1. (30 points) Write an algorithm that will, given a DFA, determines if there is some string that is accepted by it. Do the algorithm from scratch. For example, do not say something like view the DFA as a graph and use Kruskal’s Algorithm to find a spanning tree.

SOLUTION

(a) Input $M$ a DFA. Let $n$ be the number of states.
(b) Let $A_0 = \{s\}$. (Our plan is that $A_i$ will be the set of states that can be reached with a $\leq i$-long word.)
(c) For $i = 1$ to $n$: $A_i = A_{i-1} \cup \{q : (\exists (p,\sigma) \in A_{i-1} \times \Sigma) : \delta(p,\sigma) = q\}$
(d) If $F \cap A_n \neq \emptyset$ output Y, else output N.

The key is that if $M$ accepts any string then it accepts a string of length $\leq n$.

END OF SOLUTION
2. (30 points—5 points each) In this problem we use WS1S notation. (Make sure to label states A for accept, R for Reject, S for Stupid.)

(a) Draw a DFA for

\[ \{(x, X) : x + 1 \in X\}. \]

(b) Draw a DFA for

\[ \{(x, X) : x + 1 \notin X\}. \]

(c) Draw a DFA for

\[ \{(x, X) : x + 100 \in X\}. \]

(You may use DOT DOT DOT notation.)

(d) Draw a DFA for

\[ \{X : X \text{ has exactly 3 elements}\}. \]

(e) Draw a DFA for

\[ \{X : X \text{ does NOT have exactly 3 elements}\}. \]

(f) Draw a DFA for

\[ \{X : X \text{ has exactly 100 elements}\}. \]

(You may use DOT DOT DOT notation.)

**SOLUTION**
(a) This is the DFA.

\[
\begin{array}{ccc}
S & \xrightarrow{0} & R \\
 & \xrightarrow{1} & \xrightarrow{*} A \\
 & \xrightarrow{*} & \xrightarrow{0} R \\
 & \xrightarrow{*} & \xrightarrow{*} A \\
\end{array}
\]

(b) This is the DFA.

\[
\begin{array}{ccc}
S & \xrightarrow{0} & A \\
 & \xrightarrow{1} & \xrightarrow{*} R \\
 & \xrightarrow{*} & \xrightarrow{0} A \\
 & \xrightarrow{*} & \xrightarrow{*} R \\
\end{array}
\]

(c) OMITTED.

(d) This is the DFA.

\[
\begin{array}{cccccc}
R & \xrightarrow{0} & 1 & \xrightarrow{0} & 1 & \xrightarrow{0} R \\
 & \xrightarrow{0} & 1 & \xrightarrow{1} & 1 & \xrightarrow{1} A \\
 & \xrightarrow{1} & 1 & \xrightarrow{1} & 1 & \xrightarrow{*} R \\
\end{array}
\]

(e) This is the DFA.
(f) OMITTED.

END OF SOLUTION

GOTO NEXT PAGE
3. (40 points) Let $M_1, M_2, \ldots$, be a list of Turing Machines.

(a) (20 points) Let $TOT$ be the set of all Turing Machines that halt on ALL inputs.
Find a computable set $B$ of ordered triples such that:

$$TOT = \{ e : (\forall x)(\exists y) [(e, x, y) \in B] \}$$

(This means that $TOT$ is in $\Pi_2$.)

(b) (20 points)
Let $HALT_{100}$ be the set of all Turing Machines that halt on at least 100 inputs.
Find a computable set $B$ of ordered pairs such that:

$$HALT_{100} = \{ e : (\exists x) [(e, x) \in B] \}$$

(HINT: $x$ itself will code many numbers. You need to say how you do that coding.)

SOLUTION

(a) Let $TOT$ be the set of all Turing Machines that halt on ALL inputs.
Find a computable set $B$ of ordered triples such that:

$$TOT = \{ e : (\forall x)(\exists y) [(e, x, y) \in B] \}$$

(This means that $TOT$ is in $\Pi_2$.)

ANSWER

$$TOT = \{ e : (\forall x)(\exists y) [M_e(y) \text{ halts within } y \text{ steps}] \}$$

So

$$B = \{(e, x, y) : M_e(x) \text{ halts within } y \text{ steps}\}$$
Let $HALT_{100}$ be the set of all Turing Machines that halt on at least 100 inputs.

Find a computable set $B$ of ordered pairs such that:

$$HALT_{100} = \{ e : (\exists x)[(e, x) \in B] \}$$

(HINT: $x$ itself will code many numbers. You need to say how you do that coding.)

**ANSWER**

Let $p_1, \ldots, p_{101}$ be the first 101 primes. We code 101 numbers $(a_1, \ldots, a_{101})$ by $\prod_{i=1}^{101} p_i^{a_i}$.

To put this another way, the function $\pi_i(x)$ is the exp of $p_i$ in $x$.

$$HALT_{100} = \{ e : (\exists x)[\text{for all } 1 \leq i \leq 100 \text{ } M_e(\pi_i(x)) \text{ halts in } \leq \pi_{101}(x) \text{ steps}] \}$$

**END OF SOLUTION**