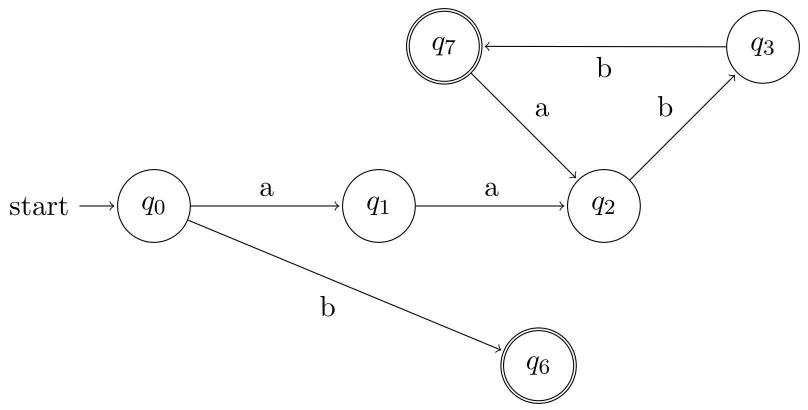
ICCS310: Assignment 2

Possawat Sanorkam
possawat2017@hotmail.com

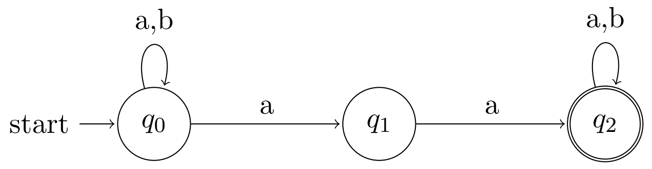
January 26, 2021

1: Regex to NFA/DFA

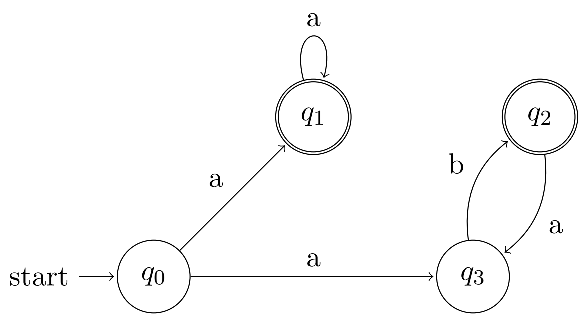
(1) $a(abb)^* + b$



(2) $(a + b)^*aa(a + b)^*$



(3) $a^+ + (ab)^+$



2: Finite-State Machines to Regex

(1) $\varnothing^*$ (Rejecting any input)

(2) $a^* + a^*b^+a^+b$ (Contains only as or any pattern of as to bs to as to bs.)

3: Binary Addition

$$A = \{w \in \Sigma^* \mid \text{ the bottom row of w is the sum of the top two rows } xy \in L_1\}$$

Prove that A is regular.

Proof: Since, A only accept column vectors of size 3 such that the bottom row of w is the sum of the top two rows, it is difficult create a machine that recognize A directly. We know that binary addition start by summing the least significant bits, including transferring a carry when the bit overflow. So, reading the string in reverse will be simpler since we can transfer the carry from the last column to the next column on the left directly.

From the lemma, if L is a regular language, then L^R is also regular. We want to show that A^R is a regular language. Let $L(M) = A^R$. So, M is a machine that recognize A^R, else we can call it a binary adder.

M is a machine that translate each vector whether it produce carry bit and send it to the next state. It only accepts the string that follow its calculation (as shown in the transition function). If there is a carry bit, we send it to the state which contains a carry bit. So, there are 2 states, $q_0,$ and q_1.

q_0 is the accepting state with no carry bit transferred.

q_1 is the state with a carry bit. (Not done with the addition yet)

We can mathematically define a function of LSB addition as $a \odot b = c$ and a function of carry bit as $carry(x, y, z)$ where $a, b, c, x, y, z \in \Sigma$. Besides, $a \odot b = c$ works the same way as XOR logic operator.

$$1 \odot 1 = 0$$
$$0 \odot 0 = 0$$
$$1 \odot 0 = 1$$
$$0 \odot 1 = 1$$

The transition of this machine follows that $(x_1 \odot x_2) \odot$ "carry bit" $= x_3$ must be satisfied and the next transition depends on the carry bit from $carry(x_1, x_2,$ "carry bit" $)$ whether it will be q_0 or q_1.

The transition function is defined as

$$q_j = \delta(q_i, \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix})$$

whenever $(x_1 \odot x_2) \odot i = x_3$ and $carry(x_1, x_2, i) = j$.

Hence, A^R is regular and that makes A regular also from the lemma.

Therefore, A is regular. □

4: Division Operation?

$$\frac{L_1}{L_2} = \{x \mid \exists y \in L_2 \text{ s.t. } xy \in L_1\}$$

Prove that if L_1 and L_2 are regular, then $\frac{L_1}{L_2}$ is also regular.

Proof: From the lemma, for every regular expression R, there is a DFA that recognizes the language $L(R)$. Suppose L_1 and L_2 are regular, then we let $M_1 = (Q, \Sigma, \delta, q_0, F_1)$ which accepts L_1 and DFA $M_2 = (Q, \Sigma, \delta, q_0, F_2)$ which accepts L_2. We want to show that there exist M_3 that can recognize $L_3 = \frac{L_1}{L_2}$.

L_3 then can be recognized by some DFA $M_3 = (Q, \Sigma, \delta, q_0, F_3)$. First, each state in Q must make $\delta^*(q_i, y) \in F_1, \forall q_i \in Q, \exists y \in L_2$. By changing the starting state to each state in Q, we will have $M_i = (Q, \Sigma, \delta, q_i, F)$, and the machine will be able to recognize some language when using $y \in L_2$. Also, if $L_2 \cap L(M_3) \neq \emptyset$, then $q_i \in F_3$. After that, $x \in \frac{L_1}{L_2}$ implies that $x \in L(M_3)$ and $\exists y \in L_2$ such that $xy \in L_1$. Since $xy \in L_1$, then $\delta(q_0, x) = s, \exists s \in Q$ and $\delta(s, y) \in F$. Since we used x to change state, $s \in F_3$ and M_3 accepts x. Thus, M_3 exists.

From the observation, L_1, which is regular, contains accepting states that made of xy from L_2 and L_3. Also, L_2, which is regular, recognized by M_2 and we can choose any number to be an accepting state in L_2 (As long as we accept at least a number). Since L_3 can be any state (number) also, there always exist x that will satisfied $xy = z$. In fact, if $x \in L(M_3)$, then $x \in \frac{L_1}{L_2}$. Hence, $\frac{L_1}{L_2}$ is regular.

Therefore, if L_1 and L_2 are regular, then $\frac{L_1}{L_2}$ is also regular. □

5: Does It Accept Everything?

Let $M = (Q, \Sigma, \delta, q_0, F)$.

Solution: Every finite-state machine is simply a directed graph, then we can check it by performing this algorithm.

Let $strs = \Sigma^*$, given size of Σ is a constant. Also, E = edges that we can refer to δ. If it is impossible to reject, then M accepts everything. If we can get other states that are not included in F, then we are guaranteed that some strings will be rejected for sure. We can simply do a BFS to check it.

The algorithm goes like this.

```
1. Create a graph of DFA from M, using Q as vertices and delta as edges.
2. Perform an efficient Breadth-First-Search and check if there are accepting states.
        - BFS start from any s in sigma (Start with any alphabet).
        - Visisted states will be skipped and not visited again.
        - Stop when there are no other states to go next.
3. If the number of accepting states is less than the number of states,
   then some strings will be rejected. Else, M will accept anything.
```

The time complexity of this algorithm is $O(|Q|)$ since the cost of BFS depends on the number of states. The only trade-off is that there must be enough memory to store all the states, so we will assume that we have unlimited resources this time.

6: All The Same?

Multiplication and power simply make them equivalent like $2 * 4 = 4 * 2 === 2^4 = 4^2$.