Special Topics in Complexity Theory, Spring 2006, Yap
Homework 1
Due: Wed Feb 8

INSTRUCTIONS

- Please download the latest copy of Chapter 0 (Intro. to Computability). It should be dated after Feb 1, 2006. These exercises are based on material there. The sections you are responsible for are:
- REMEMBER OUR HOMEWORK RULES: you prepare a solution in writing, and then we will have appointments in which you need to give an oral defence of your solution. I will grade BOTH the oral defence AND your written solution.
- In the following questions, when you describe a dfa or nfa, please use state transition diagrams only. In particular, do not explicitly describe the δ function. Use our convention for indicating start states and final states.

---

1. (10+10 Points)
Construct dfa's to accept the following languages:
(i) $\{w \in \{0,1\}^* : w \text{ has } 0101 \text{ as substring}\}$.
(ii) $\{w \in \{0,1\}^* : w \text{ has neither } 00 \text{ nor } 11 \text{ as substring}\}$.

SOLUTION (i) See Figure 1 for the dfa.

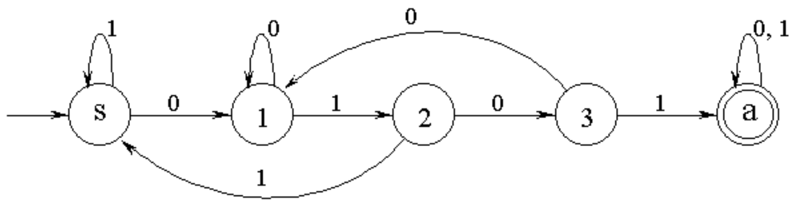

Figure 1: DFA looking for 0101 as substring

Explanation: State q_i ($i = 0, \ldots, 3$ says that the string that has been scanned so far has a maximum length suffix of length i that is equal to a prefix of 0101. E.g., state q_2 says that the string scanned so far contains a suffix of the form 01 (which is a prefix of 0101). An easy to commit error is to jump from q_3 to q_0 when you see a 0.

(ii) See Figure 2 for the dfa.

♠

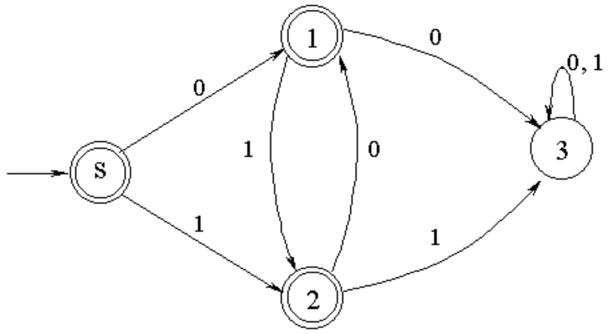

Figure 2: DFA avoiding 00 and 11 as substring

2. (10+10 Points)
   Write regular expressions for (i) and (ii) in the previous question.

   SOLUTION (i) $(0+1)^*0101(0+1)^*$
   (ii) $(01)^* + (01)^*0 + (10)^* + (10)^*1$ ♠

3. (10x4 Points)
   Prove or disprove that each of the following languages is regular:
   (i) $\{1^n : n \text{ is divisible by } 7\}$.
   (ii) $\{w \in \{0,1\}^* : w \text{ is a binary number divisible by } 7\}$.
   (iii) $\{0^n 1 0^m 1 0^{m+n} : n, m \geq 1\}$.
   (iv) $\{ww' \in \{0,1\}^* : w' \text{ is obtained from } w \text{ by interchanging 0 and 1}\}$.

   SOLUTION (i) It is regular, as it is described by the following regular expression (1111111)∗.

   (ii) It is regular as it is recognized by the following DFA M2.

   To understand this dfa, note that the state labeled i (for $i = 0, 1, \ldots, 6$) indicates the binary string read so far is equal to i mod 7. It is easy to verify that each of the transitions is correct. For instance, from state 1, if we see 0, then we go to state 2. This is because if the current string w has a binary value of $7m + 1$, then the new string $w0$ has value $14m + 2$ which is congruent to 2 mod 7.

   COMMENTS: 1. Student's solution may be slightly defective in that their dfa accepts the empty string, which does not represent any binary number. 2. Some students solve a slightly different problem, by assuming that the dfa reads the binary number from the least significant bit (lsb) instead of the msb. We do not take off any points for this, but your dfa will have more states that the above.

   (iii) Use the pumping lemma to prove the language A is not regular. Assume to the contrary that the language A is regular. Let p be the

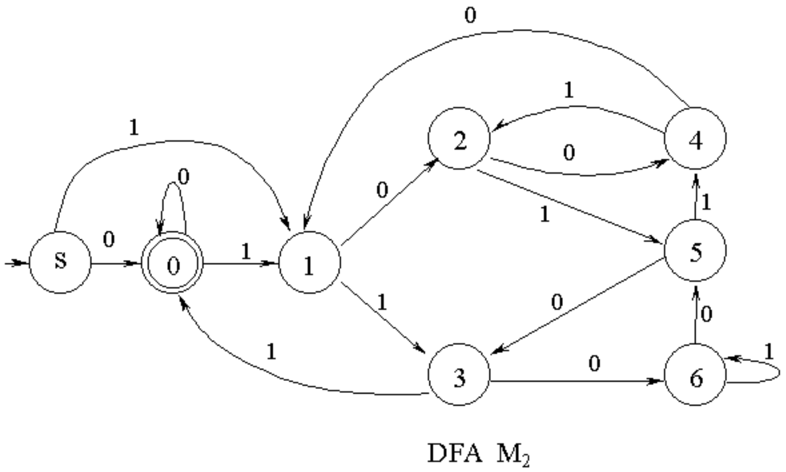

Figure 3: DFA counting mod 7

pumping length given by the pumping lemma. Choose s to be the string $0^p10^m10^{p+m}$. Because s is a member of A and s has length more than p, the pumping lemma guarantees that s can be split into three pieces, $s = xyz$, where for any $i \geq 1$, the string xy^iz is in A. By conditin 3, y consists only of 0's, let $y = 0^k, k > 0$. The string $xy^2z = 0^{p+k}10^m10^{p+m}$ can not be a member of A, as $p + k + m \neq p + m$ for any $k > 0$. Thus we obtain a contradiction.

(iv) Use the pumping lemma to prove the language B is not regular. Assume to the contrary that B is regular. Let p be the pumping length given by the pumping lemma. Choose s to be the string 0^p1^p. Because s is a member of B and s has length more than p, the pumping lemma guarantees that s can be split into three pieces, $s = xyz$, where for any $i \geq 0$ the string xy^iz is in B. By condition 3, y consists only 0's, let it be 0^k where $k > 0$. The string $xy^2z = 0^{p+k}1^p$ can not be a member of B. Thus we obtain a contradiction. ♠

4. (10 Points)
Verify that the STM in Figure 6 of Chapter 0 accepts the marked duplicate language $L^{\#}_{\text{dup}}$.

SOLUTION Actually, it turns out that there is a bug in Figure 6 (that is part of the reason I wanted you to do this exercise).

The idea of the STM in Figure **??** is to "cross out" the bits ($b = 0, 1$) that has been processed, replacing them by an X. Assume the original input has the form $u\#v$ ($uv \in \{0, 1\}^*$). At the beginning of the ith stage, the following conditions hold:

- We are in state q_0.
- The tape contents is $X^i u' \# X^i v'$ where u', v' are suffixes of u, v that remain to be crossed out.
- The tape head is scanning the first symbol of the substring $u'\#$.
- There is a string w such that $wu' = u$ and $wv' = v$.

Now consider the operations of the STM in stage i: If $u' = \epsilon$, then we enter state $q_\sqcup$ and it is easy that we finally enter the accept state iff $v' = \epsilon$. This acceptance is correct. If $u' = 0u''$ then we enter state $q_{0\#}$ which finally moves us past the last X while in state q_{00}. If the next symbol is not 0, we get stuck. Otherwise, we see 0, turn it into an X and move the tape head back back to state q_0. Thus the hypothesis for stage $i+1$ holds. If $u' = 1u''$ then we have a symmetrical situation. In any case, we either enter stage $i+1$, or accept or reject (get stuck).

This proof shows that, at the end of stage i, we accept iff $u' = v' = \epsilon$. This acceptance is clearly correct. Otherwise, we get stuck (also correct) or enter stage $i+1$.

Hence this STM accepts $L_{\mathtt{dup}}$, *assuming* the input has the form $u'\#u$ (stage 0). But it would also accept $\#X^n$. To fix this, we want to first check that the input string has the form $u'\#u$. This can be done by adding the additional states and transitions shown in Figure **??**. If we now make q_{-2} the start state (a violation of our convention of using q_0 as start state), it is then clear that our STM will accept the language $L_{\mathtt{dup}}$.

♠

5. (15 Points)
(i) Let $K_0 = \{L \in RE : \epsilon \in L\}$. Show that $L(K_0)$ is r.e..
(ii) Let $K_1 = \{A_1\}$ where A_1 is any r.e. set. Show that $L(K_1)$ is not r.e..

SOLUTION (i): Easy.
(ii): We reduce a non-r.e. co-HALT to $L(K_1)$. Assume N accepts A_1. For any $i \in \mathbb{N}$, construct M_i such that on input x: runs $U_i(i)$ for x steps. If U_i does not accept, we simulate N on x. If U_i accepts, we "simulates some language that is guaranteed to be different from A_1".

If $t(i)$ is the Gödel number of M_i, we see that $i \notin$ HALT iff $\phi_i(i) \uparrow$ iff $t(i) \in L(K_1)$. ♠

6. (10 Points)
Students often assume that if A is r.e., then any enumeration $\alpha : \mathbb{N} \to A$ is recursive. To dispel this idea, prove the following: if A is infinite and r.e., then there exists a non-recursive denumeration

$$f : \mathbb{N} \to A.$$

REMARK: If A is non-recursive, there is a simple argument. So it may be enough to assume A is recursive.

HINT: Let $\alpha : \mathbb{N} \to A$. be a recursive denumeration of A. Now split A into $A_0 \uplus A_1$, and define $f : \mathbb{N} \to A$ such that $f(i) \in A_0$ iff $i \in$ HALT.

SOLUTION If A is non-r.e. then $f(i)$ can simply be $i + 1$st largest number in A. It is easy to see that $f(i)$ is not r.e., as observed in the REMARK.

Define $\alpha_0(i) = \alpha(2i)$ and $\alpha_1(i) = \alpha(2i + 1)$. Thus $\alpha_j : \mathbb{N} \to A_j$ (for $j = 0, 1$) is a denumeration. We know that there exists denumerations

$$h : \mathbb{N} \to \text{HALT}, \qquad \overline{h} : \mathbb{N} \to \text{co-HALT}.$$

Finally define

$$f(i) = \begin{cases} \alpha_0(h^{-1}(i)) & \text{if } \ i \in \text{HALT} \\ \alpha_1(\overline{h}^{-1}(i)) & \text{else.} \end{cases}$$

Then f is a recursive denumeration of A. It remains to show that if f is recursive, then HALT is recursive. This is because $i \in$ HALT iff $f(i) \in A_0$. To check if $f(i) \in A_0$, we simply dovetail the computations

$$\{\alpha_0(j), \alpha_1(j) : j \in \mathbb{N}\}$$

until we see $f(i)$ being output. Then we know whether $f(i) \in A_0$ or not. ♠

7. (15 Points)
   Use the Rice-Shapiro theorem to show the following sets to be non-r.e.:
   (i) TOT $= \{i \in \mathbb{N} : \phi_i$ is total$\}$
   (ii) co-TOT.

   SOLUTION Here is an attempt to do (i): $TOT = L(K)$ where $K = REC$. Since DIAG $\notin K$ but any finite subset of DIAG is in K, this means that K does not pass the Rice-Shapiro condition. Hence TOT cannot be r.e..

   Unfortunately, this is not quite correct. The opening statement is wrong: TOT is not really of the form $L(K)$ for any class of languages K. So this question is aborted, but you will be graded generously.

   I imagine that some variation of Rice-Shapiro may allow us to prove this problem as stated. We will grade generously for this problem. ♠