

# CS273: Theory of Computation, Summer 2008

## Homework 3

Due 3:00PM Monday, June 30, 2008 at SC 3303

Revision due 3:00PM Thursday, July 3, 2008 at SC 3303

1. [40 points; 10 per part] (Parts from Sipser.) Prove that the following languages are not regular. (You may use all available tools, including the pumping lemma, closure properties of regular languages, and the Myhill–Nerode theorem.)

(a)  $\{0^n 1^m 0^n \mid m, n \geq 0\}$

**Solution:** We can use the pumping lemma game:

- Adversary choose  $p \geq 0$ .
- We'll chose  $w = 0^p 10^p$ , which is in the language.
- The adversary splits  $w$  into  $w = xyz$ . By the constraints of the game, however,  $y$  must be contained within the first block of 0's.
- By choosing  $i = 2$ , we will obtain the string  $xyyz = 0^{p+|y|}10^p$ . Since  $|y| \neq 0$ , the first and second blocks of 0s have different lengths, so the string is not in the language. ■

(b)  $\{w \mid w \in \{0, 1\}^* \text{ is not a palindrome}\}$

**Solution:** Suppose this language is regular. Then its complement (the set of all palindromes over  $\{0, 1\}$ ) is also regular. But we proved in an earlier homework that the set of palindromes is not regular, a contradiction. Therefore, the original language must not have been regular. ■

(c)  $\{wtw \mid w, t \in \{0, 1\}^+\}$

**Solution:** Let  $A$  be the above language, and suppose it is regular. Then  $A' = A \cap L(0^*10^*1)$  is also regular by the closure properties of regular languages.

To see what strings are in  $A'$ , take any string  $x = 0^n 10^m 1 \in L(0^*10^*1)$ , and suppose it can also be written in the form  $x = wtw$ . Note that the last character of  $w$  must be a 1, as it is the last character of  $x$ . Then because  $w$  is also a prefix of  $x$ , the only choice for  $w$  is  $w = 0^n 1$ . Thus  $x = w0^{m-n}w$ , and the only choice for  $t$  is  $0^{m-n}$ .

Thus we observe that  $A' = \{0^n 10^m 1 \mid n < m\}$ . It is straight-forward to show that this language is not regular, which would result in a contradiction to the assumption that  $A$  is regular. A quick way to show that  $A'$  is not regular is to observe that strings of the form  $0^i 1$  must be in different Myhill-Nerode equivalence classes, because  $0^{i+1} 1$  is a distinguishing suffix for  $0^i 1$  and  $0^j 1$ , when  $i < j$ . ■

(d)  $\{xy \in \{0, 1\}^* \mid x = x_1 x_2 \cdots x_n, y = y_1 y_2 \cdots y_n, \text{ and } x_j = y_j \text{ for some } 1 \leq j \leq n\}$

**Solution:** Let  $A$  be the above language, and suppose it is regular. Then the complement of  $A$  must also be regular. It is tricky to write the complement of this language because of the  $xy$  part of the definition, as well as the “for some  $j$ ” condition. Our first step is to rewrite  $A$  more clearly.

First, suppose  $x = x_1 \cdots x_n$  and  $y = y_1 \cdots y_n$ . Then the following conditions are logically equivalent:

$$(\exists j : x_j = y_j) \iff \neg(\forall j : x_j \neq y_j) \iff \neg(x = \bar{y}) \iff x \neq \bar{y}$$

In the last two conditions, we use  $\bar{y}$  to denote the bitwise complement of  $y$ . This makes the definition of  $A$  much simpler. We can also rewrite the  $xy$  condition a little more explicitly, too:

$$\begin{aligned} A &= \{xy \in \{0,1\}^* \mid |x| = |y| \text{ and } x \neq \bar{y}\} \\ &= \{w \mid \text{there exists } x, y \in \{0,1\}^* \text{ such that } w = xy, |x| = |y|, \text{ and } x \neq \bar{y}\} \end{aligned}$$

Now the complement of  $A$  is:

$$\begin{aligned} \bar{A} &= \{w \mid \text{for all ways to write } w = xy, |x| \neq |y| \text{ or } x = \bar{y}\} \\ &= \{w \mid \text{if } w = xy \text{ for } |x| = |y|, \text{ then } x = \bar{y}\} \end{aligned}$$

By our assumption, this is regular. Then if we intersect it with the set of even length strings, we obtain the following language, which must also be regular:

$$A' = \{x\bar{x} \mid x \in \{0,1\}^*\}$$

In other words,  $A'$  is the set of the strings whose first half is the bitwise complement of the second half. We can intersect  $A'$  further with  $L(0^*1^*)$ , and see that the result is  $\{0^n1^n \mid n \geq 0\}$ , which we know is non-regular. This is a contradiction, so the original language must not have been regular. ■

2. [20 points; 10 per part] (Parts from Sipser.) For each of the following languages, decide if the language is regular or not regular. Prove your answers are correct.

(a)  $\{wtw^R \mid w, t \in \{0,1\}^+\}$

**Solution:** This language is regular because it is equal to

$$\{x \in \{0,1\}^+ \mid |x| \geq 3 \text{ and the first and last characters of } x \text{ are the same}\}.$$

Note that if a string has the form  $wtw^R$ , for non-empty  $w, t$ , then the first and last characters are necessarily the same. Likewise, if a string has first and last characters the same, then we can set  $w$  to be that character, and  $t$  to be all but the first and last characters, to write the string as  $wtw^R$ .

A regular expression for this language is  $1(0+1)^*1 + 0(0+1)^*0$ . ■

- (b)  $\{w \mid w \text{ contains the same number of occurrences of } 01 \text{ and } 10 \text{ as substrings}\}$ . (For example, 011010 has two occurrences of 01 (011010) and two occurrences of 10 (011010.)

**Solution:** This language is also regular, and (surprisingly) is equal to:

$$\{x \in \{0, 1\}^* \mid x = \varepsilon \text{ or the first and last characters of } x \text{ are the same}\}$$

Note that the language trivially contains all single-character strings (the first and last positions are the same position).

Each time the string contains a substring 01, this means that the string switches (while reading left-to-right) from a block of 0s to a block of 1s. Similarly, a substring 10 means that we switch from a block of 1s to a block of 0s. Since strings can only contain 0s and 1s, these switches between types of blocks must alternate.

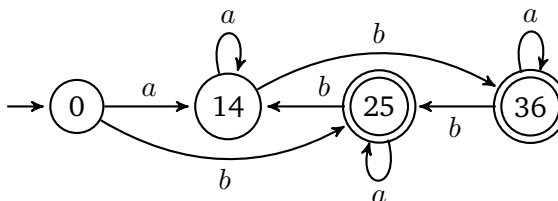
Suppose a string starts with a block of 0s. Then if it is to end with a 0 as well, the string has to switch from a 1-block to a 0-block the same number of times it switched from a 0-block to a 1-block. ■

3. Let  $M$  be the DFA described by the following table:

	$a$	$b$
$\rightarrow 0$	1	5
1	1	6
$F$ 2	2	4
$F$ 3	6	5
4	4	3
$F$ 5	5	4
$F$ 6	3	2

- (a) [15 points] Give the state diagram (or a formal description) of a DFA  $M'$  that is equivalent to  $M$  but has as few states as possible.
- (b) [10 points] Make sure that each state in  $M'$  is labeled with its corresponding set of states in  $M$ .

**Solution:** The only equivalences are:  $1 \equiv 4$ ,  $2 \equiv 5$ , and  $3 \equiv 6$ , so our equivalence classes are  $\{0\}$ ,  $\{1, 4\}$ ,  $\{2, 5\}$ ,  $\{3, 6\}$ . The minimal DFA is as follows:

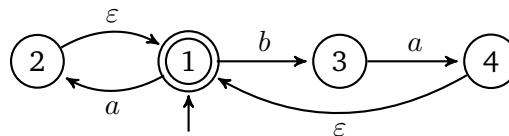


4. [30 points; 15 per part] For these two problems, assume  $\Sigma = \{a, b\}$ . Please show any intermediate steps.

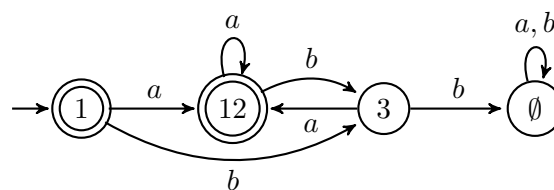
- (a) Give a regular expression for the complement of  $L((a + ba)^*)$ .

**Solution:** Our strategy is to convert between representations (NFA, DFA, regex) as appropriate. The only representation amenable to complementation is the DFA. So we must convert the regular expression to an NFA, and then to a DFA. Then we can complement the language by swapping accept and reject states, then convert the result back into a regular expression.

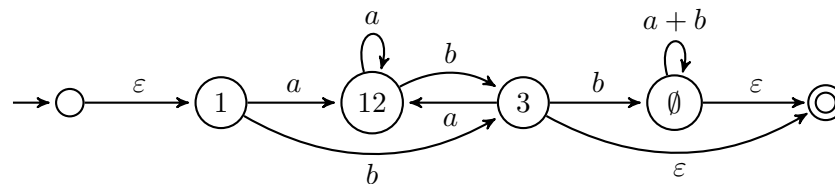
Here's what I came up with when converting this regular expression to an NFA:



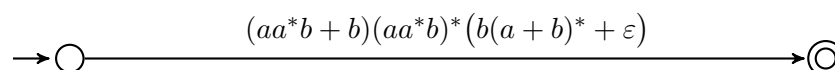
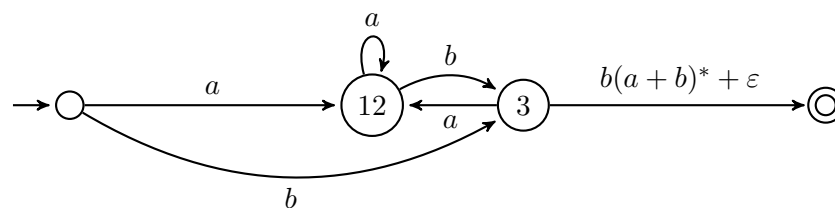
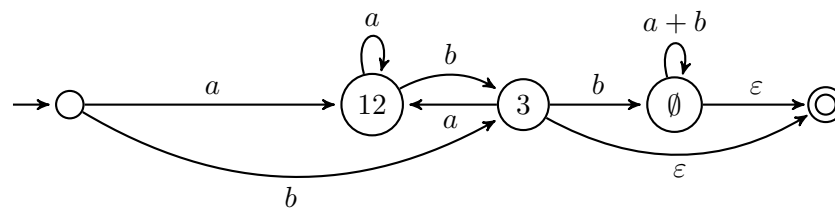
Converting this to a DFA yielded the following (state names correspond to sets of states from the NFA):



Swapping accept and reject state, and then converting to a GNFA yielded:



Finally, removing the states in order 1, empty set, 12, 3 yielded the final result:



Of course, your final answer may have been different depending on how you converted to NFA and how you converted the GNFA to a regular expression. ■

(b) Give a regular expression for the reverse of  $L(((a + b)bb^*)^* + b)$ .

**Solution:** It is possible to perform a similar sequence of conversions, as in part (a). In this case, it is only necessary to convert to an NFA, since we can directly reverse the language by modifying the NFA. However, it is also possible to observe the following inductive procedure to reverse a regular expression:

- $L(\epsilon)^R = L(\epsilon)$
- $L(\emptyset)^R = L(\emptyset)$ .
- $L(a)^R = L(a)$ , for  $a \in \Sigma$
- $L(\alpha\beta)^R = L(\beta)^R \circ L(\alpha)^R$ . (note that  $\alpha, \beta$  switch positions)
- $L(\alpha^*)^R = (L(\alpha)^R)^*$
- $L(\alpha + \beta)^R = L(\alpha^R + \beta^R)$

Note that  $L(\alpha)^R$  can always be expressed in terms of languages  $L(\beta)^R$ , for smaller regular expressions  $\beta$ .

Applying this procedure to the given regular expression, we have

$$L(((a + b)bb^*)^* + b)^R = L((b^*b(a + b))^* + b)$$

■

5. [Honors; 20 points] Let  $A$  be a regular language. Consider the two languages:

$$\begin{aligned} &\{x \mid x^n \in A \text{ for some } n \geq 0\} \\ &\{x^n \mid x \in A \text{ and } n \geq 0\} \end{aligned}$$

One is necessarily regular and one is not. Which is which? Give a proof for one and counterexample for the other.

**Solution:** The first language is regular, and the second is non-regular.

To show that the second language is non-regular, take  $A = L(0^*1)$ , which is regular. Let  $A'$  be the resulting language  $\{x^n \mid x \in A \text{ and } n \geq 0\}$ , and suppose it is regular. Then  $A' \cap L(0^*10^*1)$  is also regular, but it is equal to  $\{0^n10^{n+1} \mid n \geq 0\}$ . It is straight-forward to show that this language is in fact not regular, which is a contradiction. Therefore  $A'$  must not have been regular.

Now, suppose  $A$  is regular and is recognized by a DFA  $M = (Q, \Sigma, \delta, q_0, F)$ . Suppose  $Q = \{q_0, \dots, q_{k-1}\}$ . We will construct a DFA for  $\{x \mid x^n \in A \text{ for some } n \geq 0\}$ .

Our constructed DFA is  $M' = (Q', \Sigma, \delta', q'_0, F')$ , where:

- $Q' = Q^k$ , the tuples of  $k$  states from  $M$ .
- $\delta'((r_0, \dots, r_{k-1}), a) = (\delta(r_0, a), \dots, \delta(r_{k-1}, a))$
- $q'_0 = (q_0, \dots, q_{k-1}) \in Q'$ .

All that remains is to define the set of accepting states  $F'$ . But first, let's discuss what  $M'$  does. After reading input  $w$ , the current state tells us  $r_i = \widehat{\delta}(q_i, w)$  for every  $q_i \in Q$ . Note that we can follow the computations of  $w^n$  in  $M$ , for any  $n$ , by advancing ahead by chunks of  $w$  at a time:  $\widehat{\delta}(q_0, w^n) = \widehat{\delta}(\dots \widehat{\delta}(\widehat{\delta}(q_0, w), w) \dots), w$ .

Therefore, we set  $F'$  to be the set of tuples  $(r_1, \dots, r_k)$  such that there exists indices  $i_1, \dots, i_n$  such that

- $i_1 = 0$
- $r_{i_j} = q_{i_{j+1}}$
- $r_{i_n} \in F$

That is, if there is a way to follow paths  $q_i \rightarrow r_i$  (i.e., read in chunks of  $w$  at a time) from the start state to an accept state.

■