

CS273: Theory of Computation, Summer 2008

Homework 1

Due 3:00PM Monday, June 16, 2008 at SC 3303
Revision due 3:00PM Friday, June 20, 2008 at SC 3303

1. [8 points; 2 per part] Set definitions review. Let $A = \{1, 2, 3\}$, $B = \{1, \{2, 3\}\}$, and $C = \{(x, y) \in \mathbb{Z} \times \mathbb{Z} \mid x \neq y\}$.

(a) What is $A \times B$?

Solution: $\{(1, 1), (1, \{2, 3\}), (2, 1), (2, \{2, 3\}), (3, 1), (3, \{2, 3\})\}$ ■

(b) What is $\mathcal{P}(A) - B$?

Solution: $\{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{1, 2, 3\}\}$ ■

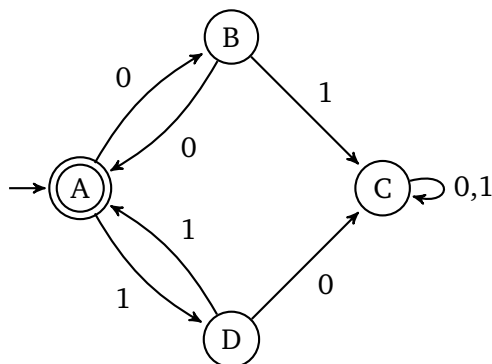
(c) What is $\mathcal{P}(A) - \mathcal{P}(B)$?

Solution: $\{\{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$ ■

(d) What is $A \times A - C$?

Solution: $\{(1, 1), (2, 2), (3, 3)\}$ ■

2. [10 points] Let M be the following DFA.



(a) [3 points] Which state is M in after reading 0011000?

Solution: State B ■

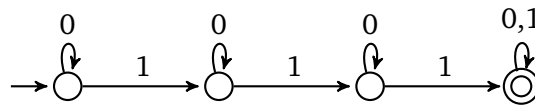
(b) [7 points] What is $L(M)$? (In other words, give a short, self-contained description of $L(M)$ that does not refer to M . You may use set notation or the English language to convey your description.)

Solution: $L(M) = \{00, 11\}^* = \{w_1 \cdots w_{2n} \in \{0, 1\}^* \mid w_{2i} = w_{2i-1} \text{ for } 1 \leq i \leq n\}$ ■

3. [50 points; 10 per part] (Parts (a)-(d) from Sipser.) Give state diagrams of DFAs recognizing the following languages; in all cases, the alphabet is $\{0, 1\}$.

(a) $\{w \mid w \text{ contains at least three 1s}\}$

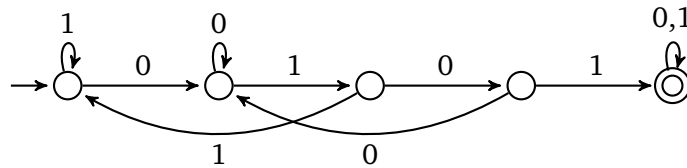
Solution:



■

(b) $\{w \mid w \text{ contains the substring 0101}\}$

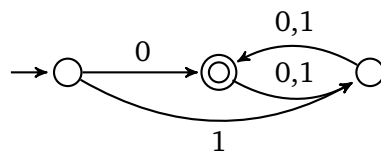
Solution:



■

(c) $\{w \mid w \text{ starts with 0 and has odd length, or starts with 1 and has even length}\}$

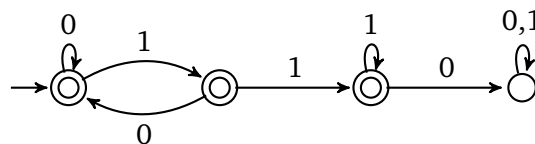
Solution:



■

(d) $\{w \mid w \text{ does not contain the substring 110}\}$

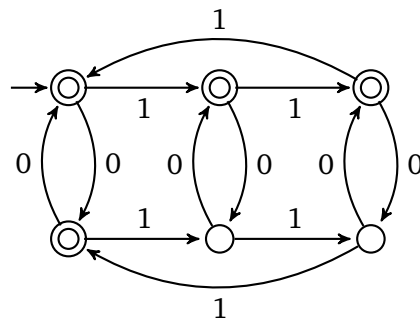
Solution:



■

(e) $\{w \mid w \text{ has an even number of 0s or the number of 1s is divisible by 3}\}$

Solution:



■

4. [30 points; 15 per part] (From Sipser.) For each language A , define

$$\text{NOPREFIX}(A) = \{w \mid w \in A \text{ and no proper prefix of } w \text{ is a member of } A\}$$

$$\text{NOEXTEND}(A) = \{w \mid w \in A \text{ and } w \text{ is not a proper prefix of any string in } A\}.$$

Let A be a regular language that is recognized by a DFA M .

(a) Describe how to modify M to create a DFA M_1 that recognizes the language $\text{NOPREFIX}(A)$.

Solution: The main observation is that once M visits an accepting state for the first time, there is no way any continuation of the input will be in $\text{NOPREFIX}(A)$.

Thus we modify M by adding a new reject state q_{new} with self-loops on every input, and change every outgoing transition from each accepting state to go to q_{new} now instead. More formally, if $M = (Q, \Sigma, \delta, q_0, F)$, then the new DFA is $M_1 = (Q_1, \Sigma, \delta_1, q_0, F)$, where

- $Q_1 = Q \cup \{q_{\text{new}}\}$
- $\delta_1(q, a) = \begin{cases} q_{\text{new}} & \text{if } q \in F \cup \{q_{\text{new}}\} \\ \delta(q, a) & \text{otherwise} \end{cases}$

Alternatively, we can simply *remove* all outgoing transitions from each of M 's accepting states. This results in an NFA (some transitions are now missing) which can then be converted into a DFA. ■

(b) Describe how to modify M to create a DFA M_2 that recognizes the language $\text{NOEXTEND}(A)$.

Solution: Suppose that by reading a string w , we end up in state q in M . The main observation here is that there is a proper extension of w (i.e., a string y such that w is a proper prefix of wy) accepted by the machine if and only if there is a path from q to an accept state (viewing the state diagram as a directed graph).

We modify M by changing its set of accept states to

$$F_2 = \{q \in F \mid \text{there is no path in } M \text{ from } q \text{ to any } q' \in F\}.$$

■

5. [15 points] (From Sipser.) For any string $w = w_1w_2 \cdots w_n$, the *reverse* of w , written w^R , is the string w in reverse order, $w_n \cdots w_2w_1$. For any language A , let $A^R = \{w^R \mid w \in A\}$. Prove that if A is regular, then A^R is also regular.

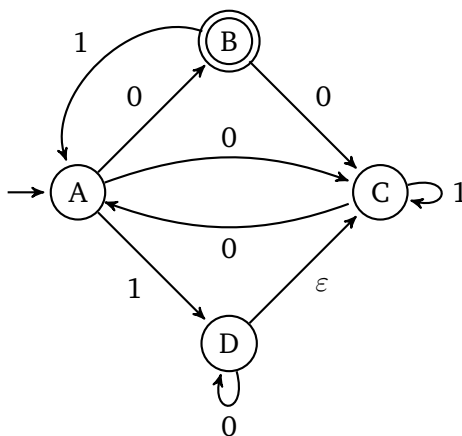
Solution: The idea here is to “run the DFA for A in reverse” – start in the accept states and end in the start state, taking transitions in the reverse direction. The result of “flipping” the transitions may no longer be a DFA, so we will define an NFA. Since we may have multiple accept states but cannot have multiple start states, we have to add a new start state as well.

Formally, if $M = (Q, \Sigma, \delta, q_0, F)$ is a DFA for A , then we define the following NFA $M' = (Q', \Sigma, \delta', q'_0, F')$:

- $Q = Q \cup \{q'_0\}$
- $\delta'(q, a) = \{r \mid \delta(r, a) = q\}$ for $q \in Q$ and $a \in \Sigma$.
- $\delta'(q, \varepsilon) = \emptyset$ for $q \in Q$.
- $\delta'(q'_0, a) = \emptyset$ for $a \in \Sigma$.
- $\delta'(q'_0, \varepsilon) = F$.
- $F' = \{q_0\}$, the original start state.

Our new start state q'_0 does nothing but ε -transition to the original accept states. ■

6. [15 points] Let N be the NFA below.



(a) [12 points] Give a state diagram for a DFA M that recognizes the same language as N .

Solution:

δ	0	1
$\rightarrow \{A\}$	$\{B, C\}$	$\{C, D\}$
$\{B, C\}$	$\{A, C\}$	$\{A, C\}$
$\{C, D\}$	$\{A, C, D\}$	$\{C\}$
$\{A, C\}$	$\{A, B, C\}$	$\{C, D\}$
$\{A, C, D\}$	$\{A, B, C, D\}$	$\{C, D\}$
$\{C\}$	$\{A\}$	$\{A\}$
$\{A, B, C\}$	$\{A, B, C\}$	$\{A, C, D\}$
$\{A, B, C, D\}$	$\{A, B, C, D\}$	$\{A, C, D\}$

States $\{B, C\}$, $\{A, B, C\}$, and $\{A, B, C, D\}$ are accepting. ■

(b) [3 points; 1 per part] Which of the following strings are in $L(N)$?

- i. 00100
- ii. 1001
- iii. 000110

[Hint: Use this part to help check your answer to part (a)].

Solution: Only the first string 00100 is accepted. ■

7. [15 points] For any language A , define

$$\text{EVERYOTHER}(A) = \{w_1w_2 \cdots w_n \mid \text{there exist } x_1, \dots, x_n \text{ such that } x_1w_1x_2w_2 \cdots x_nw_n \in A\}.$$

Prove that if A is regular, then $\text{EVERYOTHER}(A)$ is also regular.

Solution: The idea is that after reading a character w_i , we would like to simulate the machine for A as if it had read two characters x_iw_i . Since we are not given x_i as input in the language $\text{EVERYOTHER}(A)$, and the x_i 's could be any character, we try all possibilities nondeterministically.

Formally, let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA for A . We construct an NFA $M' = (Q, \Sigma, \delta', q_0, F)$ as:

- $\delta'(q, a) = \{\delta(\delta(q, b), a) \mid b \in \Sigma\}$, for all $q \in Q, a \in \Sigma$

- $\delta'(q, \varepsilon) = \emptyset$ for all $q \in Q$.

■

8. [Honors; 20 points] (From Sipser.) If A is any language, we define

$$\text{HALF}(A) = \{x \mid \text{there exists } y \text{ with } |y| = |x|, xy \in A\}.$$

Prove that if A is regular, then $\text{HALF}(A)$ is regular.

Solution: A DFA cannot count characters, so after reading all of x , it cannot be expected to know how many characters were read in order to determine where all suitable y strings might lead. Instead, however, it can simulate one character of y for each character of x it reads, to be sure that its simulation of y used only the correct number of characters.

We do not know *a priori* where the computation of x will end (i.e., where the computation of y should start). However, we can guess this value nondeterministically, and later ensure that we only accept if the computation of x indeed ended at our guess.

Formally, let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA for A . We construct an NFA $M' = (Q', \Sigma, \delta', q'_0, F')$ as follows:

- $Q' = (Q \times Q \times Q) \cup \{q'_0\}$. We will store a triple of states, representing (1) the current computation of x , (2) our guess for where the computation of x will end, and (3) the current computation of y .
- $\delta'(q'_0, \varepsilon) = \{(q_0, r, r) \mid r \in Q\}$. In (other) words, at the very beginning we guess a state r for where the computation of x might end. We start the computation of x at the start state q_0 , and start the computation of y at our guess r .
- $\delta'((p, q, r), a) = \{(\delta(p, a), q, \delta(r, b)) \mid b \in \Sigma\}$. In (other) words, when reading a character, we advance the computation of x by that character, maintain our guess for the end of x 's computation, and advance the computation of y by any nondeterministically chosen character.
- $F' = \{(q, q, r) \mid q \in Q, r \in F\}$. In (other) words, we accept whenever the computation for x ended where we guessed it might end, and the computation of y also ended in an accept state.

An alternative solution (using fewer states) is to imagine that for each character of x (processed left-to-right), we simulate one character in the computation of y , but right-to-left. At the very beginning, we guess an accepting state where y might end. We keep track of the current state of the x -computation, and the left-to-right y -computation, guessing the characters of y each time and taking transitions in reverse (as in problem 5). Whenever these two computations are in the same state, these two paths through the DFA have “met up”, and we can accept. ■