

LECTURE
5

Kleene's Theorem

Regular Expression to Finite State Automata

Introduction

The class of languages that can be represented by regular expressions is equivalent to the class of languages accepted by finite state automata -- the regular languages.

1. Kleene's Theorem :

- For any regular expression (RE), r , that represents language $L(r)$, there is a finite automaton that accepts that same language.
- For any finite automaton, M , that accepts language $L(M)$, there is a regular expression that represents the same language.

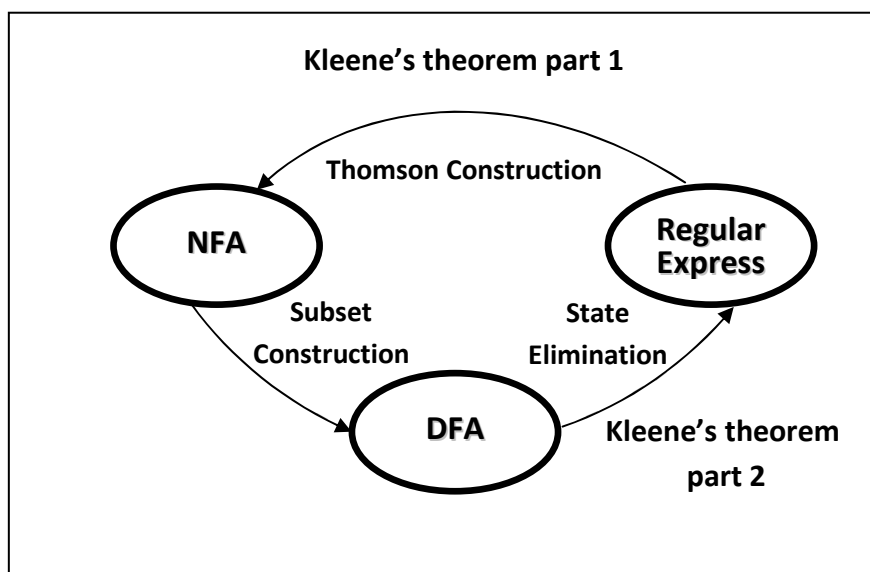


Fig. 1: Kleene's Theorem

It is clear that DFA can be converted into RE and vice versa using some algorithms or techniques. For converting RE to DFA, first we convert RE to NFA (Thomson Construction) and then NFA is converted into DFA (Subset construction). Also, every DFA can be converted to RE using State elimination method.



2. Regular Expression (RE):

One of the most useful ways to define a regular language is by means of a regular expression. A regular expression can consist of the following constructions:

Regular expression could be the empty string, represented by ϵ .

Regular expression could be any symbol (α) from the alphabet of the language.

Regular expression could be selection of regular expressions r_1 and r_2 , written as $r_1|r_2$.

Regular expression could be a sequence of regular expressions r_1 and r_2 , written as r_1r_2 .

Regular expression could be zero or more iterations of a regular expression r , written as r^* .

3. Language Defined by a Regular Expression:

With every regular expression we can associate a regular language. Conversely, every regular language can be obtained from a regular expression. For example, the different regular languages over $\Sigma=\{a, b\}$ according to the regular expressions are listed in the following tables:



Table 1: Regular Language Corresponding to Regular Expression

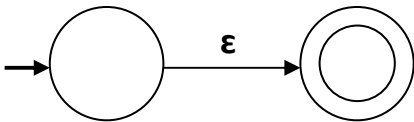
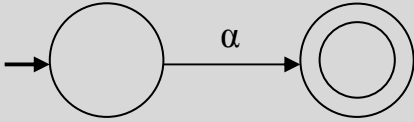
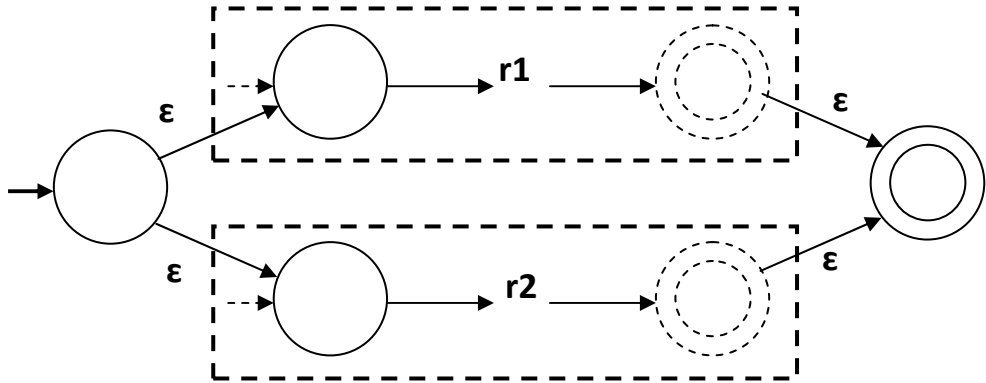
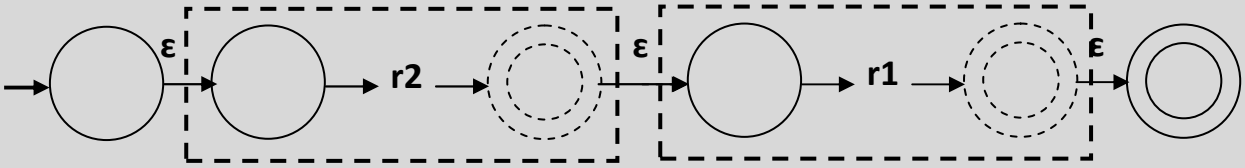
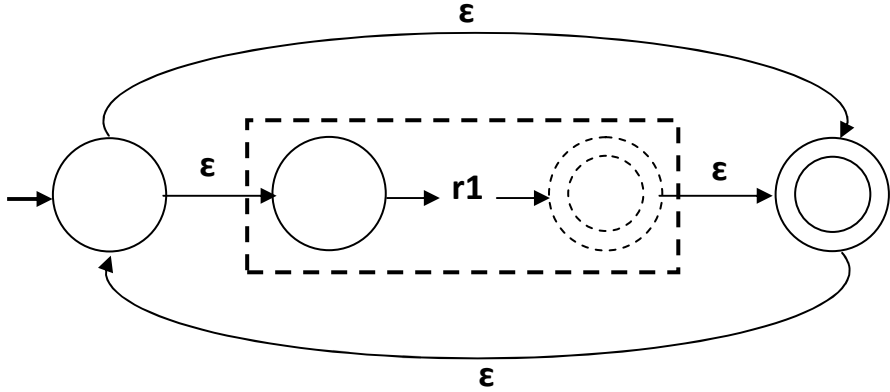
	Regular Expression	Regular Language
1.	a^*	$\{\epsilon, a, aa, aaa, aaaa, \dots\}$
2.	$(aa)^*$	$\{\epsilon, aa, aaaa, aaaaaa, aaaaaaaa, \dots\}$
3.	aa^*	$\{a, aa, aaa, aaaa, aaaaa, \dots\}$
4.	a^+	$\{a, aa, aaa, aaaa, \dots\}$
5.	ba^*	$\{b, ba, baa, baaa, \dots\}$
6.	$a b$	$\{a, b\}$
7.	$(a b)^*$	$\{\epsilon, a, b, aa, bb, ab, ba, aaa, bbb, aab, bba, \dots\}$
8.	$a b^*$	$\{a, \epsilon, b, bb, bbb, \dots\}$
9.	$(ba)^*$	$\{\epsilon, ba, baba, bababa, babababa, \dots\}$
10.	$a(a b)^*$	$\{a, aa, ab, aaa, abb, aba, abaa, \dots\}$
11.	aa^*b	$\{ab, aab, aaab, aaaab, \dots\}$

Table 2: Regular Expression Corresponding to Regular Language

	Regular Language	Regular Expression
1.	Any number of 0, followed any number of 10 and 11.	$0^*(10 11)^*$
2.	Any number of copies of 10 (including the null string)	$(10)^*$
3.	Any string beginning with 0	$0(1 0)^*$
4.	Any string not ending with 0	$(1 0)^*1$
5.	Any number of 1s followed by a 0	1^*0
6.	Any number of 1s followed by one or more 0s	1^*00^*
7.	String of any number of 1s or 00s or some of each in a row	$(1 00)^*$
8.	Set of bit string of length at least 3 that ends with 00	$(0 1)(0 1)^*00$
9.	Set of bit strings of odd length	$(0 1)((0 1)(0 1))^*$
10.	Set of bit strings that contain exactly one 1	0^*10^*
11.	Set of bit strings ending in 1 and not containing 000	$(1 01 001)^*$
12.	Set of bit strings begin with 0 end with 10.	$0(0 1)^*10$
13.	Set of bit strings doesn't contain any consecutive 0's.	$(1 01)^*$

4. Converting RE to FSA (Thomson Construction):

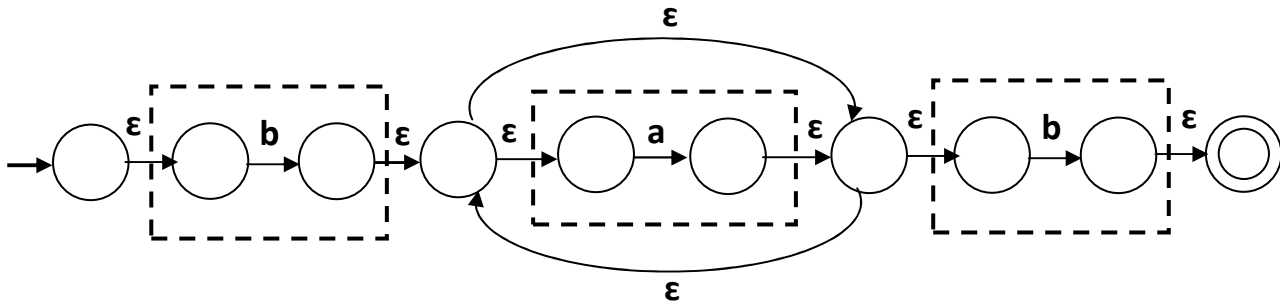
Every regular expression can be converted to FSM and vice versa. The conversion of basic regular expressions to FSM is shown as in the following table:

Table 1: Conversion of The Basic RE to FSM	
RE	FSM
ϵ	
a	
$r1 r2$	
$r1r2$	
$r1^*$	

Example 1:

Convert the regular expression ba^*b into a FSM

Solution:

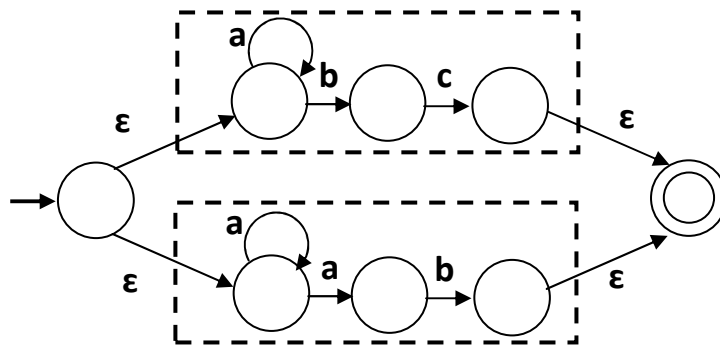


This NFA is converted DFA using subset construction (as explained in lecture 4)

Example 2:

Convert the regular expression $a^*bc|a^*ab$ into a FSM

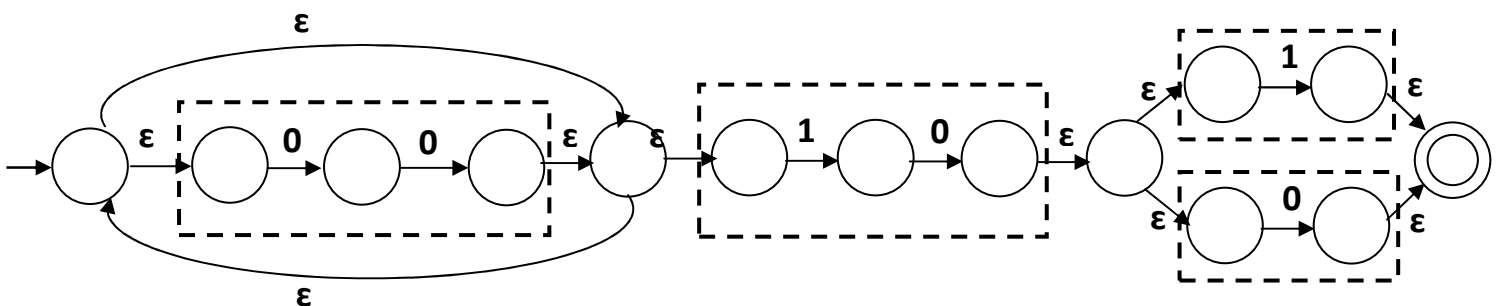
Solution:



Example 3:

Convert the following regular expression to FSM: $(00)^*10(1|0)^*$

Solution:





5. Homework:

HW 1:

Convert the following regular expression $(aab)^*ab \mid (a \mid b)^*$ into a FSM.

HW 2:

Write down regular expression representing the following languages over the alphabet $\{0, 1\}$. The language describes by the following words: (ϵ or a string that ends with 1 and has one or more 0s before each 1).