12: Equivalence of RE and FA

Learning Outcomes: At the conclusion of the chapter, the student will be able to:

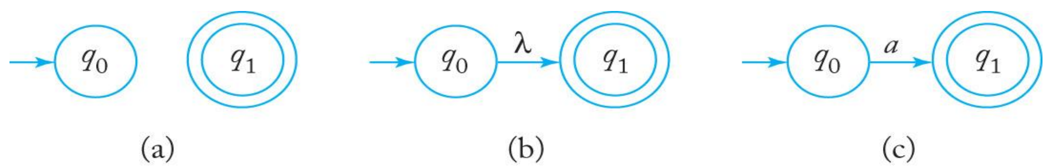
- Construct a finite automaton to accept the language denoted by a regular expression

For any regular expression r, there is a nondeterministic finite automaton that accepts the language denoted by r. Since nondeterministic and deterministic accepters are equivalent, regular expressions are associated precisely with regular languages. A constructive proof provides a systematic procedure for constructing an ε-NFA that accepts the language denoted by any regular expression.

1. Conversion of given RE to FA

The method is also known as Thompson's Construction

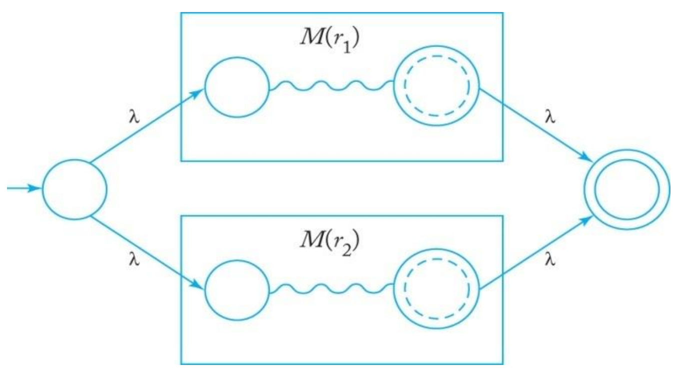
Basis of Construction:



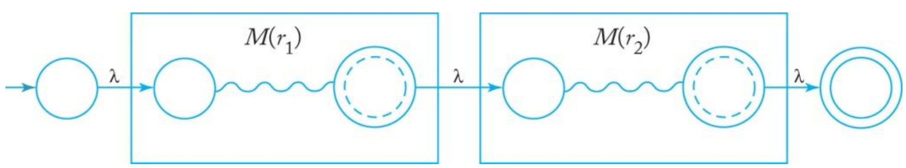
(a) nfa accepts ∅. (b) nfa accepts {λ}. (c) nfa accepts {a}

Inductive steps:

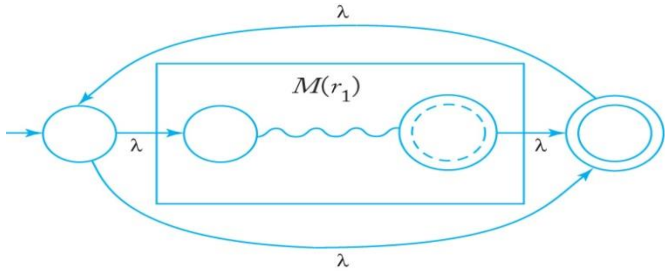
(a) Given schematic representations for automata designed to accept $L(r_1)$ and (r_2), an automaton to accept $L(r_1 + r_2)$ can be constructed as shown in following figure



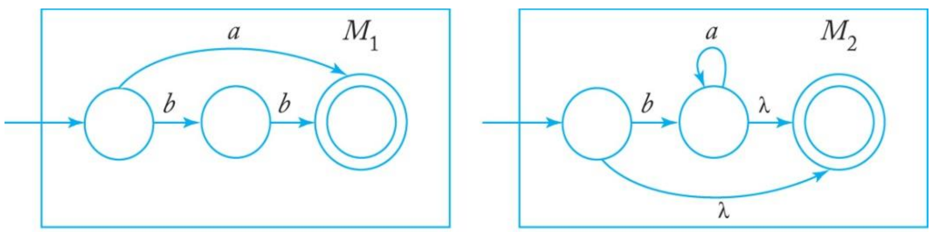
(b) Given schematic representations for automata designed to accept $L(r_1)$ and (r_2), an automaton to accept $L(r_1r_2)$ can be constructed as shown in following figure



(c) Given a schematic representation for an automaton designed to accept $L(r_1)$, an automaton to accept $L(r_1*)$ can be constructed as shown in following figure

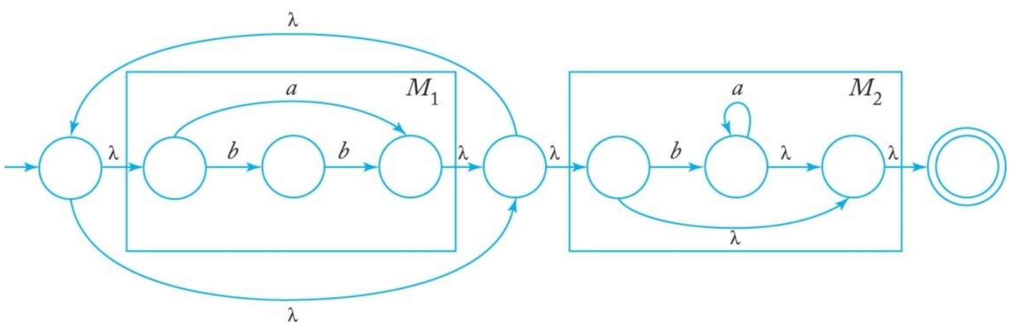


Example 12.1 Given the regular expression r = (a + bb)* (ba* + λ), a nondeterministic FA to accept L(r) can be constructed systematically as follows



(a) (b)

(a) M_1 accepts L (a + bb). (b) M_2 accepts L (ba∗ + λ).

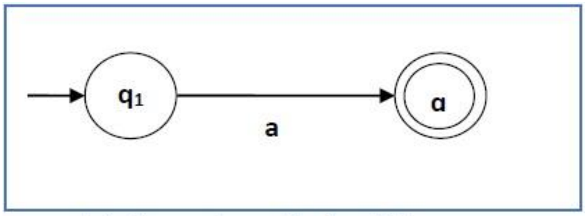


Automaton accepts L((a + bb)* (ba* + λ)).

We can use Thompson's Construction to find out a Finite Automaton from a Regular Expression. We will reduce the regular expression into smallest regular expressions and converting these to NFA and finally to DFA.

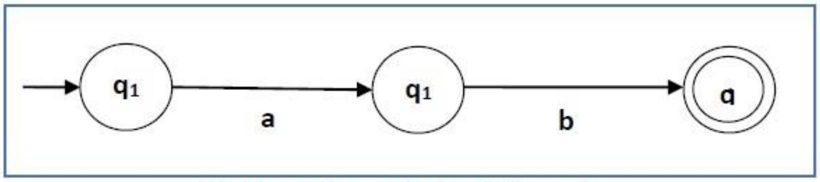
Some basic RA expressions are the following −

Case 1 − For a regular expression ‘a’, we can construct the following FA −



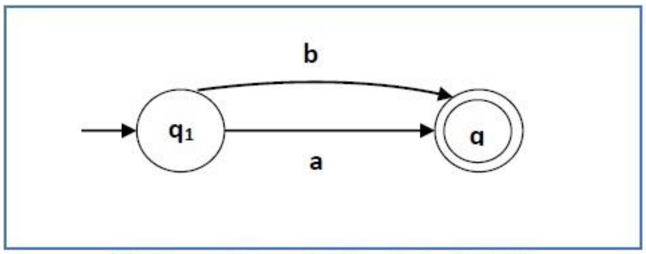
Finite automata for RE = a

Case 2 − For a regular expression 'ab', we can construct the following FA −



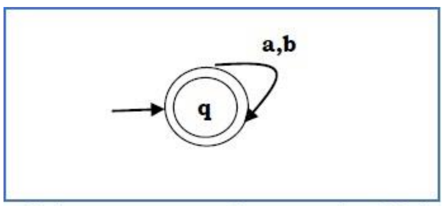
Finite automata for RE = ab

Case 3 − For a regular expression (a+b), we can construct the following FA −



Finite automata for RE= (a+b)

Case 4 − For a regular expression (a+b)*, we can construct the following FA −



Finite automata for RE= (a+b)*

Method

Step 1 Construct an NFA with Null moves from the given regular expression.

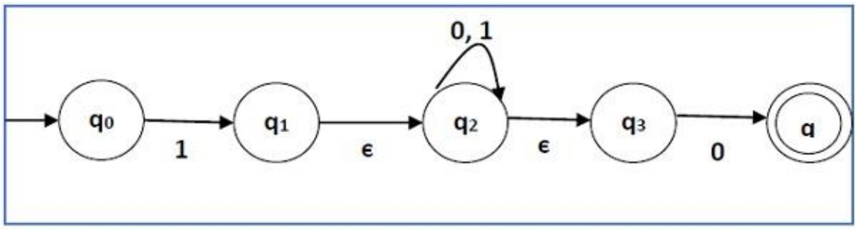
Step 2 Remove Null transition from the NFA or convert it into its equivalent DFA.

Example 12.2

Convert the following RA into its equivalent DFA − 1 (0 + 1)* 0

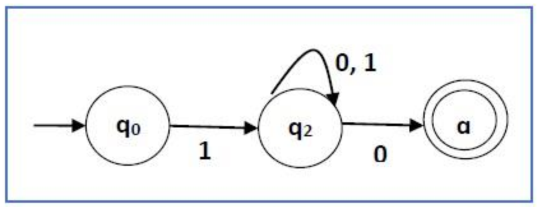
Solution

We will concatenate three expressions "1", "(0 + 1)*" and "0"



NDFA with NULL transition for RA: 1 (0 + 1)* 0

Now we will remove the **ε** transitions. After we remove the **ε** transitions from the NDFA, we get the following −



NDFA without NULL transition for RA: 1 (0 + 1)* 0

It is an NFA corresponding to the RE − 1 (0 + 1)* 0. If you want to convert it into a DFA, simply apply the method of converting **ε**-NFA to DFA (or NFA to DFA) as discussed in earlier chapters.