T-79.1001 **Syksy 2006**
Introduction to Theoretical Computer Science (T)
Session 8
Answers to demonstration exercises

4. **Problem**: Prove that the class of context-free languages is not closed under intersections and complements. (*Hint:* Represent the language $\{a^k b^k c^k \mid k \geq 0\}$ as the intersection of two context-free languages.)

   Solution: Let $L = \{a^k b^k c^k \mid k \geq 0\}$. This language has been proven to be not context-free. We can prove that context-free languages are not closed under intersection by finding two context-free languages L_1 and L_2 such that $L = L_1 \cap L_2$. Languages $L_1 = \{a^i b^k c^k \mid i, k \geq 0\}$ and $L_2 = \{a^k b^k c^i \mid i, k \geq 0\}$ fulfill this condition.

   A direct corollary is that the class of context-free languages cannot be closed under complementation, either, since they are closed under union and $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

   Finally, we prove that L_1 and L_2 are context-free by presenting context-free grammars that generate them. The language L_1 is generated by $G_1 = (\{S, A, B, a, b, c\}, \{a, b, c\}, P_1, S)$, where $P_1 = \{S \to AB, A \to aA \mid \varepsilon, B \to bBc \mid \varepsilon\}$. Similarly, L_2 is generated by $G_2 = (\{S, A, B, a, b, c\}, \{a, b, c\}, P_2, S)$, $P_2 = \{S \to AB, A \to aAb \mid \varepsilon, B \to cB \mid \varepsilon\}$.

5. **Problem**: Design a pushdown automaton corresponding to the grammar $G = (V, \Sigma, P, S)$, where

$$
\begin{aligned}
V &= \{S, (,), {}^*, \cup, \emptyset, a, b\} \\
\Sigma &= \{(,), {}^*, \cup, \emptyset, a, b\} \\
P &= \{S \to (SS), S \to S^*, S \to (S \cup S), \\
&\qquad S \to \emptyset, S \to a, S \to b\}
\end{aligned}
$$

   Solution: For any context-free grammar $G = (V, \Sigma, R, S)$, the corresponding nondeterministic pushdown automaton $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ can be formed as follows:

$$
\begin{aligned}
Q &= \{q_0, q_1, q_{\text{acc}}\} \\
\Gamma &= V \cup \{\bot\} \\
F &= \{q_{\text{acc}}\} \\
\delta &= \{\big((q_0, \varepsilon, \varepsilon), (q_1, S\bot)\big), \big((q_1, \varepsilon, \bot), (q_{\text{acc}}, \varepsilon)\big)\} && (1) \\
&\quad \cup \{\big((q_1, \varepsilon, A), (q_1, \alpha)\big) \mid (A \to \alpha \in P\} && (2) \\
&\quad \cup \{\big((q_1, \sigma, \sigma), (q_1, e)\big) \mid \sigma \in \Sigma\} && (3)
\end{aligned}
$$

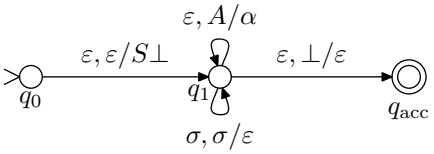
   Here we use a new symbol $\bot$ to denote the bottom of the stack.

   For the grammar given in this exercise, the construction produces the following automa-

ton:

$$
\begin{aligned}
Q =& \{q_0, q_1, q_{\text{acc}}\} \\
\Sigma =& \{(,), ^*, \cup, \emptyset, a, b\} \\
\Gamma =& \{S, (,), ^*, \cup, \emptyset, a, b, \bot\} \\
F =& \{q_{\text{acc}}\} \\
\delta =& \big\{\big((q_0, \varepsilon, \varepsilon), (q_1, S\bot)\big), \big((q_1, \varepsilon, \bot), (q_{\text{acc}}, \varepsilon)\big), \big((q_1, \varepsilon, S), (q_1, (SS))\big), \\
& \big((q_1, \varepsilon, S), (q_1, S^*)\big), \big((q_1, \varepsilon, S), (q_1, (S \cup S))\big), \big((q_1, \varepsilon, S), (q_1, \emptyset)\big), \\
& \big((q_1, \varepsilon, S), (q_1, a)\big), \big((q_1, \varepsilon, S), (q_1, b)\big), \\
& \big((q_1, (, ()), (q_1, \varepsilon)\big), \big((q_1,), ())\big), (q_1, \varepsilon)\big), \big((q_1, ^*, ^*), (q_1, \varepsilon)\big), \\
& \big((q_1, \cup, \cup), (q_1, \varepsilon)\big), \big((q_1, \emptyset, \emptyset), (q_1, \varepsilon)\big), \big((q_1, a, a), (q_1, \varepsilon)\big), \\
& \big((q_1, b, b), (q_1, \varepsilon)\big)\big\}
\end{aligned}
$$

The automaton is of the form:



Let us look at how the automaton handles input $(a \cup b^*)$:

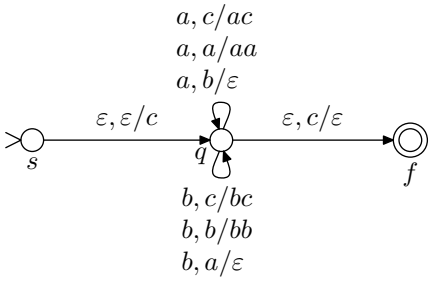
State	Input	Stack	
q_0	$(a \cup b^*)$	ε	
q_1	$(a \cup b^*)$	$S\bot$	(1)
q_1	$(a \cup b^*)$	$(S \cup S)\bot$	(2) $(S \to (S \cup S))$
q_1	$a \cup b^*)$	$S \cup S)\bot$	(3)
q_1	$a \cup b^*)$	$a \cup S)\bot$	(2) $(S \to a)$
q_1	$\cup b^*)$	$\cup S)\bot$	(3)
q_1	$b^*)$	$S^*)\bot$	(2) $(S \to S^*)$
q_1	$b^*)$	$b^*)\bot$	(2) $(S \to b)$
q_1	$^*)$	$^*)\bot$	(3)
q_1	$)$	$)\bot$	(3)
q_1	ε	$\bot$	(3)
q_{acc}	ε	ε	(1)

Note: language $L(G)$ defines all syntactically well-formed regular expressions formed over alphabet $\Sigma = \{a, b\}$.

6. **Problem**: Form the grammar corresponding to the pushdown automaton M, where $M = (Q, \Sigma, \Gamma, \delta, s, F)$:

$$
\begin{aligned}
Q =& \{s, q, f\} \\
\Sigma =& \{a, b\} \\
\Gamma =& \{a, b, c\} \\
F =& \{f\} \\
\delta =& \big\{\big((s, \varepsilon, \varepsilon), (q, c)\big), \big((q, a, c), (q, ac)\big), \big((q, a, a), (q, aa)\big) \\
& \big((q, a, b), (q, \varepsilon)\big), \big((q, b, c), (q, bc)\big), \big((q, b, b), (q, bb)\big) \\
& \big((q, b, a), (q, \varepsilon)\big), \big((q, \varepsilon, c), (f, \varepsilon)\big)\big\}
\end{aligned}
$$

Solution: As a state diagram M look like this::



Determining the context-free grammar corresponding to a given pushdown automaton is a rather tedious task. The algorithm that we use here works only with *simple* pushdown automata that satisfy the following two requirements:

- If $\big((q, u, \beta), (p, \gamma)\big)$ is a transition in the pushdown automaton, then $|\beta| \leq 1$.
- If $\big((q, u, e), (p, \gamma)\big) \in \Delta$, then $\big((q, u, A), (p, \gamma A)\big) \in \Delta$ for all $A \in \Gamma$.

The requirements do not, however, reduce the expressive power of pushdown automata, since every pushdown automaton can be converted into an equivalent simple pushdown automaton (see the book for details).

The goal is to construct a grammar with nonterminals $\langle q, A, p\rangle$, where $q, p \in K$ and $A \in \Gamma \cup \{e\}$. Intuitively, the nonterminal $\langle q, A, p\rangle$ will generate all input strings on which the automaton can move from the state q to the state p while removing the symbol A from the stack.

There are four kinds of grammar rules:

1. For all $f \in F$ there is a rule $S \to \langle s, e, f\rangle$.
2. For all transitions $\big((q, u, A), (r, B_1 \dots B_n)\big) \in \Delta$, where $q, r \in K, u \in \Sigma^*, n > 0, B_1, \dots B_n \in \Gamma$ and $A \in \Gamma \cup \{e\}$, there is a rule

   $$\langle q, A, p\rangle \to u\langle r, B_1, q_1\rangle\langle q_1, B_2, q_2\rangle \dots \langle q_{n-1}, B_n, p\rangle$$

   for all $p, q_1, \dots, q_{n-1} \in K$.
3. For all transitions $\big((q, u, A), (r, e)\big) \in \Delta$, where $q, r \in K, u \in \Sigma^*$ and $A \in \Gamma \cup \{e\}$, there is a rule

   $$\langle q, A, p\rangle \to u\langle r, e, p\rangle$$

4. For all $q \in K$ there is a rule $\langle q, e, q\rangle \to e$.

The first rule encodes the goal to reach some final state from the initial state such that the stack is finally empty. The rules of the last form tell that no computation is needed if the automaton does not change its state. Rules of type 2 represent a sequence of transitions that move the automaton from the state q to the state p while removing the symbol A from the stack. The right side of the rule constructs the transition sequence one transition at a time. Rules of type 3 are analogous to rules of type 2.

Grammar $G = (V, \Sigma, P, S), V = \Sigma \cup \{S\} \cup \{\langle q, A, p\rangle \mid q, p \in K, A \in \Gamma \cup \{e\}\}$

$$
\begin{aligned}
P = \{ & S \to \langle s, e, f\rangle, && (1.)\\
& \langle s, e, s\rangle \to e, \langle q, e, q\rangle \to e, \langle f, e, f\rangle \to e, && (4.)\\
& \langle s, e, s\rangle \to e\langle q, c, s\rangle, && (2./tr.1)\\
& \langle s, e, q\rangle \to e\langle q, c, q\rangle, && (2./tr.1)\\
& \langle s, e, f\rangle \to e\langle q, c, f\rangle, && (2./tr.1)\\
& \langle q, c, s\rangle \to a\langle q, a, s'\rangle\langle s', c, s\rangle && (2./tr.2)\\
& \langle q, c, q\rangle \to a\langle q, a, q'\rangle\langle q', c, q\rangle && (2./tr.2)\\
& \langle q, c, f\rangle \to a\langle q, a, f'\rangle\langle f', c, f\rangle && (2./tr.2)\\
& \langle q, a, s\rangle \to a\langle q, a, s'\rangle\langle s', a, s\rangle && (2./tr.3)\\
& \langle q, a, q\rangle \to a\langle q, a, q'\rangle\langle q', a, q\rangle && (2./tr.3)\\
& \langle q, a, f\rangle \to a\langle q, a, f'\rangle\langle f', a, f\rangle && (2./tr.3)\\
& \langle q, b, s\rangle \to a\langle q, e, s\rangle && (3./tr.4)\\
& \langle q, b, q\rangle \to a\langle q, e, q\rangle && (3./tr.4)\\
& \langle q, b, f\rangle \to a\langle q, e, f\rangle && (3./tr.4)\\
& \langle q, c, s\rangle \to b\langle q, b, s'\rangle\langle s', c, s\rangle && (2./tr.5)\\
& \langle q, c, q\rangle \to b\langle q, b, q'\rangle\langle q', c, q\rangle && (2./tr.5)\\
& \langle q, c, f\rangle \to b\langle q, b, f'\rangle\langle f', c, f\rangle && (2./tr.5)\\
& \langle q, b, s\rangle \to b\langle q, b, s'\rangle\langle s', b, s\rangle && (2./tr.6)\\
& \langle q, b, s\rangle \to b\langle q, b, q'\rangle\langle q', b, q\rangle && (2./tr.6)\\
& \langle q, b, s\rangle \to b\langle q, b, f'\rangle\langle f', b, f\rangle && (2./tr.6)\\
& \langle q, a, s\rangle \to b\langle q, e, s\rangle && (3./tr.7)\\
& \langle q, a, q\rangle \to b\langle q, e, q\rangle && (3./tr.7)\\
& \langle q, a, f\rangle \to b\langle q, e, f\rangle && (3./tr.7)\\
& \langle q, c, s\rangle \to e\langle f, e, s\rangle && (3./tr.8)\\
& \langle q, c, q\rangle \to e\langle f, e, q\rangle && (3./tr.8)\\
& \langle q, c, f\rangle \to e\langle f, e, f\rangle && (3./tr.8)
\end{aligned}
$$

Many of these rules are redundant. The rules that need to be included in the grammar can be found by starting from the rule $S \to \langle s, e, f\rangle$ and checking which rules can ever be used in a derivation. This results in the following set of rules:

$$
\begin{aligned}
P = \{ & S \to \langle s, e, f\rangle\\
& \langle s, e, f\rangle \to e\langle q, c, f\rangle\\
& \langle q, c, f\rangle \to a\langle q, a, q\rangle\langle q, c, f\rangle\\
& \langle q, c, f\rangle \to b\langle q, b, q\rangle\langle q, c, f\rangle\\
& \langle q, c, f\rangle \to e\langle f, e, f\rangle\\
& \langle q, a, q\rangle \to a\langle q, a, q\rangle\langle q, a, q\rangle\\
& \langle q, a, q\rangle \to b\langle q, e, q\rangle\\
& \langle q, b, q\rangle \to b\langle q, b, q\rangle\langle q, b, q\rangle\\
& \langle q, b, q\rangle \to a\langle q, e, q\rangle\\
& \langle q, e, q\rangle \to e\\
& \langle f, e, f\rangle \to e\}
\end{aligned}
$$

The grammar can still be simplified. Let $\langle q, c, f\rangle = S, \langle q, b, q\rangle = B, \langle q, a, q\rangle = A$. This gives the result

$$\begin{aligned} P = \{&S \to aAS \mid bBS \mid \varepsilon \\ &A \to aAA \mid b, \\ &B \to bBB \mid a\} \end{aligned}$$