ECS120 Fall 2006

Discussion Notes

October 10, 2006

Announcements

- **Homework 1 Grades:** If you have any questions or comments on your homework 1 grades, please see me during office hours. If you haven't received your homework, please see Professor Filkov to pick it up.
- **Homework 2 Problem 4:** Since we did not get to regular expressions during Tuesday's lecture, you do not have to provide a regular expression for homework 2 problem 4. (Furthermore, I will not grade or correct any regular expressions provided for that problem.)
- **A Note on Notation:** You are expected to have read Chapter 0 in your book, which introduces the notation used for this class. If you see notation that you are unfamiliar with, Chapter 0 is a good place to look. You might also want to consult the assigned reading in the book.
- **Show versus Prove:** There is a difference between *showing* that something is true versus *proving* something true. When showing something true, you need to provide a discussion or diagram illustrating the truth of the claim, whereas a proof is formal.

  Your argument still needs to be correct, and ideally it should be able to be transformed into a formal proof.

Homework 2 Hints

Problem 1

- Just provide a paragraph (or so) explaining why this is true.

Problem 2

- Essentially, Σ^* is the notation for any string, and Σ^n is the notation for any string of length n. More formal definitions of these operations can be found in your book (try pages 6, 13-14, 44, 64).
- **Part (a):** Try drawing a few cases (for example $n = 1, n = 2$ etc.) and generalize what happens from there.
- **Part (b):** You will want to try and do something similar to proving that a DFA is minimal. The general components of this type of proof includes:
  - Assume there exists a DFA with fewer states.
  - Use the pigeonhole principle to state that at least two states must be combined into one.
  - For every possible pair of strings, show that $\widehat{\delta}(q_0, x_i) \in L_n$ but $\widehat{\delta}(q_0, x_j) \notin L_n$ resulting in a contradiction.

Problem 3

- **Part (a):** The important thing is to provide a proper formal definition. This includes defining the function f_M.
- **Part (b):** It is important that you present the general idea. We will not require 100% correctness of your Moore machine. It must be consistent with your formal definition, and your conventions must be well-defined.

  You may also want to refer to example 1.13 and 1.15 in the book (pages 39-40).

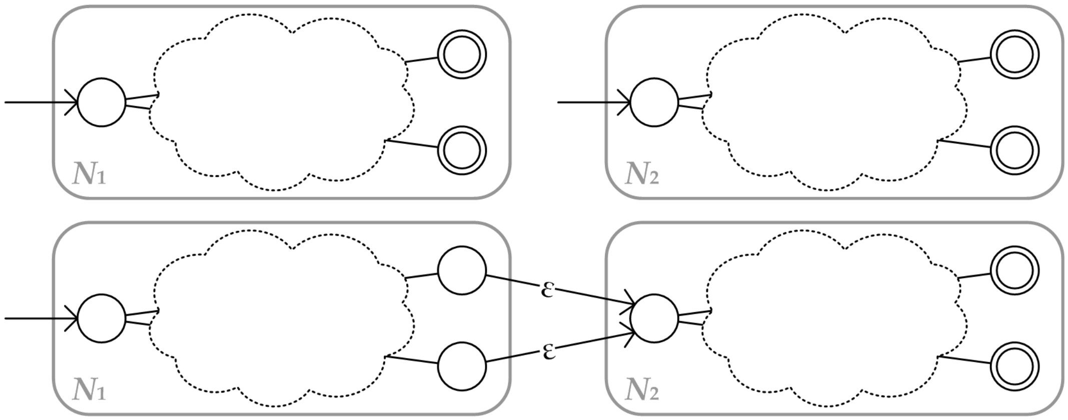
Problem 4

- If you want partial credit for an incorrect DFA, be sure to give me your work!
- The algorithm for doing this was given in class. You can also find it on page 55 of your book.
- As mentioned before, you do not need to give a regular expression.

Bonus Problem

- I will not answer any questions on the bonus problem, unless it is for clarification of the problem itself.
- Chapter 0 introduces the notation x^R, which denotes the **reverse** of x (page 14).

Concatenation



Let $L_1(N_1)$ and $L_2(N_2)$ where $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

We construct the NFA N_3 such that $L_3(N_3) = L_1 \circ L_2$ as follows:

- $Q_3 = Q_1 \cup Q_2$ is set of states
- Σ is the alphabet
- δ_3 is the transition function such that for any $q \in Q_3$ and $a \in \Sigma_\epsilon$:

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{where } q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & \text{where } q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q,a) \cup \{ q_2 \} & \text{where } q \in F_1 \text{ and } a = \epsilon \\ \delta_2(q,a) & \text{where } q \in Q_2 \end{cases}$$

- q_1 is the start state, and
- F_2 is the set of accept states

Thus we have $N_3 = (Q_3, \Sigma, \delta_3, q_1, F_2)$. This is also given in the book on page 61.

Other Set Operations

Why don't we talk about other set operations?

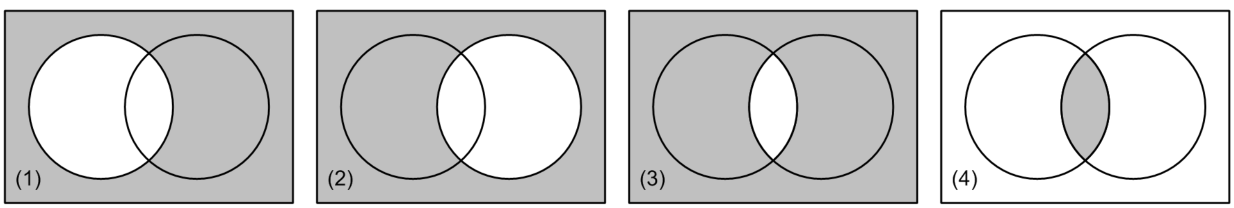
Complement

We already showed how to take the complement of a DFA earlier in class.

Intersection

Suppose we have two languages L_1 and L_2. We can determine $L_1 \cap L_2$ by:

1. Find the complement of L_1.
2. Find the complement of L_2.
3. Find the union of the complement of L_1 with the complement of L_2.
4. Take the complement of the union.



DFAs versus NFAs

A **deterministic finite automaton** (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:

1. Q is a finite set of states
2. Σ is a finite set of symbols called the alphabet
3. $\delta : Q \times \Sigma \to Q$ is the transition function
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accept states

A **nondeterministic finite automaton** (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:

1. Q is a finite set of states
2. Σ is a finite set of symbols called the alphabet
3. $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ is the transition function
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accept states

* The **power set** $\mathcal{P}(Q)$ is the set of all subsets of Q, as described in the book on page 6.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA and $w = w_1 w_2 \cdots w_n$ be a string where each $w_i \in \Sigma$.

Then M **accepts** w if a sequence of states $r_0, r_1, \ldots, r_n$ where each $r_i \in Q$ exists with three conditions:

1. $r_0 = q_0$
2. $r_{i+1} = \delta(r_i, w_{i+1})$ for $i = 0, \ldots, n-1$
3. $r_n \in F$

Let $N = (Q, \Sigma, \delta, q_0, F)$ be a NFA and $w = y_1 y_2 \cdots y_m$ be a string where each $y_j \in \Sigma_\epsilon$.

Then N **accepts** w if a sequence of states $r_0, r_1, \ldots, r_m$ where each $r_j \in Q$ exists with three conditions:

1. $r_0 = q_0$
2. $r_{j+1} \in \delta(r_j, y_{j+1})$ for $j = 0, \ldots, m-1$
3. $r_m \in F$

Example 1.13 and 1.15

(Sometimess it may be easier to define the transition function directly versus drawing a state diagram.)

Let $\Sigma = \{ \langle\text{RESET}\rangle, 0, 1, 2 \}$. For each $i \geq 1$ let A_i be the language of all strings where the sum of the numbers is a multiple of i, except that the sum is reset to 0 whenever the symbol $\langle\text{RESET}\rangle$ appears.

For example if $i = 3$ we want to accept strings such as $11\langle\text{RESET}\rangle12$. This essentially accepts strings whose sum is $0 \mod 3$.

Let each state represent the sum modulo 3. Therefore the start state q_0 represents $0 \mod 3$. If we see a 0 our sum is still $0 \mod 3$ so we stay in q_0. If we see a $\langle\text{RESET}\rangle$, the sum is reset to 0 so we stay in q_0. If we see a 1 our sum is now $1 \mod 3$ so we must move to q_1. If we see a 2 our sum is now $2 \mod 3$ so we must move to q_2.

Eventually, we end up with the following:

δ	0	1	2	$\langle\text{RESET}\rangle$
q_0	q_0	q_1	q_2	q_0
q_1	q_1	q_2	q_0	q_0
q_2	q_2	q_0	q_1	q_0

Notice that on a 0 we always stay in the same state, and on a $\langle\text{RESET}\rangle$ we always go back to q_0.

To generalize this, we build a machine B_i as follows:

- $Q_i = \{ q_0, q_1, \ldots, q_{i-1} \}$ is the set of states
- $\Sigma = \{ \langle\text{RESET}\rangle, 0, 1, 2 \}$ is the alphabet
- For each $q_j \in Q_i$ we define:

$$\begin{aligned}
\delta_i(q_j, 0) &= q_j \\
\delta_i(q_j, 1) &= q_k \text{ where } k = j + 1 \mod i \\
\delta_i(q_j, 2) &= q_k \text{ where } k = j + 2 \mod i \\
\delta_i(q_j, \langle\text{RESET}\rangle) &= q_0
\end{aligned}$$

- q_0 is the start state
- $F_i = \{ q_0 \}$ is the set of accept states

Thus giving us $B_i = (Q_i, \Sigma, \delta_i, q_0, F_i)$.