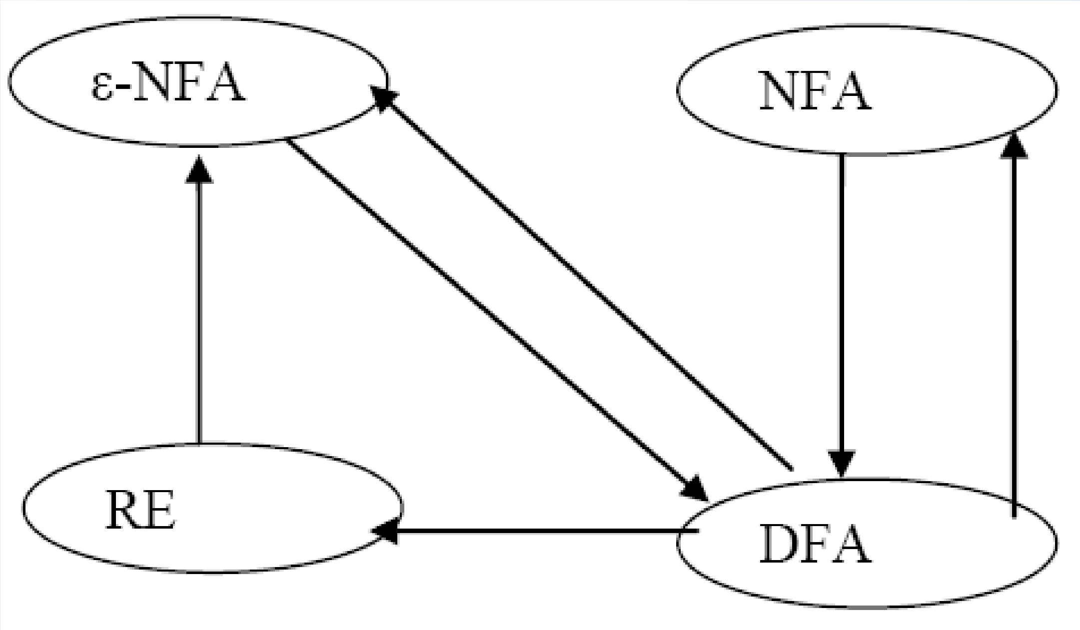
CSC-257
Theory Of Computation

(BSc CSIT, TU)

Ganesh Khatri
kh6ganesh@gmail.com

Finite Automata and Regular expressions



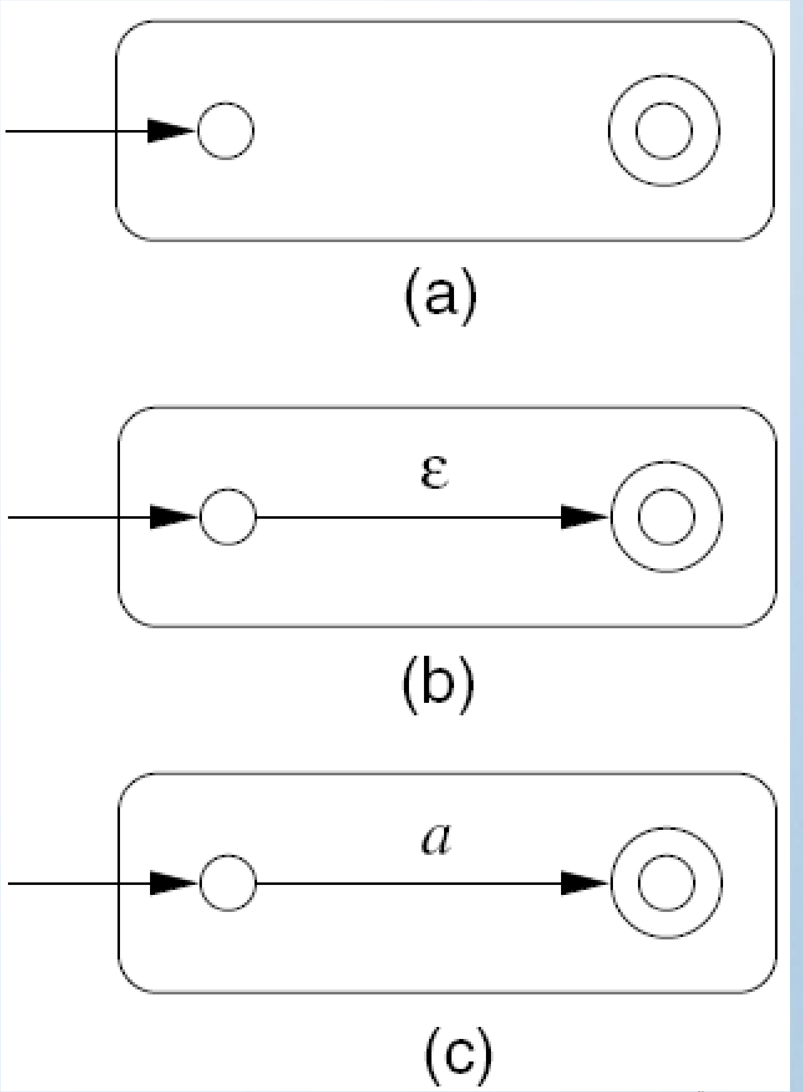
- The regular expression approach for describing language is fundamentally different from the finite automaton approach.
- However, these two notations turn out to represent exactly the same set of languages, which we call regular languages
- In order to show that the RE define the same class of language as Finite automata, we must show that :
  - Any language defined by one of these finite automata is also defined by RE.
  - Every language defined by RE is also defined by any of these finite automata
- We can proceed as :

Conversion of RE to Є-NFA

- **Theorem :** Every language defined by a regular expression is also defined by a finite automaton. [For any regular expression r, there is an Є-NFA that accepts the same language represented by r]
- **Proof :** Let L =L(r) be the language for regular expression r, now we have to show there is an Є-NFA E such that L (E) =L
- The proof can be done through structural induction on r, following the recursive definition of regular expressions
- **Basis :** For this we know
  a) Φ - represents language {Φ} : an empty language
  b) Є - represents language {Є} : language for empty strings
  c) ‘a’ - represents language {a}

Conversion of RE to Є-NFA

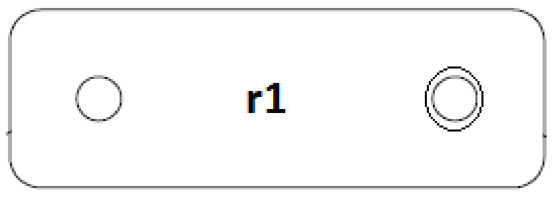
- The Є-NFA accepting these languages can be constructed as;
- a) r = Φ
- b) r = Є
- c) r = a

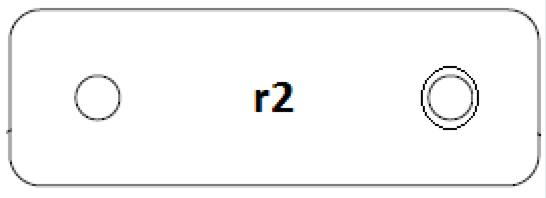


This forms the basis steps

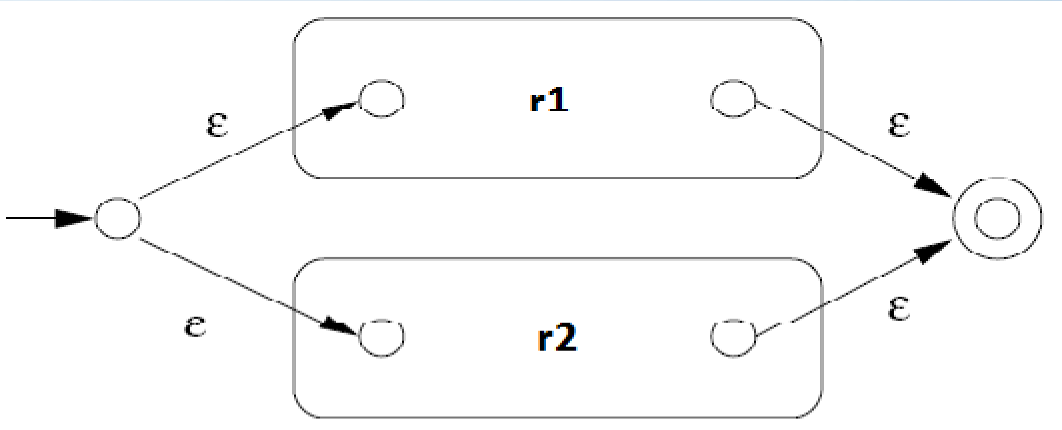
Conversion of RE to Є-NFA

- **Induction :** Let r be a regular expression representing language L(r) and r1,r2 be regular expressions for languages L(r1) and L(r2) respectively.
- There are for cases :
- **1. For union(+):** From basis step we can construct Є-NFA's for r1 and r2. Let the Є-NFA's be M1 and M2 respectively



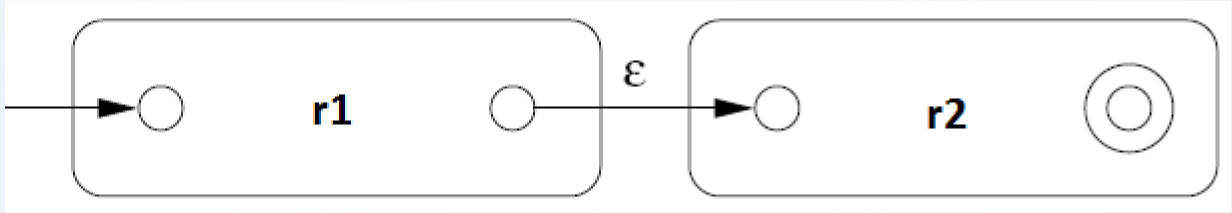


- Then, r=r1+r2 can be constructed as:
- The language of this automaton is L(r1) U L(r2) which is also the language represented by expression r1+r2



Conversion of RE to Є-NFA

- **2. For concatenation(.) :** Now, r = r1.r2 can be constructed as;

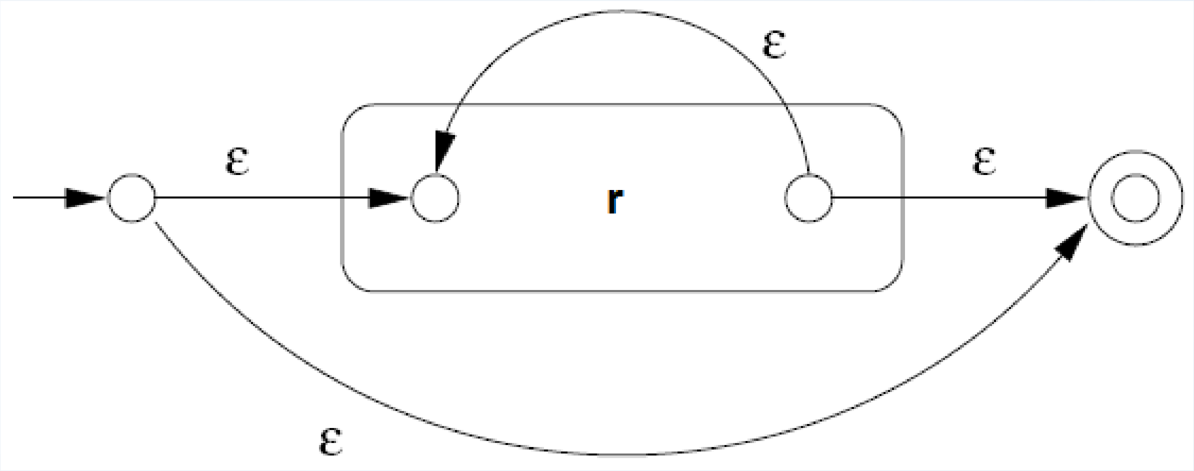


- Here, the path from starting to accepting state go first through the automaton for r1, where it must follow a path labeled by a string in L(r1), and then through the automaton for r2, where it follows a path labeled by a string in L(r2).

- Thus, the language accepted by above automaton is L(r1).L(r2).

Conversion of RE to Є-NFA

- **3. For Kleene closure(*) :** Now, r* Can be constructed as;



- Clearly language of this Є-NFA is L(r*) as it can also just Є as well as string in L(r), L(r)L(r), L(r)L(r)L(r) and so on.
- Thus covering all strings in L(r*).

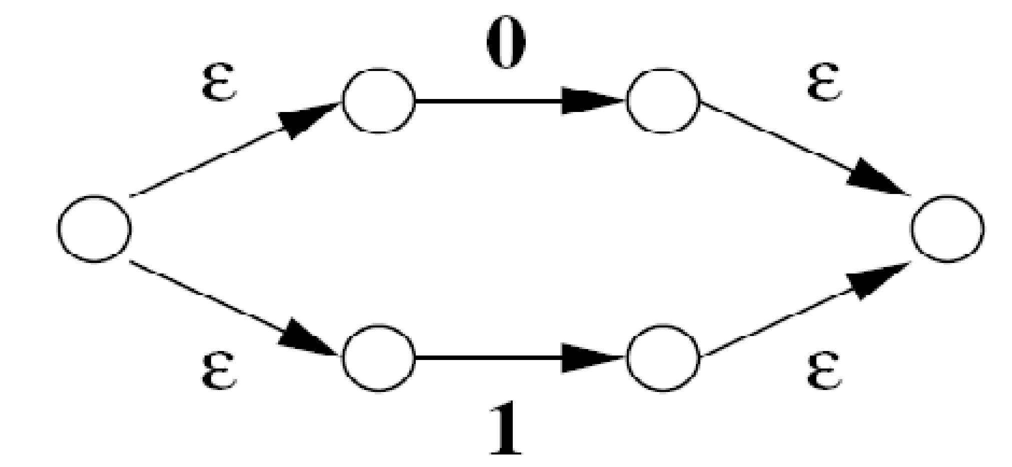
Conversion of RE to Є-NFA

- **4. Finally, for regular expression (r) :** the automaton for r also serves as the automaton for (r), since the parentheses do not change the language defined by the expression

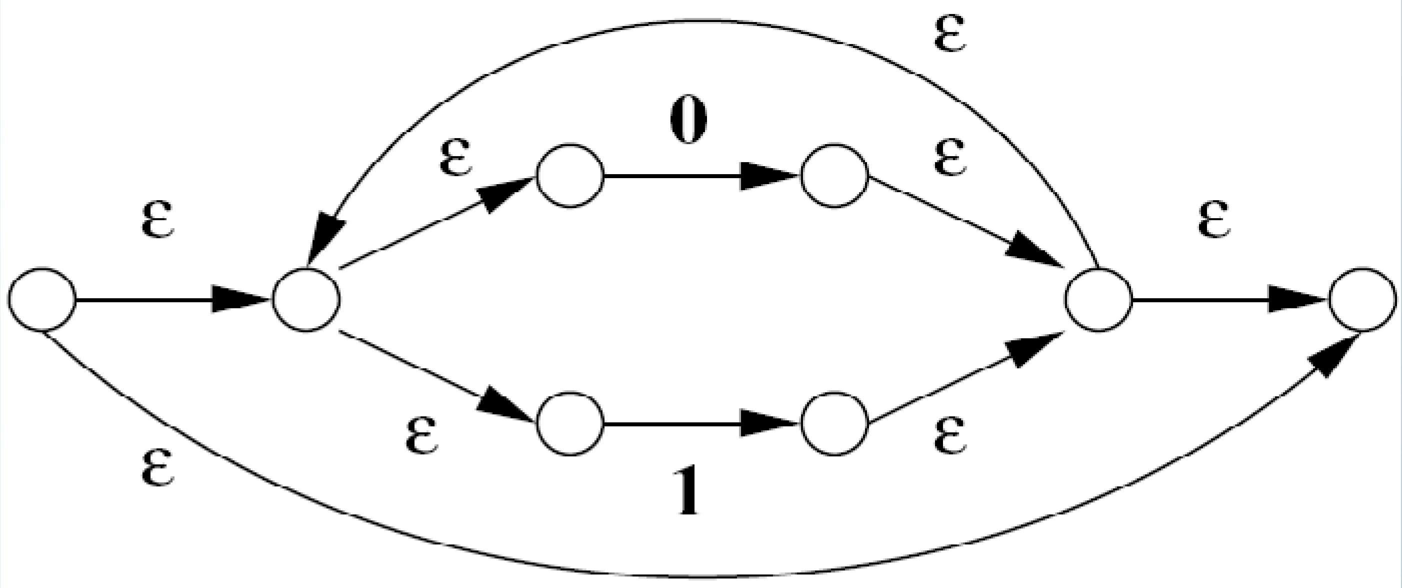
- This completes the proof.

Conversion of RE to Є-NFA

- Convert the regular expression (0+1) into Є-NFA

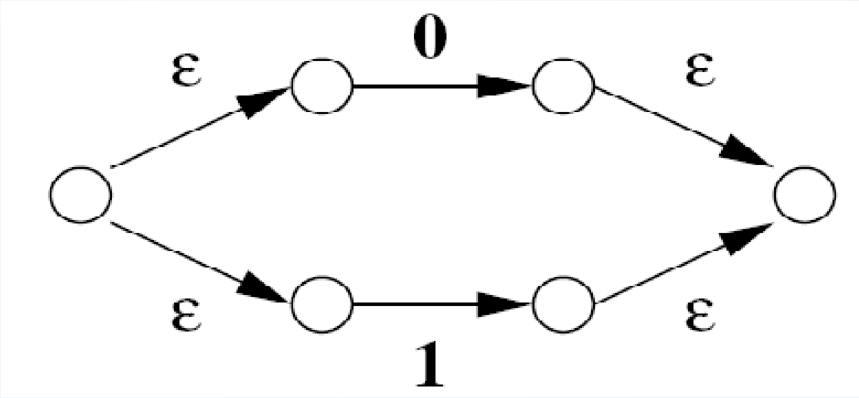


- Convert the regular expression (0+1)* into Є-NFA

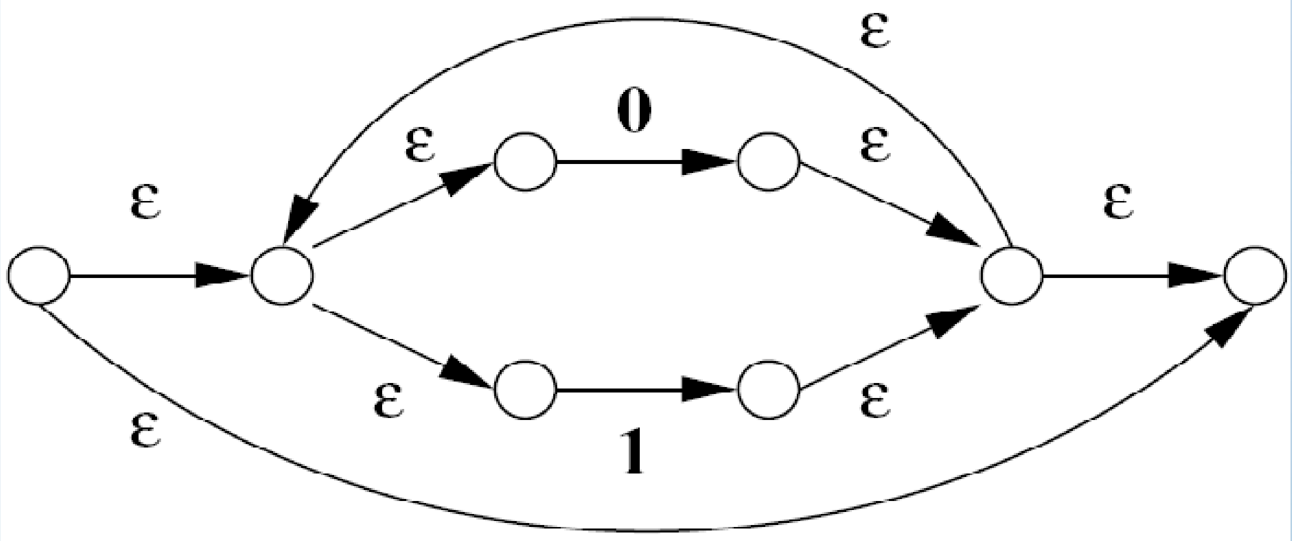


Conversion of RE to Є-NFA

- Convert the regular expression (0+1)*1(0+1) into Є-NFA
- Here, Є-NFA for (0+1) is

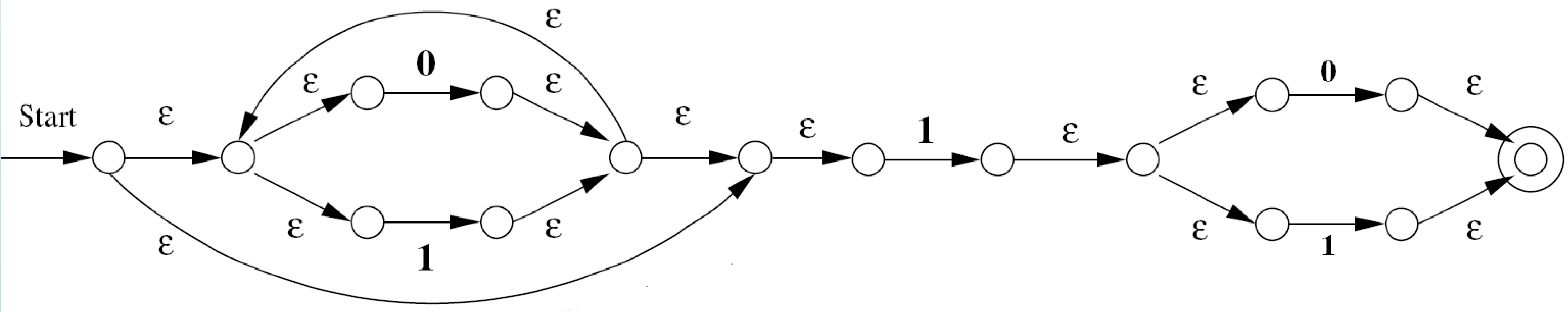


- And Є-NFA for (0+1)* is



Conversion of RE to Є-NFA

- Convert the regular expression (0+1)*1(0+1) into Є-NFA
- Now final Є-NFA of given RE is
- Note : Union, Concatenation, and Kleene closure used.



Conversion of RE to Є-NFA : Exercises

1. Convert the regular expression 01* into Є-NFA
2. Convert the regular expression (0+1)01 into Є-NFA
3. Convert the regular expression 00(0+1)* into Є-NFA
4. Convert the regular expression 0*10* into Є-NFA
5. Convert the regular expression 1*(01*01*) into Є-NFA
6. Convert the regular expression 0*10*10* into Є-NFA
7. Convert the regular expression (1+0)*11 into Є-NFA