Weighted Tree Automata

Exercise 6 (Deterministic bu-ta I)
Let $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}, \beta^{(0)}\}$. For each of the following tree languages give a deterministic bu-ta which accepts exactly that language.

1. $L_1 = \{ \xi \in T_\Sigma \mid \xi \text{ contains at least one } \alpha \text{ and one } \beta \}$,
2. $L_2 = \{ \xi \in T_\Sigma \mid \xi \text{ contains an even number of } \alpha \text{ symbols} \}$.

Exercise 7 (Deterministic bu-ta II)
Let $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}, \beta^{(0)}\}$. For every $k \in \mathbb{N}$, we define the tree $\zeta_k \in T_\Sigma$ by $\zeta_0 = \beta$ and $\zeta_{k+1} = \sigma(\alpha, \zeta_k)$. Moreover, let us consider the deterministic bu-ta $A = (Q, \Sigma, \delta, F)$ where $Q = \{l, r, \bot\}$, $F = \{r\}$, $\delta_\alpha() = l$, $\delta_\beta() = r$, and for every $(q_1, q_2) \in Q \times Q$$\delta_\sigma(q_1q_2) = \begin{cases} r & \text{if } (q_1, q_2) = (l, r), \\ \bot & \text{otherwise}. \end{cases}$

Prove the following statements.

1. $P \subseteq L(A)$.
2. $L(A) \subseteq P$.

Exercise 8 (Nondeterministic bu-ta)
Let $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}, \beta^{(0)}, \gamma^{(0)}\}$. Give a nondeterministic bu-ta which accepts exactly the language of all trees in $T_\Sigma$ that contain a $\beta$-leaf somewhere between an $\alpha$- and a $\gamma$-leaf, reading leaves left-to-right or right-to-left. Try to use as few states and transitions as possible.

Exercise 9 (Nondeterminism and determinization)
Let $\Sigma = \{\alpha^{(0)}, \gamma^{(1)}, \sigma^{(2)}\}$. Consider the language $L = \{ \xi \in T_\Sigma \mid \forall w \in \text{pos}(\xi): \xi(w) = \gamma \Rightarrow \xi|_w \in T_{\Sigma \setminus \Sigma^{(2)}} \}$ of all trees where for every path from the root to some leaf, the rank of the nodes along the path is monotonic decreasing.

1. Construct a bu-ta $A$ that accepts $L$ using the strategy of guessing and checking. Your automaton should guess whether it has already seen every gamma on the path to the root and choose the successor state accordingly. Later on, it should verify that the guess was correct. Naturally, if the guess was “no”, then a $\sigma$ should lead to failure.
2. Use the powerset construction to obtain a deterministic version of $A$. 

Prof. Dr. H. Vogler / J. Osterholzer
http://www.orchid.inf.tu-dresden.de
Lehrstuhl Grundlagen der Programmierung, Fakultät Informatik, TU Dresden