Theory of Computation - CSE 105

Regular Languages

Solutions to Homework 1

Problem 1

Design finite state automata and regular expressions for the following languages:

$L_1 = \{w : w \text{ does not contain the substring } 110\}$

$L_2 = \{w : w \text{ contains an even number of 0's or exactly two 1's}\}$

Note: Since we want to design some finite state automata for the above languages, we can choose to use DFAs or NFAs for that purpose. There are naturally several correct constructions possible, but choosing a suitable finite state automaton makes the task easier and the construction more apparent. The "suitable" choice depends on the given language. Generally however you will find it easier to design a DFA for a language that is the complement of some language whose DFA is easy to constuct, while properties like union, closure, concatenation (and infact most operations that involve modification of a DFA or NFA) are easier to capture using NFAs.

It is clear that it is easy to construct a DFA for the complement of L_1. so all that we need to do is to construct a DFA for $\overline{L_1}$ and then switch the role of the accept/reject states. We need just 4 states: q_1 is the start state, q_2 records that the last symbol seen was a 1 (and that either the symbol before that was a 0 or that the last symbol read was the first symbol of the string), q_3 records that (atleast) the last two symbols seen have been 1's and q_4 records that the substring 110 has been seen.

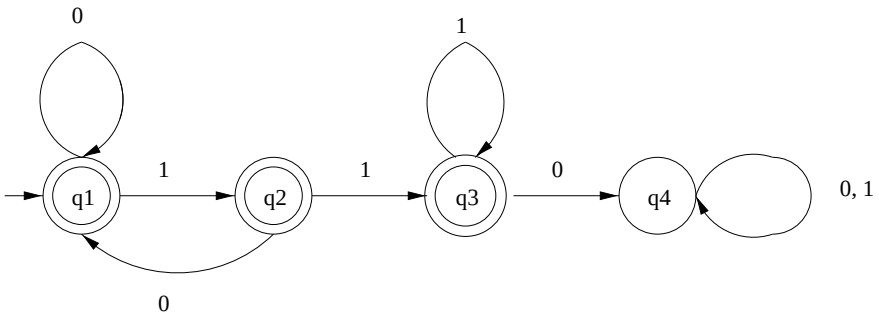

Figure 1: *A DFA recognizing* L_1

The regular expression representing L_1 is given by:

$$(0 \cup 10)^* 1^*$$

This can be derived either directly or by the translation algorithm given in Lemma 1.32, page 69 of text.

In this problem we need to essentially perform the union of 2 languages - one that has an even number of 0's and the other that has exactly two 1's. There may have been some ambiguity here as to whether we meant "contains exactly two 1's in the string" (intended interpretation!) or whether "the language is the string 11". In either case the use of nondeterminism here leads to an easy to verify construction. We need a 2-state NFA (one state records that it has seen an even number of 0's and the other does the same for an odd number of 0's) for the language that contains an even number of 0's. We need a 3-state NFA (each state recording the number of 1's seen in the string) for the language of exactly two 1's. The union of these can be accomplished using ϵ-transitions in the straightforward way.

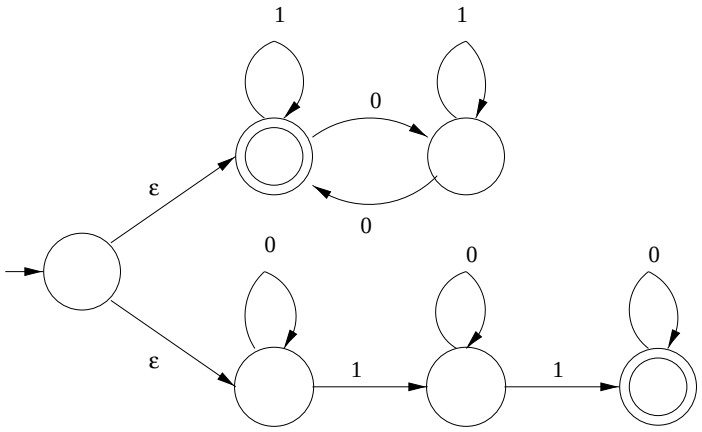

Figure 2: *An NFA recognizing L_2*

The regular expression representing L_2 is given by:

$$(1 \cup 01^*0)^* \cup (0^*10^*10^*)$$

Problem 2

Let

$$\sum_2 = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}.$$

Here, $\sum_2$ contains all columns of 0's and 1's of height two. A string of symbols in $\sum_2$ gives two rows of 0's and 1's. Consider each row to be a binary number and let

$$C = \{w \in \sum_2 | \text{the bottom row of } w \text{ is three times the top row}\}.$$

Show that C is regular.

Proof idea:

In order to to prove that C is regular, we need to construct a DFA or NFA or RE (Regular Expression) that recognizes this language. To do so, we can use the property that regular languages are closed under reversal and read the input backward, i.e. start with the low order bits, and remember that multiply by 3 in binary is equivalent to add the multiplicand with the result of shifting the multiplacand itself to the left by 1 bit. For example, three times 111 (7) is equal to 111 plus 1110 (14) which is 10101 (21). So, if we represent the top and bottom rows of a string w that is in the language by $T_n\ T_{n-1}\ \ldots\ T_1\ T_0$ and $B_n\ B_{n-1}\ \ldots\ B_1\ B_0$, respectively, they must satisfy the following condition:

$$\begin{array}{ccccc} T_n & T_{n-1} & \ldots & T_1 & T_0 \\ T_{n-1} & T_{n-2} & \ldots & T_0 & 0 \\ \hline B_n & B_{n-1} & \ldots & B_1 & B_0 \end{array}$$

As we can see, any valid input string must have $B_0 = T_0$. However, the relations between B_i and T_i for $i \geq 1$ are not as simple as for $i = 0$. Actually, a valid value for B_i will depend on the values of T_i and T_{i-1} as well as whether or not there exists a carry-in c_i^{in} (which is equal to the carry-out c_{i-1}^{out} from the previous sum). The following logical equations describe exactly how they are related (check it out!):

$$\begin{array}{lcl} B_i & = & T_i \oplus T_{i-1} \oplus c_i^{in} \\ c_{i+1}^{in} & = & (T_i \wedge T_{i-1}) \ \vee \ (T_i \vee T_{i-1}) \wedge c_i^{in} \quad = \quad c_i^{out} \end{array}$$

Besides that, it can be easily seen that a valid input also requires that $c_{n+1}^{in} = c_n^{out} = 0$ and $T_n = 0$.

Therefore, in order to prove that C is regular, we need a DFA with a total of 4 states (not taking into account the sink state) to keep track of all possible combinations of c_i^{in} and T_{i-1} and, for each of these states, there will be exactly two possible valid output transitions depending on whether T_i is equal to 0 or 1 (the corresponding value of B_i will be given by the above equation).

Proof:

Let q_{ij} denote the state for which the carry-in is equal to i and the previous symbol seen at the top is equal to j and let q_{sink} denote the sink state. Then, the following DFA recognizes C.

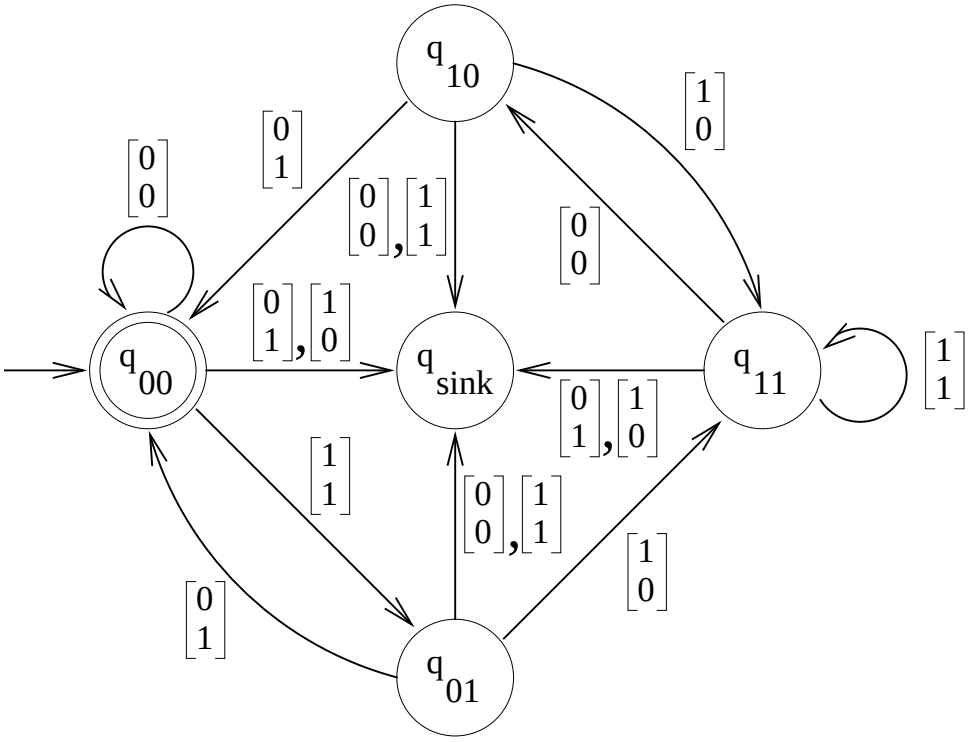

Figure 3: DFA for C (the transition from the sink state to itself was omitted)

Proof of correctness:

In order to prove the correctness of this construction, we need to consider two cases:

- $w \notin C$ and there exists some i such that $B_i \neq T_i \oplus T_{i-1} \oplus c_i^{in}$

  In this case, as soon as this occurs, the DFA will reach the sink state and the input string w will be rejected.

- $w \notin C$ and for all i, $B_i = T_i \oplus T_{i-1} \oplus c_i^{in}$

  In this case, either the carry-out from the last sum $c_n^{out} = c_{n+1}^{in} = 1$ or the last symbol seen at the top $T_n = 1$. In both cases, the DFA will end up in one of the states q_{01}, q_{10}, or q_{11} and the input string w will be rejected.

- $w \in C$ This can only happen if for all i, $B_i = T_i \oplus T_{i-1} \oplus c_i^{in}$ and $c_n^{out} = c_{n+1}^{in} = T_n = 0$. But, whenever this occurs, the DFA will end up in the state q_{00} and the input string w will be accepted.

Because the given DFA will accept all strings in C and reject all strings not in C, we just proved C is regular.

Problem 4.a

Show that $PRIMES = \{a^n \mid n \text{ is a prime}\}$ is non-regular.
Proof: We're going to proceed by contradiction as usual. Assume $PRIMES$ is regular and then pick some string $s \in E$ whose length is at least p, the pumping length. In this case, we don't have much of a choice since alphabet is unary. So, let $s = 0^{p'}$ where p' is some prime number greater than or equal to p. Because $s \in E$ and $|s| = p' \geq p$, s can be broken into three parts, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0, xy^iz \in E$;
2. $|y| > 0$; and
3. $|xy| \leq p$.

Because $|xy| \leq p$ and $|y| > 0$, $y = 0^j$ for some $0 < j \leq p$. The idea here is to choose i, the number of times you're going to pump up, so that $|xy^iz|$ is not a prime. Let $i = p' + 1$ (we're adding y p' times). Then $xy^{p'+1}z = 0^{p'+p'j} = 0^{p'(1+j)} \notin PRIMES$ since its length is a not a prime. As a result, $PRIMES$ cannot be regular.

Problem 4.b

Let

$$\Sigma_2 = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}.$$

Here, Σ_2 contains all columns of 0's and 1's of height two. A string of symbols in Σ_2 gives two rows of 0's and 1's. Consider each row to be a binary number and let

$$E = \{w \in \Sigma_2 \,|\text{the bottom row of } w \text{ is the reverse of the top row}\}.$$

Show that E is not regular.

Proof: Let us assume towards a contradiction that E is regular. We will use the pumping lemma to derive a contradiction.

Refer to Theorem 1.37, page 78 of the text. Let p be the "critical length" given by the theorem. Let us choose the string $s = \begin{bmatrix} 0 \\ 1 \end{bmatrix}^p \begin{bmatrix} 1 \\ 0 \end{bmatrix}^p$ This string is, by definition of the language, contained in the language. Now since $s \in A$ and clearly $|s| > p$, the theorem says that it may be written as $s = xyz$ such that the three conditions of the theorem hold. Condition 3 says that $|xy| \leq p$. Since $s = \begin{bmatrix} 0 \\ 1 \end{bmatrix}^p \begin{bmatrix} 1 \\ 0 \end{bmatrix}^p$, it must be that $xy = \begin{bmatrix} 0 \\ 1 \end{bmatrix}^k$ and $z = \begin{bmatrix} 0 \\ 1 \end{bmatrix}^{p-k} \begin{bmatrix} 1 \\ 0 \end{bmatrix}^p$ for some $k \leq p$. Say $x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}^{k-m}$ and $y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}^m$. Condition 1 of the theorem holds for any $i \geq 0$. We choose $i = 2$. Now consider the string $s' = xy^2z$. This is $s' = \begin{bmatrix} 0 \\ 1 \end{bmatrix}^{k-m} \begin{bmatrix} 0 \\ 1 \end{bmatrix}^{2m} z = \begin{bmatrix} 0 \\ 1 \end{bmatrix}^{k+m} \begin{bmatrix} 0 \\ 1 \end{bmatrix}^{p-k} \begin{bmatrix} 1 \\ 0 \end{bmatrix}^p$, and condition 1 says it is in A. But condition 2 of the theorem tells us $|y| > 0$, meaning $m > 0$. But then it is clear that s' is in fact not in A, because $p + m > p$ and s' will have more leading $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$'s than $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$'s. This means that the bottom row will have more 1's than the top row (and that the top row will have more 0's than the bottom row). This means that the bottom row cannot be the reverse of the top row. This is a contradiction. Hence it must be that E is not regular.