Tik-79.148 **Spring 2001**
Introduction to Theoretical Computer Science
Tutorial 7
Solutions to demonstration exercises

1. Let M be the pushdown automaton $M = (K, \Sigma, \Gamma, \Delta, s, F)$, where K, Σ, s and F are defined as for a finite automaton, Γ is the set of stack symbols, and $\Delta \subset \big(K \times (\Sigma \cup \{e\}) \times \Gamma^*\big) \times (K \times \Gamma^*)$ is a finite transition relation.

   A pushdown automaton can be seen as a nondeterministic finite automaton augmented with an infinite stack, which is used as storage during computation. All stack operations refer to the top of the stack.

   The transition $\big((q_1, a, \alpha), (q_2, \beta)\big)$ moves the automaton from the state q_1 to the state q_2 if the current input symbol is a and the symbol currently on the top of the stack is α. The transition removes α from the stack and replaces it with β, which is pushed on the stack.

   The nondeterministic pushdown automaton M accepts a word w (i.e., $w \in L(M)$), if there exists a sequence of transitions from the initial state to some final state such that the stack is empty at the end of computation (i.e., if $(s, w, e) \vdash_M^* (f, e, e)$ for some $f \in F$).

   In the exercise:

   a)

$$\begin{aligned}
M =& (K, \Sigma, \Gamma, \Delta, s, F)\\
K =& \{s, f\}\\
F =& \{f\}\\
\Sigma =& \{a, b\}\\
\Gamma =& \{c, d\}\\
\Delta =& \big\{\big((s, a, e), (s, cd)\big), \big((s, e, e), (f, e)\big),\\
& \big((f, b, c), (f, e)\big), \big((f, b, d), (f, e)\big)\\
& \big((f, e, d), (f, e)\big)\big\}
\end{aligned}$$

   The automaton first reads all symbols a and (in effect) stores their number on the stack. Because n may vary in the interval $m \leq n \leq 2m$, two different stack symbols are used. The automaton checks that for each c on the stack there is a corresponding b in the input. However, the d's can be removed from the stack even without reading any input.

b)

$$
\begin{aligned}
M =& (K, \Sigma, \Gamma, \Delta, s, F) \\
K =& \{s, f\} \\
F =& \{f\} \\
\Sigma =& \{a, b\} \\
\Gamma =& \{a, b\} \\
\Delta =& \big\{\big((s, a, e), (s, a)\big), \big((s, b, e), (s, b)\big), \\
& \big((s, a, e), (f, e)\big), \big((s, b, e), (f, e)\big), \\
& \big((s, e, e), (f, e)\big), \big((f, a, a), (f, e)\big), \\
& \big((f, b, b), (f, e)\big)\big\}
\end{aligned}
$$

The automaton operates in two stages:

i. The automaton first reads one half of the word and uses the stack to store the symbols that have been read.

ii. The automaton "guesses" nondeterministically when one half of the word has been processed. After this the automaton removes symbols from the stack one by one and checks that the stack symbols match the symbols remaining in the input.

2. Let $M = (K, \Sigma, \Gamma, \Delta, s, F)$ be a pushdown automaton. Define the language

$$L_f(M) = \{w \in \Sigma^* \mid (s, w, e) \vdash_M^* (f, e, \alpha) \text{ for some } f \in F, \alpha \in \Gamma^*\}$$

The language $L(M)$ consists of words on which the pushdown automaton has an execution ending in some final state such that all input has been processed and the stack is empty. $L_f(m)$ adds to this language those words for which the stack is not empty at the end of computation. The goal of this exercise is to show that $L(M)$ and $L_f(M)$ are equally expressive.

a) First, we have to show that for every pushdown automaton M there exists a pushdown automaton M' such that $L(M') = L_f(M)$.
M' can be constructed by augmenting M with a new final state, which is used to empty the stack at the end of computation.

$$
\begin{aligned}
M' =& (K', \Sigma, \Gamma, \Delta', s, F') \\
K' =& K \cup \{f'\}, f' \notin K \\
F' =& \{f'\} \\
\Delta' =& \Delta \cup \big\{\big((f, e, e), (f', e)\big) \mid f \in F\big\} \\
& \cup \big\{\big((f', e, \alpha), (f', e)\big) \mid \alpha \in \Gamma\big\}
\end{aligned}
$$

Now, if $w \in L_f(M)$, then $(s, w, e) \vdash_M^* (f, e, \alpha)$ for some $\alpha \in \Gamma^*$ and $f \in F$. Then (since M' includes all M's transitions) $(s, w, e) \vdash_{M'}^* (f, e, \alpha) \vdash_{M'} (f', e, \alpha) \vdash_{M'}^* (f', e, e)$, and thus $w \in L(M')$.
For the opposite direction, if $w \in L(M')$, then $(s, w, e) \vdash_{M'}^* (f, e, \alpha) \vdash_{M'}^* (f', e, e)$, and therefore $(s, w, e) \vdash_M^* (f, e, \alpha)$, i.e., $w \in L_f(M)$.

b) We then have to show that there is a pushdown automaton M'' such that $L_f(M'') = L(M)$. One possible way to construct M'' is to push a symbol γ on the bottom of the stack in the beginning of the execution (where γ is a symbol not in Γ), and then operate just as M would operate. Now, each input word $w \in L(M)$ leads the automaton to the configuration (f'', e, γ), so $w \in L_f(M'')$. More formally,

$$\begin{aligned} M'' =& (K'', \Sigma, \Gamma'', \Delta'', s'', F'') \\ K'' =& K \cup \{f'', s''\}, f'', s'' \notin K \\ \Gamma'' =& \Gamma \cup \{\gamma\}, \gamma \notin \Gamma \\ F'' =& \{f''\} \\ \Delta'' =& \Delta \cup \big\{\big((s'', e, e), (s, \gamma)\big), \big((f, e, \gamma), (f'', \gamma)\big) \mid f \in F\big\} \end{aligned}$$

If $w \in L(M)$, i.e., if $(s, w, e) \vdash_M^* (f, e, e)$ for some $f \in F$, then also $(s'', w, e) \vdash_{M''} (s, w, \gamma) \vdash_{M''}^* (f, e, \gamma) \vdash_{M''} (f'', e, \gamma)$, and thus $w \in L_f(M'')$.

For the opposite direction, if $w \in L_f(M'')$, i.e., if $(s'', w, e) \vdash_{M''} (s, w, \gamma) \vdash_{M''}^* (f, e, \gamma) \vdash_{M''} (f'', e, \gamma)$, then also $(s, w, e) \vdash_M (f, e, e)$, i.e., $w \in L(M)$.

The proof relies on the fact that $\gamma \notin \Gamma$. This ensures that γ will not be used in any transition in Δ, so there is exactly one γ in the stack at each step of the computation.

3. Determining the context-free grammar corresponding to a given pushdown automaton is a rather tedious task. The first edition of the course textbook considers only **simple** pushdown automata that satisfy the 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).

   In the second edition, these requirements are not given explicitly; instead, they are implicitly included in the grammar construction algorithm. This presentation follows the approach taken in the first edition of the textbook.

   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.

The given pushdown automaton is

$$\begin{aligned}
M =&(K, \Sigma, \Gamma, \Delta, s, F)\\
K =&\{s, q, f\}\\
\Sigma =&\{a, b\}\\
\Gamma =&\{a, b, c\}\\
F =&\{f\}\\
\Delta =&\big\{\big((s, e, e), (q, c)\big), \big((q, a, c), (q, ac)\big), \big((q, a, a), (q, aa)\big)\\
&\big((q, a, b), (q, e)\big), \big((q, b, c), (q, bc)\big), \big((q, b, b), (q, bb)\big)\\
&\big((q, b, a), (q, e)\big), \big((q, e, c), (f, e)\big)\big\}
\end{aligned}$$

Let G be the grammar $G = (V, \Sigma, R, S)$, where $V = \Sigma \cup \{S\} \cup \{\langle q, A, p\rangle \mid$

$q, p \in K, A \in \Gamma \cup \{e\}\}$, and

$$
\begin{aligned}
R = \{ & 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:

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

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

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