CS331: Homework #2

Due on February 7, 2013 at 11:59pm

Professor Zhang 9:00am

Josh Davis

Problem 1

Let $\Sigma = \{0, 1\}$. Construct a DFA A that recognizes the language that consists of all binary numbers that can be divided by 5.

Let the state q_k indicate the remainder of k divided by 5. For example, the remainder of 2 would correlate to state q_2 because $7 \mod 5 = 2$.

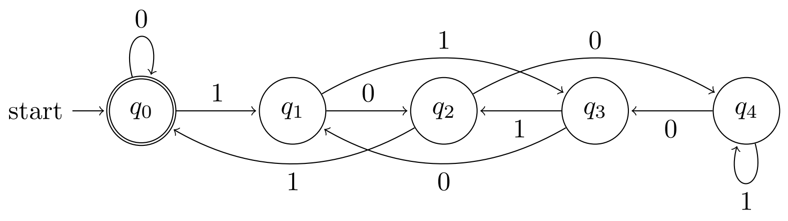

Figure 1: DFA, A, this is really beautiful, ya know?

Justification

Take a given binary number, x. Since there are only two inputs to our state machine, x can either become $x0$ or $x1$. When a 0 comes into the state machine, it is the same as taking the binary number and multiplying it by two. When a 1 comes into the machine, it is the same as multipying by two and adding one.

Using this knowledge, we can construct a transition table that tell us where to go:

	$x \mod 5 = 0$	$x \mod 5 = 1$	$x \mod 5 = 2$	$x \mod 5 = 3$	$x \mod 5 = 4$
$x0$	0	2	4	1	3
$x1$	1	3	0	2	4

Therefore on state q_0 or ($x \mod 5 = 0$), a transition line should go to state q_0 for the input 0 and a line should go to state q_1 for input 1. Continuing this gives us the Figure 1.

Problem 2

Let $w = \sigma_1 \cdots \sigma_n$ be a word on an alphabet Σ. By w^R we mean the word is the reverse of w.
Define L^R such that

$$L^R = \{w \in \Sigma^* : w^R \in L\}.$$

Proof. We will show that if L is regular, then so is L^R.

Suppose L is regular. Since L is regular, that means we can create a NFA for it. Let this NFA, A, be like so:

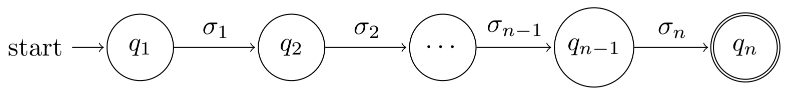

Figure 2: NFA, A

Now taking this NFA, we can keep the same alphabet Σ, states Q, but just reverse it to arrive at this NFA, B:

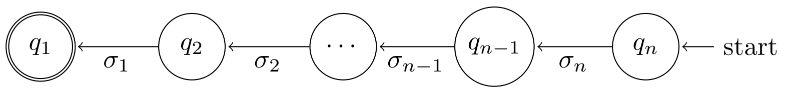

Figure 3: NFA, B

Since this is a valid NFA, it is exactly the same as Automata A except it has its edges reversed. It only accepts words such that $w \in L$ and w is reversed.

Therefore if L is regular (it can be represented by a NFA) then L^R is also regular. □

Problem 3

Let $\Sigma = \{0, 1\}$. Construct a DFA that recognizes the following:

$$\{w \in \Sigma^* : |w|_{01} = |w|_{10}\}.$$

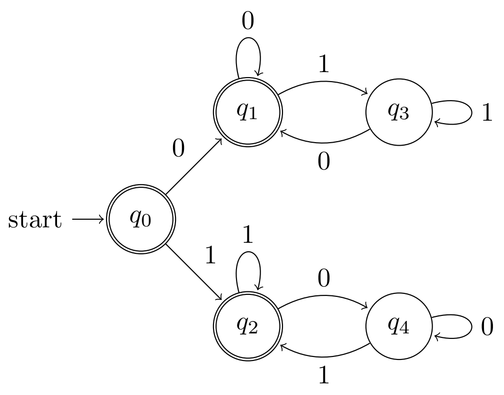

Figure 4: NFA, B

This DFA can be shown to be true by using the pumping lemma.

Problem 4

Prove that the class of regular languages is closed under imperfect shuffle.

Proof. Consider two regular languages, L_A and L_B. To show that these two languages are closed under the imperfect shuffle, we will construct a NFA that can handle these two languages because doing so with a DFA is too complicated (similar to how the book uses an NFA to show closure under union and concatenation).

Note: One thing to note is that this can be shown using a proof by induction but NFAs are more fun ;)

Consider the two NFAs, N_A and N_B such that:

$$N_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$$
$$N_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$$

N_A and N_B describe two regular languages, L_A and L_B respectively.

The new automata that will show that the two languages are closed under the imperfect shuffle can be defined as follows:

$$N = (Q, \Sigma, \delta, q, F)$$

where

1. Two extra states to differentiate between which input to expect next $Q = Q_1 \times Q_2 \times \{A, B\}$
2. The alphabet is the same, $\Sigma = \Sigma$
3. The start is in the start state for N_A and N_B and we expect an input from A, $q = (q_A, q_B, A)$.
4. The final states include both sets and ends with input last from B, $F = F_A \times F_B \times \{A\}$
5. Define δ so that
$$\delta((q_1, q_2, S), a) = \begin{cases} Q_1 \times Q_2 \times \{B\} \to \mathcal{P}(Q) & S = A \\ Q_1 \times Q_2 \times \{A\} \to \mathcal{P}(Q) & S = B \end{cases}$$

□

Problem 5

Co-determinism.

Proof. **Show that every CDFA is a DFA.**

To do this, we will look at what the differences are between a DFA and NFA. A NFA has the following differences:

1. Alphabet = Σ_ϵ, allows epsilon transitions
2. Can have 0 or more transitions for any given input and state

A CDFA has already been defined for us. Therefore it has these differences from a NFA:

1. Allows only input from Σ, no epsilon
2. Cannot have multiple transitions with the same input going into the same state, or $\delta(q, a) \cap \delta(q', a) = \emptyset$

It can be seen that a CDFA and NFA do not share the same properties. Therefore a CDFA is just a more specific DFA. □

Show that every DFA is a CDFA.

Counterexample To show that this is not true, we can take the given DFA:

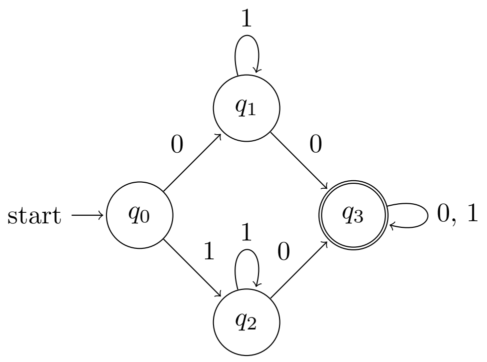

Figure 5: NFA, B

The Figure 5 is a counterexample because it has a state, q_3, that has two incoming arrows for state 0. Thus it violates the definition of co-determinism.

Proof. **Show that every NFA can be converted into an equivalent CDFA.**

Using the theorem 1.39 out of Sipser's book, it is proven that every NFA can be converted into an equivalent DFA.

Taking that same DFA, one could add more states to (maximizing) it and it would be possible to reduce all same input transisitions to a single state down to none such that $\delta(q, a) \cap \delta(q', a) = \emptyset$.

Once this has been achieved, the state machine would then be co-deterministic. □