

Solutions for Homework 1  
Course: CS 509  
Foundations of Computer Science

Lecturer: Joe Kilian  
Department of Computer Science  
Rutgers, The State University of New Jersey

September 19, 2006

**Problem 1 (Sipser 1.37).** *Let  $C_n = \{x \mid x \text{ is a binary number that is a multiple of } n\}$ . Show that  $C_n$  is regular for any integer constant  $n \geq 1$ .*

*Proof.* Given a constant  $n$ , design a DFA for  $C_n$  as follows (We read the digits from left to right):

1. The alphabet  $\Sigma = \{0, 1\}$ .
2. There are  $n$  states, denoted as  $s_0, s_1, s_2, \dots, s_{n-1}$ .
3. We define the transition function  $\delta$  for each state  $s_i$  as:

$$\begin{aligned}\delta(s_i, 0) &= s_{2i \bmod n} \\ \delta(s_i, 1) &= s_{2i+1 \bmod n}\end{aligned}$$

4. The start state is  $s_0$  and the only final state is  $s_0$  as well.

**Fact 1.** a number is a multiple of  $n$  if and only if its remainder mod  $n$  equals to 0.

After we explain the transition function, it might be easy to see that current state captures the remainder of "partial number" mod  $n$ . "Partial number" is the number represented by the digits between the left end and

current input head, denoted as  $u$ . As the next digit is exposed, we obtain an updated "partial number", either  $2u$  or  $2u + 1$ , then we update the corresponding remainder. Therefore, according to fact 1, a number is a multiple of  $n$  if and only if it stay at  $s_0$  finally. □

**Problem 2 (Sipser 1.57).** *If  $A$  is any language, let  $A_{\frac{1}{2}-}$  be the set of all first halves of strings in  $A$  so that*

$$A_{\frac{1}{2}-} = \{x \mid \text{for some } y, |x| = |y| \text{ and } xy \in A\}$$

*Show that, if  $A$  is regular, then so is  $A_{\frac{1}{2}-}$ .*

*Proof.* To prove that  $A_{\frac{1}{2}-}$  is regular, it suffices to construct a NFA for it.

Denote the DFA  $M$  for  $A$  as a tuple of five elements,  $(\Sigma, S, \delta, s, F)$ , where  $\Sigma$  is the alphabet,  $S$  is the set of states,  $\delta$  is the transition function,  $s$  is the start state and  $F$  is the set of final states.

The idea is to nondeterministically guess the middle state  $m$  at which  $M$  stays after reading  $x$  as well as a string  $y$  that leads  $M$  to traverse from  $m$  to some final state, and verify these guesses finally.

The NFA looks like:

1. The alphabet is the same.
2. The the set of states contains a special start state  $s'$  and a cross-product  $S \times S \times S$ , where the first part tracks the performance of  $M$  on  $x$ , the second part does the same thing for  $y$  while the third part tests whether the guess of  $m$  is consistent.
3. The set of final states is  $\{ \langle s_i, s_j, s_k \rangle \mid s_i = s_k, s_j \in F \}$
4. The rules of transition function  $\delta'$  are as follows:

$$\begin{aligned} \forall i, \delta'(s', \lambda) &= \{ \langle s, s_i, s_i \rangle \} \\ \forall i, j, k \in [1, |S|], \forall a, \delta'(\langle s_i, s_j, s_k \rangle, a) &= \\ & \{ \langle s'_i, s'_j, s_k \rangle \mid s'_i = \delta(s_i, a), s'_j \in \{ \delta(s_j, b) \mid b \in \Sigma \} \} \end{aligned}$$

□

**Problem 3 (Converse of Sipser 1.57).** *Prove or disprove: If  $A_{\frac{1}{2}-}$  is regular, then  $A$  is regular.*

*Proof.* False. Consider  $A_{\frac{1}{2}-} = \{a^n \mid n \in \mathbb{N}\}$  and  $A = \{a^n b^n \mid n \in \mathbb{N}\}$ .  $\square$

**Problem 4.** *Suppose we have a single-counter finite state machine (it may increment or decrement a counter and may detect whether the counter is 0).*

1. *Show that such a class of automata can accept  $a^n b^n$ .*
2. *Show that such a class of automata can not accept  $ww^R$ .*

*Proof (credit to Joe).* For part 1, the automaton for  $a^n b^n$  is simple. First of all, we could test a string  $x \in a^* b^*$  by DFA. Moreover, when reading an  $a$ , the automaton increments the counter by 1, and does the opposite for  $b$ . At the end, we accept only if the counter is 0.

For part 2, prove it via contradiction. Suppose we have such an automaton and assume that it has only  $k$  states. Consider those strings of length  $k$ , there are  $2^k$  equivalence classes among them with respect to  $ww^R$ . Yet how many equivalence classes could this automaton bring to us at this point? The upper bound for the counter is  $O(k)$ , so combining the number of states, there are only  $O(k^2)$  number of equivalence classes, which is contradictory to the previous conclusion.  $\square$

**Problem 5.** *Show that  $L = a^n b^n$  is accepted by a two-way probabilistic finite automata with acceptance gap  $(\frac{1}{4}, \frac{3}{4})$ .*

*Proof (credit to Joe).* We will mainly talk about the idea and omit the details of designing the actual PFA.

As Joe explained in lecture 2, the PFA basically does two jobs given a string  $x$ . First of all, it checks that whether  $x$  is of form  $a^n b^m$  and  $n = m \pmod{100}$ . If not, PFA just blindly rejects, otherwise, PFA will simulate the game from lecture 2 (see notes.) until both of the following conditions hold:

1. PFA read the whole input.
2. Exactly one group of player wins, where group 1 consists of A and B while C and D form group 2.

Then accept if group 2 wins and reject otherwise.

If the conditions above are not satisfied, PFA will use the two-way property to rewind the input head all the way back to the starting point and do the simulation again till the game ends appropriately.

Therefore, the acceptance probability actually depends on the conditional probability as follows:

$$\begin{aligned} & Pr[\text{PFA accepts} \mid n = m \pmod{100}] \\ = & \frac{Pr[\text{Group 2 wins in the end}]}{Pr[\text{Group 1 wins in the end}] + Pr[\text{Group 2 wins in the end}]} \end{aligned}$$

By the discussion in lecture 2, we know that this would guarantee us an acceptance gap.

□