Program Optimisation
Solutions of Homework 9

1. We have the following set of operators and constants.

   $\mathsf{BinOp} = \{:, +\}$
   $\mathsf{UnOp} = \{\mathsf{M}\}$
   $\mathsf{Leaf} = \{\mathsf{int}, []\}$

   We treat non-terminals ("registers") as leaves.

   a) The set of all lists (inner nodes: ":") of int's with an even number of elements (in particular the leftmost node is "[]") is generated by the following grammar:

   $R \rightarrow []$
   $R \rightarrow (R : \mathsf{int}) : \mathsf{int}$

   b) The set all trees in which an M always has, directly below itself, a "+", is generated by the following grammar:

   $R \rightarrow []$
   $R \rightarrow \mathsf{int}$
   $R \rightarrow R : R$
   $R \rightarrow R + R$
   $R \rightarrow \mathsf{M}(R + R)$

   c) The set of all trees in which an M never has (directly or indirectly) below itself a ":" is generated by the following grammar:

   $R \rightarrow []$
   $R \rightarrow \mathsf{int}$
   $R \rightarrow R : R$
   $R \rightarrow R + R$
   $R \rightarrow \mathsf{M}(S)$
   $S \rightarrow []$
   $S \rightarrow \mathsf{int}$
   $S \rightarrow S + S$
   $S \rightarrow \mathsf{M}(S)$

2. a) G contains the following rules:

   $R \rightarrow a(A, B)$
   $A \rightarrow b(A)$
   $A \rightarrow c(B)$
   $B \rightarrow d$

We have:

$$R \Rightarrow a(A, B) \Rightarrow a(c(B), B) \Rightarrow a(c(d), B) \Rightarrow a(c(d), d)$$

Hence we have $a(c(d), d) \in L(G, R)$ so that $L(G, R) \neq \emptyset$.

b) Let G be the given grammar. We iteratively compute a set $\mathcal{N}$ of non-terminals such that each non-terminal $N \in \mathcal{N}$ has the property that $L(G, N) \neq \emptyset$. Initially we set $\mathcal{N} = \emptyset$. At any point during the iteration, if there is a rule $A \to \alpha$ in G such that all non-terminals occurring in α have already been added to $\mathcal{N}$ then we add the non-terminal A to $\mathcal{N}$ (if A is not already present in $\mathcal{N}$). We keep adding new non-terminals to $\mathcal{N}$ in this way till no more non-terminals can be added. In the end $R \in \mathcal{N}$ iff $L(G, R) \neq \emptyset$. This algorithm runs in polynomial time. By using clever data structures we can also do it in linear time. We don't detail this here.

3. If α is a term (built from terminals as well as non-terminals) then we define the depth of α as:

$$dp(a) = 1 \quad (a \text{ is a constant (i.e. zero-ary)})$$
$$dp(A) = 0 \quad (A \text{ is a non-terminal})$$
$$dp(f(\alpha_1, \ldots, \alpha_n)) = 1 + max\{dp(\alpha_1), \ldots, dp(\alpha_n)\}$$

Define the size n of a grammar as $n = \sum_{A \to \alpha \in G} dp(\alpha)$. If A is a non-terminal and t is a term (consisting only of terminal symbols) such that $A \Rightarrow^* t$ then we can label positions in the term t with rules that were used in the corresponding derivation. For example in the exercise 2(a) we have $R \Rightarrow^* a(c(d), d)$ and we can label the term $a(c(d), d)$ as in Abbildung 1.

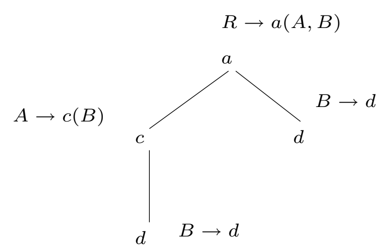

Abbildung 1: Labeling a term according to the derivation

a) Now if the tree has a path on which two distinct positions are labeled by the same rule $A \to \alpha$ then we can obtain a shorter tree as shown in Abbildung 2. Hence whenever we have $R \Rightarrow^* t$ where t contains only terminals then we also have $R \Rightarrow^* t'$ where t' contains only terminals and in the labeling corresponding to the derivation of t' from R, no rule occurs twice on the same path. Then we can check that $dp(t') \leq n$.

b)
- Suppose $L(G, R)$ is infinite. Then $L(G, R)$ has terms of arbitrarily large depth. Hence there is also a term t such that $n < dp(t)$. If $dp(t) \leq 2n$ then we are done. If $dp(t) > 2n$ then by using contractions as in Abbildung 2, we can get a term $t' \in L(G, R)$ such that $n < dp(t') \leq 2n$.

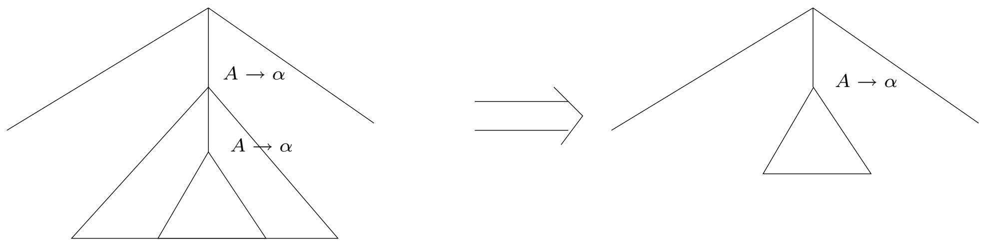

Abbildung 2:

- Suppose there is some $t \in L(G, R)$ with $n < dp(t) \leq 2n$. Since $n < dp(t)$ some path in the labeling corresponding to the derivation of t has two occurrences of the same rule (by arguments as in part 3(a)). Then by using steps as in Abbildung 3 we can terms in $L(G, R)$ of larger and larger depth.

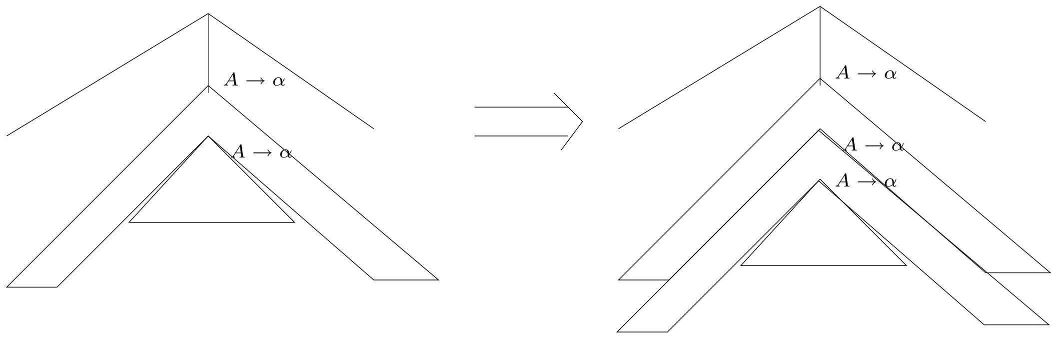

Abbildung 3: