5. Übung (2019-05-15)

# Formale Baumsprachen

***Task 11 (regular tree grammars)***

Let $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$ be a ranked alphabet. Give regular tree grammars $G_1$ and $G_2$ with

(a) $L(G_1) = \{\xi \in T_\Sigma \mid \xi \text{ contains exactly one } \sigma\}$ and

(b) $L(G_2) = \{\xi \in T_\Sigma \mid \xi \text{ contains the pattern } \sigma(\_, \gamma(\_)) \text{ at least twice}\}$.

***Task 12 (unranked tree automata in Haskell)***

In the lecture we considered automata for ranked trees, i.e. trees where each terminal symbol has a fixed number of successors (the rank). In practice it is sometimes desirable for trees to allow an arbitrary number of successors for the terminal symbols, such trees are called unranked trees. In HTML, for example, it is possible for the <**ul**>-tag (unordered list) to have arbitrarily many successors of the form <**li**>…</**li**> (list item). Automata over unranked trees are called unranked tree automata.

(a) Give a definition of unranked tree automata and deterministic unranked tree automata.

(b) Devise a polymorphic data type `DUTA q t` in Haskell for deterministic unranked tree automata. The type variables `q` and `t` represent the states and terminal symbols, respectively.

(c) Implement a function `recognize :: DUTA q t -> Tree t -> Bool` that returns **`True`** if and only if the first parameter recognizes the second parameter.

(d) Test `recognize`.

(e) Devise a polymorphic data type `NUTA q t` in Haskell for unranked tree automata.

(f) Implement a function `recognize' :: NUTA q t -> Tree t -> Bool` that returns **`True`** if and only if the first parameter recognizes the second parameter.

(g) Test `recognize'`.