COMP 330 - Assignment 1

Denis R[redacted]

20/01/21

Question 1

We need to prove: Reflexivity, Symmetry and Transitivity

A) Reflexivity

xRx if $\forall z \in \Sigma^* \quad xz \in L \iff xz \in L$

Obviously true $\square$

B) Symmetry

$xRy \Rightarrow yRx$

$xRy = (xz \in L \iff yz \in L) = (yz \in L \iff xz \in L) = yRx \quad \square$

C) Transitivity

let $\forall\, w, y, z, x \in L,\ xRw \wedge wRy \Rightarrow xRy$

$xRw \Rightarrow xz \in L \Leftrightarrow wz \in L$ (I)

$wRy \Rightarrow wz \in L \Leftrightarrow yz \in L$ (II)

$(\Rightarrow)$ Assume $xz \in L$ then by (I) $wz \in L$. If $wz \in L$ then by (II) $yz \in L$. $\therefore\ xz \in L \Rightarrow yz \in L$.

$(\Leftarrow)$ Assume $yz \in L$ then by (II) $wz \in L$. If $wz \in L$ then by (I) $xz \in L$. $\therefore\ yz \in L \Rightarrow xz \in L$.

$\therefore\ \forall\, w, x, y, z \in \Sigma^*\ xRw, wRy \Rightarrow xRy$ $\square$

Question 2

We need to prove Reflexivity, Anti-symmetry, and transitivity.

A) Reflexivity

$$\langle m, n \rangle \sqsubseteq \langle m, n \rangle \Rightarrow (m < m) \vee ((m = m) \wedge (n \leq n))$$

Clearly true $\square$

B) Anti-Symmetry

$$(x \leq y) \wedge (y \leq x) \Rightarrow (x = y)$$

$\langle m, n \rangle \sqsubseteq \langle m', n' \rangle$. Then $((m = m' \wedge n \leq n') \vee (m < m'))$

$\langle m', n' \rangle \sqsubseteq \langle m, n \rangle$ then $((m' = m \wedge n' \leq n) \vee (m' < m))$

$\hookrightarrow (m = m') \wedge (n \leq n') \wedge (n' \leq n)$

Clearly $m = m' \wedge n = n'$ $\square$

C) Transitivity

$$\underbrace{(\langle m, n\rangle \sqsubseteq \langle m', n'\rangle)}_{①} \wedge \underbrace{(\langle m', n'\rangle \sqsubseteq \langle m'', n''\rangle)}_{②}$$

$$\Rightarrow \langle m, n\rangle \sqsubseteq \langle m'', n''\rangle$$

① $\Rightarrow (m < m') \vee ((m = m') \wedge (n \leq n'))$

② $\Rightarrow (m' < m'') \vee ((m' = m'') \wedge (n' \leq n''))$

Combining ① and ②

Combining ① and ②

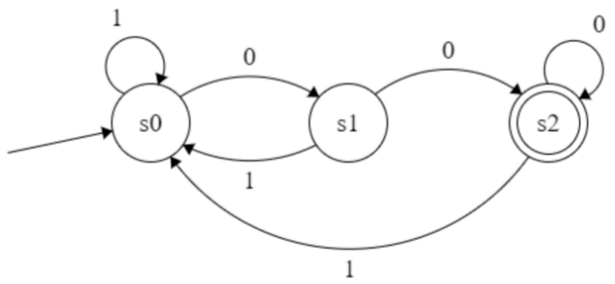
$$(m < m' < m'') \vee ((m = m' = m'') \wedge (n \leq n' \leq n''))$$

Clearly it follows: $(m < m'') \vee ((m = m'') \wedge (n \leq n''))$

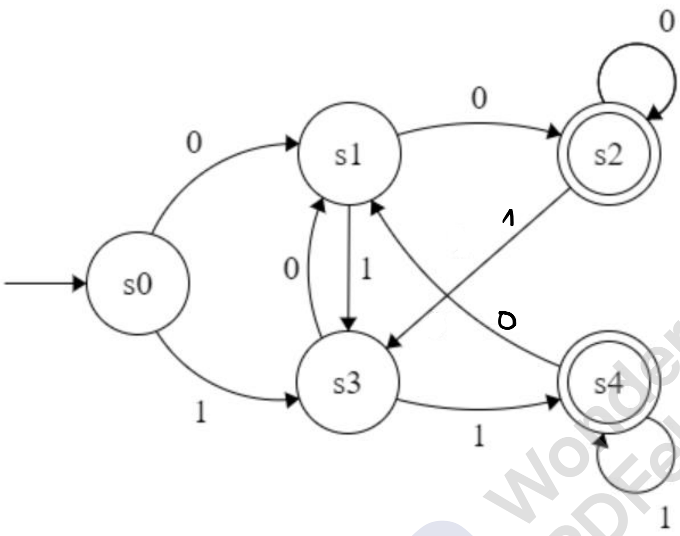
$\therefore \langle m, n\rangle \sqsubseteq \langle m'', n''\rangle$

Question 3

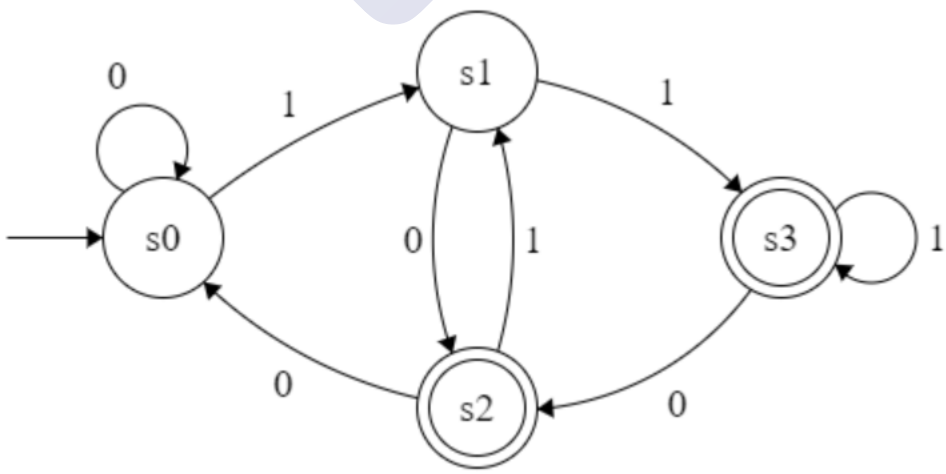
1)



2)



3)



Question 4

Let the DFA N that recognizes L be defined:

$$N = (Q, q_0, \Delta, F)$$

We can define a NFA M to recognize Left-half(L):

$$M = (Q', q_0', \Delta', F')$$

- $Q' = Q \times Q$
- $q_0' = \{(q_0, f) \mid f \in F\}$
- $\Delta'((s,t),a) = \{(\Delta(s,a), t') \mid t = \Delta(t', b) \text{ for some } b \in \Sigma\}$
- $F' = \{(s,s) \mid s \in Q\}$

The states in M are pairs of states from N. For an input string $w=w_1w_2$, the first coordinate is the state N would be after reading the first character from *w*. The second coordinate corresponds to the state a machine reading *w* in reverse (i.e., read backwards) would end up being. One can say M reads a string *w* from both ends in parallel.

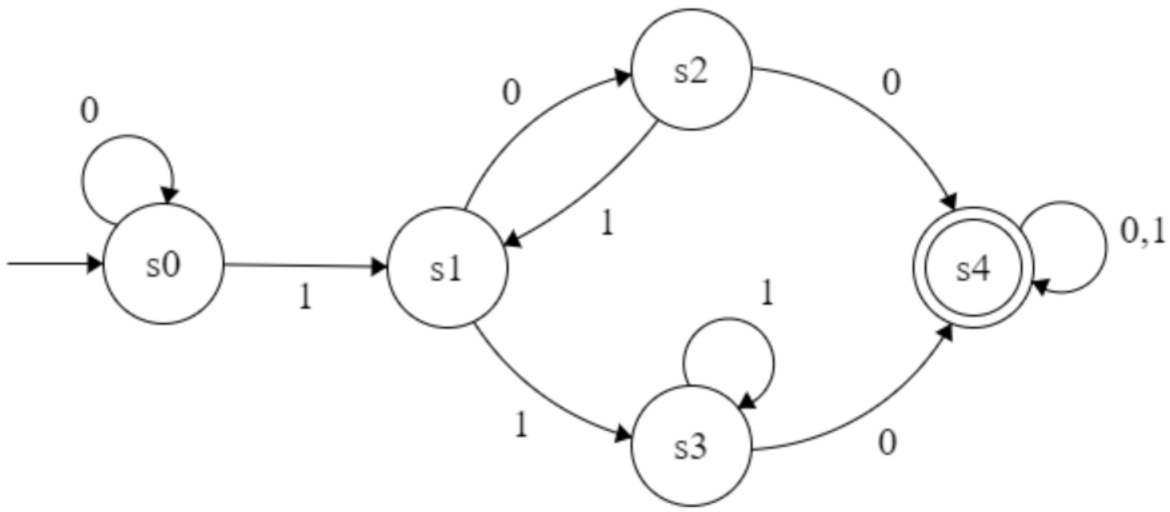
If there exists a run in which the machine ends up in a state pair (s, s) it implies that in N there is a path of length l_w that goes from a starting state to an accept state. Let this path be *w'*. Thus, $w'=w_1w_2$ is in L.

Furthermore, because a run reading *w* from both ends has ended in a state (s, s) it clearly has wielded a path of same length for both parts. Therefore, $|w_1|=|w_2|$.

Left half is regular. ■

Question 5

1)



2) Let S be the set of minimal length accepted strings are 3 bits long.

S = {000, 001, 010, 011, 100, 101, 110, 111}

Proof by contradiction:

Assume a machine with 4 states that accepts L exists, then all these cases must fit in those 4 states.

- 110, 100 are in an accept state. Next input being 0 or 1 doesn't change their state. Name this state *s4*. **We have 1 state so far.**
- 010, 011, 111 go to *s4* if the next bit is 0. However, if the next bit is 1:
  - 011, 111 stay in the same state. Name this state *s3*. **We have 2 states so far.**
  - 010 goes to a new reject state. 2 states are thus required: let the one holding 010 be *s2* and the one holding 0101 be *s1*. **We have 4 states so far.**
- 001, 101 are in *s1* (i.e., the same state as 0101). If the next bit is 0, they both go to *s4*; whereas if the next bit is 1, they go to *s3*. **We still have 4 states.**
- 000 loops back to itself if the next bit is 0. If the next bit is 1 it goes to *s1*. Clearly a new state is required to hold 000. **We have 5 states so far. CONTRADICTION.**

Therefore, at least 5 states are required for any DFA recognizing this language.