**T-79.1001** **Syksy 2007**
**Introduction to Theoretical Computer Science (T)**
**Session 7**
**Answers to demonstration exercises**

4. **Problem**: *Pattern expressions* are a generalisation of regular expressions used e.g. in some text editing tools of UN*X operating systems. In addition to the usual regular expression constructs, a pattern expression may contain string variables, including the constraint that any two appearances of the same variable must correspond to the same substring. Thus e.g. $abXb^*Xa$ and $aX(a \cup b)^*YX(a \cup b)^*Ya$ are pattern expressions over the alphabet $\{a, b\}$. The first one of these describes the language $\{awb^nwa \mid w \in \{a, b\}^*, n \geq 0\}$. Prove that pattern expressions are a proper generalisation of regular expressions, i.e. that pattern expressions can be used to describe also some nonregular languages.

   **Answer**:

   Consider the pattern expression $XX$. This expression denotes the language $L = \{zz \mid z \in \{a, b\}^*$. Suppose that $L$ is regular. Then, the pumping lemma for regular languages holds for it:

   **Lemma**: *If $L$ is a regular language, then there exists an integer $n > 0$ such that for each string $x \in n$ it holds that if $|x| \geq n$, then $x = uvw$ where (1) $|uv| \leq n$, (2) $|v| > 0$, and (3) $uv^kw \in L$ for every $k \in \mathbb{N}$.*

   Let us examine the string $x = a^nba^nb \in L$. As $|x| = 2n+2 > 0$, there has to be a partition of $x$ into three parts such that all three conditions of the lemma are satisfied.

   All partitions that satisfy (1) are of the form:

   $$u = a^i$$
   $$v = a^j$$
   $$w = a^{n-(i+j)}ba^nb$$

   where $i + j \leq n$. From (2) we know that $j > 0$. Next we examine if we can find some values for $i$ and $j$ such that (3) also holds for $k = 0$:

   $$uv^0w = uw = a^ia^{n-(i+j)}ba^nb = a^{p-j}ba^nb \ .$$

   Since $j > 0$, $p - j < p$ so $uv^0w \notin L$ for any choice of $i$ and $j$. Thus, $L$ is not regular.

   Since we can define $L$ using pattern expressions, we now know that pattern expressions are strictly more expressive than regular expressions.

5. **Problem**: Prove that the language $L = \{w \mid w$ contains equally many $a$'s as $b$'s $\}$ is not regular.

   **Solution**:

   **Lemma**: *If $L$ is a regular language, then there exists an integer $n > 0$ such that for each string $x \in n$ it holds that if $|x| \geq n$, then $x = uvw$ where (1) $|uv| \leq n$, (2) $|v| > 0$, and (3) $uv^kw \in L$ for every $k \in \mathbb{N}$.*

   Consider $x = a^nb^n \in L$. If $L$ is regular, then we can divide $x$ into three parts $u$, $v$, and $w$ such that all three conditions of the lemma hold. All partitions that satisfy (1) are of the form:

   $$u = a^i$$
   $$v = a^j$$
   $$w = a^{n-(i+j)}b^n$$

where $i + j \leq n$. From (2) we know that $j > 0$. Next we examine if we can find some values for $i$ and $j$ such that (3) also holds for $k = 0$:

$$uv^0w = uw = a^i a^{n-(i+j)} ba^n b = a^{p-j} b^n \notin L \ .$$

Since $uv^0w \notin L$ for any $i$ and $j$, $L$ is not regular.

6. **Problem**: Design an algorithm for testing whether a given a context-free grammar $G = (V, \Sigma, P, S)$, generates a nonempty language, i.e. whether any terminal string $x \in \Sigma^*$ can be derived from the start symbol $S$.

   **Solution**:

   The following procedure ?GENERATESNONEMPTYLANGUAGE($G$) takes a context-free grammar $G$ as its input and it returns the value `true`, if the language $L(G)$ is not empty.

   ?GENERATESNONEMPTYLANGUAGE($G = (V, \Sigma, P, S)$: context-free grammar)
   $T \leftarrow \Sigma$
   **repeat** $|V - \Sigma|$ **times**
     **for each** $A \rightarrow X_1 \cdots X_k \in P$
       **if** $A \notin T \wedge X_1 \cdots X_k \in T^k$
         $T \leftarrow T \cup \{A\}$
   **if** $S \in T$
     return **true**
   **else**
     return **false**

   The basic idea is to start from the set $T = \Sigma$ of terminal symbols and then check whether it is possible to "retreat" to $S$ using productions of $P$ reversed. At each step a nonterminal $A$ is added to the set $T$ if there exists some rule for $A$ such that all symbols in the right side belong to $T$. These steps are repeated $|V - \Sigma|$ times.

   To see why $|V - \Sigma|$ steps are enough, let us consider the word $z \in L(G)$ such that $z$ has the smallest parse tree of all words in $L(G)$. If $z$ has has a derivation of the form:

$$S \rightarrow^* uAy \rightarrow^* uvAxy \rightarrow^* uvwxy$$

   where $u, v, w, x, y \in \Sigma^*$, then also $z' = uwy$ can be derived using the rules of the grammar[1]. In that case, the parse tree of $z'$ is smaller than that of $z$ contradicting our earlier assumption. Now we see that in the minimal parse tree of $z$ it is not possible to have two occurrences of a nonterminal $A$ in a single branch so we have to iterate over the set $T$ only as many times as there are nonterminals in the grammar.

   Consider the grammar $G$:

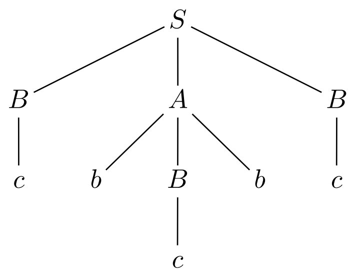
$$\begin{aligned} S &\rightarrow BAB \mid ABA \\ A &\rightarrow aAS \mid bBa \\ B &\rightarrow bBS \mid c \end{aligned}$$

   The computation of $T$ proceeds as follows:

$$\begin{aligned} T_0 &= \{a, b, c\} && \\ T_1 &= \{a, b, c, B\} && (B \rightarrow c) \\ T_2 &= \{a, b, c, A, B\} && (A \rightarrow bBa) \\ T_3 &= \{a, b, c, A, B, C, S\} && (S \rightarrow BAB, S \rightarrow ABA) \end{aligned}$$

   Since $|V - \Sigma| = 3$, the algorithm terminates and $T = T_3$ so $L(G)$ is not empty. The smallest parse-tree of a $z \in L(G)$ is:

[1] Compare this with the pumping theorem of context-free languages.




**Appendix: Chomsky normal form and CYK-algorithm**

Let's change the grammar:

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

into Chomsky normal form, and check with CYK-algorithm whether words *abb* and *abba* belong to language $L(G)$.

A grammar is in Chomsky normal form, if the following conditions are met:

1. Only the initial symbol $S$ can generate an empty string.
2. The initial symbol $S$ does not occur in the right hand side of any rule.
3. All rules are of form $A \to BC$ or $A \to a$ (where $A$, $B$ ja $C$ are nonterminals and $a$ a terminal symbol), except for rule $S \to \varepsilon$ (if such a rule exists).

The grammar is put into the normal form in phases.

1. **Initial symbol is removed from right side of the rules**.
   Because there are rules $S \to aAS$ and $S \to bBS$ in the grammar, let's add a new starting symbol $S'$ and a rule $S' \to S$. The resulting set of rules is

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

2. **$\varepsilon$-productions are removed**.
   Because in the Chomsky normal form only the initial symbol $S'$ may generate $\varepsilon$, other $\varepsilon$ rules must be removed from the grammar. We start by computing the set of erasable nonterminals: NULL:

$$\begin{aligned} \mathrm{NULL}_0 &= \{S\} && (S \to \varepsilon) \\ \mathrm{NULL}_1 &= \{S, S'\} && (S' \to S) \\ \mathrm{NULL}_2 &= \{S, S'\} = \mathrm{NULL} \end{aligned}$$

   Next, the rules $A \to X_1 \cdots X_n$ are replaced by a set of rules

$$A \to \alpha_1 \cdots \alpha_2, \qquad \text{where } \alpha_i = \begin{cases} X_i, X_i \notin \mathrm{NULL} \\ X_i \text{ or } \varepsilon, X_i \in \mathrm{NULL} \end{cases}$$

   Finally, we remove all rules of form $A \to \varepsilon$ (except for rule $S' \to \varepsilon$). As the result we get rule set[2]:

[2] To be exact, now we should add a new initial symbol $S''$ and rules $S'' \to \varepsilon | S'$, but in this case we can use $S'$ as the starting symbol without problems.

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

3. **Unit productions are removed.**
   Next we remove from the grammar all rules of form $A \to B$ where both $A$ and $B$ are nonterminals.
   First, we compute sets $F(A)$ for all $A \in V - \Sigma$:

$$
\begin{aligned}
F(A) = F(B) = F(S) = \emptyset \\
F(S') = \{S\}
\end{aligned}
$$

   Nonterminal $B$ belongs to set $F(A)$ exactly when we can derive $B$ from $A$ using only unit productions:
   Rule $A \to B$ is replaced by $\{A \to w \mid \exists C \in F(B) \cup \{B\} : C \to w \in P\}$. As the result we get a set of rules

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

4. **Too long productions are removed.**
   In the last phase we add into the grammar a new nonterminal $C_\sigma$ and a rule $C_\sigma \to \sigma$ for all $\sigma \in \Sigma$ and divide all rules $A \to w$ ($|w| > 2$) into a chain of rules, all of which consist of exactly two symbols.
   The Chomsky normal form for the given grammar is the following set of rules:

$$
\begin{aligned}
S' &\to C_a S'_1 \mid C_a A \mid C_b S'_2 \mid C_b B \mid \varepsilon \\
S'_1 &\to AS \\
S'_2 &\to BS \\
S &\to C_a S_1 \mid C_a A \mid C_b S_2 \mid C_b B \\
S_1 &\to AS \\
S_2 &\to BS \\
A &\to C_a A_1 \mid b \\
A_1 &\to AA \\
B &\to C_a B_1 \mid a \\
B_1 &\to BB \\
C_a &\to a \\
C_b &\to b
\end{aligned}
$$

Using CYK-algorithm we can check whether word $x = x_1 \cdots x_n$ belongs to the language defined by grammar $G$. During the progress of algorithm we compute nonterminal sets $N_{i,j}$. Set $N_{i,j}$ includes all those nonterminals, which can be used to derive substring $x_i \cdots x_j$. We can apply dynamic programming for computing the sets:

$$
\begin{aligned}
N_{i,i} &= \{A \mid (A \to x_i) \in P\} \\
N_{i,i+k} &= \{A \mid \exists B, C \in V - \Sigma \text{ s. t. } (A \to BC) \in P \text{ and} \\
&\qquad \exists j : i \le j < i + k \text{ s. e } B \in N_{i,j} \wedge C \in N_{j+1,i+k}\}
\end{aligned}
$$

Let's look at the grammar we got above and word *abba*. First we compute sets $N_{i,i}$, $i \leq 4$:

| | | $i \rightarrow$ | | | |
|---|---|---|---|---|---|
| | $N_{i,i+k}$ | $1:a$ | $2:b$ | $3:b$ | $4:a$ |
| $k \downarrow$ | 0 | $\underline{a}bba$ $\{B, C_a\}$ | $a\underline{b}ba$ $\{A, C_b\}$ | $ab\underline{b}a$ $\{B, C_a\}$ | $abb\underline{a}$ $\{A, C_b\}$ |

On each square of the array it has been denoted, which substring the square corresponds to.

Next we compute $N_{1,2}$. Now the only possible $j = 1$, so we look at sets $N_{1,1} = \{B, C_a\}$ ja $N_{2,2} = \{A, C_b\}$. The only rules of form $A \rightarrow BC$, $B \in N_{1,1}$ and $C \in N_{2,2}$, are: $\{S' \rightarrow C_aA, S \rightarrow C_aA\}$, so $N_{1,2} = \{S', S\}$. The same way we can compute sets $N_{2,3} = \{A_1\}$ and $N_{3,4} = \{S', S\}$, so the second row of the array is

| | | $i \rightarrow$ | | | |
|---|---|---|---|---|---|
| | $N_{i,i+k}$ | $1:a$ | $2:b$ | $3:b$ | $4:a$ |
| $k \downarrow$ | 0 | $\underline{a}bba$ $\{B, C_a\}$ | $a\underline{b}ba$ $\{A, C_b\}$ | $ab\underline{b}a$ $\{B, C_a\}$ | $abb\underline{a}$ $\{A, C_b\}$ |
| | 1 | $\underline{ab}ba$ $\{S', S\}$ | $a\underline{bb}a$ $\{A_1\}$ | $ab\underline{ba}$ $\{S', S\}$ | |

At square $N_{1,3}$ we have to look at two alternatives,

$$j = 1 \quad \Rightarrow \quad \begin{array}{l} N_{1,1} = \{C_a, B\} \\ N_{2,3} = \{A_1\} \end{array} \qquad\qquad j = 2 \quad \Rightarrow \quad \begin{array}{l} N_{1,2} = \{S', S\} \\ N_{3,3} = \{C_b, A\} \end{array}$$

The nonterminal set corresponding to case $j = 1$ is $\{A\}$ ($A \rightarrow C_aA_1$) and that of case $j = 2$ is $\emptyset$, so $N_{1,3} = \{A\}$. We can continue the same way and and get the final table

| | | $i \rightarrow$ | | | |
|---|---|---|---|---|---|
| | $N_{i,i+k}$ | $1:a$ | $2:b$ | $3:b$ | $4:a$ |
| | 0 | $\underline{a}bba$ $\{B, C_a\}$ | $a\underline{b}ba$ $\{A, C_b\}$ | $ab\underline{b}a$ $\{B, C_a\}$ | $abb\underline{a}$ $\{A, C_b\}$ |
| | 1 | $\underline{ab}ba$ $\{S', S\}$ | $a\underline{bb}a$ $\{A_1\}$ | $ab\underline{ba}$ $\{S', S\}$ | |
| $k \downarrow$ | 2 | $\underline{abb}a$ $\{A\}$ | $a\underline{bba}$ $\{S'_1, S_1\}$ | | |
| | 3 | $\underline{abba}$ $\{S', S, A_1\}$ | | | |

Since $S' \in N_{1,4}$, $abba \in L(G)$. But, $S' \notin N_{1,3}$, so $abb \notin L(G)$.