

# CS 475: Formal Models of Computation

## Homework 1

Due *in class* on Thursday, February 05, 2009

1. [6 points] Let  $L_1$ ,  $L_2$ , and  $L_3$  be some languages over an alphabet  $\Sigma$ . Define

$$\text{Vote}(L_1, L_2, L_3) = \{w \mid w \text{ is in at least two of } \{L_1, L_2, L_3\}\}.$$

Prove that  $\text{Vote}(L_1, L_2, L_3)$  is regular if  $L_1$ ,  $L_2$ , and  $L_3$  are regular.

**Answer.** Regular sets are closed under union and intersection. Therefore, if  $L_1$ ,  $L_2$ , and  $L_3$  are regular, then the language  $\text{Vote}(L_1, L_2, L_3) = (L_1 \cap L_2) \cup (L_1 \cap L_3) \cup (L_2 \cap L_3)$  is regular too.

2. [10 points] A regular expression is in *disjunctive normal form* if it is of the form  $(\alpha_1 \cup \alpha_2 \cup \dots \cup \alpha_n)$  for some  $n \geq 1$ , in which none of the  $\alpha_i$ , for  $1 \leq i \leq n$ , contains an occurrence of  $\cup$ . Show that every regular language can be represented with a disjunctive normal form.

**Answer.** Use induction on the structure of a regular expression  $R$ :

*Base cases:* For  $R = \epsilon$ ,  $R = \emptyset$ , or  $R = a$ , for any  $a \in \Sigma$ ,  $R$  is in disjunctive normal form since there is not an occurrence of  $\cup$  in  $R$ .

*Induction assumption:* Assume  $R_i = \alpha_1 \cup \alpha_2 \cup \dots \cup \alpha_p$  and  $R_j = \beta_1 \cup \beta_2 \cup \dots \cup \beta_q$  are in disjunctive normal form.

*Inductive step:*

- If  $R = R_i \cup R_j$ , then  $R = \alpha_1 \cup \alpha_2 \cup \dots \cup \alpha_p \cup \beta_1 \cup \beta_2 \cup \dots \cup \beta_q$ , which is in disjunctive normal form.
- If  $R = R_i R_j$ , then  $R = \alpha_1 \beta_1 \cup \alpha_1 \beta_2 \cup \dots \cup \alpha_1 \beta_q \cup \dots \cup \alpha_p \beta_1 \cup \alpha_p \beta_2 \cup \dots \cup \alpha_p \beta_q$  is in disjunctive normal form.
- The case  $R = (R_i \cup R_j)^*$  follows easily using the identity  $R = (R_i \cup R_j)^* = (R_i^* R_j^*)^*$ , which do not contain any  $\cup$ . By the induction assumption,  $R_i^*$  and  $R_j^*$ , can be written in disjunctive normal form, which completes the proof.

3. [12 points] Let  $p > 1$  be an integer, and let  $\Sigma_p = \{0, 1, 2, \dots, p-1\}$ , that is, the set of all non-negative integers less than  $p$ . Let  $f_p : \mathbb{N} \cup \{0\} \rightarrow \Sigma_p^*$  be a function that maps a non-negative integer to its representation in base  $p$  as a string. For example, if  $p = 3$ , and  $m = 5$ , both in decimal base, then  $f_p(m) = f_3(5) = 12$ . Assume  $f_p$  maps to the shortest representation in base  $p$ , that is, with no leading zeroes.

Show that for any integers  $q > 0$  and  $p > 1$ , and for any integer  $r$  such that  $0 \leq r < q$ , that there is a DFA that accepts the following language:

$$L(p, q, r) = \{w = f_p(m) \mid m \in \mathbb{N} \cup \{0\}, m \equiv r \pmod{q}\}.$$

Assume that the first character of the string feed to the DFA is the most significant digit.

**Answer.** In the following, we use the following notation: a number  $x \in \mathbb{Z}$ , has a string representation in base  $p$  as  $\mathbf{x}$ . This distinction is important, since  $xy$  means  $x \times y$ , while  $\mathbf{xy}$  is concatenation of strings.

We construct a DFA  $M = (Q, \Sigma, \delta, q_{start}, \{q_{accept}\})$  that decides  $L$ . The alphabet  $\Sigma$  is the set of digits in base  $p$ . Let  $Q = \{q_0, q_1, \dots, q_{q-1}, q_{start}\}$  be a set of  $q$  states. For every  $q_i \in Q - q_{start}$ , and every  $\mathbf{a} \in \Sigma$ , we define the transition function  $\delta$  as follows:

$$\delta(q_i, \mathbf{a}) = q_j, \text{ with } j \equiv ip + a \pmod{q}.$$

Also, for  $q_{start}$ , and every  $\mathbf{a} \in \Sigma$ :

$$\delta(q_{start}, \mathbf{a}) = q_a.$$

To complete the construction of  $M$ , we define  $q_{accept} = q_r$ .

We need to prove that  $M$  decides  $L$ . Let  $\mathbf{w}$  be the input string. In the following we assume that  $|\mathbf{w}| > 0$ , since if  $|\mathbf{w}| = 0$ ,  $w$  is undefined, the state of  $M$  is  $q_{start}$ , and  $M$  correctly rejects  $\mathbf{w}$ .

We prove that after reading the input string  $\mathbf{w}$ ,  $w \equiv j \pmod{q}$  if and only if  $M$  ends up in state  $q_j \in Q - q_{start}$ .

- If  $w \equiv j \pmod{q}$ , then the final state of  $M$  is  $q_j$ : By induction on the length of  $\mathbf{w}$ .
  - *Base case:* If  $|\mathbf{w}| = 1$ , then  $\mathbf{w} = \mathbf{a}$  for some  $\mathbf{a} \in \Sigma$ , and  $w \equiv a \pmod{q}$ . From the transition function we have  $\delta(q_{start}, \mathbf{a}) = q_a$ .
  - *Induction assumption:* Let  $\mathbf{w} = \mathbf{xa}$ , for  $0 \leq a < p$ . If  $x \equiv r_x \pmod{q}$ , then after reading  $\mathbf{x}$ ,  $M$  is in state  $q_{r_x}$ .
  - *Inductive step:* Since  $x \equiv r_x \pmod{q}$ , for some  $0 \leq r_x < q$ , we can write  $x = dq + r_x$ , for some  $d \in \mathbb{Z}$ . The string  $\mathbf{w} = \mathbf{xa}$  represents the number  $w = xp + a = dqp + r_xp + a$ , and observe that  $w \equiv r_xp + a \pmod{q}$ . From the transition function of  $M$ , we have

$$\delta(q_{r_x}, \mathbf{a}) = q_j, \text{ with } j \equiv r_xp + a \pmod{q}.$$

Therefore,  $w \equiv j \pmod{q}$ , and  $M$  is in state  $q_j$ .

- If the final state of  $M$  is  $q_j$ , then  $w \equiv j \pmod q$ :

This follows from the construction of  $M$ , and we can prove it by contradiction. Let  $w$ , with  $n = |w|$ , be one of the smallest length strings for which the claim fails. When  $n = 1$ ,  $w = a$ , for some  $a \in \Sigma$ , and the final state of  $M$  is  $q_a$ . Therefore,  $n > 1$ . Let  $q_i$  be the state after  $n - 1$  characters have been read from  $w$ . We can write  $w = xa$ , with  $|x| = n - 1$ , and  $a \in \Sigma$ . By assumption,  $x \equiv i \pmod q$ . Since  $x \equiv i \pmod q$ , then  $xp + a \equiv ip + a \pmod q$ . With  $j = ip + a \pmod q$ , automaton  $M$  goes to state  $q_j$  from state  $q_i$  after reading  $a$ . The final state is  $q_j$ , and  $w \equiv j \pmod q$ , which is a contradiction that proves the original claim to be true.

4. [14 points] Toad's kingdom is being attacked by a weasel. King Toad believes that if all of the army of toads croaks at the same time, the chorus produced may sound like the roar of a lion that should scare the weasel away. For this plan to work, the toads have to coordinate quietly in the battle field, and King Toad has thought of the following strategy:

The army of toads has been arranged in a row. King Toad is not quite sure how many toads there are in the army, but it figures it has to be a finite number. In order not to alert the weasel, it has been ordered that a toad can only talk to the toad immediately to its left, and the toad immediately to its right. At the far left of the row there is King Toad himself, and on the far right there is a toad sergeant. Each toad, but the King, is modeled as the finite automaton  $M$  (that is, if there are  $n$  toads, we have  $n$  copies of  $M$ , including the sergeant). The state of each copy of  $M$  is set as a function of its previous state and the state of its immediate left and right neighbors (assume the sergeant knows its role by recognizing the missing neighbor). King Toad is also modeled after automaton  $M$ , but with an extra input that indicates the presence of the weasel. Automaton  $M$  has a special state, called the *CROAKING* state. All toads, start in the same state, called *WAITING*. The first time King Toad detects the weasel, it enters the state *CROAK WHEN READY*.

Assume that all of the toads transition to a new state at the same time. Design the set of states and transition function for  $M$  for each toad and the King, such that for the initial condition described, no matter how many toads are in the King Toad's army, all toads will, at some future time, enter their *CROAKING* states simultaneously once the weasel is detected. (The set of states of  $M$  contains at least the states *CROAKING*, *WAITING*, and *CROAK WHEN READY*, but other states are necessary.)

**Answer.** The original problem is called *the firing squad problem*, which changes King Toad for a general, and the toads for soldiers [ref]. Instead of *CROAKING*, the soldiers have to fire at the same time.

Consider the toads numbered from 1 to  $n$ . Number 1 is the "King Toad", and number  $n$  is the "sergeant", both identified for having only one neighbor. This solution uses a divide-and-conquer approach: if  $n$  is even, then the two middle toads are identified, and each one assumes the role of King Toad for a corresponding half; if  $n$  is odd, then the middle toad assumes the role of King Toad for both halves. After the election of new kings, the previous kings assume the role of a sergeant. Since the problem is now divided in two parts of equal size, if both halves begin synchronized, and the two sub-problems are solved with the same method (and hence the same delay), then all toads will enter the *CROAKING* state simultaneously. In the base case of the recursion, the last kings cannot choose new kings, since all their neighbors are sergeant already. Finally, a sergeant croaks when he detects that his neighbors (or neighbor) have become sergeant too.

The problem is then to determine the middle toads. One approach is to have the king send two “messages”, one of them three times slower than the other, requiring that the sergeant on the other extreme bounce back the incoming message; thus the two messages will meet at the middle soldiers.

In the DFA states, we need to keep record of such conditions as king, sergeant, holding a message going in a specific direction, count the delay of the slow message (a constant), *CROAKING*, *WAITING*, and *CROAK WHEN READY*. The number of states required is limited by possible combinations of these conditions. It is clear that all these functions can be implemented with a finite number of states independent of  $n$ .

There are other ways to determine the middle toads. One way is for each toad (one at a time) to send messages in opposite directions, and check when they return. If the messages return at the same time (or two steps apart), then that toad is the middle toad (or one of the two). Otherwise that toad (after waiting for the second message to return) tells its neighbor to give it a try).