CSCI 510 **Models of Computation**

Homework 1

Solution Key

Problem 1.1

Draw a circuit and a straight-line program for the Boolean function whose value is **True** exactly when x or y or both is **True** and z is **False**.

The circuit should simply use 2 gates to compute $(x \vee y) \wedge \bar{z}$.

Straight line program:

```
(1 READ x)
(2 READ y)
(3 READ z)
(4 1 OR 2)
(5 NOT 3)
(6 4 AND 5)
```

The following questions are lab problems. Please remember to prepare a solution for your assigned problem before going to your lab section.

Lab Problem 1

When it was first discovered that there are problems that cannot be solved by computers, it was a very surprising result. An equally unsettling discovery by Gödel was that some mathematical statements have no proofs. In this problem, we will do a high-level proof of this fact. Instead of using mathematical statements, we will simply use English sentences.

This is the sentence we are trying to prove:

There exists a true sentence for which there is no proof.

We will argue this by contradiction.[1] That is, we will assume the following statement, then obtain a contradiction:

[1] We are being informal here, because we have not given a mathematical formalization of the statement we are trying to prove.

Assumption: All true sentences have proofs.

You may assume that any statement for which there exists a proof is true.

Let S be the sentence: "This sentence has no proof."

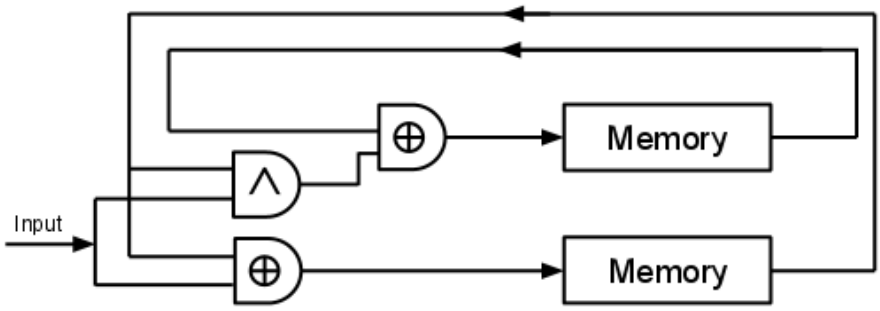
There are two cases to examine:

- Suppose S is true. Using our assumption, there exists a proof for S. But S states that it has no proof, so S must be false.
- Suppose that S is false. Then there exists a proof for S, so S must be true.

Whether S is true or false, we obtain a contradiction. Thus, our assumption is false.

Lab Problem 2

Convert the following circuit diagram for an FSM into a state diagram (where each state is represented by a vertex and each transition is a directed edge).



In the above diagram, $\bigoplus$ is exclusive OR and $\bigwedge$ is AND.

Hint: Think of the memory bits as a single 2-bit binary integer.

This is an FSM with 4 states. Let the values of the bottom and top cells be m_0 and m_1, respectively. Let the new values be m_0' and m_1' and the input be a. Then, $m_0' = m_0 \oplus a$ and $m_1' = m_1 \oplus m_0 \wedge a$. This machine doesn't change the value of the state if $a = 0$ but adds 1 to the result modulo 4.

Lab Problem 3

Vending Machines and FSMs: Finite state machines can be used for many purposes. Here you'll design a finite state machine to model your own vending machine. Your vending machine will take quarters and distribute two flavors of juice, Apple Juice and Orange Juice, each for $.50.

In order to model the vending machine as an FSM, the input and action alphabets must be defined. Your vending machine has four possible inputs at each step, **Q, O, A**, and **C** which have the following meanings:

Q: A quarter ($.25) has been inserted into the machine.

O: The *Orange Juice* button has been pressed.

A: The *Apple Juice* button has been pressed.

C: The *Return Change* button has been pressed.

Note: We assume that multiple buttons are not pushed at the same time and that a quarter cannot be inserted at the same time as a button pressed.

Your FSM has four possible actions, **DO, DA, N**, and **R**. These have the following meanings:

DO: Distribute an Orange Juice.

DA: Distribute an Apple Juice.

N: Do nothing.

R: Return one quarter.

You should think about your design carefully. Make sure that your machine behaves reasonably. An example of an unreasonable behavior would be to distribute an Orange Juice when someone presses the *Apple Juice* button. Also note that the *Return one quarter* action will have the effect of returning at most one quarter. That is, *Return one quarter* after $.50 has been inserted will result in returing a single quarter, not two quarters.

The FSM you submit should be given as a graph (circles and lines). Briefly discuss any significant design decisions that you make.

The solution depends on the machine. Be reasonable.

Because it is impossible to have two actions on a single transition, and there is no "nothing is happening" input, the "Return change" button must be such that you only get back one quarter at a time. (Unless you use non-determinism, which has not yet been covered.) So essentially, it's a "Return one quarter" button to correspond to the "Return one quarter" action.

It is also impossible for this machine to hold an infinite amount of change, because there are only a finite amount of states, and the user must be able to get back all of the money that they put in. To that end, we limit the amount of money to 50 cents, and spit out a quarter every time the user tries to put more in. This means that we need three states, each representing a different amount of change in the vending machine: a 0 cent state, which we'll call q_0; a 25 cent state, which we'll call q_1; and a 50 cent state, which we'll call q_2. We connect these states as follows: