Harvard University
Computer Science 121

Problem Set 1

Tuesday September 22, 2015 at 11:59pm

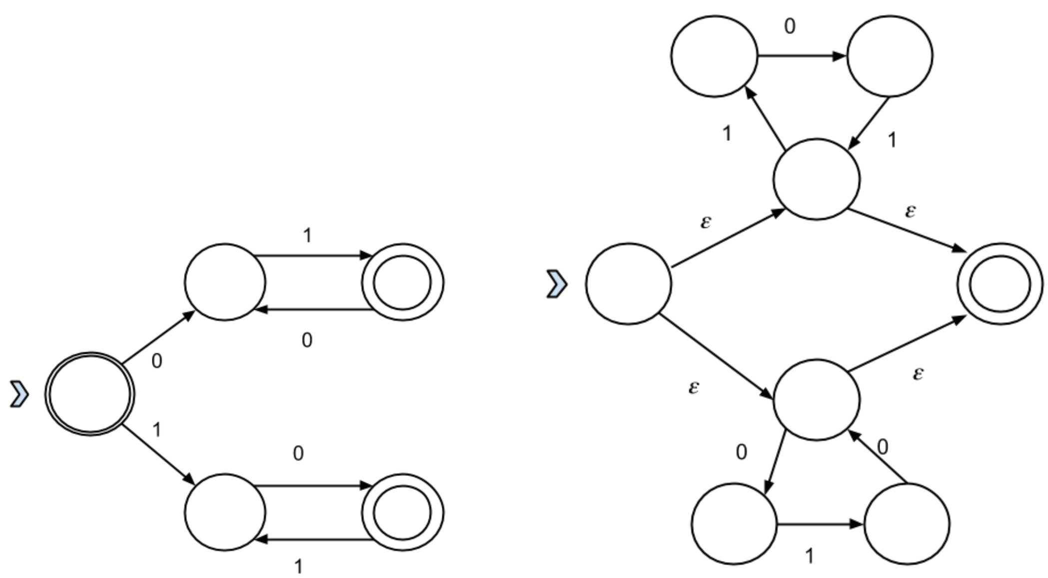
Problem set by Kevin Zhang

Collaboration Statement: I collaborated on this assignment with Tomoya Hasegawa, Jason Shen, and David DiCurcio, got help from no one else, and referred to http://madebyevan.com/fsm/ to draw automata.

Note: Finite automata (FA) drawings may be done by hand or using an online drawing tool.

PART A (Graded by Sam and Serena)

PROBLEM 1 (2+2+1 points, suggested length of 3 lines)



(A) Describe informally L_1, the language accepted by the NFA on the left.

(B) Describe informally L_2, the language accepted by the NFA on the right.

(C) Write down $L_1 \cap L_2$.

Solution.

(A). L_1 is the language of all even-length strings of alternating 0's and 1's.

(B). L_2 is the language of all strings that consist of repeating 101's or 010's.

(C). Since any string in L_2 longer than 3 "letters" must be at least 6 letters, and the first six letters of any string longer than 6 letters in L_2 are either 010010 or 101101 (either repeating 101's or 010's), any string in L_2 six letters or longer cannot also be in L_1 (since there is either a repeating 0 or a repeating 1 in the string). Therefore, the only strings that could potentially be in both L_1 and L_2 are those of length 3 or 0, or $\{010, 101, \varepsilon\}$. However, since L_1 consists only of even-length strings, neither 101 or 010 can be in L_1. Therefore, the only string in both L_1 and L_2 is $\varepsilon : L_1 \cap L_2 = \{\varepsilon\}$.

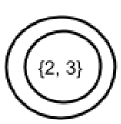
PROBLEM 2 (3+6+(2) points, suggested length of 1 page)

Let $S_n = \{1, 2, 3, ..., n\}$. We can represent subsets of S_n using strings of length n by setting the i^{th} character to 1 if element i is in the subset, and 0 if it is not. For example, for $n = 3$, the subset $\{3\}$ would be represented by the string 001, and the subset $\{2, 3\}$ would be represented by the string 011. Let $f_n : \{0, 1\}^n \to P(S_n)$ be a function that produces the subset represented by a string, e.g. $f_3(001) = \{3\}$ and $f_3(011) = \{2, 3\}$.

(A) Is f_n injective for all n? Surjective? Bijective? Informally explain each of your answers.

(B) In this question, we will use the machinery of DFAs to "compute" a function. We will label accept states with values, and the state that the DFA halts at after processing an input string will correspond to the output of the function it computes. Your task is to draw a DFA that computes the function f_3. *Notes:*

- If the DFA is given a string of length greater than 3, it should enter a labeled "error" non-accept state.
- If the DFA is given a string of length 3, it should halt in an accept state that corresponds to the correct subset of S_3. Thus you should have $2^3 = 8$ accept states.
- No additional explanation or description of the DFA is required beyond the drawing.
- Label the accept states in your drawing. An example of a labelled accept state that should appear in your drawing is below:



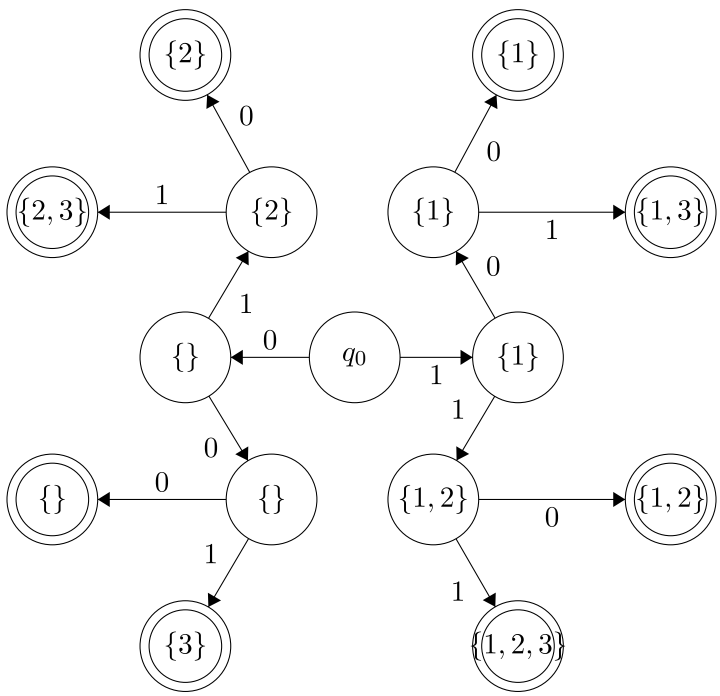
(C) (Challenge!! Not required; worth up to 2 extra credit points.) How many states are required for a DFA that computes f_n? Prove that your answer is a lower bound.

Note: On every problem set we will provide a challenge problem, generally significantly more difficult than the other problems in the set, but worth only a few points. It is recommended that you *attempt these problems, but only after completing the rest of the assignment.*

Solution.

(A). f_n is injective; if two length-n strings result in the same sets, then they must have the exact same letters as well (if they differ at one place, the element which corresponds to that digit is in one set and not in the other). f_n is also surjective; it's easy to see that every subset of S_n is representable by a string of 0's and 1's. Thus, f_n is bijective, since it is both injective and surjective.

(B). Note: all transitions 0's and 1's leaving final states go to the error state (not drawn).



(C). The initial state counts for 1 state; after reading each input, twice as many states are possible (i.e. 2 states are possible after the first input, 4 after the second, and so on). No set can count for more than one state (i.e. the empty set {} cannot count for both intermediate states while reading 001 or 010), because we need to keep track of the inputs leading up to that point to make sure we read no more than n letters. In addition, there is at least one error state. Therefore, we have the sum $\sum_{i=0}^{n} 2^i + 1 = 2^{n+1}$ minimum states.

PART B (Graded by Juan and Varun)

PROBLEM 3 (5+5 points, suggested length of 1/2 page)

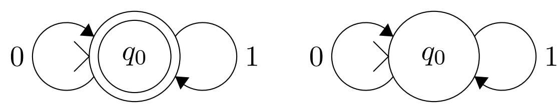
Two FAs are "equivalent" if the languages that they accept are the same. If two FAs are not equivalent, they are "distinct". For this question, you may assume that the alphabet is $\{0, 1\}$.

(A) How many distinct DFAs are there with 1 state? Draw it/them. Describe informally the languages recognized by each.

(B) How many distinct NFAs are there with 1 state? Draw it/them. Describe informally the languages recognized by each.

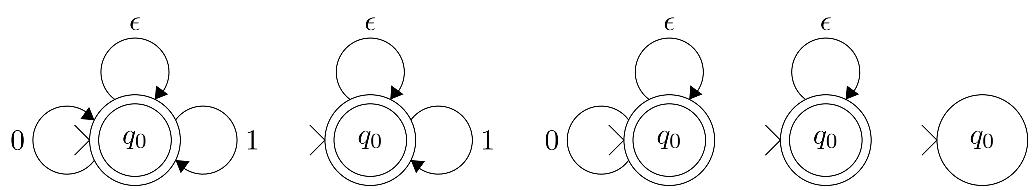
Solution.

(A). There are two DFAs with only one state; one that accepts all strings (including the empty string), and one that rejects all strings (including the empty string).



(B). There are five NFAs with exactly 1 state; they are diagrammed below.

- The first accepts any string of 0's and 1's, including the empty string,
- the second accepts only the empty string and strings of 1's,
- the third accepts only the empty string and strings of 0's,
- the fourth accepts only the empty string, and
- the fifth accepts no strings at all.



PROBLEM 4 (3+3+3 points, suggested length of 1/3 page)

Are the following statements true or false? Justify your answers with a proof or counterexample.

(A) $(L_1 \cap L_2)^* = L_1^* \cap L_2^*$

(B) $(L_1 \cup L_2) \cdot L_3 = (L_1 \cdot L_3) \cup (L_2 \cdot L_3)$, where $\cdot$ is concatenation.

(C) $\{\varepsilon\} \cdot L_1 = \emptyset \cdot L_1$

Solution.

(A). **False**. Let L_1 be $\{1, 0\}$ and L_2 be $\{10\}$. The left side is just $\emptyset^*$, while the right side at least contains 10.

(B). **True**. Let $s \in (L_1 \cup L_2) \cdot L_3$. Let elements of L_1 be denoted as a_i for some i, elements of L_2 be denoted as b_k for some k, and elements of L_3 be denoted as c_j for some j. Then s is either of the format $(a_i)^* \cdot (c_j)^*$ or $(b_i)^* \cdot (c_j)^*$. The former case is an element of $(L_1 \cdot L_3)$, while the latter is an element of $(L_2 \cdot L_3)$. Thus, every element in $(L_1 \cup L_2) \cdot L_3$ is also in $(L_1 \cdot L_3) \cup (L_2 \cdot L_3)$.

We also need to prove that if $s \in (L_1 \cdot L_3) \cup (L_2 \cdot L_3)$, then $s \in (L_1 \cup L_2) \cdot L_3$. This is straightforward to see; in either case (whether $s \in (L_1 \cdot L_3)$ or $s \in (L_2 \cdot L_3)$), s is composed of either a string of a_i or b_j and a string of c_j. This means that s can be divided into a substring that is either in L_1 or L_2, and a substring that is in L_3. This is exactly $(L_1 \cup L_2) \cdot L_3$.

(C). **False**. The left side consists of L_1 itself; prepending any element of L_1 by the only element in $\{\varepsilon\}$, ε does nothing. However, the right side consists of no elements at all, since the empty set contains nothing and therefore there is nothing to concatenate. These two sets are obviously not equal.