(1)$), odstranění konfliktů níže uvedenými postupy, ověření výsledné gramatiky na $LL(1)$ (při nalezení konfliktů opakování předchozího kroku).

Odstranění levé rekurze

4.1 Odstraňte levou rekurzi v následující gramatice:

$G = (\{S, A, B\}, \{b, a, c\}, P, S)$, kde
$P = \{\ S \rightarrow SAb \mid SBa \mid Scc \mid AaB \mid bc,$
$\quad A \rightarrow aAa \mid \varepsilon,$
$\quad B \rightarrow BbB \mid b\ \}$

Rohová substituce

Nechť $G = (N, T, P, S)$ je bezkontextová gramatika s pravidlem $A \rightarrow B\alpha$, a nechť všechny $B-$pravidla jsou $B \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_m$.
Nechť $G_1 = (N, T, P, S)$ je gramatika, která vznikla z G vyloučením pravidla $A \rightarrow B\alpha$ a přidáním pravidel $A \rightarrow \alpha_1\alpha \mid \alpha_2\alpha \mid \ldots \mid \alpha_m\alpha$.

Tato transformace nese název rohová substituce. Je zvláštní tím, že její opakovanou alpikací je možné z gramatiky, která je $LL(1)$ a nemá ε-pravidla udělat jednoduchou $LL(1)$ gramatiku.

4.2 Aplikujte rohovou substituci na následující gramatiku:

$G = (\{S, A, B\}, \{a, b, c, d\}, P, S)$, kde
$P = \{\ S \rightarrow AB,$
$\quad A \rightarrow ab \mid Ba,$
$\quad B \rightarrow c \mid dB\ \}$

4.3 Aplikujte rohovou substituci na následující gramatiku:

$G = (\{S, A, B\}, \{a, b\}, P, S)$, kde
$P = \{\ S \rightarrow AB,$
$\quad A \rightarrow aA \mid \varepsilon,$
$\quad B \rightarrow bA \mid \varepsilon\ \}$

Levá faktorizace

Pro $LL(1)$ gramatiku musí platit:

$$A \rightarrow \alpha, A \rightarrow \beta \Rightarrow FIRST_1(\alpha) \cap FIRST_1(\beta) = \emptyset$$

nesplnění této podmínky se označuje jako konflikt $FIRST - FIRST$.

Kolizi může způsobit přítomnost pravidel tvaru $A \rightarrow \alpha\alpha_1 \mid \alpha\alpha_2 \mid \ldots \mid \alpha\alpha_n$, kde $\alpha \neq \varepsilon$. Kolizi odtsraníme jestliže uvedená pravidla nahradíme pravidly tvaru $A \rightarrow \alpha A'$, $A' \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$.

Této transformaci se říká levá faktorizace. Většinou je nutné před touto transformací aplikovat na gramatiku rohovou substituci, aby konflikty uvedeného typu byly v požadované formě.

4.4 Aplikujte levou faktorizaci na gramatiku:

$G = (\{A, B, C\}, \{a, b, c, d\}, P, A)$, kde
$P = \{\ A \rightarrow abB \mid acC \mid acdB \mid bBb \mid bbC,$
$\quad B \rightarrow bbaB \mid bbadC \mid cBC,$
$\quad C \rightarrow aA \mid b\ \}$

4.5 Aplikujte levou faktorizaci na gramatiku:

$G = (\{A, B, C\}, \{a, x, y, z\}, P, A)$, kde
$P = \{\ A \rightarrow aBxx \mid aCyy \mid zy \mid zx,$
$\quad B \rightarrow aBx \mid z,$
$\quad C \rightarrow yCy \mid z\ \}$

4.6 Odstraňte konflikt first-first v následující gramatice.

$G = (\{S, A, B\}, \{c, a, b\}, P, S)$, kde
$P = \{\ S \rightarrow A \mid B,$
$\quad A \rightarrow cA \mid a,$
$\quad B \rightarrow cB \mid b\ \}$

4.7 Odstraňte konflikt first-first v následující gramatice.

$G = (\{A, B, C\}, \{a, b, c, d\}, P, A)$, kde
$P = \{\ A \rightarrow aB \mid CB,$
$\quad C \rightarrow aC \mid bB,$
$\quad B \rightarrow cB \mid d\ \}$

4.8 Odstraňte konflikt first-first v následující gramatice.

$G = (\{A, B, D\}, \{c, d, b, x, y, z\}, P, A)$, kde
$P = \{\ A \rightarrow Bc \mid Dd,$
$\quad B \rightarrow bx \mid y,$
$\quad D \rightarrow Bz\ \}$

4.9 Odstraňte konflikt first-first v následující gramatice.

$G = (\{S, A, B\}, \{b, c, a, d\}, P, S)$, kde
$P = \{\ S \rightarrow A \mid AbcB \mid bc,$
$\quad A \rightarrow a,$
$\quad B \rightarrow A \mid dd\ \}$

4.10 Odstraňte konflikt first-first v následující gramatice.

$G = (\{S, A, B\}, \{a, b, c\}, P, S)$, kde
$P = \{\ S \to A \mid B,$
$\quad A \to aAb \mid \varepsilon,$
$\quad B \to aBc \mid \varepsilon\ \}$

Pohlcení terminálního symbolu

Pro $LL(1)$ gramatiku musí platit:

$$A \to \alpha, A \to \varepsilon \Rightarrow FIRST_1(\alpha) \cap FOLLOW_1(A) = \emptyset$$

nesplnění této podmínky se označuje jako konflikt $FIRST - FOLLOW$.

Nechť $\{a\} \subseteq FIRST_1(\gamma_i) \cap FOLLOW_1(A)$. Může to být způsobeno tím, že v gramatice je pravidlo následujícího tvaru: $X \to \alpha A a \beta$ a přitom $A-$pravidla jsou tyto: $A \to \gamma_1 \mid \ldots \mid \gamma_n$. Konflikt se můžeme pokusit odstranit tak, že pravidlo $X \to \alpha A a \beta$ nahradíme pravidlem $X \to \alpha [Aa] \beta$ a pro nový neterminál $[Aa]$ přidáme pravidla $[Aa] \to \gamma_1 a \mid \ldots \mid \gamma_n a$. Tím jsme z množiny $FOLLOW_1(A)$ vyloučili symbol a (za předpokladu, že tam není z jiného důvodu).

4.11 Řešte kolizi FIRST-FOLLOW v následující gramatice:

$G = (\{A, B, C\}, \{a, c, b\}, P, A)$, kde
$P = \{\ A \to BaC,$
$\quad B \to \varepsilon \mid aaC,$
$\quad C \to c \mid bC\ \}$

4.12 Řešte kolizi FIRST-FOLLOW v následující gramatice:

$G = (\{A, B, C\}, \{a, c\}, P, A)$, kde
$P = \{\ A \to BaC,$
$\quad B \to aB \mid \varepsilon,$
$\quad C \to c\ \}$

Extrakce pravého kontextu

Pohlcení terminálního symbolu je možné udělat pouze tehdy, vyskytuje-li se přímo za problematickým neterminálem. Situce ale může vypadat například takto: $X \to \alpha A Y \beta$ a $Y\beta \Rightarrow^+ a\gamma$. V tomto případě nelze přímo použít transformaci pohlcení terminálního symbolu, můžeme se však pokusit (opakovaně) substituovat za neterminál bezprostředně vpravo od neterminálu A a obdržet tak konflikt v požadované formě. Této transformaci říkáme extrakce pravého kontextu.

4.13 Pokuste se provést extrakci pravého kontextu a zamyslete se:

$G = (\{S, A, B\}, \{d, a, b, c\}, P, S)$, kde
$P = \{\ S \to Ad,$
$\quad A \to aAB \mid b \mid bbc \mid \varepsilon,$
$\quad B \to A\ \}$

4.14 Řešte kolizi FIRST-FOLLOW v následující gramatice:

$G = (\{S, B, A, C\}, \{a, b, d, c\}, P, S)$, kde
$P = \{\ S \to aBAa \mid bABb,$
$\quad A \to \varepsilon \mid aC \mid BCd,$
$\quad B \to bB \mid CdC,$
$\quad C \to cC \mid d\ \}$

4.15 Transformujte na $LL(1)$ následující gramatiku:

$G = (\{E, T, F, S\}, \{o, n, a, (,)\}, P, E)$, kde
$P = \{\ E \to EoT \mid T,$
$\quad T \to ToF \mid F,$
$\quad F \to nS \mid S,$
$\quad S \to a \mid (E)\ \}$

4.16 Ověřte, zda je následující gramatika $LL(1)$ gramatika. Pokud není, použijte standardní transformace na úpravu gramatiky na $LL(1)$ a o výsledné gramatice dokažte, že je $LL(1)$ gramatikou.

$G = (\{S, A\}, \{b, c, a\}, P, S)$, kde
$P = \{\ S \rightarrow bcAa \mid bb,$
$\quad A \rightarrow \varepsilon \mid acAb\ \}$

LR(0) a SLR(k) analyzátory

$LR(0)$ analyzátor

5.1 Zkonstruujte $LR(0)$ analyzátor

$G = (\{S, A\},\{a, b, c\},P, S)$, kde
$P = \{\ S \rightarrow aAa,$
$S \rightarrow aAb,$
$A \rightarrow abA,$
$A \rightarrow bA,$
$A \rightarrow ac\ \}$

a analyzujte slovo `aabbacb`

5.2 Dokažte, nebo vyvrťte, že následující gramatika je $LR(0)$

$G = (\{S, A, B, C\},\{a, b, c, d\},P, S)$, kde
$P = \{\ S \rightarrow aAa,$
$S \rightarrow bBb,$
$S \rightarrow cCc,$
$S \rightarrow dCd,$
$A \rightarrow a \mid aA,$
$B \rightarrow bA \mid Cc,$
$C \rightarrow cA\ \}$

5.3 Zkonstruujte $LR(0)$ analyzátor a analyzujte libovolné slovo

$G = (\{S, A, C, B, D\},\{a, b, c, d\},P, S)$, kde
$P = \{\ S \rightarrow aAa,$
$S \rightarrow bAb,$
$S \rightarrow cCc,$
$S \rightarrow dCd,$
$A \rightarrow B,$
$B \rightarrow b,$
$C \rightarrow D,$
$D \rightarrow d\ \}$

$SLR(k)$ analyzátor

5.4 Zkonstruujte $SLR(1)$ a $SLR(2)$ analyzátor

$G = (\{S, A, B\},\{a, b\},P, S)$, kde
$P = \{\ S \rightarrow AB,$
$A \rightarrow aAb,$
$A \rightarrow \varepsilon,$
$B \rightarrow bB,$
$B \rightarrow b\ \}$

a analyzujte slovo `aabbbb` a `aaabbb`

5.5 Najděte všechny $SLR(2)$ konflikty

$G = (\{S, P, R\}, \{a, b, c, d\}, P, S)$, kde
$P = \{\ S \rightarrow PP,$
$P \rightarrow Ra,$
$P \rightarrow bRc,$
$P \rightarrow dc,$
$P \rightarrow bda,$
$R \rightarrow d\ \}$

5.6 Dokažte, že gramatika z příkladu 5.5 není $SLR(k)$ pro žádné k

5.7 Rozhodněte, zda následující gramatika je $SLR(k)$

$G = (\{X, S, L, R\}, \{=, i, d, *\}, P, X)$, kde
$P = \{\ X \rightarrow S,$
$S \rightarrow L = R,$
$S \rightarrow R,$
$L \rightarrow id,$
$L \rightarrow *R,$
$R \rightarrow L\ \}$

5.8 Zkonstruujte $SLR(1)$ analyzátor:

$G = (\{X, S, A\}, \{+, (,), a\}, P, X)$, kde
$P = \{\ X \rightarrow S,$
$S \rightarrow S + A,$
$S \rightarrow A,$
$A \rightarrow (S),$
$A \rightarrow a(S),$
$A \rightarrow a\ \}$

$LR(0)$ a $SLR(k)$ gramatiky

5.9 Zkonstruujte $LR(0)$ gramatiky
$L_1 = \{1^n\ a\ 0^n \mid n \geq 0\} \cup \{1^n\ b\ 0^{2n} \mid n \geq 0\}$
$L_2 = \{a\ 1^n\ 0^n \mid n \geq 0\} \cup \{b\ 1^n\ 0^{2n} \mid n \geq 0\}$
$L_3 = \{1^n\ 0^m \mid m > n > 0\}$
$L_4 = \{w \# w^R \mid w \in \{0, 1\}^*\}$

5.10 Dokažte, že následující gramatika není $LR(0)$ ale je $SLR(1)$

$G = (\{X, S, A\}, \{+, (,), a\}, P, X)$, kde
$P = \{\ X \rightarrow S,$
$S \rightarrow S + A,$
$S \rightarrow A,$
$A \rightarrow (S),$
$A \rightarrow a(S),$
$A \rightarrow a\ \}$

5.11 Nalezněte co možná nejjednoduší gramatiku takovou, že

a) není $LR(0)$
a) není $SLR(1)$ protože má konflikt čtení-redukce
a) není $SLR(1)$ protože má konflikt redukce-redukce

5.12 Ověřte, zda je následující gramatika $SLR(1)$ či $SLR(2)$ gramatika.

$G = (\{X, S, B\}, \{a, b\}, P, X)$, kde
$P = \{\ X \rightarrow S,$
$S \rightarrow aB,$
$S \rightarrow a,$
$B \rightarrow bSb\ \}$

LR(k) a LALR(k) analyzátory

6.1 Rozhodněte, zda je gramatika $LR(0)$, $SLR(1)$, $LALR(1)$, $LR(1)$

$G = (\{X, S, A, B\}, \{a, d, b, e, c\}, P, X)$, kde
$P = \{\ X \rightarrow S,$
$S \rightarrow aAd,$
$S \rightarrow bBd,$
$S \rightarrow aBe,$
$S \rightarrow bAe,$
$A \rightarrow c,$
$B \rightarrow c\ \}$

6.2 Rozhodněte, zda je gramatika $LR(0)$, $SLR(1)$, $LALR(1)$, $LR(1)$

$G = (\{X, S, A, B, C, D, E\}, \{a, b\}, P, X)$, kde
$P = \{\ X \rightarrow S,$
$S \rightarrow AB,$
$A \rightarrow a,$
$B \rightarrow CD,$
$B \rightarrow aE,$
$C \rightarrow ab,$
$D \rightarrow bb,$
$E \rightarrow bba\ \}$

6.3 Zkonstruujte $LALR(1)$ analyzátor

$G = (\{X, S, A, B\}, \{a, b, c\}, P, X)$, kde
$P = \{\ X \rightarrow S,$
$S \rightarrow aAb,$
$S \rightarrow c,$
$S \rightarrow cB,$
$A \rightarrow bS,$
$A \rightarrow Bc,$
$B \rightarrow c\ \}$

6.4 Zkonstruujte iniciální stav $LR(2)$ automatu

$G = (\{S, R\}, \{+, (,), a\}, P, S)$, kde
$P = \{\ S \rightarrow R,$
$R \rightarrow RR,$
$R \rightarrow R + R,$
$R \rightarrow (R),$
$R \rightarrow a\ \}$

6.5 Proveďte $LR(1)$ analýzu

$G = (\{X, S, A\}, \{a, b\}, P, X)$, kde
$P = \{\ X \rightarrow S,$
$S \rightarrow aAS,$
$S \rightarrow \varepsilon,$
$A \rightarrow bb\ \}$

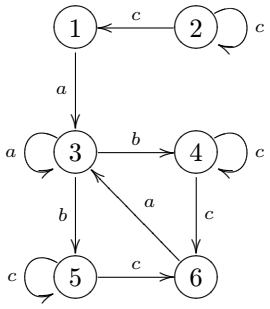
6.6 Zkonstruujte analyzátor a analyzujte slova

$G = (\{S, A, B\}, \{0, 1\}, P, S)$, kde

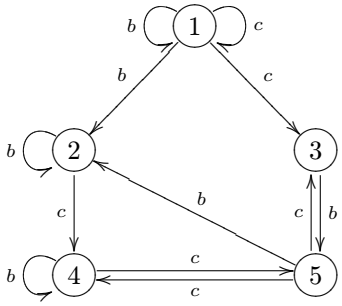
$$
\begin{aligned}
P = \{\ & S \rightarrow AB, \\
& A \rightarrow 0A1, \\
& A \rightarrow \varepsilon, \\
& B \rightarrow 1B, \\
& B \rightarrow 1\ \}
\end{aligned}
$$

Bisimulace

7.1 Pro daný přechodový systém najděte všechny dvojice bisimulačně ekvivalentních stavů metodou hry. Pomocí bisimulačního kolapsu k němu zkonstruujte ekvivalentní přechodový systém.



7.2 Pro daný přechodový sytém najděte maximální bisimulaci metodou postupných aproximací.

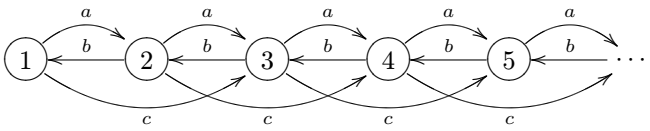


7.3 Je dán přechodový systém P_1 (nekonečně stavový). Zkonstruujte přechodový systém P_2 takový, aby platilo:

(i) P_2 má stav I takový, že $I \sim 1$

(ii) P_2 je konečně stavový

Jaká je maximální bisimulace pro P_1?
P_1:



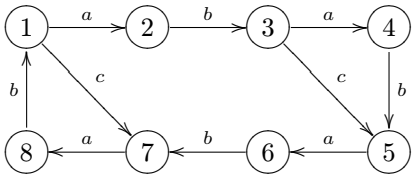
7.4 Najděte konečné automaty A_1, A_2 bez ε-přechodů takové, aby splňovaly všechny tři následující podmínky:

(i) $\mathcal{L}(A_1) = \mathcal{L}(A_2)$

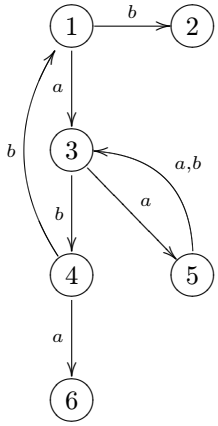
(ii) $\mathcal{L}(A_1)$ je nekonečný

(iii) Počáteční stavy automatů A_1, A_2 nejsou bisimulačně ekvivalentní

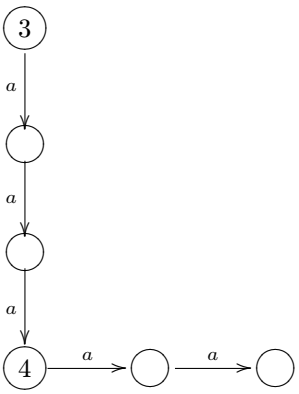
7.5 Dokažte nebo vyvraťte: $2 \sim 8$.
Najděte maximální bisimulaci.



7.6 Najděte nejmenší n, takové aby platilo $3 \not\sim_n 4$, ale $3 \sim_{n-1} 4$. Najděte maximální bisimulaci. Faktorizujte podle relace bisimulace.



7.7 Najděte nejmenší n, takové aby platilo $3 \not\sim_n 4$, ale $3 \sim_{n-1} 4$. Najděte maximální bisimulaci. Faktorizujte podle relace bisimulace.



Přechodové systémy BPA a BPP

8.1 Otázky:

- Je daný proces popsaný deterministickým konečným automatem. Jak zjistím, které stavy jsou bisimulačně ekvivalentní?
- Najděte postačující podmínku na to, aby pro normovaný přechodový systém, byla jazyková ekvivalence shodná s bisimulací.
- Pro normované BPA ověřte: $ABC \sim DBC \Rightarrow A \sim D$
- Najděte nutnou podmínku, aby pro BPA platilo $AA \sim A$.

8.2 Dána BPA algebra. Najděte přechodový systém, který je určen

$$X \xrightarrow{a} XBB$$
$$X \xrightarrow{c} \varepsilon$$
$$B \xrightarrow{a} BBB$$
$$B \xrightarrow{b} \varepsilon$$

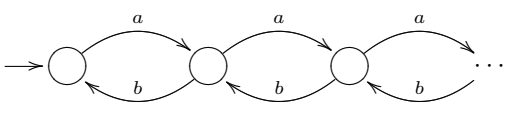
8.3 Jaký jazyk generuje následující BPA (z proměnné X)?

$$X \xrightarrow{a} XA$$
$$X \xrightarrow{b} XB$$
$$X \xrightarrow{c} \varepsilon$$
$$A \xrightarrow{a} \varepsilon$$
$$B \xrightarrow{b} \varepsilon$$

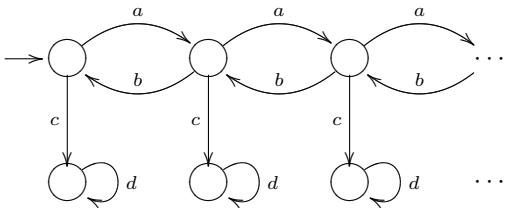
8.4 Nakreslete přechodový systém určený BPP algebrou

$$X \xrightarrow{a} X|B$$
$$X \xrightarrow{c} X|D$$
$$B \xrightarrow{b} \varepsilon$$
$$D \xrightarrow{d} \varepsilon$$
$$(X \xrightarrow{e} \varepsilon\ )$$

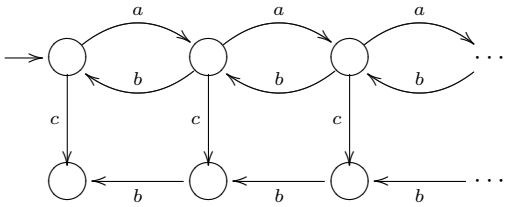
8.5 Vyjádřete daný přechodový systém BPA i BPP syntaxí:



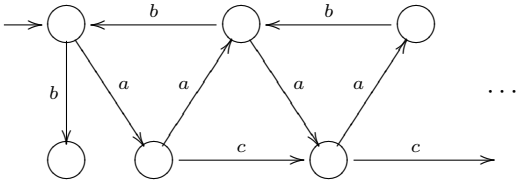
8.6 Vyjádřete daný přechodový systém BPA syntaxí:



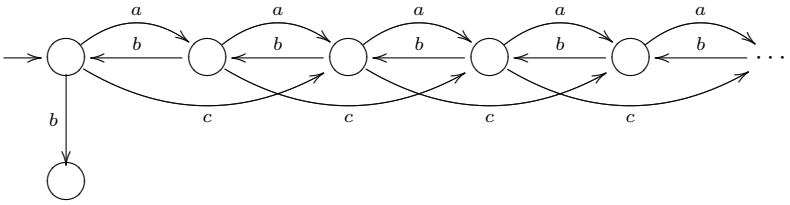
8.7 Vyjádřete daný přechodový systém BPP syntaxí:



8.8 Vyjádřete daný přechodový systém BPA syntaxí:



8.9 Vyjádřete daný přechodový systém BPP syntaxí:



Konstrukce tabel pro BPA

Tablo pro $\gamma = \delta$ se skládá z podtabel. Každé podtablo je strom. Podtablo s kořenem $X\alpha = Y\beta$ vytvoříme následovně. Nechť $k = min\{\|X\|, \|Y\|\}$. Na uzel $X\alpha = Y\beta$ aplikujeme k-krát trojici pravidel (REC, SUM, PREFIX). Po k-aplikacích jsou některé uzly reziduály, jeden z nich označíme jako vytčený a podle něj aplikujeme na ostatní uzly v aktuálním patře pravidlo SUB (respektive SUBL, nebo SUBR). Reziduálem je uzel ve tvaru $\alpha = \gamma\beta$ (použijeme pravdilo SUBL) nebo $\gamma\alpha = \beta$ (použijeme pravdilo SUBR), případně $\alpha = \beta$ (použijeme pravdilo SUB (tj. SUBL, nebo SUBR, na straně nezáleží)). Po aplikaci odpovídajícího pravidla SUB (*a jen tehdy*) obdržíme listy podtabla. Po dokončení podtabla, zkontrolujeme všechny jeho listy. U každého listu nastává jeden z následujících případů:

- List je úspěšný. (Netřeba pro něj budovat podtablo.)
- List je neuspěšný. Pak je celé tablo neúspěšné.
- List není ani úspěšný ani neúspěšný, stává se kořenem nového podtabla.

Jestliže v průběhu tvorby kteréhokoliv podtabla se některý uzel (nemusí to být nutně list) ukáže být neúspěšným podle níže uvedených kritérií, je neúspěšné celé tablo.

Každé podtablo je téměř vyvážený strom. Všechny cesty v něm jsou stejně dlouhé, liší se maximálně o jedna (po aplikaci pravidla SUB,SUBL nebo SUBR se pod vytčeným reziduálem cesta neprodlouží, sám reziduál slouží jako list, a tedy případně i jako kořen dalšího podtabla, zatímco pod ostatními uzly na stejném patře se cesta prodlouží o jedna (díky aplikaci odpovídajícího pravidla SUB) , a teprve tyto uzly (o patro níž než je reziduál) tvoří listy daného podtabla).

Kritéria úspěšnosti:

1. $\alpha = \alpha$ (tj. levá a pravá strana jsou shodné, α zde nemá nic společného s α použitou výše či níže)
2. $\alpha = \beta$ a na cestě od tohoto listu do kořene *celého tabla* se vyskytuje uzel se stejným označením $\alpha = \beta$ a mezi nimi je alespoň jednou použito pravidlo PREFIX (uzel $\beta = \alpha$ se nepočítá!)

Kritéria neúspěšnosti :

1. $a.\alpha = b.\beta$,kde a je různé od b
2. $\alpha = \beta$,kde $|\alpha|$ je různá od $|\beta|$

Poznámka:

1. Výše uvedená metoda funguje pouze pro NORMOVANÉ BPA !!!
2. Poznámka o zápisu: XY = X.Y

Pravidla

$$\frac{X\alpha = Y\beta}{E\alpha = F\beta} REC \qquad \text{kde } X \stackrel{def}{=} E \text{ a } Y \stackrel{def}{=} F$$

$$\frac{a\alpha = a\beta}{\alpha = \beta} PREFIX$$

$$\frac{(\sum_{i=1}^{m} a_i\alpha_i)\alpha = (\sum_{j=1}^{n} b_j\beta_j)\beta}{\{a_i\alpha_i\alpha = b_{f(i)}\beta_{f(i)}\beta\}_{i=1}^{m}\{a_{g(j)}\alpha_{g(j)}\alpha = b_j\beta_j\beta\}_{j=1}^{n}}SUM$$

kde $f : \{1,\ldots,m\} \to \{1,\ldots,n\}$
$g : \{1,\ldots,n\} \to \{1,\ldots,m\}$
$m, n \geq 1$

$$\frac{\alpha_i\alpha = \beta_i\beta}{\alpha_i\gamma = \beta_i}SUBL$$ kde $\alpha = \gamma\beta$ je vytčený residuál

$$\frac{\alpha_i\alpha = \beta_i\beta}{\alpha_i = \beta_i\gamma}SUBR$$ kde $\gamma\alpha = \beta$ je vytčený residuál

9.1 Zkonstruujte důkaz $PQQ = SU$

$P = aPQQ + bRQQ + c$
$Q = c$
$R = bP$
$S = aSU + bT + c$
$T = bSU$
$U = cV$
$V = c$

9.2 Zkonstruujte důkaz $FX = A$, $XCB = BCX$, $FBX = AB$, $CXB = XBB$

$A = aBCX + aB$
$B = aC$
$C = aD$
$D = bD + c$
$F = a + aXC$
$X = aY$
$Y = aD$

9.3 Dokažte $DH \sim DFG$, $AH \sim AGF$, $EF \not\sim D$, $BA \not\sim DG$

$A = aBC + aD + aEF$
$B = b$
$C = c$
$D = bH + bC$
$E = bG$
$F = c$
$G = bA$
$H = cBA$

9.4 Zkonstruujte důkaz $Y = C$

$X = aYX + b$
$Y = bX$
$A = aC + b$
$C = bAA$

9.5 Zkonstruujte důkaz $X = A$

$X = aXX + b + cY$
$Y = aYX + b + cX$
$A = aAA + b + cA$

9.6 Zkonstruujte důkaz $A = E$

$$
\begin{aligned}
A &= aBC + aH \\
B &= b \\
C &= d + aDE \\
D &= bF \\
F &= c \\
E &= aC + aGH \\
G &= b \\
H &= d + aI \\
I &= bK \\
K &= cA
\end{aligned}
$$

9.7 Zkonstruujte důkaz $A = X$

$$
\begin{aligned}
X &= d + aXX + bY + cZ \\
Y &= bX + aYY + cZ + d \\
Z &= bX \\
A &= d + aAA + bA + cB \\
B &= bA
\end{aligned}
$$

9.8 Zkonstruujte důkaz $AB = XYB$

$$
\begin{aligned}
A &= aB + aC \\
X &= a + aZ \\
B &= aB + a \\
Y &= aY + a \\
Z &= bZ + bX \\
C &= bC + bA
\end{aligned}
$$

9.9 Zkonstruujte důkaz $FBI = AB$

$$
\begin{aligned}
A &= aBCI + aB \\
B &= aC \\
C &= aD \\
D &= bD + c \\
F &= a + aIC \\
I &= aK \\
K &= aD
\end{aligned}
$$

Büchiho automaty

10.1 Navrhněte nedeterministický BA pro jazyk všech ω-slov nad abecedou $\Sigma = \{a, b, c\}$, která obsahují nekonečný počet symbolů b a c.

10.2 Navrhněte nedeterministický BA pro jazyk všech ω-slov nad abecedou $\Sigma = \{a, b, c\}$, která obsahují nekonečný počet symbolů b a c a pro která platí, že pokud libovolný konečný prefix slova obsahuje lichý počet symbolů b pak obsahuje také sudý počet symbolů c.

10.3 Navrhněte nedeterministický BA pro jazyk
$L = \{w = w_1 w_2 w_3 \ldots \mid w_i \in \{a, b\} \text{ pro } i \geq 1,\ \exists \text{ nekonečně mnoho } j \in \mathbf{N} \text{ takových, že } w_j = w_{j+4}\}$

10.4 Nechť $\Sigma = \{0, 1, \#\}$. Pro slovo $w = a_0 a_1 a_3 \ldots \in \Sigma^\omega$ a dvě čísla $i, j \in \mathbf{N}$, $0 \leq i \leq j$ označme $w[i, j]$ podslovo $a_i \ldots a_j$. Pro pevně zvolené $n \in \mathbf{N}$ definjme jazyk:

$L_n = \{w \mid w \in ((0+1)^{n-1}\#)^\omega$, a pro nekonečně mnoho $i \geq 0$ platí:
$w[2in, (2i+1)n - 1] \neq w[(2i+1)n, (2i+2)n - 1]\}$.

Popište dva nedeterministické BA A_1, A_2 velikosti $O(n)$ takové, aby $L_n = L(A_1) \cap L(A_2)$.

10.5 Dokažte, nebo vyvraťte: Ke každému NBA A lze zkonstruovat NBA A' s jediným počátečním stavem.

10.6 Nechť jazyk $L = \{w \in \{0, 1\}^\omega \mid \text{buď } 0 \notin inf(w), \text{ nebo } 1 \notin inf(w)\}$.

a) zkonstruujte nedeterministický BA

b) diskutujte, zda je možné sestrojit pro daný jazyk NBA s jedním koncovým stavem

c) dokažte, že pro daný jazyk nelze sestrojit NBA s jedním koncovým stavem

10.7 Konstruujte NBA pro následující jazyky L nad abecedou $\{a, b, c, d\}$.

a) $L = \{w \mid inf(w) = \{a\}\}$

b) $L = \{w \mid inf(w) = \{a, b\}\}$

c) $L = \{w \mid a \in inf(w)\}$

d) $L = \{w \mid a \in inf(w) \wedge c \notin inf(w)\}$

e) $L = \{w \mid \{a, b\} \subseteq inf(w) \wedge d \notin inf(w)\}$

f) $L = \{w \mid \{a, b\} \subset inf(w)\}$

10.8 Buď A konečný automat. Označme $L(A)$ množinu slov, kterou akceptuje konečný automat A. Označme $L_\omega(A)$ množinu ω-slov, kterou akceptuje Büchiho automat A. Najděte automat A tak, aby platilo

a) $(L(A))^\omega = L_\omega(A)$

b) $(L(A))^\omega \neq L_\omega(A)$