CS372
FORMAL LANGUAGES & THE THEORY OF COMPUTATION

Dr. Nguyen Thi Thu Huong.
Phone: +84 24 38696121, Mobi: +84 903253796
Email: huongnt@soict.hust.edu.vn,
huong.nguyenthithu@hust.edu.vn

Unit 3

Closure Properties, Regular Expressions

Closure Properties

- We carry out operations on one or more languages to obtain a new language
- It’s helpful when interesting properties are preserved
- A variety of operations which preserve regularity, i.e. the universe of regular languages is closed under these operations: union, intersection, complementation, star, concatenation ….

Regular operations

- Union: $\cup$

  Example: $0 \cup 1$;

- Concatenation: ° (In writing regular expressions, no character is used to represent this operation)

  Example: 0 1

- Star*

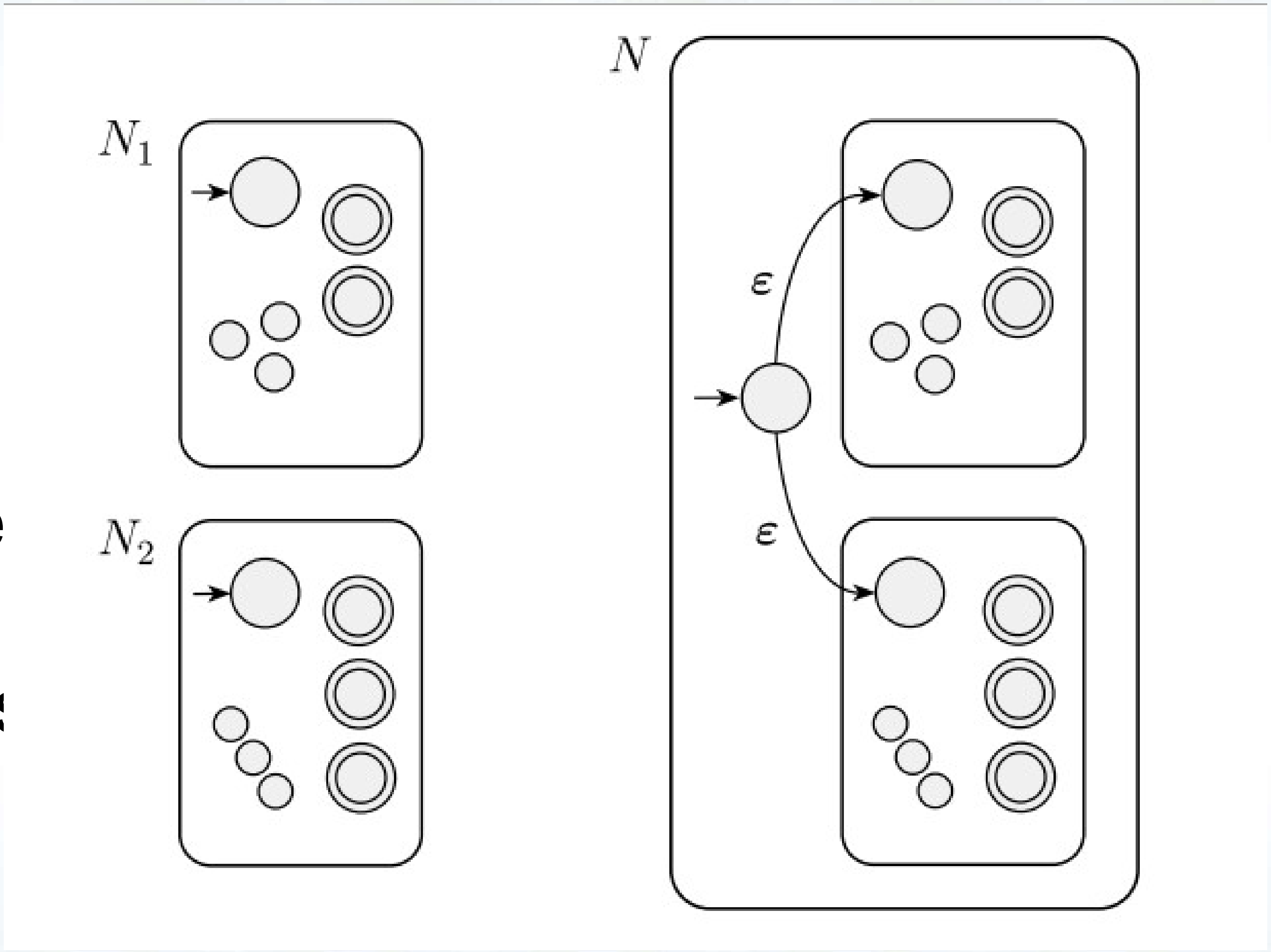
  Example: 0^*

Theorem

- The class of regular languages is closed under the union operation.
- PROOF IDEA
  - Given regular languages L_1 and L_2 and want to prove that $L_1 \cup L_2$ is regular.
  - The idea is to take two NFAs, N_1 and N_2 for L_1 and L_2, and combine them into one new NFA, N

Construction of an NFA N to recognize $L_1 \cup L_2$

- Add a new start state that branches to the start states of the old machine with ε arrows



Formal Description

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize L_1,

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize L_2.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $L_1 \cup L_2$.

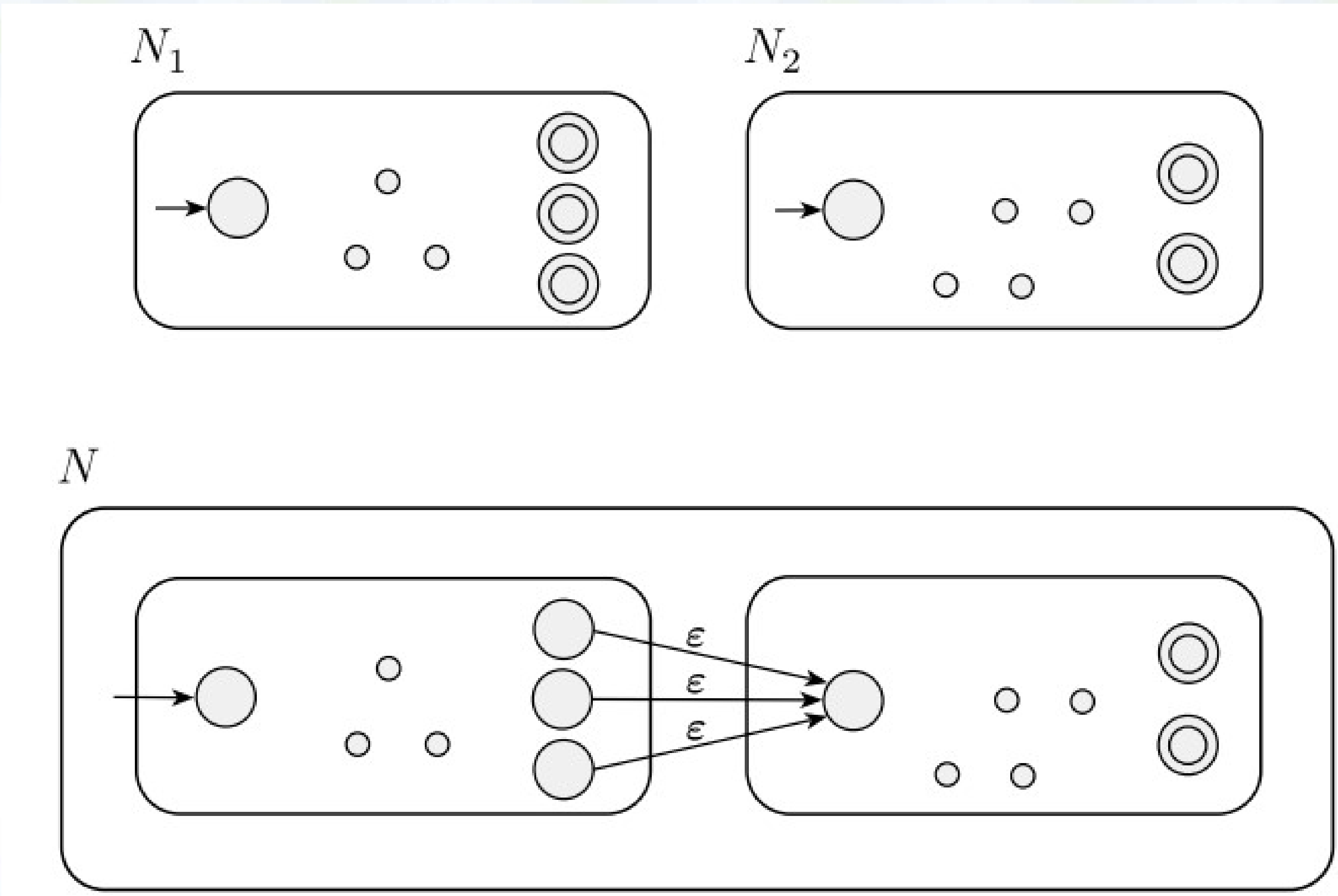
1. $Q = \{q_0\} \cup Q_1 \cup Q_2$.
2. The state q_0 is the start state of N.
3. The set of accept states $F = F_1 \cup F_2$.
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma \cup \varepsilon$,

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & \text{if } q \in Q1 \\ \delta_2(q, a) & \text{if } q \in Q2 \\ \{q_1, q_2\} & \text{if } q = q_0 \text{ and } a = \varepsilon \\ \emptyset & \text{if } q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$

Theorem

- The class of regular languages is closed under the concatenation operation
- PROOF IDEA Take two NFAs, N_1 and N_2 for L_1 and L_2, and combine them into a new NFA N
  - Assign N's start state to be the start state of N_1.
  - The accept states of N_1 have additional ε arrows to the start state of N_2
  - The accept states of N are the accept states of N_2
  - N accepts when the input can be split into two parts, the first accepted by N_1 and the second by N_2

Construction of N to recognize L_1 L_2



Formal description

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize L_1,

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize L_2.

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $L_1 L_2$.

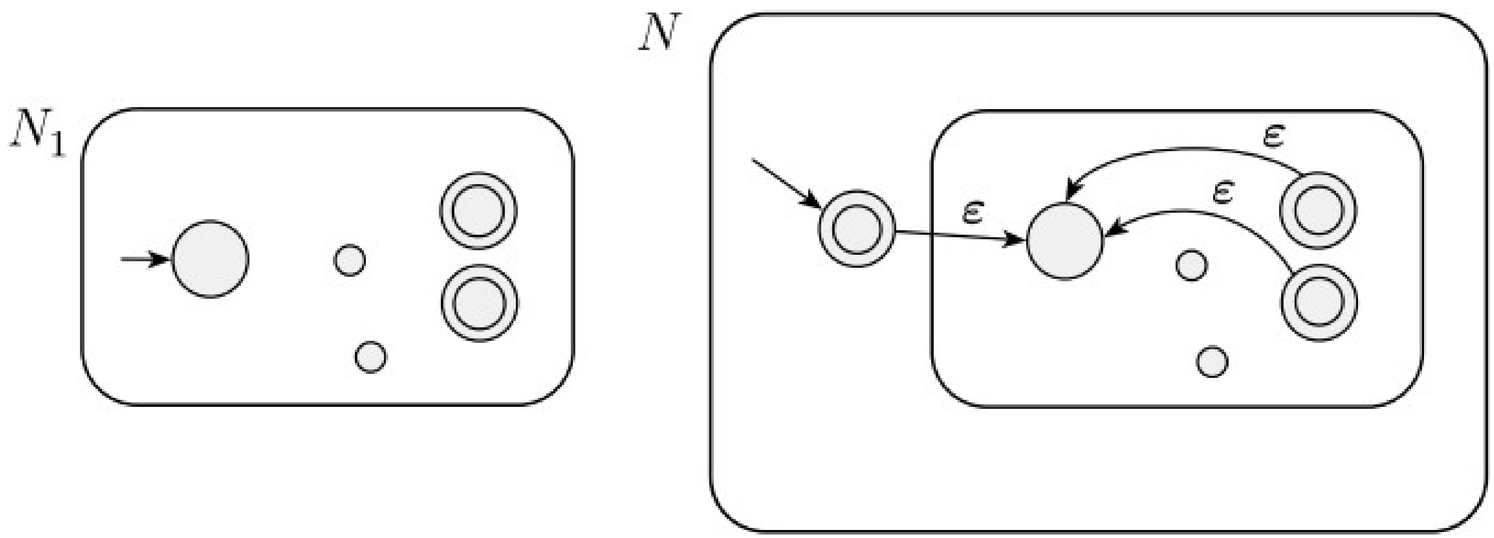
1. $Q = Q_1 \cup Q_2$.
2. The start state q_1 is the same as the start state of N_1.
3. The accept states F_2 are the same as the accept states of N_2.
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma \cup \varepsilon$,

$$\delta(q, a) = \begin{array}{ll} \delta_1(q, a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q, a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q, a) & q \in Q_2. \end{array}$$

Theorem

- The class of regular languages is closed under the star operation.
- Proof idea
  - Take an NFA N_1 for L_1 and modify it to recognize $L_1{}^*$
  - N like N_1 with additional ε arrows returning to the start state from the accept states.
  - Modify N so that it accepts ε

Construction of N to recognize L_1*



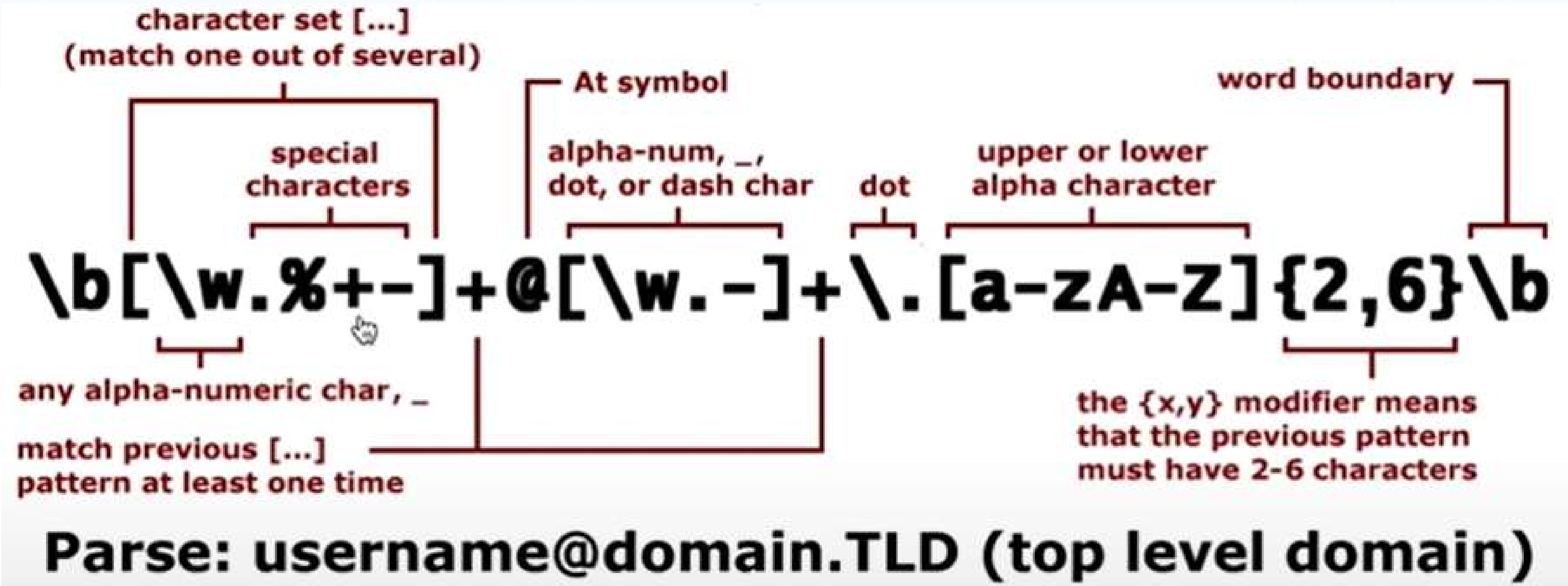
Formal description

- Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F1)$ recognize L_1.
  Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize L_1^* .
  1. $Q = \{q_0\} \cup Q_1$.
  2. The state q_0 is the new start state.
  3. $F = \{q_0\} \cup F_1$.

  $$\delta(q, a) = \begin{cases} \delta_1(q, a) \text{ if } q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a) \text{ if } q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q, a) \cup \{q_1\} \text{ if } q \in F_1 \text{ and } a = \varepsilon \\ \{q1\} \text{ if } q = q_0 \text{ and } a = \varepsilon \\ \emptyset \text{ if } q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$

Regular expression

- Have you ever seen a regular expression?



Regular expression

Files

index.js

index.js saved

```
const phoneNumbers = [
  '097.123.1234',
  '091-303-0001',
  '0123 123 324'
];

function sanitize(phoneNumbers) {
  return phoneNumbers.map(str => {
    return str.replace(/[. -]/g, '');
  });
}

sanitize(phoneNumbers);

/*
Expected:
[
  '0971231234',
  '0913030001',
  '0123123324'
]
*/
```

https://CaringGlumInterchangeability.nhim175.repl.run

```
node v10.15.2 linux/amd64
>
=> [ '0971231234', '0913030001', '0123123324' ]
>
```

Regular expression

- Similar to arithmetic expression, we can use the regular operations to build up expressions describing languages, which are called regular expressions.
- Example is: $(0 \cup 1)0^*$.
- Applications
  - Patterns for searching
  - Description of tokens for scanner generators
  - Pattern in programming languages like Python, in tools of UNIX like owk, grep

FORMAL DEFINITION OF A REGULAR EXPRESSION

Say that R is a regular expression if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,

Assume r_1 and r_2 are regular expressions denote languages R_1 and R_2

4. $(r_1 + r_2)$, is the regular expression denotes $R_1 \cup R_2$
5. $(r_1 r_2)$, is the regular expressions denotes $R_1 \circ R_2$
6. (r_1^*), is the regular expression denotes R_1^*

Precedence of regular operations

- Precedence
  - Stars
  - Concatenations
  - Unions
  - Example

    $$((0((0+1)^*))0) = 0(0+1)^*0$$
- $rr^* = r^*r = r^+$

Examples

- Assume that the alphabet Σ is{0,1}
  1. $0^*10^* = \{w|w$ contains a single 1}.
  2. $(0+1)^*1(0+1)^* =\{w|w$ has at least one 1}.
  3. $\Sigma^*001\Sigma^* =\{w|w$ contains the string 001 as a substring}.
  4. $1^*(0+1)^* = \{w|$ w begin with at least one 1}.

Equivalence of regular expressions and finite automata

- Theorem: A language is regular if and only if some regular expression describes it.
- To prove the theorem, let's divide it into 2 lemmas:
  - Lemma: If a language is described by a regular expression, then it is regular
  - Lemma: If a language is regular, then it is described by a regular expression

Conversion of Regular Expression to Finite Automata

- Given a regular expression R describing some language A.
- Convert R into an NFA recognizing A.

Conversion of Regular Expression to Finite Automata

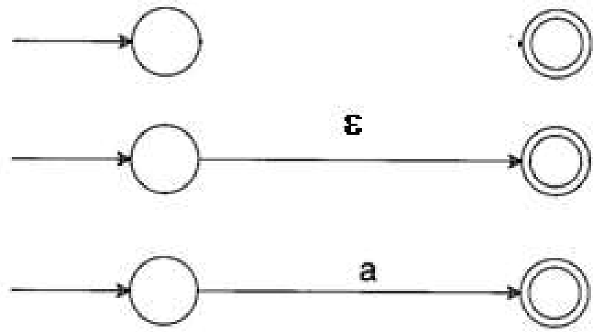
- We consider the six cases in the formal definition of regular expressions.

1. ε (empty string)
2. a (a is a symbol)
3. A+B (union)
4. AB (concatenation)
5. A* (star)

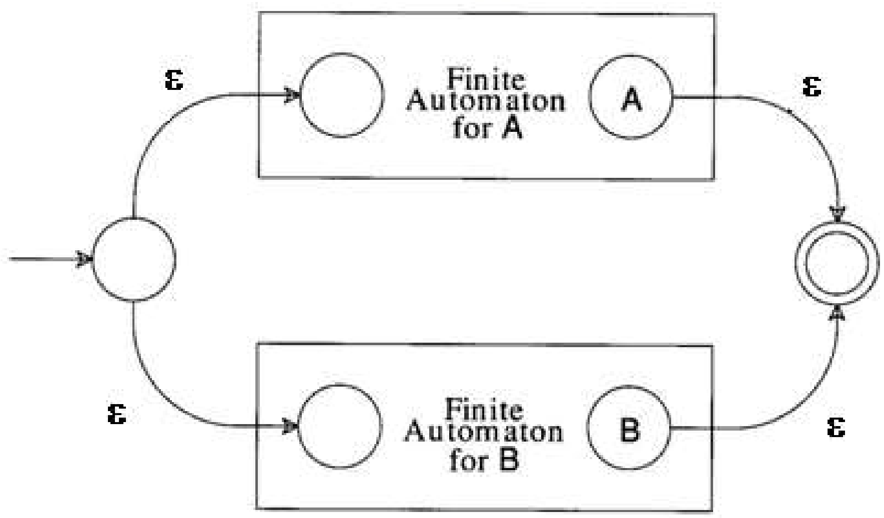
Conversion of Regular Expression to Finite Automata

- Without loss of generality, we show that language L defined by regular expression E is accepted by for some NF A with
  1. Exactly one accepting state
  2. No arcs into the initial state
  3. No arcs out of the accepting state
- The proof is by structural induction on R, following the recursive definition of regular expressions.

Conversion of Regular Expression to Finite Automata

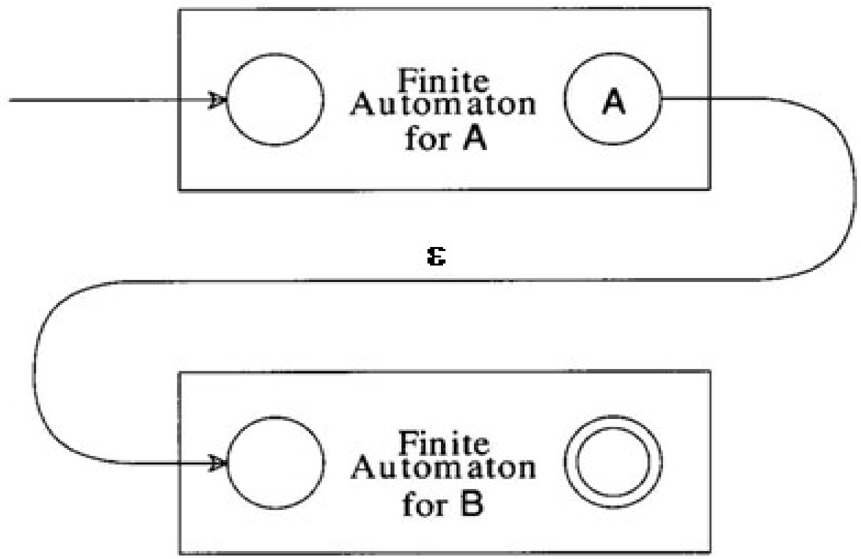


NFAs for the empty set, a and ε

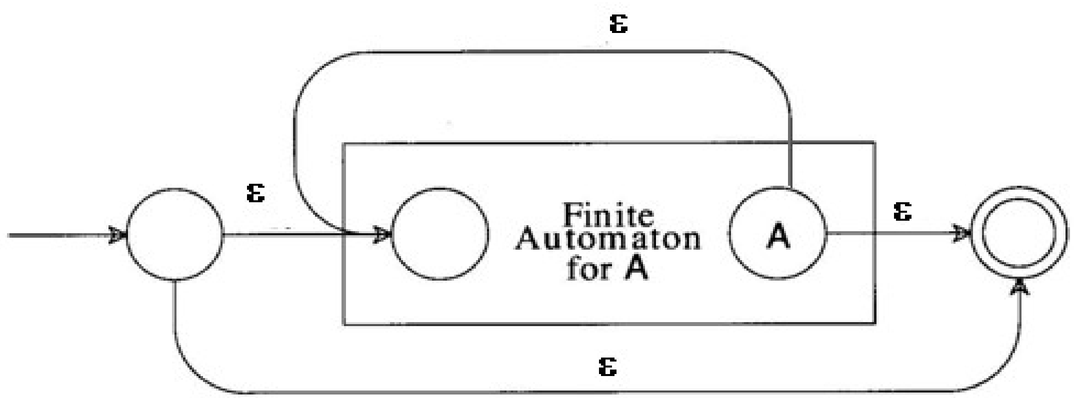


An NFA for A + B

Conversion of Regular Expression to Finite Automata

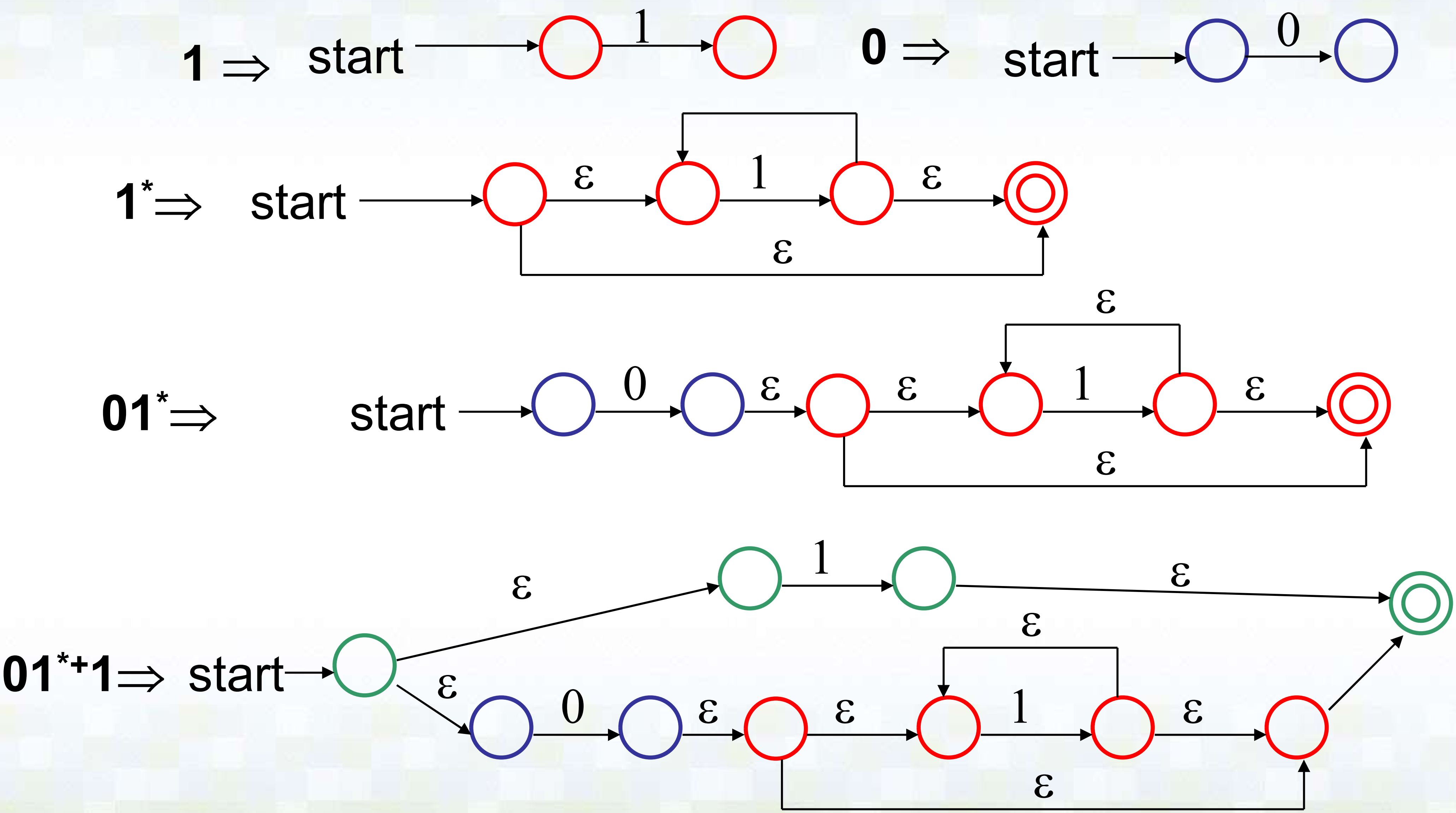


An NFA for A B

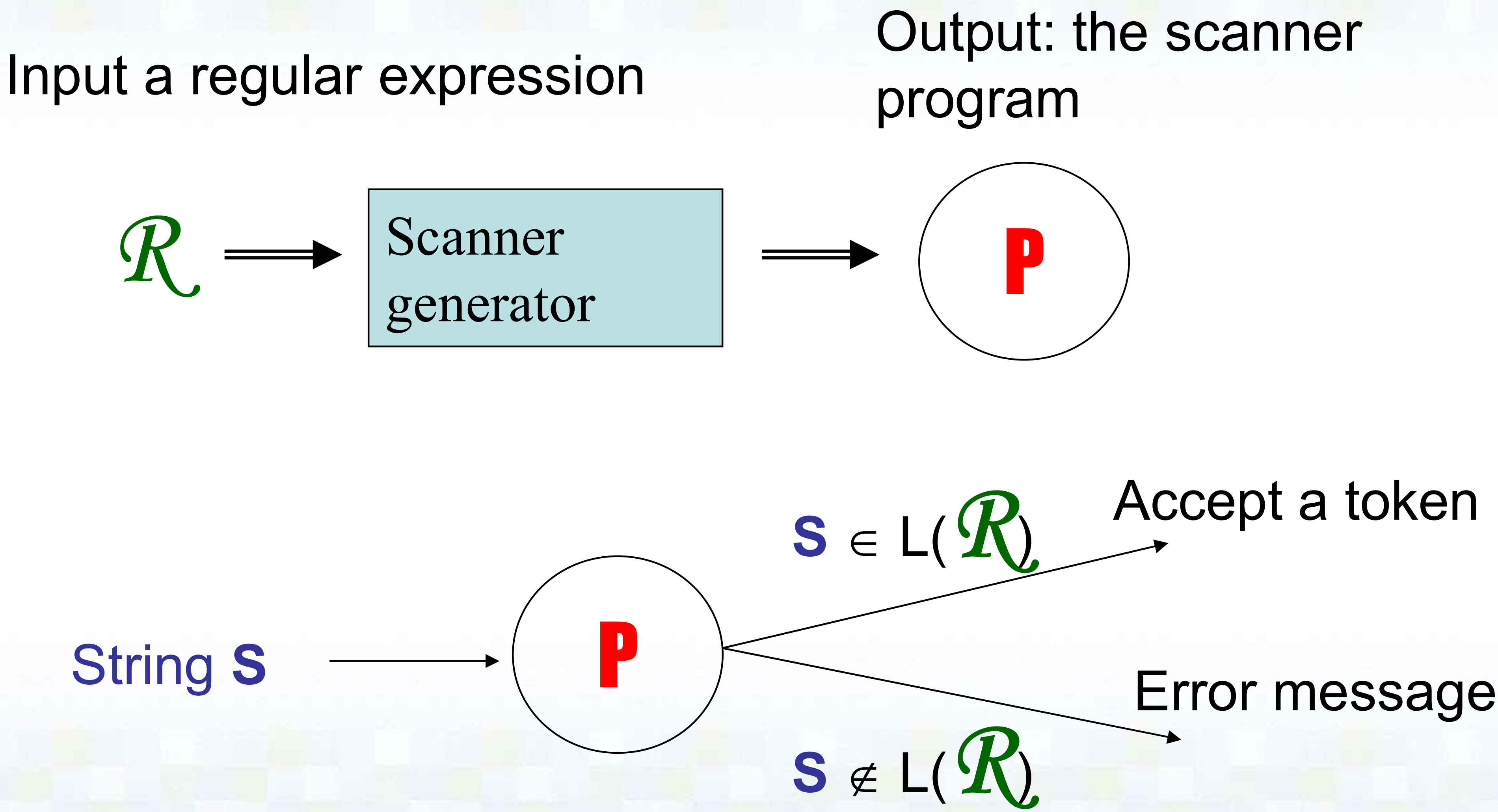


An NFA for A^*

Example: Building NFA for 01^*+1

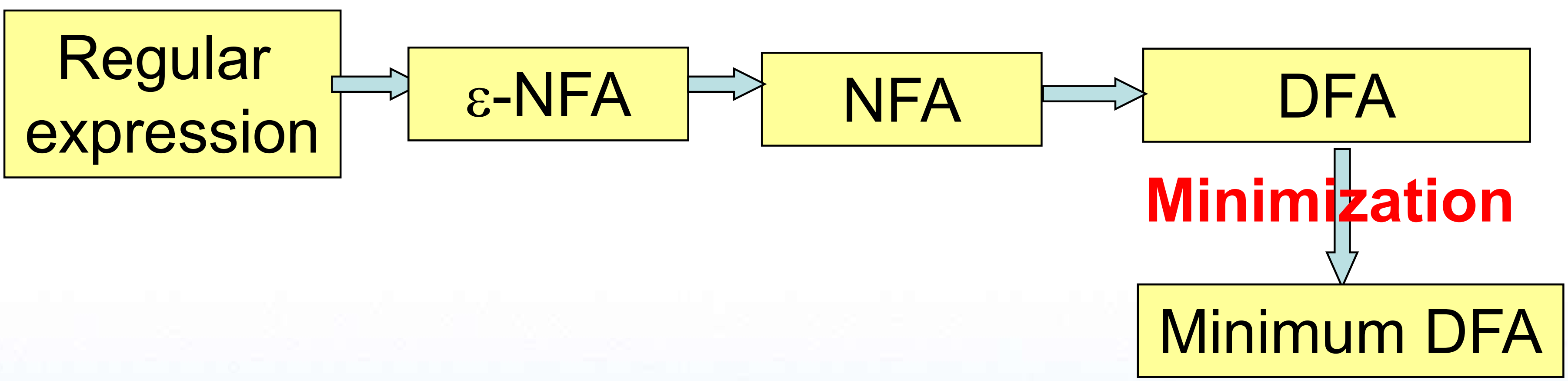


Model of a scanner generator



How a scanner program is built?

- The scanner program is built from an deterministic finite automaton
- We need a process to convert from a set of regular expression to a deterministic finite automaton



Conversion of Finite Automata to Regular Expression

- Using the concept of generalized nondeterministic finite automaton, GNFA in a special form:
  - The start state has transition arrows going to every other state but no arrow coming in from any other state.
  - There is only a single accept state, and it has arrows coming in from every other state but no arrow going to any other state.
  - The accept state is not the same as the start state.
  - Except for the start and accept states, one arrow goes from every state to every other state and also from each state to itself.

Method

- Convert a DFA into a GNFA in the special form
- Convert a GNFA into a regular expression
- Method details will be left for the presentation (pp 70-76, Sipser's book)