Computer Science 170 (Summer 2011) Problem Set 1

Introduction to the Theory of Computation

Due Wednesday, June 8

Exercise 1. Let $\Sigma = \{0, 1\}$. For each of the following languages, construct a deterministic finite automaton that accepts the language:

a. $\{w \in \Sigma^* \mid \text{each 0 in } w \text{ is immediately preceded and immediately followed by a 1}\}$

b. $\{w \in \Sigma^* \mid w \text{ has 0101 as a substring}\}$

c. $\{w \in \Sigma^* \mid \text{the 10th symbol from the right end of } w \text{ is a 1}\}$. (Hint: this one may be hard to draw out explicitly as a graph—try and describe pieces of the graph. If the DFA describes a *program*, what is its underlying structure?)

Exercise 2. Let $\Sigma = \{0, 1\}$. Construct a deterministic finite automaton accepting all strings which, when interpreted as binary numbers, are divisible by 5. How many states are in your DFA, and can you identify what they are encoding? (Hint: think about the process of long division.)

Exercise 3. Let $L \subseteq \Sigma^*$ be an arbitrary language. We define

$$\begin{aligned}
L^R &= \{x^R = x_n x_{n-1} \cdots x_1 \mid \exists x = x_1 x_2 \cdots x_n \in L, x_i \in \Sigma\} \\
\mathit{prefix}(L) &= \{x' = x_1 x_2 \cdots x_j \mid \exists x = x_1 x_2 \cdots x_n \in L, x_i \in \Sigma, 0 \leq j \leq n\} \\
\mathit{suffix}(L) &= \{x' = x_j x_{j+1} \cdots x_n \mid \exists x = x_1 x_2 \cdots x_n \in L, x_i \in \Sigma, 1 \leq j \leq n+1\}
\end{aligned}$$

Prove that if L is a regular language, so too are L^R, $\mathit{prefix}(L)$, and $\mathit{suffix}(L)$. Can you define $\mathit{suffix}(L)$ purely in terms of the other two definitions? If the answer is "yes," do so. (Nondeterminism makes this exercise easier.)

Exercise 4. Let L_1 and L_2 be regular languages. Define the *shuffle* of the two languages to be defined as

$$L_1 \parallel L_2 = \{x_1 y_1 x_2 y_2 \cdots x_n y_n \mid x_1 x_2 \cdots x_n \in L_1, y_1 y_2 \cdots y_n \in L_2\}.$$

Show that the regular languages are closed under the shuffle operator. *(Hint: try messing around with the DFAs recognizing the two languages to get a DFA recognizing the shuffle. If Q_i are the states in the DFA recognizing L_i, try $Q = Q_1 \times Q_2 \times \{1, 2\}$, and use that last "bit" $\{1, 2\}$ of the state to do "switching" between the two machines, and consequently, the two languages.)* You should be able to solve this problem with a deterministic DFA.

Exercise 5. Let $\Sigma = \{a, b\}$. Construct *nondeterministic* finite automata accepting each of the following languages:

a. The set of strings in Σ^* such that some two as are separated by a string whose length is $4i$, for some $i \geq 0$.

b. The set of all strings of as and bs such that the 10th symbol from the right end is a b.

Exercise 6. Now let's try another definition of the shuffle of two regular languages L_1 and L_2: we let $z = z_1 z_2 \cdots z_{m+n} \in L_1 \parallel L_2$ if there exist $x = x_1 x_2 \cdots x_m \in L_1$ and $y = y_1 y_2 \cdots y_n \in L_2$ where x and y can be "merged" or "shuffled" to form z.

The shuffled word $z = z_1 z_2 \cdots z_{m+n}$ can be defined more formally (but maybe not as intuitively) as: there exists a decomposition of the set $S = \{1, 2, \ldots, m+n\}$ into disjoint $I = \{i_1 < i_2 < \cdots < i_m\}$ and $J = \{j_1 < j_2 < \cdots < j_n\}$ so that $I \cup J = S$, $I \cap J = \emptyset$, and $x = z_{i_1} z_{i_2} \cdots z_{i_m}$, $y = z_{j_1} z_{j_2} \cdots z_{j_n}$. Are the regular languages still closed under this definition of shuffle? Show how to construct an NFA for this language that *guesses* whether the next symbol comes from a word in L_1 or a word in L_2. (Nondeterminism helps.)

Some Additional Stuff

Exercise 7. Prove for any integer $n \geq 1$ that there are languages accepted by an NFA with $n+1$ states, but *requiring* DFAs with *at least* 2^n states. *(Difficult, but not too hard if you follow this hint: let $\Sigma = \{0, 1\}$, and consider the language given by the regular expression $(0 \cup 1)^* 1 (0 \cup 1)^{n-1}$. Look at an arbitrary DFA with less than 2^n states purporting to accept this language, and use the Pumping Lemma or a similar pigeonhole-type argument, watching what the DFA does with the 2^n inputs of length n.)*

Exercise 8. Suppose $\delta : Q \times \Sigma \longrightarrow Q$ is the transition function of a DFA, and $\hat{\delta} : Q \times \Sigma^* \longrightarrow Q$ is its extension defined as

$$\begin{aligned} \hat{\delta}(q, \epsilon) &= q, \\ \hat{\delta}(q, wa) &= \delta(\hat{\delta}(q, w), a) \end{aligned}$$

for $a \in \Sigma$, $w \in \Sigma^*$. Prove that for any strings $x, y \in \Sigma^*$, $\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$. (Hint: use induction on $|y|$.)

Exercise 9. A set of integers is *linear* if it is of the form

$$\{c + pi \mid i = 0, 1, 2, 3, \ldots\} = \{c, c+p, c+2p, c+3p, \ldots\}$$

for fixed integers $c, p \geq 0$. A set is *semilinear* if it is the union of a *finite* number of linear sets. Let R be a regular language over the *one*-letter alphabet $\Sigma = \{0\}$. Prove that $\{j \mid 0^j \in R\}$ is semilinear. *Hard way: prove by induction on regular expressions. Easy way: what can a DFA for this language look like if the alphabet has only one letter? Try drawing the DFA—what does it look like? How many edges go out of a node? Where are the final states?*