Decision Problems for Regular Languages

Emptiness Problem

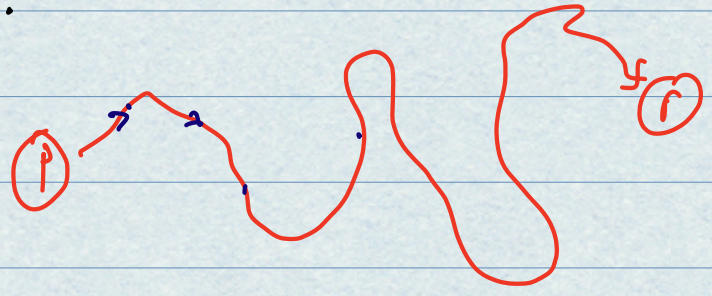
Input: A DFA M

Decide: Is $L(M) = \emptyset$?

Solution:

We first define a notion of **reachability** (accessibility).

Definition: Given $M = (Q, \Sigma, \delta, q_0, F)$ and $p, r \in Q$. We say "**r is reachable from p**" if there is a path from p to r in M.



For a state $p \in Q$, define

- $Acc(p) = \{ r \mid r$ is reachable from p via a path of length $\geq 0 \}$

- $Acc_+(p) = \{r \mid r$ is reachable from p via a path of length $\geq 1\}$

Note: • $p \in Acc(p)$
• $Acc_+(p) \subseteq Acc(p)$
• $Acc(p) = Acc_+(p) \cup \{p\}$

Q: How to compute $Acc(p)$ and $Acc_+(p)$ effectively for a state p?

Algorithm: to compute $Acc_+(p)$
Idea: Compute inductively.

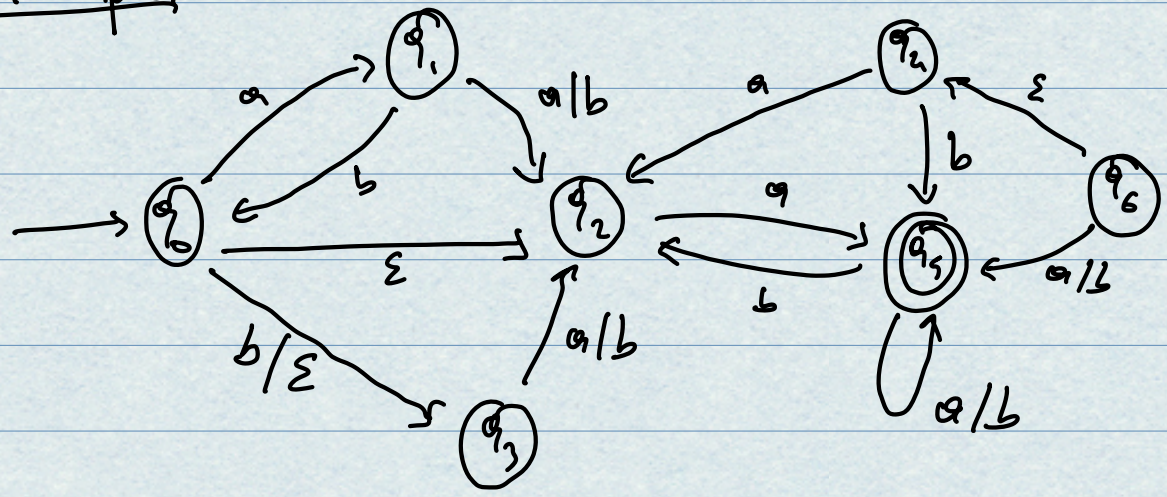
- Collect states that are reachable from p in one step.

- Recursively collect states that are reachable from the newly added states

- Repeat the above until no more states are added.

Example:

M:



- Compute $Acc(q_0)$ and $Acc_t(q_0)$

$A_1 = \{q_1, q_2, q_3\}$

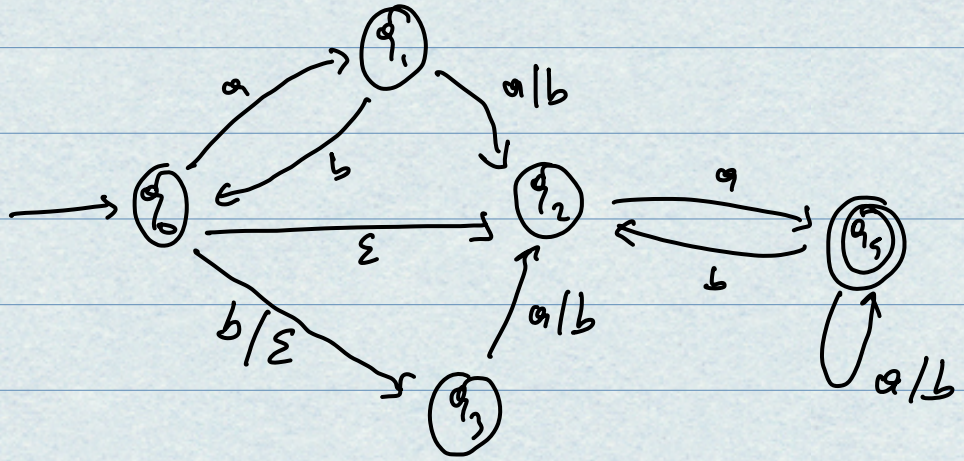
$A_2 = \{q_1, q_2, q_3, q_0, q_5\}$

$A_3 = \{q_1, q_2, q_3, q_0, q_5\} = Acc_t(q_0)$

$\Rightarrow$ q_4 and q_6 are not reachable from q_0.

$\Rightarrow Acc(q_0) = Acc_t(q_0) \cup \{q_0\}$

Now q_4 and q_6 can be removed from M.



Proposition: Every FA can be converted to an equivalent FA in which all states are reachable from q_0, the initial state.

* Emptiness problem is solvable.

$L(m) \neq \emptyset \iff Acc_+(q_0) \cap F \neq \emptyset$.

Finiteness Problems (FP)

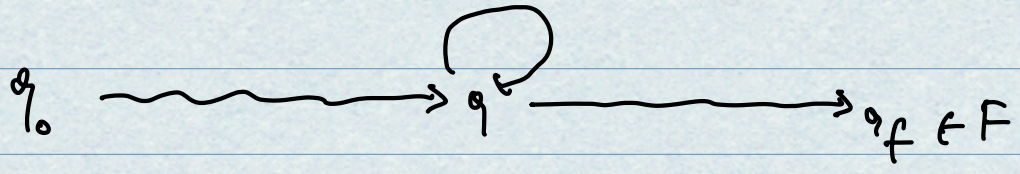
Input: A DFA m

Decide: Is $L(m)$ finite?

Proposition: FP is solvable

- Suppose $L(m)$ is infinite.
  $\Rightarrow$ there exist strings in $L(m)$ that are arbitrarily long.

- Let n = # of states of m
  By pumping lemma, a string $z \in L(m)$ with $|z| \geq n$ can be accepted by a computation



Note:

$L(M)$ is infinite $\Rightarrow$
- $q \in Acc(q_0)$
- $q \in Acc_+(q)$
- $Acc(q) \cap F \neq \emptyset$.

Also the converse of the above holds.

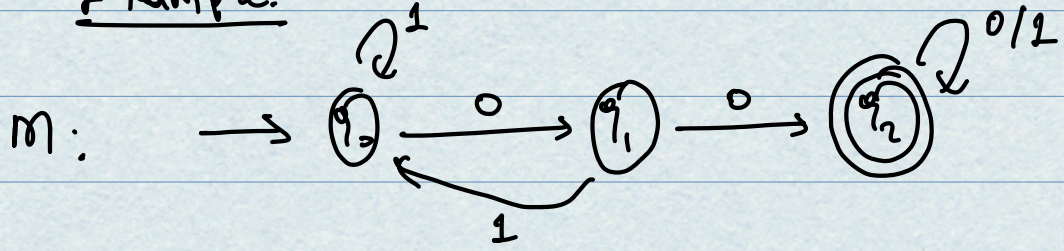
Therefore,

$L(M)$ is infinite $\Leftarrow$
- $q \in Acc(q_0)$ (1)
- $q \in Acc_+(q)$ (2)
- $Acc(q) \cap F \neq \emptyset$. (3)

To check if $L(M)$ is infinite
- Compute $Acc(q_0)$
- Find a state $q \in Q$
- Compute $Acc(q)$ and $Acc_+(q)$
- Check if (1), (2) and (3) hold.

If some $q \in Q$ can be found, then $L(M)$ is infinite, otherwise not!

Example:



- Let us pick q_2.

(1) $q_2 \in Acc_+(q_0)$

(2) $q_2 \in Acc(q_0)$

(3) $Acc_+(q_2) \cap F \neq \emptyset$

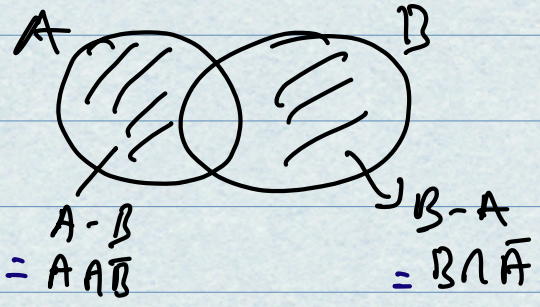
$\Rightarrow L(M)$ is infinite.

Equivalence Problem:

Input: Two DFAs M_1 and M_2

Decide: $L(M_1) = L(M_2)$?

Recall: $A = B \iff A - B = \emptyset \wedge B - A = \emptyset$



$\Rightarrow A = B \iff A \cap \bar{B} = \emptyset \wedge B \cap \bar{A} = \emptyset.$

Hence $L(M_1) = L(M_2)$ iff

$$\underbrace{L(M_1) \cap \overline{L(M_2)} = \emptyset}_{(*)} \wedge \underbrace{L(M_2) \cap \overline{L(M_1)} = \emptyset}_{(**)}$$

We can construct DFAs for (*) and (**) and check if their languages are empty, which we know how to solve.

Example:

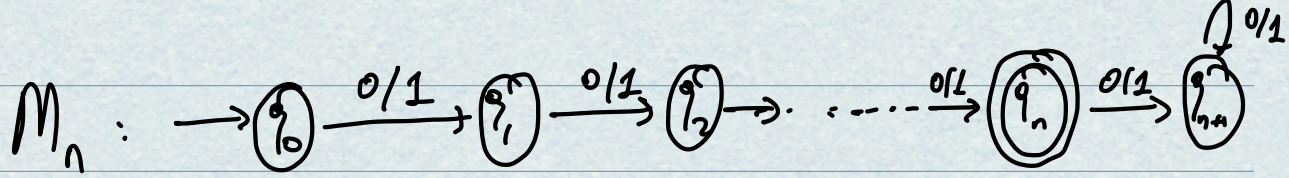
Input: A DFA M and $n \geq 0$.

Question: Does M accept a string z with $|z| = n$.

Show that this problem is solvable.

Proof:

- Given: M, $n \geq 0$
- Construct a DFA M_n that accepts strings in Σ^n (set of strings of length n)



Note: - $L(M)$ is regular

- $L(M_n)$ is regular

- Define a DFA $M \times M_n$ to accept $L(M) \cap L(M_n)$.

$\Rightarrow$ M accepts a string of length n iff $L(M) \cap L(M_n) \neq \emptyset$.

(solvable)