T-79.148 **Spring 2003**
Introduction to Theoretical Computer Science
Tutorial 13
Solutions to the demonstration problems

4 **Problem**: Show that all context-sensitive languages can be recognised by linear-bounded automata. (Make use of the fact that in applying the grammar's production rules, the length of the sentential form under consideration can never decrease, except in the special case of the empty string.) Deduce from this result the fact that all context-sensitive languages are recursive.

Solution:

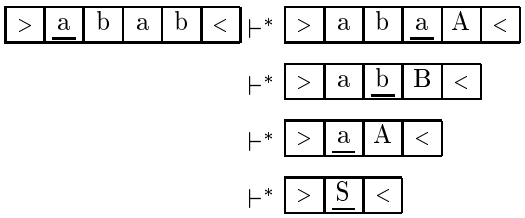
By definition, all rules of a context-sensitive languages are of the form where the right side is at least as long as the left side (apart from the possible rule $S \to \varepsilon$).

A context-sensitive language can be recognized by a linear-bounded automaton that non-deterministically moves into some place of the input and applies one of the rules of the grammar from right to left. Since the string may only become shorter, no new space is needed. Also, if symbols are removed, then it is trivial to construct a machine that removes all extra space. If we have only the initial symbol S on the tape at the end, the word is accepted.

Consider the following context-sensitive grammar:

$$\begin{aligned} S &\to aA \mid bB \\ aA &\to abB \mid ab \\ bB &\to baA \mid ba \end{aligned}$$

A linear-bounded automaton recognizing this language would work as follows for the input $abab$:



The machine described above is not total, since it is possible that it ends in an infinite loop. For example, if the grammar in question is:

$$\begin{aligned} S &\to \varepsilon \\ ab &\to ba \\ ba &\to ab, \end{aligned}$$

then all computations with a non-empty input fail to terminate.

We can fix the problem by noting that since the length of the tape may not increase, the number of possible configurations is finite. This number is of the magnitude $q \times n \times |\Gamma|^n$, where q is the number of states, n the length of input, and $|\Gamma|$ the size of the tape alphabet.

We can totalize the machine by adding a counter for it that counts the number of steps taken. The easiest way to do this is to have a two-track machine that keeps the counter on the second track encoded as a binary number. This number is then increased with each step of the original machine. When the counter reaches the limit, we can reject the word as the machine is in a loop.

Finally, we have to check that we can implement the counter without breaking the linear space bound. To encode a number $q \times n \times |\Gamma|^n$ we need $k = \log_2(q) + \log_2(n) + n\log(|\Gamma|)$ bits that is linear with respect to n. Even though $k > n$, we can squeeze it into the available space by encoding it in a suitable base.

1. **Problem**:

   Show that every language generated by an unrestricted grammar can also be generated by a grammar where no terminal symbols occur on the left hand side of any production.

   Solution: We can systematically construct a grammar G' that fulfills the conditions and generates the same language as a given grammar G by adding a new nonterminal A_a for each symbol $a \in \Sigma$, replacing a by A_a in each rule of the grammar, and finally adding a rule $A_a \to a$.

   Formally: Let G be an unrestricter grammar $G = (V, \Sigma, P, S)$. We construct a grammar $G' = (V', \Sigma, P', S')$, where

   $$V' = V \cup \{A_a \mid a \in \Sigma\}$$

   Each rule $r = x_1 \cdots x_n \to x_{n+1} \cdots x_{n+m}$ of G where $x_i \in V$ is transformed into:

   $$c(r) = y_1 \cdots y_n \to y_{n+1} \cdots y_{n+m}$$

   where

   $$y_i = \begin{cases} x_i, & x_i \in V - \Sigma \\ A_{x_i}, & x_i \in \Sigma \end{cases} .$$

   Now the set of rules P' can be defined as follows:

   $$P' = \{c(r) \mid r \in P\} \cup \{A_a \to a \mid a \in \Sigma\} .$$

   Consider the grammar from the Exercise 4:

   $$\begin{aligned} S &\to aA \mid bB \\ aA &\to abB \mid ab \\ bB &\to baA \mid ba \end{aligned}$$

   By using the construction, we get a grammar:

   $$\begin{aligned} S &\to A_aA \mid A_bB \\ A_aA &\to A_aA_bB \mid A_aA_b \\ A_bB &\to A_bA_aA \mid A_bA_a \\ A_a &\to a \\ A_b &\to b \end{aligned}$$

6. **Problem**:

   Show that every context-sensitive grammar can be put in a normal form where the productions are of the form $S \to \varepsilon$ or $\alpha A\beta \to \alpha\omega\beta$, where A is a nonterminal symbol and $\omega \neq \varepsilon$. (S denotes here the start symbol of the grammar.)

   Solution:

   In normalizing the grammar we have three steps:

   i) Remove the initial symbol S from the right side of rules.

   ii) Remove all terminal symbols from the left sides of the rules.

   iii) Fix all rules that are of a wrong form.

   The three steps are defined as follows:

i) If S occurs in the right side of a rule, we add a new initial symbol S' and a rule $S' \to S$ to the grammar.

ii) The terminal symbols are removed using the method presented in solution of Exercise 5.

iii) Each incorrect rule

$$X_1 \cdots X_n \to Y_1 \cdots Y_m \ ,$$

where $m \geq n$ we add $n-1$ new nonterminals $(Z_1, \ldots, Z_{n-1})$ and the rule is replaced by the set of rules:

$$\begin{aligned}
X_{n-1}X_n &\to Z_1 X_n \\
Z_1 X_n &\to Z_1 Y_n \cdots Y_m \\
X_{n-2}Z_1 &\to Z_2 Z_1 \\
Z_2 Z_1 &\to Z_2 Y_{n-1} \\
&\vdots \\
Z_{n-1}Z_{n-2} &\to Z_{n-1}Y_2 \\
Z_{n-1}Y_2 &\to Y_1 Y_2
\end{aligned}$$

For example, let us consider the rule

$$ABBA \to BAABA$$

As $n = 4$ we need three new nonterminals: Z_1, Z_2, and Z_3. The corresponding set of rules is:

$$\begin{aligned}
BA &\to Z_1 A \\
Z_1 A &\to Z_1 BA \\
BZ_1 &\to Z_2 Z_1 \\
Z_2 Z_1 &\to Z_2 A \\
AZ_2 &\to Z_3 Z_2 \\
Z_3 Z_2 &\to Z_3 A \\
Z_3 A &\to BA \ .
\end{aligned}$$

Now the derivation of the original rule becomes the following derivation:

$$\begin{aligned}
AB\underline{BA} \Rightarrow AB\underline{Z_1 A} \Rightarrow A\underline{BZ_1}BA \Rightarrow A\underline{Z_2 Z_1}BA \Rightarrow \underline{AZ_2}ABA \\
\Rightarrow \underline{Z_3 Z_2}ABA \Rightarrow \underline{Z_3 A}ABA \Rightarrow BAABA \ .
\end{aligned}$$