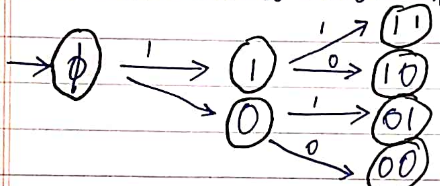
Automata Theory
Assignment-1

George Paul
2021121006

① To use an FSM as memory, each possible value ~~for the FSM~~ that can be stored in memory will have to be treated as a state. Each transition will just move to a state that has the bit to be stored, appended to it.

For example, the following is an FSM that can store two bits.



in terms of the number of states, ~~we~~ an FSM needs at least $(2^{k+1})-1$ to fully, functionally store k bits of memory.

The transition function will be:

$\delta(x, l) = x+l$, where + is concatenation
eg. $\delta(10, 1) = 101$

② Consider a DFA, A

Suppose A has read some bits already whose decimal value is x.

On reading another bit $y \in \{0,1\}$, the new equivalent decimal value is $(2x+y)$.

From modular arithmetic we know that if
$x \equiv i \pmod{n}$
then $(2x+y) \equiv (2i+y) \pmod{n}$

A can therefore be a DFA that accepts C_n if it is constructed as follows:

$$A = (\{0, 1, 2, 3, \ldots, n-1\}, \\ \{0, 1\}, \\ \delta(i, a) = (2i+a) \bmod n, \\ \{0\}, \\ \{0\})$$

(3) $\Sigma_2^0 = \{\epsilon\}$

$$\Sigma_2^1 = \left\{ \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix} \right\} = \Sigma_2 \Sigma_2^1$$

$$\Sigma_2^* = \bigcup_{i \in N} \Sigma_2^i$$

$$= \left\{ \epsilon, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}01\\01\end{bmatrix}, \begin{bmatrix}101\\010\end{bmatrix}, \ldots \right\}$$

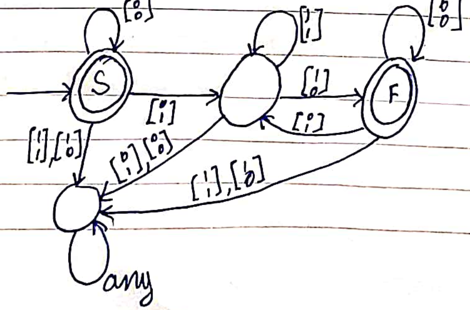
$$= \left\{ \begin{bmatrix}a\\b\end{bmatrix} \mid a, b \in \{0,1\}^* - \epsilon \right\}$$

Proof of regularity of A:

~~The language A can be represented by the regular expression:~~

$$\cancel{A = \left(\begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}1\\1\end{bmatrix} \right)^*}$$

the following ~~DFA~~ ~~solves~~ the automata solves the problem given by A:



5

regular expression ~~for~~ for the given DFA is

$R = 1 + 3$

$1 = 3a$

$2 = 1(a+b) + 3b + a^*$

$3 = 2b$

$2 = 3a(a+b) + 2b + a^*$

$= 2\,ba(a+b) + 2b + a^*$

$= 2\,[ba(a+b) + b] + a^*$

$= [ba(a+b) + b]^* + a^*$ by Arden's theorem

$\therefore R = 1 + 3$

$= 1 + [[ba(a+b)+b]^* + a^*]\,b$

$= [[ba(a+b)+b]^* + a^*]\,b\,(a+\varepsilon)$

~~8~~

7 Consider an NFA, $N = (Q, \Sigma, \delta, q_0, F)$

let $F = \{ q_i \mid q_i$ is ~~a final~~ an accept state $\}$

Adding a state p to Q such ~~tat~~ that

$\delta(q_i, \varepsilon) = p$

Now replace F with $F_2 = \{p\}$

This new automata N_2 is equivalent to N since every $q_i \in F$ can reach p.

8 1. Consider the string $\{^n \}^n$ which will have balanced parentheses and so is in L

Assuming L is regular and has a pumping length p.

$\{^p \}^p$ must satisfy the pumping condition

Since $|\{^p|$ ~~also~~ $\geq p$, the pumping string must be ~~...~~ a subset of $\{^p$.

For any subset A we select, pumping that string ~~...~~, say a times, will give

$A^a \{^p \}^p$ and thus an unbalanced number of parantheses.
hence L is not regular.

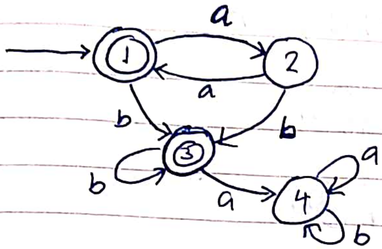
8 2. Assuming L is regular and has pumping length p

Consider the string $a^{p!} \in L$
and
any subset A of $a^{p!}$
On pumping that string A say n times, we get the string

$a^{p!+pn}$ which does not always yield a string that is in L. (i.e. ~~...~~ for all $n \neq p!$)

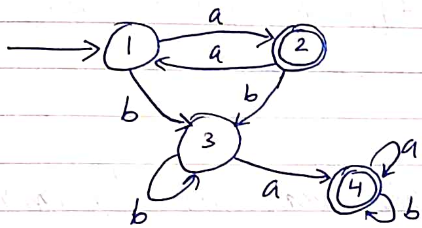
hence ~~...~~ L is not regular.

10 1.



10 2. Same as above

10 3. Consider the compliment of the above DFA (i.e. flipping the accept states)



$$1 = 2a + \varepsilon$$

$$2 = 1a$$

$$3 = 1b + 2b + 3b$$

$$4 = \cancel{a^* + b^*}\ 4a + 4b + 3a$$

$$2 = \cancel{3aa + a}\ (2a + \varepsilon)a$$

$$= 2aa + a$$

$$= a(aa)^* \qquad \therefore 1 = a(aa)^*a$$

$$\cancel{4 = 4a + 4b + 3a}$$

$$\cancel{= 4(a+b) + 1b + 2b +}$$

$$3 = 1b + 2b + 3b$$

$$= a(aa)^*ab + a(aa)^*b + 3b$$

$$= a(aa)^*(ab + b) + 3b$$

$$③ = [a(aa)^* (ab+b)]^* b$$

$$④ = ④a + ④b + ③a$$

$$= ④(a+b) + ③a$$

$$= (③a)^* (a+b)^*$$

$$= [[a(aa)^* (ab+b)]^* ba]^* (a+b)^*$$

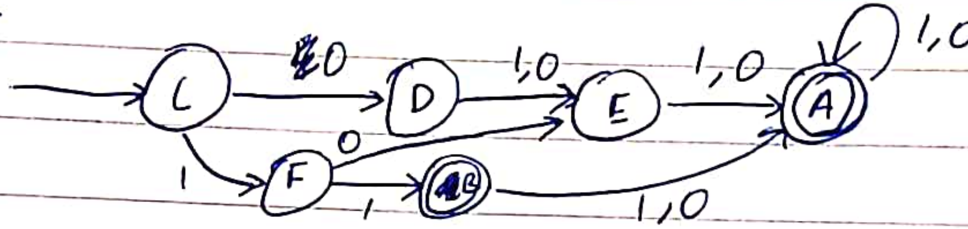
$$R' = ② + ④$$

$$= a(aa)^* + [[a(aa)^* (ab+b)]^* ba]^* (a+b)^*$$

⑨ a right linear grammar can be converted to a left linear grammar ~~by~~ as follows:

1. ~~com~~ construct an finite automata from the right linear grammar.
2. reverse the transition directions for each transition.
3. ~~In~~ Make all non-final states final and ~~a~~ all final states non-final.
4. ~~Co~~ Derive a left linear grammar from this automata.

⑥ 1. D =



Assuming that x is ~~in~~ in the big endian representation (whose 4 = 001)

⑥ 2. → 333 = 101100101 in big endian

will reach final state A and remain in a loop in this sequence: C → F → E → A

→ 420 = 001001011 in big endian

will reach final state A and remain ~~a~~ in a loop in this sequence: C → D → E → A