15-251: Great Theoretical Ideas In Computer Science

Recitation 2 Solutions

Announcements

- Congrats on finishing the first HW! :)
- Solution session for HW 1 is 12:30pm Sunday, Gates 4303
- Regrade requests on HW 1 due by Wednesday

Training Manual

- **Deterministic Finite Automaton (DFA):** A DFA $M$ is a machine that reads a finite input one character at a time in one pass, transitions from state to state, and ultimately accepts or rejects. Formally, $M$ is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where $Q$ is the finite set of states, $\Sigma$ is the finite alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the starting state, and $F \subseteq Q$ is the set of accepting states.

- **Regular language:** A language $L$ is regular if $L = L(M)$ for some DFA $M$ ($M$ recognizes $L$).

We have shown that if $L_1$ and $L_2$ are both regular languages over $\Sigma^*$, for some fixed $\Sigma$, then the following are all regular.
- $L_1$
- $L_1 \cup L_2$
- $L_1 \cap L_2$
- $L_1 L_2$ (the concatenation of two regular languages)

- **Turing Machine (TM):** A TM $M$ is a machine that can read and write to an infinite tape containing the input, transition from state to state, and ultimately accept, reject, or loop infinitely. Formally, $M$ is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$, where:
  - $Q$ is the finite set of states,
  - $\Sigma$ is the finite input alphabet with $\sqcup \notin \Sigma$,
  - $\Gamma$ is the finite tape alphabet with $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
  - $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function,
  - $q_0 \in Q$ is the starting state,
  - $q_{acc} \in Q$ is the accepting state,
  - and $q_{rej} \in Q, q_{rej} \neq q_{acc}$ is the rejecting state.

- **Decider TM:** A TM $M$ is a decider if it halts on all inputs.

- **Decidable language:** A language $L$ is decidable (or computable) if $L = L(M)$ for some decider TM $M$. 
**Drawing DFAs**

(a) Draw a DFA recognizing the language \( L \) over \( \{a, b\} \) where \( L \) is the set of strings that begin and end with the same character.

(b) Draw a DFA that recognizes the language

\[ L = \{ x : x \text{ has an even number of } 1\text{s and an odd number of } 0\text{s} \} \]

over the alphabet \( \Sigma = \{0, 1\} \).

We have 4 states, each representing having seen an even/even 0/1s, even/odd 0/1s, odd/even 0/1s, and odd/odd 0/1s, and transition accordingly.

**A Santa Lived As a Devil At NASA!**

Show that, if \(|\Sigma| > 1\), then

\[ \text{PAL} = \{ x \mid x \in \Sigma^* \text{ and } x = x^R \} \]

is an irregular language, where \( x^R \) denotes the reverse of the string \( x \).

Assume for sake of contradiction that there exists a DFA with \( k \) states that decides PAL. Consider two symbols \( a, b \in \Sigma \) (since we assumed \(|\Sigma| > 1\)). Take the strings \( b^i a \) for \( i \in \{0, \ldots, k\} \). Since there are only \( k \) states, by the Pigeonhole Principle, there must exist some \( i, j \), \( 0 \leq i < j \leq k \) such that \( b^i a \) and \( b^j a \) end in the same state. Thus, \( b^i a b^j \) and \( b^j a b^i \) must end in the same state.

However, since the first string is a palindrome while the latter is not, the two strings must end in different states. This is a contradiction.
Reversing Regular Languages

If $A$ is a regular language over $\Sigma$, then show that $A^R$ (the reversal of $A$) is regular by providing a DFA for it.

Let $A$ be given. Since $A$ is regular, we know there exists some DFA $M = (Q, \Sigma, \delta, q_0, F)$ which accepts only strings from $A$. We want to find a DFA $M' = (Q', \Sigma, \delta', q'_0, F')$ which accepts only strings from $A^R$.

Consider $Q' = \mathcal{P}(Q)$, $q'_0 = F$, and $F' = \{S \in Q'|q_0 \in S\}$. So every state in $M'$ is a subset of states from $M$, the starting state of $M'$ is the set of accepted states from $M$, and any state in $M'$ which contains the starting state from $M$ is an accepted state. The intuition here is that we want to work back from the accepted states of $M$ to the starting state of $M$ when reading our reversed string.

Define $\delta' : (Q', \Sigma) \mapsto Q'$ as:

$$\delta'(S, c) = \bigcup_{s \in S} \delta^{\leftarrow}(s, c)$$

Where $\delta^{\leftarrow}$ is the function that maps a state from $M$ and a character from $\Sigma$ to the largest set $S' \subset Q$ such that $\forall s' \in S', \delta(s', c) = s$.

Balance in All Things

Construct a TM that decides the language $L = \{x :$ the parentheses in $x$ are balanced$\}$ over the alphabet $\Sigma = \{(,)\}$.

In $q_0$, we find an unmatched right parenthesis in the input and mark it with $X$. In $q_1$, we find the matching left parenthesis to that right parenthesis we just found, and mark that with $X$ as well. We repeat the process until we reach either end of the input. If we find an unmatched right parenthesis, we reject. Otherwise, we check the tape from right to left in $q_3$, and accept iff there is no left parenthesis remaining.
Multiple Multiples (Extra Problem)

Let $\Sigma = \{0, 1\}$. For each $n \geq 1$, define

$$C_n = \{x \in \Sigma^* \mid x \text{ is a binary number that is a multiple of } n\}.$$ 

Show that $C_n$ is regular for all $n$.

At a high level, we wish to have states corresponding to different remainders modulo $n$, and for a string corresponding to binary number $w$ to end on state $q_i$ if $w \equiv i \pmod{n}$. To this end, let $Q_n = \{q_{\text{init}}, q_0, q_1, \ldots, q_{n-1}\}$, and set $F = \{q_0\}$. Define $\delta_n$ such that $\delta(q_{\text{init}}, 0) = q_0, \delta(q_{\text{init}}, 1) = q_1, \delta(q_i, 0) = q_{2i}$, and $\delta(q_i, 1) = q_{2i+1}$, where indices are taken modulo $n$. We claim that $\delta$ transitions us accordingly, i.e. $M_n = (Q_n, \Sigma, \delta_n, q_{\text{init}}, F)$ decides $C_n$.

We can show this via induction on the length $k$ of an input word (ignoring $k = 0$, which is clear). The base case $k = 1$ follows from the fact that the input 0 ends in $q_0$ and the input 1 ends in $q_1$. For induction, assume that for all strings $w$ of length $k$, running $M_n$ on $w$ will end on the state $q_i$, where $i \equiv w \pmod{n}$. Let $w = a_1a_2\ldots a_{k+1}$ be a string of length $k + 1$, and let $u = a_1a_2\ldots a_k$. Treating these as binary numbers, we have $w \equiv 2u + a_{k+1} \pmod{n}$. By induction, running $M_n$ on $u$ ends on state $q_i$, where $i \equiv u \pmod{n}$. Hence, running $M_n$ on $u0$ ends on state $q_{2i}$ and running $M_n$ on $u1$ ends on state $q_{2i+1}$; we conclude that running $M_n$ on $ua_k$ ends on state $q_{2i+a_k}$. Since $2i + a_k \equiv 2u + a_k \pmod{n}$, this proves the inductive claim, and so $M_n$ decides $C_n$, as desired.