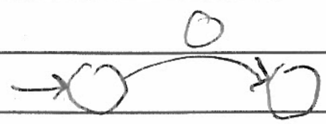
$L := \{ s \in \{0,1\}^* \mid s \text{ contains at least one } 1 \text{ and at least one } 0 \}$

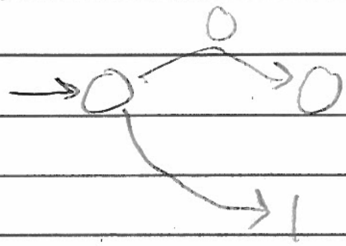
Q 1. 

From that init state, if we see a 0 then we've made progress toward verifying that the string is in L, so must introduce new state



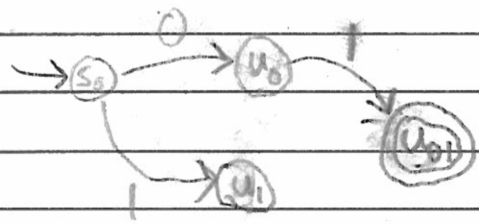
Why? Because $\varepsilon | 1 \notin L$ but $0|1 \in L$

Likewise for seeing a 1. That must be handled differently from seeing a 0.



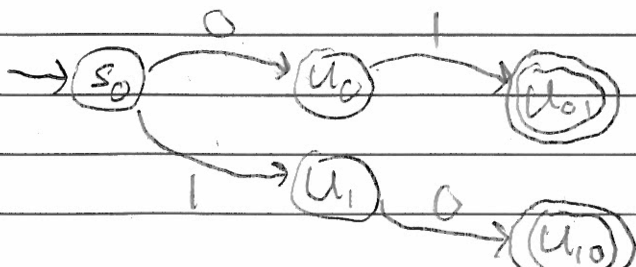
Why? Because $0|1 \in L$ but $1|1 \notin L$

Similarly, can't loop back to empty state since
$\varepsilon | 0 \notin L$
but $1|0 \in L$

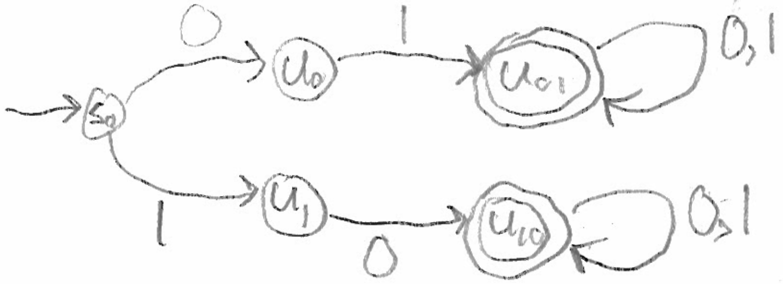


Must create a new state because the existing 3 cannot be accept states.

Clearly we can handle a 0 from u_1 in a similar way, but note I'm not arguing that we must introduce a fifth state.



If a prefix of a string is in L then the whole string is in L, so self-loop on either bit in the accept states!



At this point, we either need to add two transitions (one from u_0, the other from u_1), or else add a new state.

I see no reason for why another state would be necessary or even useful.

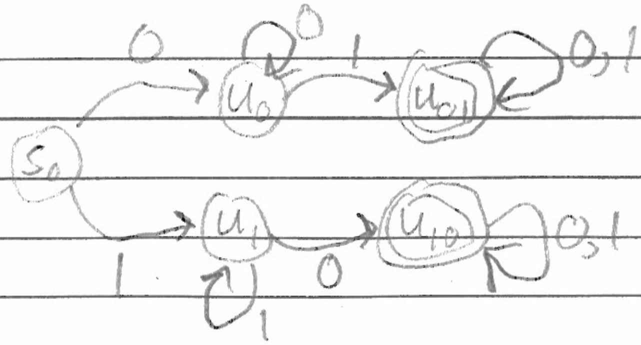
Notice: If we see 0, go to u_0, and then see another 0, we've neither made nor lost progress toward verifying that the string is in L; we're still in the position of:

(*) Having seen at least one 0 and no 1s.

Tells us to draw a self loop. 

,
After that, we can see (*) as the meaning of being in state u_0.

Clearly same reasoning applies to u_1, so we get a complete FSM that's correct by construction:

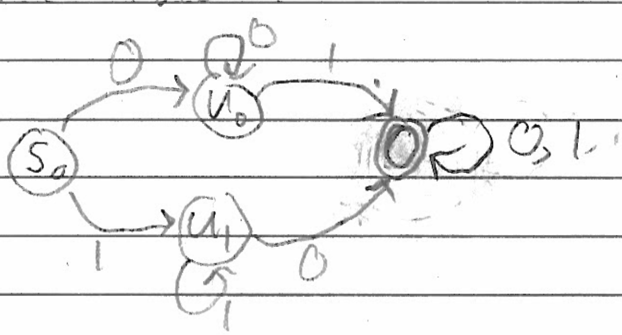


Then we're done. But note that since

u_{01} – meaning: "seen at least one 0, and at least one 1, starting with a 0"

u_{10} – meaning: " " " " " " "
" " " " " "
" 1 "

are both <u>sink states</u> – meaning they have no edges to other states. Since they're also both accept states, we can join them:

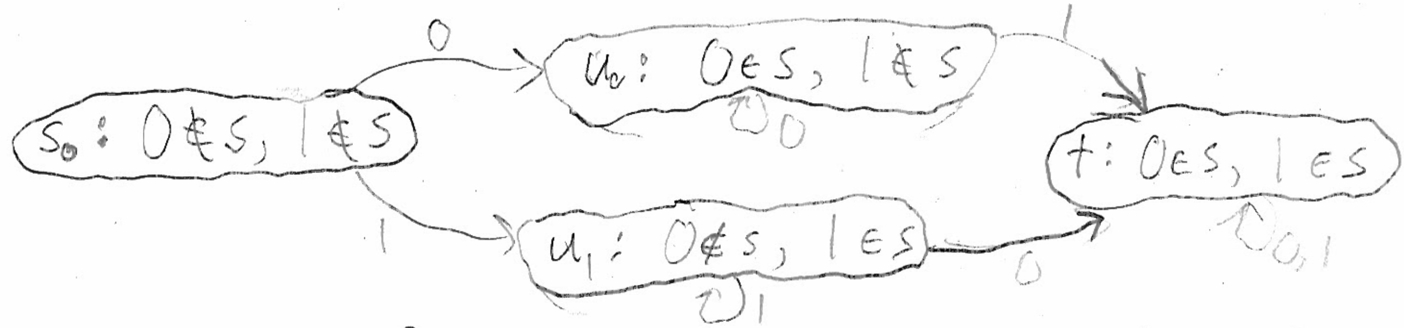


If we hadn't introduced u_{10} at all, we would be close to an argument for why there is no smaller DFA that accepts L.

Q2. For $a \in \{0,1,2\}$, and $s \in \{0,1,2\}^*$ let $a \in s$ mean "s contains at least one occurrence of a."

So the language is $L := \{ s \in \{0,1,2\}^* \mid 0 \in s, 1 \in s, 2 \in s \}$

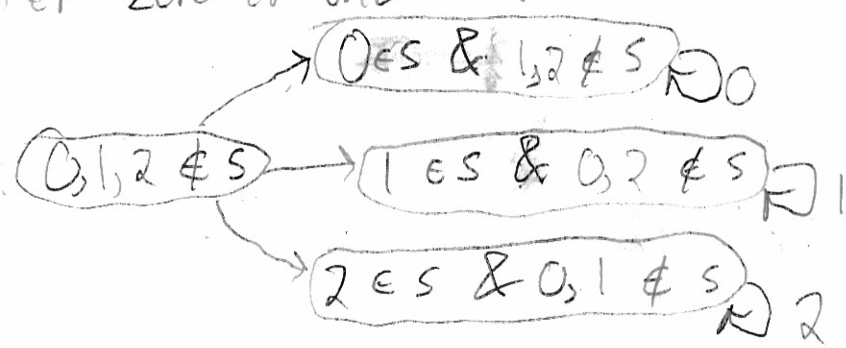
In Q1 we had states for



That is 2^2 states. Analogous idea for this L would have 2^3 states, for the $2 \times 2 \times 2$ properties:

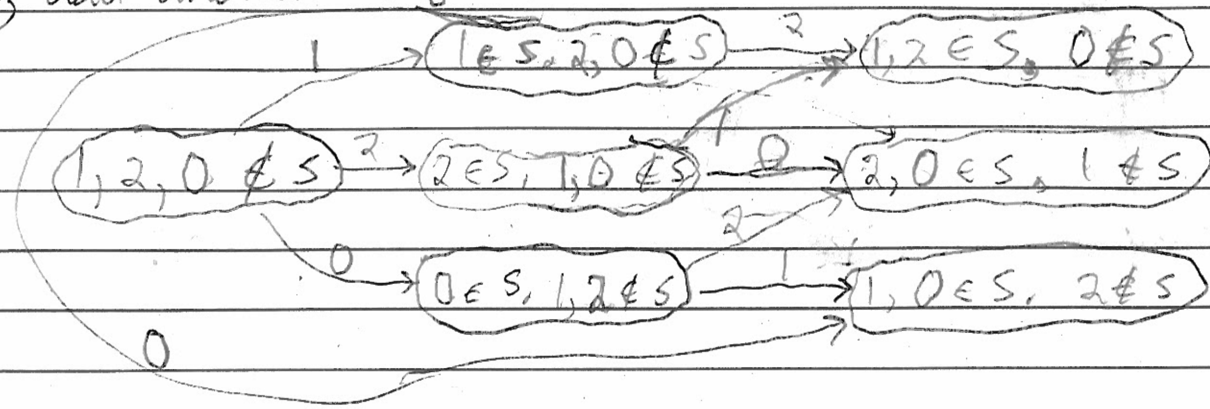
$$(0 \in s?) \times (1 \in s?) \times (2 \in s?)$$

Start with the states reachable after zero or one characters:



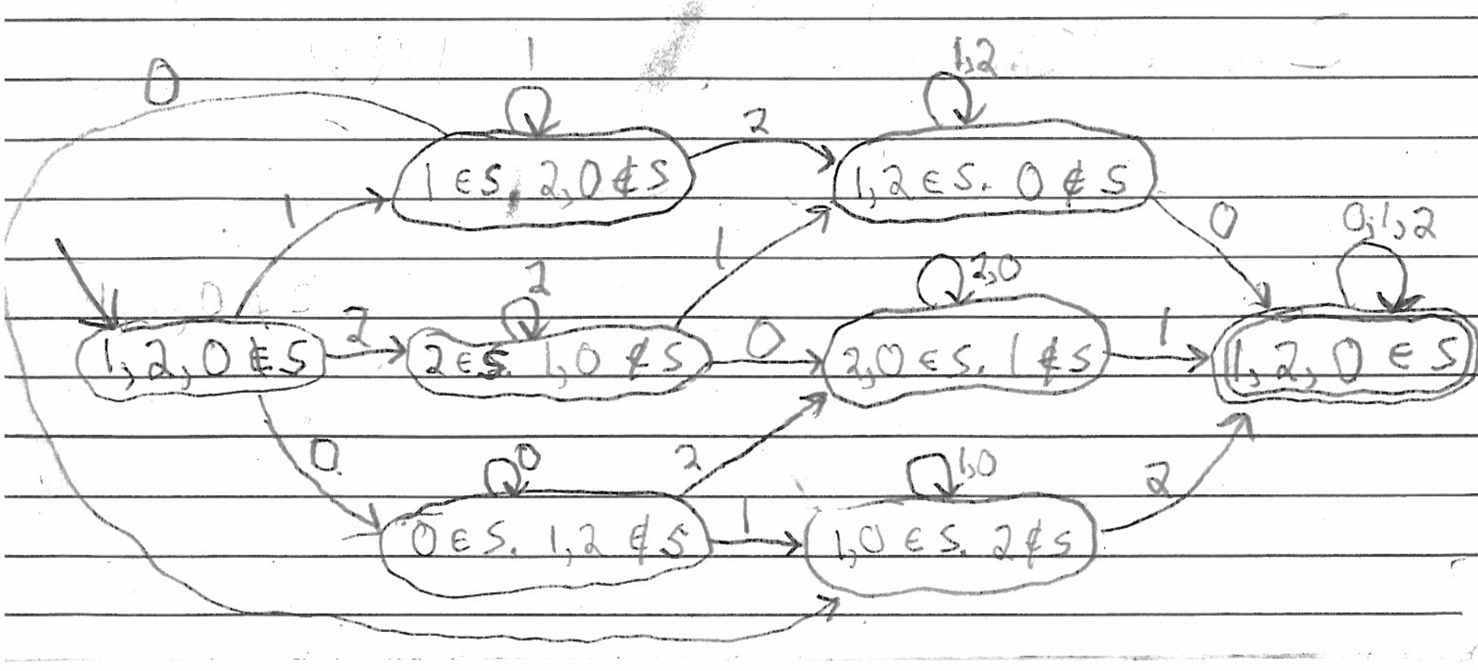
When we see the second character either it's the same as the first and we take the self-loop, or else we're in a new situation: having seen exactly 2 distinct characters.

If we see a new character next, must go to a new state. Why: after that it'd be possible with one more transition to reach an accept state, whereas all current states require at least 2 more transition to reach any accept state.
So, add another layer:



We haven't argued that the three new states must be distinct, but we also don't need to in order to answer the handout question.

Finally add the accept state, and all the self loops:



Use for A3 Q1.(b)

Myhill-Nerode Theorem:

Let Σ be an alphabet and $L \subseteq \Sigma^*$, i.e. L is a language of Σ-strings.
Suppose there are n strings $t_1, \ldots, t_n$ s.t. for each pair $t_i \neq t_j$ $\exists s \in \{0,1\}^*$ s.t. $t_i s \in L \Leftrightarrow t_j s \notin L$

Then any DFA that computes L must have at least n states.