Homework for lecture
Automata and Formal Languages II

TU München
Institut für Informatik

Dr. Peter Lammich

SS 2015 — Homework Sheet 7 — 27.05.2014

Submission: June 3

Aufgabe 7.1. **[Pretty-Printing Trees]** (10 points)
Let $\mathcal{F}$ be a ranked alphabet, and $\Sigma := \mathcal{F} \,\dot{\cup}\, \{\text{“(”}, \text{“)”}, \text{“,”}\}$ be an unranked alphabet that contains the symbol names from $\mathcal{F}$, open parentheses, closing parentheses, and comma. Consider the following function $p : T(\mathcal{F}) \to \Sigma$, that pretty-prints a tree, where $\cdot$ is string concatenation:

$$p(f(t_1, \ldots, t_n)) = f \cdot \text{“(”} \cdot p(t_1) \cdot \text{“,”} \cdot \ldots \cdot \text{“,”} \cdot p(t_n) \cdot \text{“)”}$$

Show that the image of a tree regular language under p is context free, i.e., given a tree regular language $L \subseteq T(\mathcal{F})$, show that the language $W := \{p(t) \mid t \in L\} \subseteq \Sigma^*$ is context free. Hint: Show how to convert the rules of a top-down tree automaton into a context free grammar.

Aufgabe 7.2. [Simple Tree to String Transducers] (10 points)
Generalizing the pretty-printing function from the previous question, we define a *simple tree to string transducer* from a ranked alphabet $\mathcal{F}$ to an alphabet Σ by specifying, for each $f \in F_n$, strings $r_{f,0}, r_{f,1} \ldots r_{f,n} \in \Sigma^*$. The transformation $r : T(\mathcal{F}) \to \Sigma^*$ is then defined as:

$$r(f(t_1, \ldots, t_n)) = r_{f,0} \cdot r(t_1) \cdot r_{f,1} \cdot \ldots \cdot r_{f,n-1} \cdot r(t_n) \cdot r_{f,n}$$

Intuitively, the $r_{f,i}$ are the strings that separate the pretty-prints of the arguments of f.

For example, for the ranked alphabet $+/2, */2, Suc/1, 0/0$, a simple pretty-printer, which does some parenthesis optimizations, could be defined by the simple tree to string transducer:

$$\begin{array}{rlrlrl}
r_{+,0} = & \text{“(”} & r_{+,1} = & \text{“+”} & r_{+,2} = & \text{“)”} \\
r_{*,0} = & \text{“”} & r_{*,1} = & \text{“*”} & r_{+,2} = & \text{“”} \\
r_{Suc,0} = & \text{“Suc(”} & r_{Suc,1} = & \text{“)”} & & \\
r_{0,0} = & \text{“0”} & & & &
\end{array}$$

then, we have $r(+(*(0, Suc(0)), 0)) = \text{“}(0 * Suc(0) + 0)''$.

1. Consider the alphabet $\mathcal{F} = true/0, false/0, Cons/2, Nil/0$ and $\Sigma = \{a - z, :, [,]\}$. Define a simple tree to string transducer that pretty prints a list of booleans to a representation using infix “::” for $Cons$, and [] for Nil, e.g., the list $[true, true, false]$ is printed as $true :: true :: false :: []$, and the empty list is printed as [].

2. Consider arbitrary alphabets $\mathcal{F}$ and Σ, and a transducer $r : \mathcal{F} \to \Sigma$. Show that the inverse image of a regular word language under r is tree regular, i.e., for a regular word language $W \subseteq \Sigma^*$, show that the tree language $r^{-1}(W) := \{t \mid r(t) \in W\}$ is regular.

   Hint: Similar to leaf languages (which are a special case of transducers), construct a tree automaton with states $Q \times Q$, such that a tree t is accepted in state (q_1, q_2), if $r(t)$ brings the word automaton from q_1 to q_2, i.e., $\delta(q_1, r(t)) = q_2$.

3. Show that the inverse image of a context free language W under a simple tree to string transducer r is not regular in general. Give a counterexample by specifying suitable alphabets, W and r.