Foundations of XML Types: Suggested Answers

Trees and Tree Automata

Q2. Give a bottom-up deterministic tree automaton that recognizes the tree language $L$ composed of the two trees below:

```
  a
 / \\  
 b   c
```

A2. A sample bottom-up deterministic tree automaton $B$ that recognizes $L$:

- **Alphabet($B$)**: $\{a^{(2)}, b^{(0)}, c^{(0)}\}$
- **States($B$)**: $\{q_a, q_b, q_c\}$
- **Final($B$)**: $\{q_a\}$
- **Rules($B$)**: $\{((q_b, q_c) \xrightarrow{a} q_a), (q_c, q_b) \xrightarrow{a} q_a, \epsilon \xrightarrow{b} q_b, \epsilon \xrightarrow{c} q_c\}$

Q3. Bottom-up tree automata seen during the course traverse trees from the leaves to the root. In a similar manner, one may define top-down tree automata that recognize trees by going in the opposite direction: from the root to the leaves. Specifically, a top-down tree automaton $A$ consists in:

- **Alphabet($A$)**: finite alphabet of symbols
- **States($A$)**: finite set of states
- **Rules($A$)**: finite set of transition rules
- **Initial($A$)**: finite set of initial states ($\subseteq$ States($A$))
- $q_{\text{acc}} \in$ States($A$): final state

There are two major differences with automata seen during the course:

- transition rules are either of the form: $q \xrightarrow{a} (q_1, q_2)$ where $q, q_1, q_2 \in$ States($A$) and $a \in$ Alphabet($A$) or of the form $q \xrightarrow{a} q_1$ for leaves.
- a tree is accepted if and only if there exists a run for which all the leaves are labeled with $q_{\text{acc}}$.

Give a top-down tree automaton that recognizes $L$.

A3. A sample top-down tree automaton $T$ that recognizes $L$:

- **Alphabet($T$)**: $\{a^{(2)}, b^{(0)}, c^{(0)}\}$
- **States($T$)**: $\{q_a, q_{\text{acc}}\}$
- **Initial($T$)**: $\{q_a\}$
- $q_{\text{acc}} \in$ States($T$): final state
- **Rules($T$)**: $\{q_a \xrightarrow{a} (q_b, q_c), q_a \xrightarrow{a} (q_c, q_b), q_b \xrightarrow{b} q_{\text{acc}}, q_c \xrightarrow{c} q_{\text{acc}}\}$

Q4. Do you see any interest of top-down tree automata in the context of XML stream processing where XML documents are sequentially parsed (only once) and processed on the fly? Explain.
A4. A top-down tree automaton can be used to implement on-the-fly validation of an XML stream against
a given schema. In this context, nodes of an XML document are scanned, parsed, and processed on-
the-fly starting from the root and in the order of a depth-first tree traversal. Top-down automata are
more appropriate in this setting than bottom-up automata. This is because bottom-up automata have
to wait for the full document (leaves) in order to be able to start validation. On the opposite, some
transitions of top-down automata can be triggered without having to wait for the full document, so
that they can be used to detect errors earlier with a stream (i.e., an incomplete document).

Q5. A top-down tree automaton is deterministic iff (1) there is at most one initial state and (2) for each
$q \in \text{States}(A)$ et $a \in \text{Alphabet}(A)$ there is at most one rule $q \xrightarrow{a} (q_1, q_2)$ (intuitively, there is at most
one possible transition for each state and symbol).

Is it possible to give a deterministic top-down tree automaton that recognizes $L$? Either give it or
justify.

A5. It is not possible. Indeed, let’s try to build a deterministic top-down tree automaton $DT$ that recog-
nizes $L$:

\[
\begin{align*}
\text{Alphabet}(DT) & : \{a^{(2)}, b^{(0)}, c^{(0)}\} \\
\text{States}(DT) & : \{q_a, q_{acc}\} \\
\text{Initial}(DT) & : \{q_a\} \\
q_{acc} & \in \text{States}(DT) : \text{final state}
\end{align*}
\]

The impossible part is to define $\text{Rules}(DT)$:

$DT$ must be deterministic so we have no other choice then putting only one transition rule for $q_a$ and $a$,
such as: $q_a \xrightarrow{a} (q, q)$. Then, while still keeping $DT$ deterministic, the only thing we can do is to add the
rules $q \xrightarrow{b} q_{acc}$ and $q \xrightarrow{c} q_{acc}$. However, if we define $\text{Rules}(DT) = \{q_a \xrightarrow{a} (q, q), q \xrightarrow{b} q_{acc}, q \xrightarrow{c} q_{acc}\}$
then $DT$ also recognizes the two trees below:

```
      a
    / \  \
   b   b
```
```
      a
    / \  \
   c   c
```

These two trees are not part of $L$. There does not exist any deterministic top-down tree automaton
that can recognize $L$ (and only $L$).

Q6. It is known that non-deterministic bottom-up and non-deterministic top-down automata are equally
expressive. From your answers to the previous questions, what can you conclude about the respective
expressive power of deterministic bottom-up and deterministic top-down tree automata? Justify.

A6. If we sum up: the tree language $L$ can be recognized by a bottom-up tree automata (see $B$ above), and
by a non-deterministic top-down tree automaton (see $T$ above). However, no deterministic top-down
tree automaton can recognize $L$ (see previous question). Thus, deterministic top-down tree automata
are strictly less expressive than non-deterministic top-down tree automata.