



Faculty of Informatics  
Masaryk University Brno

---

# Cvičení k předmětu IA006

## Vybrané kapitoly z teorie automatů

poslední modifikace 21. září 2016

---

Tato sbírka byla vytvořena z příkladů ke cvičení z předmětu *Teorie automatů a formálních jazyků II*, které byly původně připraveny Ivanou Černou. Na opravě nesčetných chyb a doplnění příkladů se podílelo mnoho studentů a cvičící předmětu *IA006 Vybrané kapitoly teorie automatů* Jiří Barnat, Vojtěch Řehák a Jan Strejček.

# 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.

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

## Operace $k:w$ a $L_1 \oplus_k L_2$

Nechť  $k \in \mathbb{N}$  je přirozené číslo,  $w$  je slovo a  $L_1, L_2$  jsou jazyky. Operace  $k:w$  a  $L_1 \oplus_k L_2$  definujeme následovně:

$$k:w = \begin{cases} w & \text{pokud } |w| \leq k \\ u, \text{ kde } |u| = k \text{ a } w = u.v \text{ pro nějaké } v & \text{pokud } |w| > k \end{cases}$$
$$L_1 \oplus_k L_2 = \{k:w \mid w \in L_1.L_2\}$$

Tedy  $k:w$  je prvních  $k$  písmen slova  $w$  (případně celé slovo  $w$ , je-li jeho délka kratší než  $k$ ) a  $L_1 \oplus_k L_2$  je jazyk všech slov z  $L_1.L_2$  zkrácený stejným způsobem na prvních  $k$  písmen.

## FIRST a FOLLOW

Nechť  $G = (N, \Sigma, P, S)$  je bezkontextová gramatika. Pro každé  $k \in \mathbb{N}$  jsou funkce  $FIRST_k^G$  a  $FOLLOW_k^G$  definovány následovně:

$$FIRST_k^G : (\Sigma \cup N)^* \longrightarrow 2^{\Sigma^*}$$

$$FIRST_k^G(\alpha) = \{k : w \mid \alpha \Rightarrow^* w, w \in \Sigma^*\}$$

$$FOLLOW_k^G : N \longrightarrow 2^{\Sigma^*}$$

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

Pozor na typ argumentu u jednotlivých funkcí. Argumentem funkce  $FIRST_k^G(\alpha)$  je řetězec terminálů a neterminálů ( $\alpha \in (\Sigma \cup N)^*$ ), na rozdíl od funkce  $FOLLOW_k^G(A)$ , jejíž argumentem je vždy právě jeden neterminál ( $A \in N$ ).

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

$$FIRST_k^G : 2^{(\Sigma \cup N)^*} \longrightarrow 2^{\Sigma^*}$$

$$FIRST_k^G(M) = \bigcup_{\alpha \in M} FIRST_k^G(\alpha)$$

$$FOLLOW_k^G : 2^N \longrightarrow 2^{\Sigma^*}$$

$$FOLLOW_k^G(M) = \bigcup_{A \in M} FOLLOW_k^G(A)$$

Dále budeme používat 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).

## Algoritmus pro výpočet funkce FIRST

Je dána gramatika  $G = (N, \Sigma, P, S)$  a řetězec  $\alpha = Y_1 Y_2 \dots Y_l$ , kde každé  $Y_i \in N \cup \Sigma \cup \{\varepsilon\}$ .

- 1)  $FI_k(x) = \{x\}$  pro  $x \in \Sigma$ ,  $FI_k(\varepsilon) = \{\varepsilon\}$
- 2) Výpočet  $FI_k(x)$  pro  $x \in N$ :

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

$$X_i \rightarrow Y_1^1 \dots Y_{k_1}^1 \mid Y_1^2 \dots Y_{k_2}^2 \mid \dots \mid Y_1^j \dots Y_{k_j}^j$$

Potom

$$\begin{aligned} FI_k(X_i) = & (FI_k(Y_1^1) \oplus_k FI_k(Y_2^1) \oplus_k \dots \oplus_k FI_k(Y_{k_1}^1)) \cup \\ & \cup (FI_k(Y_1^2) \oplus_k FI_k(Y_2^2) \oplus_k \dots \oplus_k FI_k(Y_{k_2}^2)) \cup \\ & \vdots \\ & \cup (FI_k(Y_1^j) \oplus_k FI_k(Y_2^j) \oplus_k \dots \oplus_k FI_k(Y_{k_j}^j)) \end{aligned}$$

Hodnoty  $FI_k(X_i)$  jsou nejmenším pevným bodem uvedené soustavy rekurzivních rovnic. Nejmenší pevný bod spočítáme tak, že nejprve nastavíme všechna  $FI_k(X_i)$  na hodnotu  $FI_k(X_i) = \emptyset$ . V první iteraci dosadíme tyto hodnoty do pravých stran rovnic. Vyhodnocením rovnic získáme nové hodnoty pro všechna  $FI_k(X_i)$ . Tyto nové hodnoty v další iteraci opět dosadíme do pravých stran rovnic. Tento postup opakujeme tak dlouho, dokud nám ve dvou po sobě jdoucích iteracích nevyjdou stejné hodnoty pro všechna  $FI_k(X_i)$ . Takto získané hodnoty jsou nejmenším pevným bodem a tedy i hledanými množinami  $FI_k(X_i)$ . Chceme-li výpočet nejmenšího pevného bodu urychlit, můžeme při dosazování do pravých stran rovnic použít namísto hodnot z předchozí iterace již známé hodnoty z aktuální iterace.

- 3)  $FI_k(\alpha) = FI_k(Y_1) \oplus_k FI_k(Y_2) \oplus_k \dots \oplus_k FI_k(Y_l)$

## Algoritmus pro výpočet funkce FOLLOW

Je dána gramatika  $G = (N, \Sigma, P, S)$ . Funkce  $FO_k$  je definována pouze pro neterminály. Opět budeme počítat hodnotu  $FO_k(X_i)$  pro všechny neterminály  $X_i$  současně. Nejprve opět sestavíme příslušné rovnice. Pro každý neterminál  $A$ , který není kořenový, klademe:

$$FO_k(A) = \bigcup_{(B \rightarrow \alpha A \beta) \in P} FI_k(\beta) \oplus_k FO_k(B)$$

Pro kořenový neterminál  $S$  klademe:

$$FO_k(S) = \{\varepsilon\} \cup \bigcup_{(B \rightarrow \alpha S \beta) \in P} FI_k(\beta) \oplus_k FO_k(B)$$

Hodnoty tvaru  $FI_k(\beta)$  spočítáme dle předchozího algoritmu a dosadíme do rovnic. Následně spočítáme všechny hodnoty  $FO_k(X_i)$  jako nejmenší pevný bod rovnic (použijeme stejný postup výpočtu jako u předchozího algoritmu).

Hodnoty výrazu  $FI_k(\alpha)$  i  $FO_k(A)$  lze často určit intuitivně, buď přímo z definice funkcí nebo s využitím příslušných rovnic (zejména pokud nejsou rekurzivní). Uvedené rovnice se často dají zjednodušit s využitím následujícího pozorování. Pokud víme, že délka nejkratšího řetězce v jazyku  $L_1$  je alespoň  $n$ , pak platí:

$$L_1 \oplus_k L_2 = \begin{cases} L_1 \oplus_k FI_{k-n}(L_2) & \text{pokud } n < k \\ \{k:w \mid w \in L_1\} & \text{pokud } n \geq k \text{ a } L_2 \neq \emptyset \\ \emptyset & \text{pokud } n \geq k \text{ a } L_2 = \emptyset \end{cases}$$

S využitím tohoto principu snadno můžeme například nahradit výraz  $FI_3(aBcD) \oplus_3 FO_3(E)$  ekvivalentním výrazem  $FI_3(aBcD) \oplus_3 FO_1(E)$ . Z podobných důvodů lze například výraz  $FI_2(aBcDeF)$  nahradit ekvivalentním výrazem  $FI_2(aBc)$ , ovšem pouze za předpokladu, že neterminály  $D, F$  jsou normované (tj. lze z nich odvodit nějaký řetězec terminálů).

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

$$G = (\{A, B, C, D\}, \{c, a, d, e\}, P, A), \text{ kde}$$

$$P = \left\{ \begin{array}{l} A \rightarrow Bc \mid a, \\ B \rightarrow A \mid C \mid d, \\ C \rightarrow B \mid d, \\ D \rightarrow e \end{array} \right\}$$

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

$$G = (\{A\}, \{a, b\}, P, A), \text{ kde}$$

$$P = \left\{ \begin{array}{l} A \rightarrow Aa, \\ A \rightarrow b \end{array} \right\}$$

**1.4** 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:

$$G = (\{S, A, B, C\}, \{a, c, b, e, d\}, P, S), \text{ kde}$$

$$P = \left\{ \begin{array}{l} S \rightarrow aAc \mid B, \\ A \rightarrow aA \mid bSCe \mid \varepsilon, \\ B \rightarrow aC \mid \varepsilon, \\ C \rightarrow d \mid \varepsilon \end{array} \right\}$$

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

$$G = (\{S, A, B, C, D\}, \{a, b, d, c, x, y, z\}, P, S), \text{ kde}$$

$$P = \left\{ \begin{array}{l} S \rightarrow aABbCD \mid \varepsilon, \\ A \rightarrow ASd \mid \varepsilon, \\ B \rightarrow SAc \mid xC \mid \varepsilon, \\ C \rightarrow Sy \mid Cz \mid \varepsilon, \\ D \rightarrow aBD \mid \varepsilon \end{array} \right\}$$

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

$$G = (\{S, B, A, D, C\}, \{a, c, b, d\}, P, S), \text{ kde}$$

$$P = \left\{ \begin{array}{l} S \rightarrow aBcB, \\ A \rightarrow aA \mid Aa, \\ B \rightarrow DAc \mid bA, \\ C \rightarrow cBc \mid aaB, \\ D \rightarrow d \mid dC \end{array} \right\}$$

**1.7** Zamyslete se, v jakých případech platí následující vztahy:

- |  |                                    |
|--|------------------------------------|
| a) $FI_k(A) = \emptyset$   | f) $FO_k(A) = \emptyset$           |
| b) $\varepsilon \in FI_k(A)$   | g) $\varepsilon \in FO_k(A)$       |
| c) $FI_k(A) \subseteq FI_{k+1}(A)$                                     | h) $FO_k(A) \subseteq FO_{k+1}(A)$ |
| d) $FI_{k+1}(A) \subseteq FI_k(A)$                                     | i) $FO_{k+1}(A) \subseteq FO_k(A)$ |
| e) $FI_k(\alpha \cdot FO_k(X)) = FI_k(\alpha) \oplus_k FO_k(X)$        | j) $FO_k(A) \subseteq FO_k(B)$     |
| k) $(L_1 \oplus_k L_2) \oplus_k L_3 = L_1 \oplus_k (L_2 \oplus_k L_3)$ |                                    |

# SLL(k) gramatiky a analyzátoři

Intuitivně řečeno, gramatika  $G$  je  $SLL(k)$ , právě když se odpovídající nedeterministický syntaktický analyzátor pro analýzu shora dolů dokáže vždy jednoznačně rozhodnout pouze na základě neterminálu na vrcholu zásobníku a prvních  $k$  písmen z dosud nepřčteného vstupu. Formálně, gramatika  $G$  je  $SLL(k)$ , právě když pro všechny neterminály  $A \in N$  a pro každá dvě různá pravidla  $A \rightarrow \beta$ ,  $A \rightarrow \gamma$  platí:

$$FI_k(\beta) \oplus_k FO_k(A) \cap FI_k(\gamma) \oplus_k FO_k(A) = \emptyset$$

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

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

**2.2** Ověřte, zda gramatika je  $SLL(3)$ :

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

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

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

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

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

# LL(k) gramatiky a analyzátoři

Gramatika je  $LL(k)$ , právě když pro dvě libovolná různá pravidla gramatiky  $A \rightarrow \beta, A \rightarrow \gamma$  a pro všechny nejlevější větne 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  $LL(2)$ :

$$G = (\{S, X, Y\}, \{b, a\}, P, S), \text{ kde}$$
$$P = \left\{ \begin{array}{l} S \rightarrow X, \\ X \rightarrow Y \mid bYa, \\ Y \rightarrow a \mid \varepsilon \end{array} \right\}$$

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

$$G = (\{S, A, B\}, \{a, b\}, P, S), \text{ kde}$$
$$P = \left\{ \begin{array}{l} S \rightarrow aAaB \mid bAbB, \\ A \rightarrow a \mid ab, \\ B \rightarrow aB \mid a \end{array} \right\}$$

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

$$G = (\{S, A, B\}, \{a, b, 0, 1\}, P, S), \text{ kde}$$
$$P = \left\{ \begin{array}{l} S \rightarrow A \mid B, \\ A \rightarrow aAb \mid 0, \\ B \rightarrow aBbb \mid 1 \end{array} \right\}$$

**3.4** Ukažte, že gramatika je  $LL(k)$ :

$$G = (\{S, T, A, B\}, \{a, b, c\}, P, S), \text{ kde}$$
$$P = \left\{ \begin{array}{l} S \rightarrow aT \\ T \rightarrow SA \mid A \\ A \rightarrow bB \mid c \\ B \rightarrow b^{k-1}d \mid \varepsilon \end{array} \right\}$$

**3.5** Zkonstruujte  $LL(2)$  analyzátoři pro gramatiku  $G$ :

$$G = (\{S, A\}, \{a, b\}, P, S), \text{ kde}$$
$$P = \left\{ \begin{array}{l} S \rightarrow \varepsilon, \\ S \rightarrow abA, \\ A \rightarrow Saa, \\ A \rightarrow b \end{array} \right\}$$

**3.6** Zkonstruujte  $LL(3)$  analyzátoři pro gramatiku  $G$ :

$$G = (\{S, A, B\}, \{a, b\}, P, S), \text{ kde}$$
$$P = \left\{ \begin{array}{l} S \rightarrow aAaB, \\ S \rightarrow bAbB, \\ A \rightarrow a, \\ A \rightarrow ba, \\ B \rightarrow aB, \\ B \rightarrow a \end{array} \right\}$$

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

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

**3.8** Navrhněte gramatiku, která je LL(2) a není SLL(2).

**3.9** Navrhněte gramatiku, která

- a) je SLL(3) a není LL(2)
- b) je SLL(2) a není LL(3)
- c) je LL(2) a není SLL(3)
- d) je LL(3) a není SLL(2)

**3.10** Navrhněte gramatiku, která není LL( $k$ ) pro žádné  $k$ .

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

## Odstranění levé rekurze

4.1 Odstraňte levou rekurzi v následující gramatice:

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

## Rohová substituce

Nechť  $G = (N, \Sigma, P, S)$  je bezkontextová gramatika s pravidlem  $A \rightarrow B\alpha$ , a necht všechny  $B$ -pravidla jsou  $B \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_m$ .

Nechť  $G_1 = (N', \Sigma, P', S)$  je gramatika, která vznikla z  $G$  vyloučením pravidla  $A \rightarrow B\alpha$  a přidáním pravidel  $A \rightarrow \alpha_1\alpha \mid \alpha_2\alpha \mid \dots \mid \alpha_m\alpha$ .

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

4.2 Aplikujte rohovou substituci na následující gramatiku:

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



4.3 Aplikujte rohovou substituci na následující gramatiku:

$$G = (\{S, A, B\}, \{a, b\}, P, S), \text{ kde}$$

$$P = \{ S \rightarrow AB,$$

$$A \rightarrow aA \mid \varepsilon,$$

$$B \rightarrow bA \mid \varepsilon \}$$

## 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  $A \rightarrow \alpha\alpha_1 \mid \alpha\alpha_2 \mid \dots \mid \alpha\alpha_n$ , kde  $\alpha \neq \varepsilon$ . Kolizi odstraníme jestliže uvedená pravidla nahradíme pravidly tvaru  $A \rightarrow \alpha A', A' \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$ .

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 Opakovaně aplikujte levou faktorizaci na gramatiku:

$$G = (\{A, B, C\}, \{a, b, c, d\}, P, A), \text{ kde}$$

$$P = \{ A \rightarrow abB \mid acC \mid acdB \mid bBb,$$

$$B \rightarrow bbaB \mid bbadC \mid cBC,$$

$$C \rightarrow aA \mid b \}$$

4.5 Aplikujte levou faktorizaci na gramatiku:

$$G = (\{A, B, C\}, \{a, x, y, z\}, P, A), \text{ kde}$$

$$P = \{ A \rightarrow aBxx \mid aCyy \mid zy \mid zx,$$

$$B \rightarrow aBx \mid z,$$

$$C \rightarrow yCy \mid z \}$$

4.6 Odstraňte konflikt FIRST-FIRST v následující gramatice.

$$G = (\{S, A, B\}, \{c, a, b\}, P, S), \text{ kde}$$

$$P = \{ S \rightarrow A \mid B,$$

$$A \rightarrow cA \mid a,$$

$$B \rightarrow cB \mid b \}$$

4.7 Odstraňte konflikt FIRST-FIRST v následující gramatice.

$$G = (\{A, B, C\}, \{a, b, c, d\}, P, A), \text{ kde}$$

$$P = \{ A \rightarrow aB \mid CB,$$

$$C \rightarrow aC \mid bB,$$

$$B \rightarrow cB \mid d \}$$

4.8 Odstraňte konflikt FIRST-FIRST v následující gramatice.

$$G = (\{A, B, D\}, \{c, d, b, x, y, z\}, P, A), \text{ kde}$$

$$P = \{ A \rightarrow Bc \mid Dd,$$

$$B \rightarrow bx \mid y,$$

$$D \rightarrow Bz \}$$

4.9 Odstraňte konflikt FIRST-FIRST v následující gramatice.

$$G = (\{S, A, B\}, \{b, c, a, d\}, P, S), \text{ kde}$$

$$P = \{ S \rightarrow A \mid AbcB \mid bc,$$

$$A \rightarrow a,$$

$$B \rightarrow A \mid dd \}$$

4.10 Odstraňte konflikt FIRST-FIRST v následující gramatice.

$$G = (\{S, A, B\}, \{a, b, c\}, P, S), \text{ kde}$$

$$P = \{ S \rightarrow A \mid B,$$

$$A \rightarrow aAb \mid \varepsilon,$$

$$B \rightarrow aBc \mid \varepsilon \}$$

## Pohlčení terminálního symbolu

Pro  $LL(1)$  gramatiku musí platit:

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

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:  $X \rightarrow \alpha A a \beta$  a přitom  $A$ -pravidla jsou tyto:  $A \rightarrow \gamma_1 \mid \dots \mid \gamma_n$ . Konflikt se můžeme pokusit odstranit tak, že pravidlo  $X \rightarrow \alpha A a \beta$  nahradíme pravidlem  $X \rightarrow \alpha [Aa] \beta$  a pro nový neterminál  $[Aa]$  přidáme pravidla  $[Aa] \rightarrow \gamma_1 a \mid \dots \mid \gamma_n a$ . 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:

$$G = (\{A, B, C\}, \{a, c, b\}, P, A), \text{ kde}$$
$$P = \left\{ \begin{array}{l} A \rightarrow BaC, \\ B \rightarrow \varepsilon \quad \mid \quad aaC, \\ C \rightarrow c \quad \quad \mid \quad bC \end{array} \right\}$$

**4.12** Řešte kolizi FIRST-FOLLOW v následující gramatice:

$$G = (\{A, B, C\}, \{a, c\}, P, A), \text{ kde}$$
$$P = \left\{ \begin{array}{l} A \rightarrow BaC, \\ B \rightarrow aB \quad \mid \quad \varepsilon, \\ C \rightarrow c \end{array} \right\}$$

## Extrakce pravého kontextu

Pohlčení terminálního symbolu je možné udělat pouze tehdy, vyskytuje-li se přímo za problematickým neterminálem. Situace ale může vypadat například takto:  $X \rightarrow \alpha AY \beta$  a  $Y \beta \Rightarrow^+ a \gamma$ . V tomto případě nelze přímo použít transformaci pohlčení 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:

$$G = (\{S, A, B\}, \{d, a, b, c\}, P, S), \text{ kde}$$
$$P = \left\{ \begin{array}{l} S \rightarrow Ad, \\ A \rightarrow aAB \mid b \mid bbc \mid \varepsilon, \\ B \rightarrow A \end{array} \right\}$$

**4.14** Řešte kolizi FIRST-FOLLOW v následující gramatice:

$$G = (\{S, B, A, C\}, \{a, b, d, c\}, P, S), \text{ kde}$$
$$P = \left\{ \begin{array}{l} S \rightarrow aBAa \mid bABb, \\ A \rightarrow \varepsilon \quad \mid \quad aC \quad \mid \quad BCd, \\ B \rightarrow bB \quad \mid \quad CdC, \\ C \rightarrow cC \quad \mid \quad d \end{array} \right\}$$

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

$$G = (\{E, T, F, S\}, \{o, n, a, (\cdot)\}, P, E), \text{ kde}$$
$$P = \left\{ \begin{array}{l} E \rightarrow EoT \mid T, \\ T \rightarrow ToF \mid F, \\ F \rightarrow nS \mid S, \\ S \rightarrow a \mid (E) \end{array} \right\}$$

**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.

$$G = (\{S, A\}, \{b, c, a\}, P, S), \text{ kde}$$
$$P = \left\{ \begin{array}{l|l} S \rightarrow bcAa & bb, \\ A \rightarrow \varepsilon & acAb \end{array} \right\}$$

# LR(0) a SLR(k) analyzátoři

Poznámka ke konstrukci analyzátoři LR:

Pro gramatiky, ve kterých se kořen gramatiky (nejčastěji neterminál  $S$ ) vyskytuje na pravé straně nějakého pravidla zavádíme nový neterminál (nejčastěji označen  $X$ ) a pravidlo  $X \rightarrow S$ , který se stává novým kořenem gramatiky. Tato modifikace je nutná pro korektní identifikaci situace, ve které má analyzátoři úspěšně akceptovat.

Pro gramatiky, ve kterých se kořen gramatiky nevyskytuje na žádné pravé straně pravidla gramatiky, není výše uvedená modifikace nutná. Přispívá však k čitelnosti výsledného analyzátoři. (K akceptování dojde vždy právě v jednom stavu položkového automatu, který je navíc při systematické konstrukci vždy druhým stavem položkového automatu.)

## LR(0) analyzátoři

### 5.1 Zkonstruujte LR(0) analyzátoři

$G = (\{S, A\}, \{a, b, c\}, P, S)$ , kde

$$P = \{ \begin{array}{l} S \rightarrow aAa, \\ S \rightarrow aAb, \\ A \rightarrow abA, \\ A \rightarrow bA, \\ A \rightarrow ac \end{array} \}$$

a analyzujte slovo **aabbacb**

### 5.2 Dokažte, nebo vyvráťte, že následující gramatika je LR(0).

$G = (\{S, A, B, C\}, \{a, b, c, d\}, P, S)$ , kde

$$P = \{ \begin{array}{l} S \rightarrow aAa, \\ S \rightarrow bBb, \\ S \rightarrow cCc, \\ S \rightarrow dCd, \\ A \rightarrow a \quad | \quad aA, \\ B \rightarrow bA \quad | \quad Cc, \\ C \rightarrow cA \end{array} \}$$

### 5.3 Zkonstruujte LR(0) analyzátoři a analyzujte libovolné slovo

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

$$P = \{ \begin{array}{l} S \rightarrow aAa, \\ S \rightarrow bAb, \\ S \rightarrow cCc, \\ S \rightarrow dCd, \\ A \rightarrow B, \\ B \rightarrow b, \\ C \rightarrow D, \\ D \rightarrow d \end{array} \}$$

## SLR(k) analyzátoři

### 5.4 Zkonstruujte SLR(1) a SLR(2) analyzátoři

$$G = (\{S, A, B\}, \{a, b\}, P, S), \text{ kde}$$

$$P = \{ S \rightarrow AB,$$

$$A \rightarrow aAb,$$

$$A \rightarrow \varepsilon,$$

$$B \rightarrow bB,$$

$$B \rightarrow b \}$$

a analyzujte slovo aabbbb a aaabbb

**5.5** Najděte všechny  $SLR(2)$  konflikty

$$G = (\{S, P, R\}, \{a, b, c, d\}, P, S), \text{ kde}$$

$$P = \{ S \rightarrow PP,$$

$$P \rightarrow Ra,$$

$$P \rightarrow bRc,$$

$$P \rightarrow dc,$$

$$P \rightarrow bda,$$

$$R \rightarrow d \}$$

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

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

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

$$P = \{ X \rightarrow S,$$

$$S \rightarrow L = R,$$

$$S \rightarrow R,$$

$$L \rightarrow id,$$

$$L \rightarrow *R,$$

$$R \rightarrow L \}$$

**5.8** Zkonstruujte  $SLR(1)$  analyzátor:

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

$$P = \{ X \rightarrow S,$$

$$S \rightarrow S + A,$$

$$S \rightarrow A,$$

$$A \rightarrow (S),$$

$$A \rightarrow a(S),$$

$$A \rightarrow a \}$$

## $LR(0)$ 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  $SLR(1)$

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

$$P = \{ X \rightarrow S,$$

$$S \rightarrow S + A,$$

$$S \rightarrow A,$$

$$A \rightarrow (S),$$

$$A \rightarrow a(S),$$

$$A \rightarrow a \}$$

**5.11** Naleznete co možná nejjednodušší gramatiku takovou, že

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

$$P = \{ \begin{array}{l} X \rightarrow S, \\ S \rightarrow aB, \\ S \rightarrow a, \\ B \rightarrow bSb \end{array} \}$$

# LR(k) a LALR(k) analyzátoři

**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

$$P = \{ \begin{array}{l} X \rightarrow S, \\ S \rightarrow aAd, \\ S \rightarrow bBd, \\ S \rightarrow aBe, \\ S \rightarrow bAe, \\ A \rightarrow c, \\ B \rightarrow c \end{array} \}$$

**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

$$P = \{ \begin{array}{l} X \rightarrow S, \\ S \rightarrow AB, \\ A \rightarrow a, \\ B \rightarrow CD, \\ B \rightarrow aE, \\ C \rightarrow ab, \\ D \rightarrow bb, \\ E \rightarrow bba \end{array} \}$$

**6.3** Zkonstruujte  $LALR(1)$  analyzátoři

$G = (\{X, S, A, B\}, \{a, b, c\}, P, X)$ , kde

$$P = \{ \begin{array}{l} X \rightarrow S, \\ S \rightarrow aAb, \\ S \rightarrow c, \\ S \rightarrow cB, \\ A \rightarrow bS, \\ A \rightarrow Bc, \\ B \rightarrow c \end{array} \}$$

**6.4** Zkonstruujte iniciální stav  $LR(2)$  automatu

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

$$P = \{ \begin{array}{l} S \rightarrow R, \\ R \rightarrow RR, \\ R \rightarrow R + R, \\ R \rightarrow a \end{array} \}$$

**6.5** Proveděte  $LR(1)$  analýzu

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

$$P = \{ \begin{array}{l} X \rightarrow S, \\ S \rightarrow aAS, \\ S \rightarrow \varepsilon, \\ A \rightarrow bb \end{array} \}$$

**6.6** Zkonstruujte analyzátor a analyzujte slova 00111 a 11110.

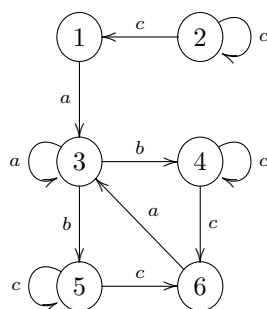
$G = (\{S, A, B\}, \{0, 1\}, P, S)$ , kde

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

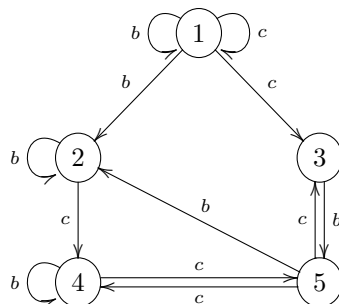


# Bisimulace

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 systém najděte maximální bisimulaci metodou postupných aproximací.

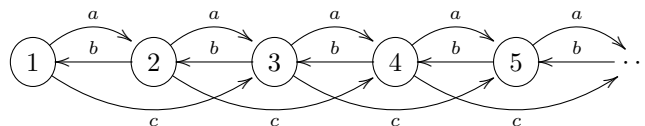


7.3 Je dán přechodový systém  $P_1$  (nekonečně stavový). Zkonstruujte přechodový systém  $P_2$  takový, aby platilo:

- (i)  $P_2$  má stav  $I$  takový, že  $I \sim 1$
- (ii)  $P_2$  je konečně stavový

Jaká je maximální bisimulace pro  $P_1$ ?

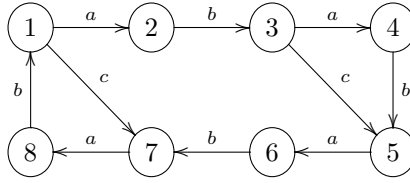
$P_1$ :



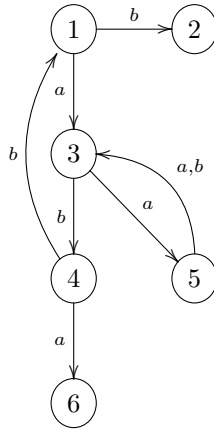
7.4 Najděte konečné automaty  $A_1, A_2$  bez  $\varepsilon$ -přechodů takové, aby splňovaly všechny tři následující podmínky:

- (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

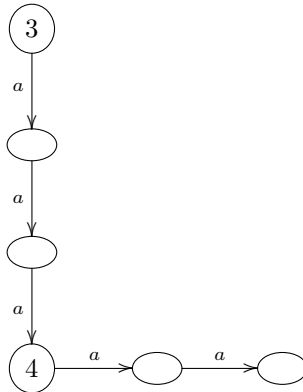
7.5 Dokažte nebo vyvráťte:  $2 \sim 8$ .  
Najděte maximální bisimulaci.



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 Najděte přechodový systém, který je určen následující BPA algebrou.

$$\begin{aligned} X &\xrightarrow{a} XBB \\ X &\xrightarrow{c} \varepsilon \\ B &\xrightarrow{a} BBB \\ B &\xrightarrow{b} \varepsilon \end{aligned}$$

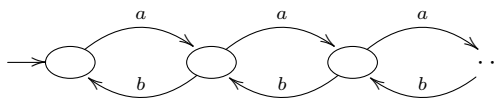
## 8.3 Jaký jazyk generuje následující BPA (z proměnné $X$ )?

$$\begin{aligned} X &\xrightarrow{a} XA \\ X &\xrightarrow{b} XB \\ X &\xrightarrow{c} \varepsilon \\ A &\xrightarrow{a} \varepsilon \\ B &\xrightarrow{b} \varepsilon \end{aligned}$$

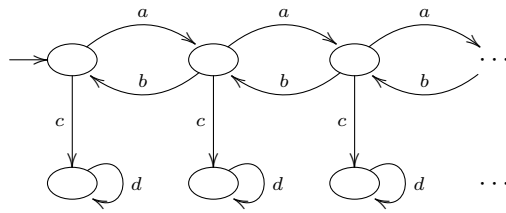
## 8.4 Nakreslete přechodový systém určený BPP algebrou

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

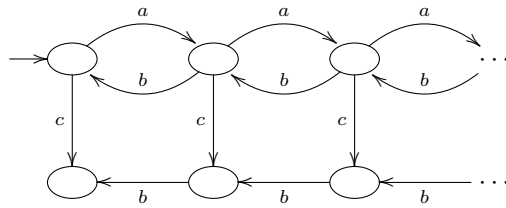
## 8.5 Vyjádřete daný přechodový systém BPA i BPP syntaxí:



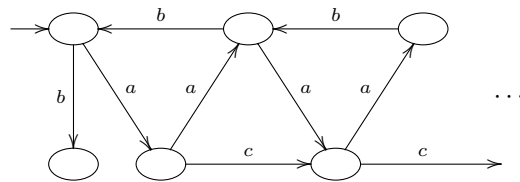
8.6 Vyjádřete daný přechodový systém BPA syntaxí:



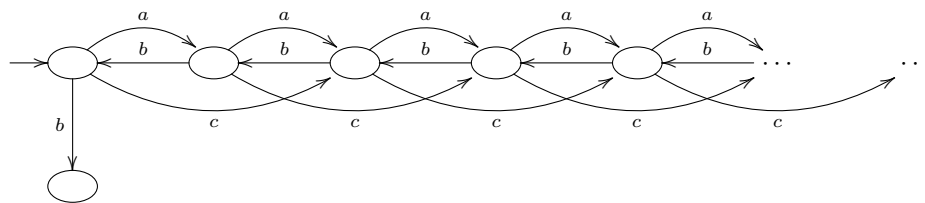
8.7 Vyjádřete daný přechodový systém BPP syntaxí:



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 pravidlo SUBL) nebo  $\gamma\alpha = \beta$  (použijeme pravidlo SUBR), případně  $\alpha = \beta$  (použijeme pravidlo 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 neúspěšný. 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.  $\alpha = \alpha$
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

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

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

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

$$\text{kde } f : \{1, \dots, m\} \rightarrow \{1, \dots, n\} \\ g : \{1, \dots, n\} \rightarrow \{1, \dots, m\} \\ m, n \geq 1$$

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

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

**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** Navrhňte nedeterministický BA pro jazyk všech  $\omega$ -slov nad abecedou  $\Sigma = \{a, b, c\}$ , která obsahuje nekonečný počet symbolů  $a$ .

**10.2** Navrhňte nedeterministický BA pro jazyk všech  $\omega$ -slov nad abecedou  $\Sigma = \{a, b, c\}$ , která obsahuje nekonečný počet symbolů  $b$  a  $c$ .

**10.3** Navrhňte nedeterministický BA pro jazyk všech  $\omega$ -slov nad abecedou  $\Sigma = \{a, b, c\}$ , která obsahuje 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  $c$ .

**10.4** Navrhňte nedeterministický BA pro jazyk  
 $L = \{w = w_1w_2w_3 \dots \mid w_i \in \{a, b\} \text{ pro } i \geq 1, \exists \text{ nekonečně mnoho } j \in \mathbb{N} \text{ takových, že } w_j = w_{j+4}\}$

**10.5** Nechť  $\Sigma = \{0, 1, \#\}$ . Pro slovo  $w = a_0a_1a_2 \dots \in \Sigma^\omega$  a dvě čísla  $i, j \in \mathbb{N}$ ,  $0 \leq i \leq j$  označme  $w[i, j]$  podslovo  $a_i \dots a_j$ . Pro pevně zvolené  $n \in \mathbb{N}$  definujme jazyk:

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

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

**10.6** Dokažte, nebo vyvráťte: Ke každému NBA  $A$  lze zkonstruovat NBA  $A'$  s jediným počátečním stavem.

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

- zkonstruujte nedeterministický BA
- diskutujte, zda je možné sestrojít pro daný jazyk NBA s jedním koncovým stavem
- dokažte, že pro daný jazyk nelze sestrojít NBA s jedním koncovým stavem

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

- $L = \{w \mid \text{inf}(w) = \{a\}\}$
- $L = \{w \mid \text{inf}(w) = \{a, b\}\}$
- $L = \{w \mid a \in \text{inf}(w)\}$
- $L = \{w \mid a \in \text{inf}(w) \wedge c \notin \text{inf}(w)\}$
- $L = \{w \mid \{a, b\} \subseteq \text{inf}(w) \wedge d \notin \text{inf}(w)\}$
- $L = \{w \mid \{a, b\} \subsetneq \text{inf}(w)\}$

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

- $(L(A))^\omega = L_\omega(A)$
- $(L(A))^\omega \neq L_\omega(A)$

**10.10** Pro jazyk  $L = \{w \in \{a, b\}^\omega \mid \text{inf}(w) = \{a, b\}\}$  navrhňte deterministický Mullerův automat, deterministický zobecněný Büchiho automat a deterministický Büchiho automat.



# Logika MSO

**11.1** Definujte základní syntax logiky MSO.

**11.2** Buď  $\Sigma$  abeceda a  $w \in \Sigma^*$  konečné slovo, buď  $\varphi$  sentence (formule bez volných proměnných) logiky MSO. Definujte, kdy  $w \models \varphi$ .

**11.3** Definujte následující syntaktické zkratky:

- a)  $x \leq y, \quad x \geq y, \quad x = y$
- b)  $\forall x \in X : \varphi$
- c)  $\exists x \in X : \varphi$
- d)  $first(x)$  —  $x$  je první pozice v neprázdném slově.
- e)  $last(x)$  —  $x$  je poslední pozice v neprázdném slově.
- f)  $succ(x, y)$  —  $x < y$  a pozice  $x$  a  $y$  bezprostředně sousedí (též značeno jako  $x <_1 y$ ).

**11.4** Uveďte, jak je sentencí MSO definován jazyk.

**11.5** Nechť  $\Sigma = \{a, b\}$ . Uveďte sentence logiky MSO, které definují následující regulární jazyky:

- a)  $\emptyset$
- b)  $\{\varepsilon\}$
- c)  $\{a\}$
- d)  $\{a\}^*$
- e)  $\{a\}^+$
- f)  $\{a\}^* \cdot \{b\}$
- g)  $\{w \in \Sigma^* \mid |w| = 3k \wedge k \geq 0\}$

**11.6** Popište jazyky nad abecedou  $\{a, b\}$ , které jsou dány následujícími sentencemi logiky MSO:

- a)  $\exists x \exists y \forall z : (x \leq z \wedge x < y \wedge (z < y \implies x = z) \wedge Q_a(y))$
- b)  $\exists X : [\forall x, y : (succ(x, y) \implies (x \in X \iff y \notin X)) \wedge \wedge \forall x : (x \in X \implies (Q_a(x) \wedge \neg first(x)))]$

**11.7** Jaká třída jazyků je popsateľná logikou MSO?

**11.8** Jaká třída jazyků je popsateľná logikou prvního řádu?

# Plán cvičení

## Cvičení 1 — FIRST, FOLLOW, SLL(k)

- Zopakujte, jak funguje nedeterministická analýza metodou shora dolů; v čem tkví nedeterminismus metody, jak ho lze řešit. (Motivace pro FIRST a FOLLOW.)
- Definujte funkci FIRST a na jednoduchých příkladech ukažte, co počítá.
  - Ukažte intuitivní výpočet hodnoty funkce FIRST "rozbalováním".
  - Vysvětlete operátor  $\oplus_k$  a uveďte, že  $FIRST_k(ABC) = FIRST_k(A) \oplus_k FIRST_k(B) \oplus_k FIRST_k(C)$ .
  - Pozor na  $A \rightarrow aA \mid b$  versus  $A \rightarrow aA$
- Definujte funkce FOLLOW a na jednoduchých příkladech ukažte, co počítá.
  - Ukažte problém, kdy pravá strana pravidla není dostatečně "dlouhá" pro výpočet hodnoty FOLLOW.
  - Vysvětlete, jak se sestavují rovnice definující hodnoty FOLLOW.
  - Připomeňte, že  $\varepsilon \in FOLLOW_k(S)$
- Procvičujte funkce FIRST a FOLLOW na příkladu 1.4
- Definujte SLL(k)
- Řešte příklad 2.1
- Řešte příklad 2.3

## Cvičení 2 — LL(k) a Transformace na SLL(1)

- Připomenutí definice LL(k).
- Demonstrace rozdílu mezi LL(k) a SLL(k) na příkladu 3.1
- Příklad 3.2
- Sestrojte analyzátor ke gramatice z příkladu 3.2, nebo řešte příklad 3.6
- Vysvětlete konflikty FI-FI a FI-FO.
- Zmínit odstranění levé rekurze (konflikt FI-FI) a odkázat na předchozí kurz.
- Odstranění konfliktu FI-FI
  - Rohová substituce  
(Vysvětlete rozdíl rohová versus obecná substituce.)
  - Levá faktorizace

- Vymyslete příklad gramatiky, na které demonstujete výše uvedené.
- Převod FI-FO na FI-FI
  - Pohlčení terminálního symbolu
  - Extrakce pravého kontextu
  - Vymyslete příklad gramatiky, na které demonstujete výše uvedené.
- Demonstujete, že algoritmické řešení nevede vždy k cíli.
- Příklad 4.6
- D.Ú. Příklad 3.9

### Cvičení 3 — LR(0) a SLR(k)

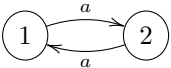
- Připomenutí principu analýzy zdola nahoru.
  - V čem je tato analýza nedeterministická?
  - Nástroje pro řešení nedeterminismu:
    - \* Nahlížení na  $k$ -symbolů na vstupu.
    - \* Analýza zásobníku (Charakteristický automat)
- Příklad 5.1
  - Sestrojení charakteristického automatu
  - Definice akcí automatu dle jednotlivých položek
    - \* čtení: položka s terminálem za tečkou
    - \* redukce: položka s tečkou na konci
  - LR(0) konflikty
  - Přepis do tabulky analyzátoru (striktně oddělujte GOTO a ACTION část tabulky!)
  - Analýza vybraného slova.
- Příklad 5.4
  - Konstruuje přímo SLR(2)
  - Po úspěšné konstrukci rozhodněte a odůvodněte SLR(1) a LR(0)
- Ve zbývajícím čase nebo na doma řešte příklad 5.11

### Cvičení 4 — LR(k) a LALR(k)

- Příklad 6.1
  - Počítejte jako LR(1)
  - Poté diskutujte LR(0) a SLR(1) (zde je demonstrován rozdíl SLR(k) a LR(k))
  - Motivace pro LALR(k)
- Příklad 6.3
  - Diskutujte, jaké LALR konflikty mohou vzniknout pro  $k = 1$ .
  - LALR není to, že sloučím stavy, které nevytvoří sloučením konflikt.

- Příklad 6.4
  - Demontrace, že je třeba poctivě počítat uzávěr z každé položky.
- Diskuze o vztazích  $SLR(k) \Rightarrow LALR(k) \Rightarrow LR(k)$ .
- Obměna předchozího.

## Cvičení 5

- Připomněte se definici relace bisimulace
- Příklad 7.1
  - Bisimulační hrou určete ekvivalentní stavy.
- Příklad 7.2
- Příklad 7.3
- Na systému  ukažte
  - různé relace bisimulace.
  - relace bisimulace, které nejsou relacemi ekvivalence.
  - Ukažte, jak intuitivně počítat bisimulační uzávěr pro vybranou dvojici (např. (1,2)).
- Příklad 8.2
- Příklad 8.4
- Příklad 8.5
- D.Ú.: Ostatní příklady z kapitoly 8.

## Cvičení 6 — Büchi automaty a MSO Logika

- Na příkladu  $\{w \in \{a, b\}^\omega \mid w \text{ má nekonečně mnoho } a\}$  vysvětlete princip Büchi automatů.
- Příklad 10.1
- Příklad 10.2
- Příklad 10.3
- Příklad 10.6
- Příklad 10.7 — čárka před nebo indikuje vylučovací poměr!
- Příklad 11.1
- Příklad 11.2
- Příklad 11.3
- Příklad 11.4
- Příklad 11.5
- Příklad 11.6
- Příklad 11.7
- Příklad 11.8

# Řešení některých příkladů

**1.2** Výpočet  $FI_2(A)$  provedeme z pedagogických důvodů zcela algoritmicky. Nejprve sestavíme příslušné rovnice:

$$\begin{aligned} FI_2(A) &= FI_2(B) \oplus_2 \{c\} \cup \{a\} \\ FI_2(B) &= FI_2(A) \cup FI_2(C) \cup \{d\} \\ FI_2(C) &= FI_2(B) \cup \{d\} \end{aligned}$$

Nyní spočítáme pevný bod rovnic tak, že v každé iteraci používáme striktně hodnoty z předchozí iterace. Poslední sloupec obsahuje výsledné hodnoty.

	0	1	2	3	4	5	6
A	$\emptyset$	$\{a\}$	$\{dc, a\}$	$\{dc, ac, a\}$	$\{dc, ac, a\}$	$\{dc, ac, a\}$	$\{dc, ac, a\}$
B	$\emptyset$	$\{d\}$	$\{a, d\}$	$\{dc, a, d\}$	$\{dc, ac, a, d\}$	$\{dc, ac, a, d\}$	$\{dc, ac, a, d\}$
C	$\emptyset$	$\{d\}$	$\{d\}$	$\{a, d\}$	$\{dc, a, d\}$	$\{dc, ac, a, d\}$	$\{dc, ac, a, d\}$

Pro zajímavost ukážeme, že pokud použijeme v každé iteraci nejnovější známe hodnoty, výpočet se může výrazně urychlit. Výsledek je však shodný.

	0	1	2	3
A	$\emptyset$	$\{a\}$	$\{ac, dc, a\}$	$\{ac, dc, a\}$
B	$\emptyset$	$\{a, d\}$	$\{ac, dc, a, d\}$	$\{ac, dc, a, d\}$
C	$\emptyset$	$\{a, d\}$	$\{ac, dc, a, d\}$	$\{ac, dc, a, d\}$

$$\begin{aligned} FI_2(A) &= \{ac, dc, a\} \\ FI_3(Ae) &= \{ae, dce, ace, dcc, acc\} \end{aligned}$$

**1.3**  $FI_2(A) = \{b, ba\}$ ,  $FI_3(A) = \{b, ba, baa\}$

**1.4**  $FI_1(BBb) = \{a, b\}$

$$FI_2(BBb) = \{ad, aa, ab, b\}$$

$$FO_1(A) = \{c\}$$

$$FO_1(S) = FO_1(B) = FO_1(C) = \{e, d, \varepsilon\}$$

$$FO_3(A) = \{cde, cec, c\}$$

$$FO_3(S) = FO_3(C) = \{ecd, ece, dec, ec, \varepsilon\}$$

$$FI_1(SAcB) = \{a, b, c\}$$

$$FI_4(SAcB) = \{aaaa, aaab, aaac, aaba, aabd, aabe, aac, aaca, aacb, aacc, abaa, abab, abac, abad, abae, abde, abec, ac, aca, acaa, acab, acac, acad, acba, acbd, acbe, acc, acca, adaa, adab, adac, adba, adbd, adbe, adc, adca, baaa, baab, baac, baba, babd, babe, bacd, bace, badd, bade, bdec, bec, beca, c, ca, cad\}$$

**1.5**  $FO_1(S) = FO_1(D) = \{\varepsilon, a, c, d, y\}$

$$FO_1(A) = \{a, b, c, d, x\}$$

$$FO_1(B) = \{\varepsilon, a, b, c, d, y\}$$

$$FO_1(C) = \{\varepsilon, a, b, c, d, y, z\}$$

$$FO_2(S) = \{\varepsilon, da, db, dc, dd, dx, c, ca, cb, cc, cd, cy, aa, ab, ac, ad, ax, y, ya, yb, yc, yd, yy, yz\}$$

**1.6** Pozor, gramatika není normovaná, a tudíž generuje prázdný jazyk.

**2.1** Není  $SLL(2)$ , konflikt nastává na pravidlech  $Y \rightarrow a \mid \varepsilon$ .

**2.2** Není  $SLL(3)$ , konflikt nastává na pravidlech  $A \rightarrow a \mid ab$ .

**2.3** Nejprve pro každé pravidlo tvaru  $A \rightarrow \alpha$  spočítáme množinu  $FI_2(\alpha) \oplus_2 FO_2(A)$ :

1	$S \rightarrow aAaA$	$FI_2(aAaA) \oplus_2 FO_2(S)$	=	$\{aa, ac\}$
2	$S \rightarrow aBaB$	$FI_2(aBaB) \oplus_2 FO_2(S)$	=	$\{ab\}$
3	$A \rightarrow aA$	$FI_2(aA) \oplus_2 FO_2(A)$	=	$\{aa, ac\}$
4	$A \rightarrow c$	$FI_2(c) \oplus_2 FO_2(A)$	=	$\{ca, c\}$
5	$B \rightarrow bD$	$FI_2(bD) \oplus_2 FO_2(B)$	=	$\{bb, ba, b\}$
6	$D \rightarrow bD$	$FI_2(bD) \oplus_2 FO_2(D)$	=	$\{bb, ba, b\}$
7	$D \rightarrow \varepsilon$	$FI_2(\varepsilon) \oplus_2 FO_2(D)$	=	$\{ab, \varepsilon\}$

Nyní snadno zkonstruujeme tabulku přechodové funkce analyzátoru. Prázdná políčka znamenají, že analyzátor v odpovídající situaci vrátí chybu, protože analyzované slovo není generováno grammatikou  $G$ .

	$aa$	$ab$	$ac$	$ba$	$bb$	$bc$	$ca$	$cb$	$cc$	$a$	$b$	$c$	$\varepsilon$
$S$	$aAaA, 1$	$aBaB, 2$	$aAaA, 1$										
$A$	$aA, 3$		$aA, 3$				$c, 4$					$c, 4$	
$B$				$bD, 5$	$bD, 5$						$bD, 5$		
$D$		$\varepsilon, 7$		$bD, 6$	$bD, 6$						$bD, 6$		$\varepsilon, 7$
$a$	ČTI	ČTI	ČTI							ČTI			
$b$				ČTI	ČTI	ČTI					ČTI		
$c$							ČTI	ČTI	ČTI			ČTI	
$\$$													AKC.

Zpravidla se uvádí pouze „zajímavá“ část tabulky, t.j. bez řádků pro terminály a pro  $\$$  a bez sloupců, které by následně zůstaly prázdné:

	$aa$	$ab$	$ac$	$ba$	$bb$	$ca$	$b$	$c$	$\varepsilon$
$S$	$aAaA, 1$	$aBaB, 2$	$aAaA, 1$						
$A$	$aA, 3$		$aA, 3$			$c, 4$		$c, 4$	
$B$				$bD, 5$	$bD, 5$		$bD, 5$		
$D$		$\varepsilon, 7$		$bD, 6$	$bD, 6$		$bD, 6$		$\varepsilon, 7$

Analýza slova  $acaac$ :

$$\begin{aligned}
 & (acaac, S\$, \varepsilon) \quad \vdash \quad (acaac, aAaA\$, 1) \quad \vdash^a \quad (caac, AaA\$, 1) \quad \vdash \\
 & \vdash \quad (caac, caA\$, 14) \quad \vdash^c \quad (aac, aA\$, 14) \quad \vdash^a \quad (ac, A\$, 14) \quad \vdash \\
 & \vdash \quad (ac, aA\$, 143) \quad \vdash^a \quad (c, A\$, 143) \quad \vdash \quad (c, c\$, 1434) \quad \vdash^c \\
 & \vdash^c \quad (\varepsilon, \$, 1434) \quad \Rightarrow \quad \text{akceptuje}
 \end{aligned}$$

Analýza slova  $abaac$ :

$$\begin{aligned}
 & (abaac, S\$, \varepsilon) \quad \vdash \quad (abaac, aBaB\$, 2) \quad \vdash^a \quad (baac, BaB\$, 2) \quad \vdash \\
 & \vdash \quad (baac, bDaB\$, 25) \quad \vdash^b \quad (aac, DaB\$, 25) \quad \Rightarrow \quad \text{zamítá}
 \end{aligned}$$

**2.4** Zajímavá část tabulky  $SLL(3)$  analyzátoru vypadá takto:

	$baa$	$bab$	$aaa$	$aab$	$aba$	$aa$	$ab$	$a$	$b$
$S$	$baAa, 1$	$baBb, 2$							
$A$			$aA, 3$	$aA, 3$	$aB, 4$	$aB, 4$	$aB, 4$		
$B$	$bA, 5$	$bA, 5$						$\varepsilon, 6$	$\varepsilon, 6$

**3.6** Nejprve zkonstruujeme pomocné LL(3) tabulky:

$T_0 = (S, \{\varepsilon\})$	$S \rightarrow aAaB$ $S \rightarrow bAbB$	$aaa, aba$ $bab, bba$	$\{aa, aaa\}, \{\varepsilon\}$ $\{ba, baa\}, \{\varepsilon\}$
$T_1 = (A, \{aa, aaa\})$	$A \rightarrow a$ $A \rightarrow ba$	$aaa$ $baa$	– –
$T_2 = (B, \{\varepsilon\})$	$B \rightarrow aB$ $B \rightarrow a$	$aa, aaa$ $a$	$\{\varepsilon\}$ –
$T_3 = (A, \{ba, baa\})$	$A \rightarrow a$ $A \rightarrow ba$	$aba$ $bab$	– –

Nyní zapíšeme tabulku přechodové funkce analyzátoru. Uvádíme pouze zajímavou část tabulky, tj. řádky popisující situaci, kdy je na vrcholu zásobníku nějaké  $T_i$ , a sloupce, které jsou v těchto řádcích tabulky neprázdné. Zbytek tabulky obsahuje pokyn  $\check{C}TI$ , je-li na vrcholu zásobníku terminál shodný s terminálem na vstupu, a  $AKCEPTUJ$ , je-li na vrcholu zásobníku jeho dno  $\$$  a celý vstup je přečtený (na vstupu je  $\varepsilon$ ). Ve všech ostatních případech (včetně prázdných buněk uvedené části tabulky) analyzátor vrátí chybu, protože analyzované slovo není generováno gramatikou  $G$ .

	$aaa$	$aba$	$bab$	$bba$	$baa$	$aa$	$a$
$T_0$	$aT_1aT_2, 1$	$aT_1aT_2, 1$	$bT_3bT_2, 2$	$bT_3bT_2, 2$			
$T_1$	$a, 3$				$ba, 4$		
$T_2$	$aT_2, 5$					$aT_2, 5$	$a, 6$
$T_3$		$a, 3$	$ba, 4$				

**4.6** Levá faktorizace na tomto příkladu „cyklí“. Jazykově ekvivalentní jednoduchá LL(1) gramatika je například  $(\{S\}, \{a, b, c\}, \{S \rightarrow a \mid b \mid cS\}, S)$ .

**4.10** V této gramatice nelze požadovanou transformací odstranit konflikt. Samozřejmě existuje jiná gramatika, která je s touto jazykově ekvivalentní a je LL(1).

**4.12**  $A \rightarrow [Ba]C, [Ba] \rightarrow a[Ba] \mid a, C \rightarrow c$

**6.3** LR(1):

0	X → .S	ε	S	1
	S → .aAb	ε	a	2
	S → .c	ε	c	3
	S → .cB	ε	c	3
1	X → S.	ε	Accept	
2	S → a.Ab	ε	A	4
	A → .bS	b	b	5
	A → .Bc	b	B	6
	B → .c	c	c	7
3	S → c.	ε	r(S → c)	
	S → c.B	ε	B	8
	B → .c	ε	c	9
4	S → aA.b	ε	b	10
5	A → b.S	b	S	11
	S → .aAb	b	a	12
	S → .c	b	c	13
	S → .cB	b	c	13
6	A → B.c	b	c	14

7	B → c.	c	r(B → c)	
8	S → cB.	ε	r(S → cB)	
9	B → c.	ε	r(B → c)	
10	S → aAb.	ε	r(S → aAb)	
11	A → bS.	b	r(A → bS)	
12	S → a.Ab	b	A	15
	A → .bS	b	b	5
	A → .Bc	b	B	6
	B → .c	c	c	7
13	S → c.	b	r(S → c)	
	S → c.B	b	B	16
	B → .c	b	c	17
14	A → Bc.	b	r(A → Bc)	
15	S → aA.b	b	b	18
16	S → cB.	b	r(S → cB)	
17	B → c.	b	r(B → c)	
18	S → aAb.	b	r(S → aAb)	

LALR(1): Položkový automat LALR(1) analyzátoru vznikne z položkového automatu LR(1) analyzátoru spojením stavů 2,12 a 3,13 a 4,15 a 7,9,17 a 8,16 a 10,18. Každý nově vzniklý stav jsme označili nejnižším číslem z původních stavů. Položkový automat tedy bude vypadat takto:

0	X → .S	ε	S	1
	S → .aAb	ε	a	2
	S → .c	ε	c	3
	S → .cB	ε	c	3
1	X → S.	ε	Accept	
2	S → a.Ab	ε, b	A	4
	A → .bS	b	b	5
	A → .Bc	b	B	6
	B → .c	c	c	7
3	S → c.	ε, b	r(S → c)	
	S → c.B	ε, b	B	8
	B → .c	ε, b	c	7

4	S → aA.b	ε, b	b	10
5	A → b.S	b	S	11
	S → .aAb	b	a	2
	S → .c	b	c	3
	S → .cB	b	c	3
6	A → B.c	b	c	14
7	B → c.	ε, b, c	r(B → c)	
8	S → cB.	ε, b	r(S → cB)	
10	S → aAb.	ε, b	r(S → aAb)	
11	A → bS.	b	r(A → bS)	
14	A → Bc.	b	r(A → Bc)	

**11.1**  $\varphi ::= Q_a(x) \mid x < y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \varphi \mid \exists X \varphi$ , kde  $a \in \Sigma$  je libovolný znak z abecedy,  $x, y$  reprezentují proměnné prvního řádu a  $X$  reprezentuje proměnnou druhého řádu.

- 11.3** a)  $\neg(y < x), \quad y \leq x, \quad \neg(x < y \vee y < x)$   
b)  $\forall x (x \in X \implies \varphi)$ , kde  $\forall x \psi \equiv \neg(\exists x \neg\psi)$   
c)  $\exists x (x \in X \wedge \varphi)$   
d)  $\forall y (x \leq y)$   
e)  $\forall y (y \leq x)$   
f)  $(x < y) \wedge \neg\exists z (x < z \wedge z < y)$

- 11.5** a)  $\exists x : (Q_a(x) \wedge Q_b(x))$   
b)  $\forall x : \neg first(x)$   
c)  $\forall x : (first(x) \wedge Q_a(x)) \wedge \exists y : first(y)$   
d)  $\forall x : Q_a(x)$   
e)  $\forall x : Q_a(x) \wedge \exists y : Q_a(y)$



$$\text{f) } \exists x : (Q_b(x) \wedge \text{last}(x) \wedge \forall z : (z < x \implies Q_a(z)))$$

$$\text{g) } \exists X : (\forall x : (\text{first}(x) \implies x \in X) \wedge$$

$$\wedge \forall x \in X : \exists y, z \notin X : (\text{succ}(x, y) \wedge \text{succ}(y, z) \wedge (\text{last}(z) \vee \exists v \in X : \text{succ}(z, v))))$$

**11.6** a)  $\{a, b\} \cdot \{a\} \cdot \{a, b\}^*$ , tj. na druhé pozici je a; b)  $\{w \in \Sigma^* \mid w \text{ má na každé sudé pozici písmeno a}\}$