Algorithms for Regular and CFG Languages

In the following, let $M = (Q, \Sigma, \delta, s, A)$ be a DFA with n states, over an alphabet Σ of constant size.

1 Describe an algorithm for deciding if $L(M) = \emptyset$.

Solution:

if there is no path in the directed graph of M from the vertex s to a state in A, then the language is empty. In graph languages, this is doing a **BFS** in the directed graph from s.

2 Describe an algorithm for deciding if $L(M) = \Sigma^*$.

Solution:

Complement the automata M: $\overline{M} = (Q, \Sigma, \delta, s, Q \setminus A)$, and check if its language is empty using the previous algorithm.

3 Describe an algorithm for deciding if $L(M)$ is finite.

Solution:

A state $q \in Q$ is ***loopy*** if there is a string $w \in \Sigma^*$ such that $\delta^*(q, w) = q$. One can check if a state is loopy, by computing all the states that are reachable from q, and all the states that can reach q (again, this is doing two **BFS**s in the graph and the reverse graph) – if the two sets have a non-empty intersection then the state is loopy. Now, compute for all the states of M if they are loopy or not. Let G_s be set of states that are loopy and reachable from s. Now check if any of the states in G_s can reach an accepting state. If so, the language of M is infinite. Otherwise, it is finite.

4 Given two DFAs M, M' decide if $L(M) \subseteq L(M')$.

Solution:

Using the production construction build an automata for the language $L(M) \setminus L(M')$, and check if it is empty. If it is empty, then the containment holds.

5 Given two DFAs M, M' decide if $L(M) = \overline{L(M')} = \Sigma^* \setminus L(M')$.

Solution:

Using the production construction check if $L(M) \cup L(M') = \Sigma^*$. Similarly, check if $L(M) \cap L(M') = \emptyset$. If both things holds, then the answer is yes.

6 Given a CFG $G = (V, T, P, S)$, decide if $L(G)$ contains any string.

Solution:

A variable $X \in V$ is ***final*** if there is a production $X \to \alpha$, an $\alpha \in T^*$ (i.e., only terminals). Scan the productions and mark all the variables that have such a final production.

Generalizing, a variable is ***final*** if their is a production $X \to \alpha$, an $\alpha \in (V \cup T)^*$, and all the variables appearing in α are final. So, scan all the productions repeatedly, and mark all variables that have such a production as final. Repeatedly do this till an iteration does not mark any new variables as final. If S is final, then $L(G)$ is not empty. Otherwise, it is.

7 Two sets $q, q' \in Q$, are distinguishable, if there exists a string $w \in \Sigma^*$, such that $\delta(q, w) \in A$ and $\delta(q', w) \notin A$ (or vice versa). Show how to compute the set D_0 of all the pairs of states that are distinguishable with strings of length 0.

Solution:

Clearly, two state are distinguishable with a string of length 0 if one is accepting, and the other one is rejecting. As such, we have:

$$D_0 = \left\{ \{p, q\} \;\middle|\; p \in A, q \in Q \setminus A \right\}.$$

8 Let D_i be all the set of pairs of states of M that are distinguishable with strings of length at most i. Show how to compute D_{i+1} from D_i. (Think about $i = 0$ first, and then $i = 1$, etc.)

Solution:

Consider the case $i = 0$. Two states q, q' are distinguishable by a string of length one (i.e., a single character $c \in \Sigma$), if $\delta(q, c) \in A$ and $\delta(q', c) \notin A$ (or vice versa). Namely, the states $\delta(q, c)$ and $\delta(q', c)$ are distinguishable by strings of length 0.

More generally, if q and q' are distinguishable with a string $w = w_1 w_2 \cdots w_i$, then $q_1 = \delta^*(q, w_1)$ is distinguishable from $q'_1 = \delta^*(q', w_1)$, with the string $w_2 w_3 \cdots w_i$. Thus, q and q' are distinguishable from a string of length i if there is a character $c \in \Sigma$, such that $\{\delta(q, c), \delta(q', c)\} \in D_{i-1}$. Formally, we have

$$D_i = D_{i-1} \cup \left\{ \{q, q'\} \;\middle|\; q, q' \in Q, \text{ and } \exists c \in \Sigma \text{ such that } \{\delta(q, c), \delta(q', c)\} \in D_{i-1} \right\}.$$

9 One can show that if $D_i = D_{i+1}$, then D_i is the set of all distinguishable pairs of states of M. Since $|D_i| \leq \binom{n}{2}$, it follows that this happens after at most $O(n^2)$ iterations of the algorithm using the above steps. Let D^* be the set of pairs the first iteration this happens – this is the set of all distinguishable pairs of states of M. Given M and D^*, show how to compute a minimal automata equivalent to M.

Solution:

We need the following two claims:

Claim 0.1. *The set D_i contains all the pairs of states that can be distinguished by strings of length at most i.*

Proof: Boring induction. Omitted. ■

Claim 0.2. *If $D_i = D_{i+1}$ then all the distinguishable pairs of states of M are in D_i.*

Proof: The set D_i contains all the pairs that are distinguishable by strings of length at most i. If the claim is false, then there are two states q, q' that are distinguishable, and their shortest distinguishing string, say $w = w_1 w_2 \cdots w_k$, has length $> i$. (If it was shorter, than the pair would already be in D_i.)

Let $q_t = \delta^*(q, w_1 \cdots w_t)$ and $q'_t = \delta^*(q', w_1 \cdots w_t)$, and observe that their shortest distinguishing string is $w_{t+1} w_{t+2} \cdots w_k$ (if there is a shorter distinguishing string then there is a shortest distinguishing string for q and q'). In particular, q_{k-i-1} and q'_{k-i-1} are distinguishable, and their shortest distinguishable string is of length $i+1$. But this implies that $\{q_{k-i-1}, q'_{k-i-1}\} \in D_{i+1} \setminus D_i$, which is a contradiction. ■

The algorithm to create the minimal automata is now straightforward – assume the states of $Q = \{q_1, q_2, \ldots, q_n\}$. For $q_i \in Q$, let $f(q_i)$ be the first state in Q that is NOT distinguishable from q_i. Formally, $f(q_i) = q_j$, if $q_1, \ldots q_{j-1}$ are distinguishable from q_i (we can test this, since this happens if $\{q_1, q_i\}, \ldots \{q_{j-1}, q_i\} \in D^*$ but $\{q_j, q_i\} \notin D^*$. Te new automata now has the state space

$$Q' = \{ f(q) \mid q \in Q \},$$

the start state $s' = f(s)$, the transition function

$$\delta'(q, c) = f\Big(\delta\big(q, c\big)\Big).$$

And the set of accepting states is $A' = \{f(q) \mid q \in A\}$.

It is straightforward to prove that $M' = (Q', \Sigma, s', \delta, A')$ has the same language is M. The proof that is minimal follows from the Myhill-Nerode theorem (or the homework problem proving it), and is omitted here.