Automata Theory - Quiz II (Solutions)

K. Subramani
LCSEE,
West Virginia University,
Morgantown, WV
{ksmani@csee.wvu.edu}

1 Problems

1. **Induction:** Let L denote the language of balanced strings over $\Sigma = \{0, 1\}$, i.e., L consists of all strings in $\{0, 1\}$, such that $n_0(w) = n_1(w)$. Given a string $w = w_1 w_2 \ldots w_n$, a *proper prefix* of w is defined to be any string $x = x_1 x_2 \ldots x_k$, such that $k < n$ and $x_i = w_i,\ i = 1, 2, \ldots, k$. Let $y = y_1 y_2 \ldots y_r$ denote an arbitrary string in L with $y_1 = y_r$. Argue that y must have a balanced proper prefix.

   Solution: We use induction on $|y|$ to prove the theorem. The following observations are in order:

   (a) $|y|$ must be even. (Straightforward proof using induction.)

   (b) $|y| \geq 4$. If $|y| = 0$, then $y = \lambda$; if $|y| = 2$, then y must be 01 or 10 and hence the first and last characters are not equal.

   Accordingly, the base case is $|y| = 4$; the restriction that $y_1 = y_4$, implies that y is either 0110 or 1001. In the former case, 01 is the balanced proper prefix, while in the latter case, 10 is. Thus, in both cases, inspection establishes the existence of a balanced proper prefix and the basis is proven.

   Assume that the theorem holds true for all balanced strings y, where $|y| \leq 4 + 2 \cdot r$, $r \geq 0$. Now consider a balanced string y of length $|y| = 4 + 2\dot{r} + 2$. Without loss of generality, assume that $y_1 = y_{4+2\cdot r+2} = 0$. Accordingly, y has the form $0w0$, where $w = w_1 w_2 \ldots w_{4+2\cdot r}$. Since y is balanced, w contains at least one position i such that $w_i = 1$. Let m denote the smallest such position, i.e., $w_i = 0,\ i = 1, 2, \ldots, m - 1$. Consider the string y' obtained from y, by eliminating the substring $z = 0w_m$. Clearly, y' is balanced; further $|y'| = 4 + 2 \cdot r$ and hence, as per the inductive hypothesis, y' has a balanced proper prefix, say y''. Reinserting z into y'' at the appropriate positions, gives us a balanced proper prefix of y.

   Applying the principle of mathematical induction, we conclude that the theorem is true.

   □

2. **Context-Free Grammars:** Consider the grammar $G = \langle V, T, S, P \rangle$, where $V = \{S\}$, $T = \{0, 1\}$, $S = S$, and the productions P are defined by:

   $$S \rightarrow 0S1 \mid 1S0 \mid SS \mid \lambda$$

   Argue that $L(G)$ contains the set of *all* balanced strings.

   Solution: Let x denote an arbitrary balanced string; we used induction on $|x|$ to show that x can be derived from S.

   BASIS: $|x| = 0$. In this case, x must be λ and since $S \rightarrow \lambda$ is a production of the grammar, we have proved that $S \Rightarrow^* x$.

   INDUCTIVE STEP: Assume that if x is a balanced string of length at most $2 \cdot k$, then $S \Rightarrow^*$ x. Now consider a string x of length $2 \cdot k + 2$, $k \geq 0$. We need to consider the following cases:

(i) $x_1 = 0$ and $x_{2\cdot k+2} = 1$ – In this case, x has the form $0w1$, where w is also balanced. However, since $|w| = 2\cdot k$, as per the inductive hypothesis, w can be derived from S. But this immediately implies that x can be derived from S; for instance, $S \Rightarrow 0S1 \Rightarrow^* 0w1 \Rightarrow x$.

(ii) $x_1 = 1$ and $x_{2\cdot k+2} = 0$ – This case is identical to the one above.

(iii) $x_1 = 0$ and $x_{2\cdot k+2} = 0$ – In this case, we know from the prefix theorem, that x must have a balanced proper prefix. Let us call this prefix y. Then x can be written as $y \cdot y'$, where y' is also balanced. Since $|y|, |y'| \leq 2 \cdot k$, we know that y and y' can be derived from S. Thus x can be derived from S as follows: $S \Rightarrow SS \Rightarrow yS \Rightarrow y \cdot y' \Rightarrow x$.

(iv) $x_1 = 1$ and $x_{2\cdot k+2} = 1$ – This case is identical to the one above.

Applying the principle of mathematical induction, we conclude that all balanced strings can be derived from S, i.e., $L(G)$ contains the set of all balanced strings. □

3. **Context-Free Grammar Parsing:** Consider the grammar $G = \langle V, T, S, P\rangle$, with $V = \{S, A, B, C\}$, $T = \{a, b\}$, $S = S$ and the productions P defined by:

$$\begin{aligned} S &\rightarrow AB \mid BC \\ A &\rightarrow BA \mid a \\ B &\rightarrow CC \mid b \\ C &\rightarrow AB \mid a \end{aligned}$$

Use the CYK algorithm to establish the membership or non-membership of $w = ababa$ in $L(G)$.
(You may not use trial and error!)

Solution:

Table (1) represents the application of the CYK algorithm to determine the membership of $w = ababa$ in $L(G)$.

Table 1: Applying the CYK algorithm

$\{S\}$				
$\{B\}$	$\{B\}$			
$\{B\}$	$\{S, C\}$	$\{B\}$		
$\{S, C\}$	$\{A, C\}$	$\{S, C\}$	$\{A, C\}$	
$\{A, C\}$	$\{B\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
a	b	a	b	a

Since the topmost leftmost entry contains the start symbol S, we conclude that $w \in L(G)$; indeed the following leftmost derivation establishes it.

$$\begin{aligned} S &\Rightarrow AB \\ &\Rightarrow aB \\ &\Rightarrow aCC \\ &\Rightarrow aABC \\ &\Rightarrow aBABC \\ &\Rightarrow abABC \\ &\Rightarrow abaBC \\ &\Rightarrow ababC \\ &\Rightarrow ababa \end{aligned}$$

□

4. **Pushdown Automaton Design:** Consider the language L defined as follows:
$L = \{w \in \{0,1\}^* \ : \ n_0(w) = 2 \cdot n_1(w)\}$. Establish that L is context-free by designing a PDA that accepts all and only those strings in L.

Solution: In design problems, it is helpful to identify the strategy first. One approach to designing an accepter for L is as follows: Mark the occurrence of a 1 with two X characters and mark the occurrence of a 0 with a single Y character. If the input string w contains twice as many 0s as 1s, then it must be the case that the number of X characters equals the number of Y characters.

Consider the following non-deterministic PDA $M = \langle Q, \Sigma, \Gamma, \delta, q_0, z, F\rangle$, where,

(i) $Q = \{q_0, q_1, q_f\}$.

(ii) $\Sigma = \{0, 1\}$.

(iii) $\Gamma = \{X, Y, z\}$.

(iv) δ is defined through the following transitions:

(a) If there is no input at all, i.e., $w = \lambda$, then $n_0(w) = 2 \cdot n_1(w)$. Accordingly, we have,

$$\delta(q_0, \lambda, z) \quad = \quad (q_f, \lambda)$$

(b) When the input character is 0, we first check if there are any 1s on the stack to match it with; if not, Y is pushed onto the stack. Accordingly, we have,

$$\begin{aligned}
\delta(q_0, 0, z) &= (q_0, Yz) \\
\delta(q_0, 0, X) &= (q_0, \lambda) \\
\delta(q_0, 0, Y) &= (q_0, YY)
\end{aligned}$$

(c) When the input character is a 1, we first check if there are any 0s on the stack to match it with; if not, we push in two X characters onto the stack. The matching portion needs to be carried out carefully, since there are various character combinations on the stack, which must be accounted for. Accordingly, we have,

$$\begin{aligned}
\delta(q_0, 1, z) &= (q_0, XXz) \\
\delta(q_0, 1, X) &= (q_0, XXX) \\
\delta(q_0, 1, Y) &= (q_1, \lambda) \\
\delta(q_1, \lambda, Y) &= (q_0, \lambda) \\
\delta(q_1, \lambda, X) &= (q_0, XX) \\
\delta(q_1, \lambda, z) &= (q_0, Xz)
\end{aligned}$$

(v) $q_0 = q_0$.

(vi) $z = z$.

(vii) $F = \{q_f\}$.

The above design reinforces the power of non-determinism; if we had chosen to design a DPDA for L, the task would have been onerous indeed. It is a straightforward exercise using induction to establish that $L = L(M)$. $\square$

5. **Turing Machine Design:** Design a Deterministic Turing Machine to decide the language of palindromes over $\{0, 1\}$. You may select the architecture of the Turing Machine, as per your tastes and convenience.

   Solution: I choose a two-tape Turing Machine in which the tapes are semi-infinite, i.e., unbounded only on the left. The left end of each tape is marked by the special character $\triangleright$. The algorithm consists of the following steps:

   (a) Copy the input string w which is on the first tape onto the second tape.

   (b) Position the head of the first tape on the first symbol after $\triangleright$, i.e., the first symbol of w and the head of the second tape on the last symbol of w.

   (c) As long as the symbols under both heads are *identical*, move the first tape head to the right and the second tape head to the left. If at any point, the tape heads read different symbols, halt and reject. Likewise, if the first tape head encounters a blank, then halt and accept. Note that the occurrence of a blank below the first tape head coincides with the occurrence of $\triangleright$ below the second tape head.

   The above design makes the following assumptions:

   (a) The input string is written on the first tape immediately after the $\triangleright$,

   (b) At commencement, both tape heads are on the $\triangleright$ symbols of their respective tapes,

   (c) The second tape is initially blank,

   (d) On each transition, the tape heads can independently move right ($\rightarrow$), left ($\leftarrow$) or stay put ($-$).

   Consider the Turing Machine $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \Box, F\rangle$, where,

   (i) $Q = \{q_0, q_1, q_2, q_3, q_4\}$.

   (ii) $\Sigma = \{0, 1\}$.

   (iii) $\Gamma = \{0, 1, \Box, \triangleright\}$.

   (iv) The program δ is defined by the following transitions:

   (a) The first set of transitions copies the input string onto the second tape. Accordingly, we have,

$$\begin{aligned}
\delta(q_0, 0, \Box) &= (q_0, 0, \rightarrow, 0, \rightarrow) \\
\delta(q_0, 1, \Box) &= (q_0, 1, \rightarrow, 1, \rightarrow) \\
\delta(q_0, \triangleright, \triangleright) &= (q_0, \triangleright, \rightarrow, \triangleright, \rightarrow)
\end{aligned}$$

   (b) The second set of transitions repositions the heads of both tapes. Accordingly, we have,

$$\begin{aligned}
\delta(q_0, \Box, \Box) &= (q_1, \Box, \leftarrow, \Box, -) \\
\delta(q_1, 0, \Box) &= (q_1, 0, \leftarrow, \Box, -) \\
\delta(q_1, 1, \Box) &= (q_1, 1, \leftarrow, \Box, -) \\
\delta(q_1, \triangleright, \Box) &= (q_2, \triangleright, \rightarrow, \Box, \leftarrow)
\end{aligned}$$

   (c) The third set of transitions accomplishes the match and move portion of our algorithm. Accordingly, we have,

$$\begin{aligned}
\delta(q_2, 0, 0) &= (q_2, 0, \rightarrow, 0, \leftarrow) \\
\delta(q_2, 1, 1) &= (q_2, 1, \rightarrow, 1, \leftarrow) \\
\delta(q_2, \Box, \triangleright) &= (q_3, \Box, -, \triangleright, -)
\end{aligned}$$

(d) The final set of transitions accounts for the case in which w is not a palindrome. Accordingly, we have,

$$\begin{array}{rcl} \delta(q_2, 0, 1) & = & (q_4, 0, -, 1, -) \\ \delta(q_2, 1, 0) & = & (q_4, 1, -, 0, -) \end{array}$$

(v) $q_0 = q_0$.

(vi) $\square = \square$.

(vii) $F = \{q_3\}$.

We observe that q_3 is the "accept" state, while q_4 is the "reject" state. It is not hard to see that the above Turing Machine accomplishes the mission of recognizing palindromes. I do expect that you will formally argue the correctness of the design using induction. $\square$