CPS 102: Discrete Mathematics **Instructor:** Bruce Maggs

Assignment 1 **Due:** Monday, September 19th, 2011

1 Designing DFAs (35 points)

Let $\Sigma = \{0, 1\}$. For each of the following languages, give the state diagram for a DFA that recognizes it.

(a) (4 pts) $L_1 = \{w{:}w$ contains the substring 010$\}$

(b) (4 pts) $L_2 = \{w{:}w$ begins and ends with an even number of 1's$\}$

(c) (5 pts) L_3 is the language that consists of all strings such that between every two 1's, there is an even number of 0's.

(d) (5 pts) L_4 is the language that consists of all strings w such that w ends in an odd number of 1's and w contains an even number of 0's.

(e) (5 pts) $\mathcal{L} = \{w \mid$ w contains an even number of 1s or exactly one 0$\}$. Examples in $\mathcal{L}$: 11, 10, 101, 00, ε. Examples not in $\mathcal{L}$: 111, 100, 1.

(f) (5 pts) $\Sigma^*\mathbf{0}\Sigma^*\mathbf{1}\Sigma^*\mathbf{0}\Sigma^*$ Examples in $\mathcal{L}$: 010, 000110, 11011001. Examples not in $\mathcal{L}$: 111, 011, 00011.

(g) (7 pts) $\mathcal{L} = \{w \mid$ w is a binary number divisible by 2, given least significant digit first$\}$. Examples in $\mathcal{L}$: 0, 01, 001, 010, 01011. Examples not in $\mathcal{L}$: 1, 111, 101.

2 DFAs can verify addition (15 points)

DFA's "can't count," but they can verify addition. Consider an alphabet with rather interesting symbols:

$$\Sigma = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\}$$

which represent columns which could appear when writing the addition and result of two binary numbers. An input of length n represents the addition of two n-bit numbers where the input is given from least to most significant bit. The first number's binary representation is in the top row, the second number's representation is in the middle row, and the result is in the bottom row. Let $\mathcal{L}$ be the language of strings over this alphabet which represent correct addition statements as described above. As an example, correctly representing $4 + 2 = 6$ in this scheme would give the following string (first input symbol on the left)

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

The following input would represent $4 + 5 = 8$, and thus this string should not be in $\mathcal{L}$:

$$\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Write a DFA which accepts $\mathcal{L}$. Keep your DFA small; machines with more than 5 states will be penalized.

3 Regular Expressions (20 points)

Regular expressions are a useful tool to computer scientists because they allow us to concisely represent a regular language[1]. In particular, it is often easier to write strings which represent a potentially infinite language than to write down a DFA for that language.

Remember that a language is just a set of strings. Formally, a regular expression over an alphabet Σ can be defined inductively:

- $\emptyset$ is a regular expression and denotes the empty set.
- ϵ is a regular expression and denotes the set containing the empty string, $\{\epsilon\}$.
- For each $a \in \Sigma$, $\mathbf{a}$ is a regular expression and denotes the set $\{a\}$.
- If r, s are two regular expressions that represent the languages R, S respectively, then $r+s$, rs, and r^* are also regular expressions. $r + s$ denotes the set $R \cup S$. rs denotes the set of strings $\{ab \mid a \in R, b \in S\}$. Think of this as the set of concatenations of a string from R and a string from S. Finally, r^* represents the set R^* which is the concatenation of any number (including none) of strings from R. The first few strings in r^* include the empty string, all of the strings of r, all strings which are the concatenation of two strings in r, and so on. Please use the following website to find out more information about the * operator: `http://planetmath.org/encyclopedia/KleeneStar.html`
- We will let Σ be shorthand for any single character in the alphabet.

As examples, $(\mathbf{0} + \mathbf{1})^*$ represents all binary strings (including the empty string), $(\Sigma\Sigma)^*$ represents all strings of even length over an alphabet Σ, and Σ^* represents all strings over the alphabet Σ.

Give a regular expression for each of the following languages over the alphabet $\Sigma = \{0, 1\}$.

(a) (6 pts) The set of strings which contain at least four consecutive zeros.

(b) (6 pts) The set of strings w such that if w has even length, w starts with a 1 and if w has odd length, it starts with a 0.

(c) (8 pts) The set of strings w which, when viewed as binary strings written in the usual way (most significant bit first, least significant bit last), represent numbers that are divisible by 3.

[1] To prove this equivalence in a reasonable manner requires an extension of DFA's called *nondeterministic finite automata* (NFA) and a little more machinery. If we measure the complexity of an NFA by the number of states it has, and the complexity of a regular expression by the number of symbols it takes to write it down, a regular expression for a language could be exponentially larger than an NFA for that language.

4 Regular languages and set operations (15 points)

It's an amazing fact that for the class of regular languages (also called regular sets) a plethora of so-called *closure properties* hold. Namely, if R and S are regular languages, then the following are also regular languages:

- $\overline{R}$, the *complement* of R, consisting of those strings not in R (sometimes denoted R^c).
- R^*, the *Kleene closure* of R as described in question 3.
- $R \cdot S$, the concatenation language of R and S.
- $R \cup S$, the union language consisting of strings s that are in R or in S
- $R \cap S$, the intersection language consisting of strings s that are in both R and S
- $R^{\mathcal{R}}$, the reversal language consisting of the reverse of all strings in R.
- $R \setminus S$, the difference language consisting of strings s that are in R but not in S.
- $R \Delta S$, the symmetric difference (xor) of two languages is the language $(R \setminus S) \cup (S \setminus R)$.

Without any additional tools beyond DFAs we can prove several of these facts, as seen in lecture. The goal here is to give a *rigorous* argument for why these constructions work- we expect you to be more detailed than we were in class!

Let me define a few additional concepts before you get going. In class, we loosely described the notion of acceptance as the machine ends in a final state when it has no more input. We can formally define this concept as follows. Suppose we are given a DFA $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$. Define the function $\delta^* : Q \times \Sigma^* \to Q$ inductively as

$$\delta^*(q, \epsilon) = q$$

$$\delta^*(q, \sigma w) = \delta^*(\delta(q, \sigma), w), \qquad \sigma \in \Sigma$$

The δ^* function takes in a state and a word and tells us where the machine ends a computation if we were to start in state q and read in the word w. Now we can formally describe the acceptance as $w \in \Sigma^*$ is accepted if and only if $\delta^*(q_0, w) \in F$.

(a) (3 pts) Prove that the complement operation is a regular operation.

(b) (7 pts) Prove that the union operation is a regular operation.

(c) (5 pts) Consider the following construction for the reversal operation: Given a regular language $\mathcal{L}$ there is some DFA $\mathcal{M}$ whose acceptance language is $\mathcal{L}$. We can reverse all of the arrows in $\mathcal{M}$ and make the start state a final state. Furthermore, collect all final states of the original machine into a single state, and make it the start state in the augmented machine. This DFA accepts the reversal language.

Does the above construction work? Either prove that it is correct or explain why it is incorrect.

5 Blue Eyes[2] (15 points)

A group of people with assorted eye colors live on an island. They are all perfect logicians – if a conclusion can be logically deduced, they will do it instantly. No one knows the color of their eyes. Every night at midnight, a ferry stops at the island. Any islanders who have figured out the color of their own eyes then leave the island, and the rest stay. Everyone can see everyone else at all times and keeps a count of the number of people they see with each eye color (excluding themselves), but they cannot otherwise communicate. Everyone on the island knows all the rules in this paragraph.

On this island there are 100 blue-eyed people, 100 brown-eyed people, and the Guru (she happens to have green eyes). So any given blue-eyed person can see 100 people with brown eyes and 99 people with blue eyes (and one with green), but that does not tell him his own eye color; as far as he knows the totals could be 101 brown and 99 blue. Or 100 brown, 99 blue, and he could have red eyes.

The Guru is allowed to speak once (let's say at noon), on one day in all their endless years on the island. Standing before the islanders, she says the following:

"I can see someone who has blue eyes."

Who leaves the island, and on what night?

There are no mirrors or reflecting surfaces, nothing dumb. It is not a trick question, and the answer is logical. It doesn't depend on tricky wording or anyone lying or guessing, and it doesn't involve people doing something silly like creating a sign language or doing genetics. The Guru is not making eye contact with anyone in particular; she's simply saying "I count at least one blue-eyed person on this island who isn't me."

And lastly, the answer is not "no one leaves."

6 Optional (25 points)

Design a DFA that accepts those binary strings w such that when you reverse w you get a binary integer that is divisible by 5. For example, your DFA should accept 10011, because when you reverse it, you get 11001, which as a binary integer is 25 in decimal, and 25 is evenly divisible by 5.

[2]The following problem is copied verbatim from `http://xkcd.com/blue_eyes.html`.