Solutions to Supplementary Problems

S11.1 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 decidable.

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

A language generated by a context-sensitive grammar G can be recognised by a linear-bounded automaton that in each phase nondeterministically moves to some position on its tape and applies one of the rules of G there "backwards", replacing a sequence of symbols on the tape that matches the right hand side of a rule by the sequence on its left hand side. Since the l.h.s. sequence is not longer than the r.h.s. sequence, no new space on the tape is needed. In cases where the l.h.s. sequence is properly shorter than the r.h.s. sequence, it is simple to adapt the machine so that it also compresses the string on its tape. If the machine eventually has only the start symbol S on its tape, it halts and accepts.

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 corresponding to this grammar would work as follows on input $abab$ (here $\vdash^*$ means that several steps are taken by the machine between two configurations):

$$\begin{aligned} &\boxed{\underline{\mathrm{a}}}\boxed{\mathrm{b}}\boxed{\mathrm{a}}\boxed{\mathrm{b}} \vdash^* \boxed{\mathrm{a}}\boxed{\mathrm{b}}\boxed{\underline{\mathrm{a}}}\boxed{\mathrm{b}} \\ \vdash^* &\boxed{\mathrm{a}}\boxed{\underline{\mathrm{b}}}\boxed{\mathrm{a}}\boxed{\mathrm{A}} \\ \vdash^* &\boxed{\underline{\mathrm{a}}}\boxed{\mathrm{b}}\boxed{\mathrm{B}} \\ \vdash^* &\boxed{\underline{\mathrm{a}}}\boxed{\mathrm{A}} \\ \vdash^* &\boxed{\underline{\mathrm{S}}} \end{aligned}$$

This construction shows that all context-sensitive languages can be recognised by (non-deterministic!) linear-bounded automata.

However, a machine constructed as above may still go 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 on an input string that contains either ab or ba as a substring fail to terminate.

This problem can be fixed by noting that since the length of the string on the tape may not increase, the number of possible machine configurations is finite. On an input of length n, this number is at most $T(n) = q \times (n+2) \times |\Gamma|^n$, where q is the number of states, $n+2$ the number of possible tape head positions, and $|\Gamma|$ the size of the tape alphabet.

We can now modify the machine to a self-timed version that counts the number of steps it has taken. The easiest way to do this is to design a two-track machine that keeps a step counter on its second track, initialised to value $T(n)$ represented in binary notation.[1] The counter value is decreased with each step of the original machine. When the counter reaches zero, this timed machine can reject the input string, since the original machine is then necessarily in a loop.

This kind of non-deterministic machine, where all computations terminate, can be simulated by a deterministic total machine with the techniques discussed earlier in the course (on a given input, the deterministic machine simulates, one-by-one, all the computations of the non-deterministic machine and checks if any of them accepts). This shows that context-sensitive languages are also decidable.

S11.2 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 fulfils the conditions and generates the same language as a given grammar G by adding a new nonterminal Y_a for each symbol $a \in \Sigma$, replacing a by Y_a in each rule of the grammar, and finally adding a rule $Y_a \to a$.

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

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

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

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

where

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

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

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

Consider the grammar from the Exercise 4:

$$S \to aA \mid bB$$
$$aA \to abB \mid ab$$
$$bB \to baA \mid ba$$

By using the construction, we get a grammar:

$$S \to Y_aA \mid Y_bB$$
$$Y_aA \to Y_aY_bB \mid Y_aY_b$$
$$Y_bB \to Y_bY_aA \mid Y_bY_a$$
$$Y_a \to a$$
$$Y_b \to b$$

[1] Or to stay strictly within the tape length limit of n positions, in k-ary notation for some $k \sim \log_2 |\Gamma|$.

S11.3 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 normalising the grammar we perform two steps:

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

ii) Manipulate all the rules into the desired form.

The steps are implemented as follows:

i) The l.h.s. terminal symbols are removed as presented in the solution to Problem S11.2.

ii) For each rule
$$X_1 \cdots X_n \to Y_1 \cdots Y_m \ ,$$
where $2 \leq n \leq m$, we add $n-1$ new nonterminals $(Z_1, \ldots, Z_{n-1})$ and replace the rule by the following new set of rules, where only one nonterminal is rewritten at each step:

$$\begin{aligned}
X_{n-1}X_n &\to Z_{n-1}X_n \\
Z_{n-1}X_n &\to Z_{n-1}Y_n \cdots Y_m \\
X_{n-2}Z_{n-1} &\to Z_{n-2}Z_{n-1} \\
Z_{n-2}Z_{n-1} &\to Z_{n-2}Y_{n-1} \\
&\vdots \\
Z_1Z_2 &\to Z_1Y_2 \\
Z_1Y_2 &\to Y_1Y_2
\end{aligned}$$

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:

$$\begin{aligned}
BA &\to Z_3A \\
Z_3A &\to Z_3BA \\
BZ_3 &\to Z_2Z_3 \\
Z_2Z_3 &\to Z_2A \\
AZ_2 &\to Z_1Z_2 \\
Z_1Z_2 &\to Z_1A \\
Z_1A &\to BA \ .
\end{aligned}$$

Now a derivation step using the original rule becomes the following derivation sequence:

$$\begin{aligned}
AB\underline{BA} \Rightarrow AB\underline{Z_3A} \Rightarrow A\underline{BZ_3}BA \Rightarrow A\underline{Z_2Z_3}BA \Rightarrow \underline{AZ_2}ABA \\
\Rightarrow \underline{Z_1Z_2}ABA \Rightarrow \underline{Z_1A}ABA \Rightarrow BAABA \ .
\end{aligned}$$