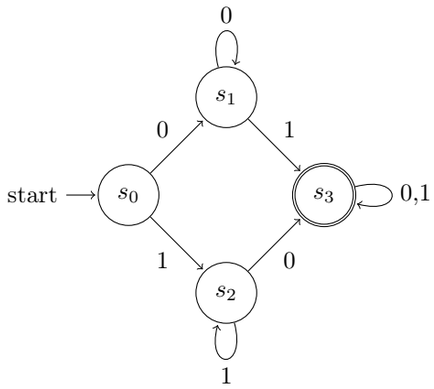
CPSC 121 - DFAS SOLUTIONS

Problem 1. Convert the following DFA to a sequential circuit.



Make sure that your sequential circuit has an output and that the output is true when the DFA is in the accepting state given the input.

Solution: We represent each state by its number in binary; the transitions (arrows) of the DFA are given by the following table:

Current State		Input	Next State	
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	0
1	1	0	1	1
1	1	1	1	1

The following is a derivation of the circuitry for the inputs into the multiplexer, which determine which state the sequential circuit should transition to given the current state and a particular input value. Note, we label the individual wires for particular bits of the state as b_0 and b_1, where b_0 corresponds to the rightmost bit, or in general the 0^{th} bit of a binary number, and b_1 corresponds to the leftmost bit, or more generally the 1^{st} bit of a binary number. Note that in the drawing of the sequential circuit these wires are labelled as 0 and 1 rather than b_0 and b_1.

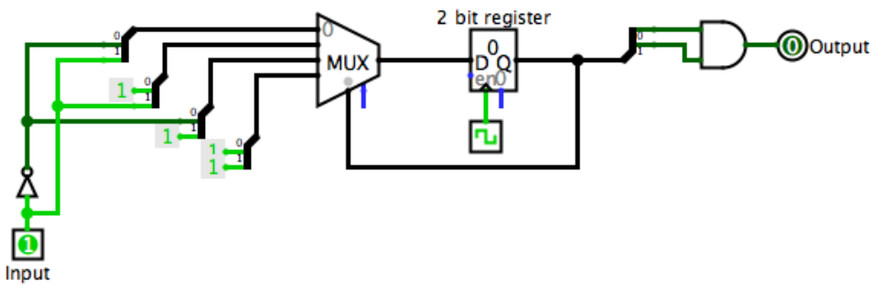
Consider the 0^{th} state, the circuitry for the sequential circuit's behaviour when in the 0^{th} state is fed into the 0^{th} entry of the multiplexer, which corresponds to the sequential circuit being in state s_0. Likewise, we feed the circuitry representing the s_1, s_2, and s_3 states into the 1^{st}, 2^{nd}, and 3^{rd} entries of the multiplexer, respectively.

Analysing the table we constructed above, in particular the first two rows which represent the sequential circuit being in the s_0. We determine the logic for each wire (or bit) separately. For the b_1 wire (bit), we notice from the table that whenever the input is 0 the next state's b_1 bit is 0 and whenever the input is 1 the next state's b_1 bit is 1. Hence, the value of the next state's b_1 bit is always the value of the input. Therefore, we feed the input into the b_1 wire of the 0^{th} input of the multiplexer. For the b_0 wire (bit), we notice from the table that whenever the input is 0 the next state's b_0 bit is 1 and whenever the input is 1 the next state's b_0 bit is 0. Hence, the value of the next state's b_0 bit is always the negation of the input. Therefore, we feed the negation of the input into the b_0 wire of the 0^{th} input of the multiplexer. We repeat an identical procedure for the remaining states.

As for the output to the circuit, the circuit should return true if and only if the sequential circuit is in the accepting state. We note from the above DFA that there is only a single accepting state s_3 which can be represented by $s_0 = 1$ (or true) and by $s_1 = 1$ (or true). Therefore, we determine whether the sequential circuit is in the accepting state by the implementing the circuitry $s_1 \wedge s_0$

Note: This implementation of a sequential circuit uses a single register instead of two D flip-flops next to one another (which was the implementation used in lab 8). Both implementations are equivalent and equally valid. Note that the two inputs to each wire-merge would go to two distinct multiplexers in the implementation that uses two D flip flops instead of a single 2 bit register.

We obtain the following circuit:



Problem 2. Describe in English which inputs will lead the DFA to an accepting state.

Solution: The DFA accepts any string that contains at least one "0" and at least one "1".

Problem 3. Design a deterministic finite-state automaton (DFA) that accepts exactly the strings over the alphabet $\{A, B, \ldots, Z\}$ that

- contain at most two E's
- contain fewer T's than E's
- and in which no L that comes after two E's is immediately followed by an Y

For instance, your DFA should accept the strings

- VADER
- MEETING
- PINKELMERFLOYD

but not the strings

- EXTREME (there are three E's)
- MERELY (two E's followed by LY)
- DINOSAUR (there are as many T's as E's (zero))
- TETRAODON (there are more T's than E's)
- VOLDEMORT (there are as many T's as E's (one of each))

Clearly indicate the meaning of each state. Hint: Our solution uses 9 states.

Solution: Here is a DFA that works. It keeps track of the number of E's that have been seen (S0/S4 mean no E's, S1/S5 mean one E, S2/S6 mean two E's) as well as the number of T's that have been seen (S0/S1/S2 mean no T's, S4/S5/S6 means one T). Once we have seen two E's, an L sends the DFA to a "watch for a Y" state (S3 if we were in S2, S7 if we were in S6). A third E, a second T or a Y after an L sends the DFA to the "garbage" state S8.