Compiler Construction
CS510

Lecture 1 Topics

- The Scanning Process
  - Tokens ,Lexemes and Patterns
  - Scanner Phases
- Regular Expressions
  - Languages
  - RE Definition
  - RE Examples
  - Extensions of RE Operations
- Finite Automata

Scanning Process (Continued)

First step for humans to understand written text is recognizing Words.

This is a sentence

Ist his ase nte nce

*** Alex Hiken Stanford2014**

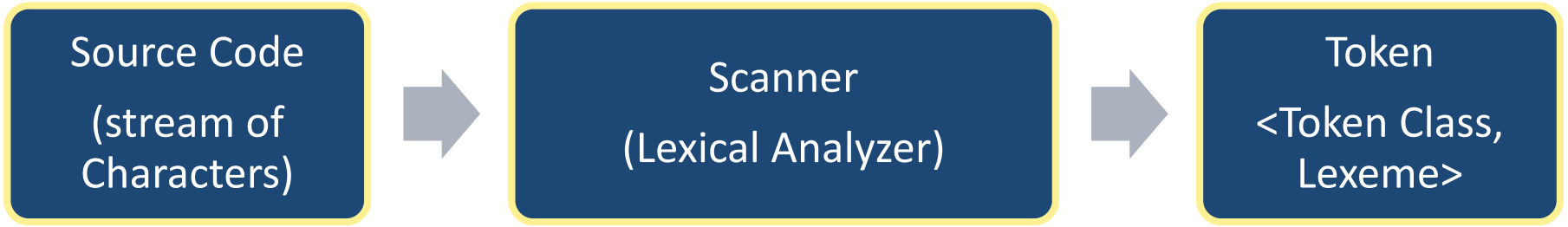
Scanning Process (Continued)

It is also helpful to identify a syntactic category

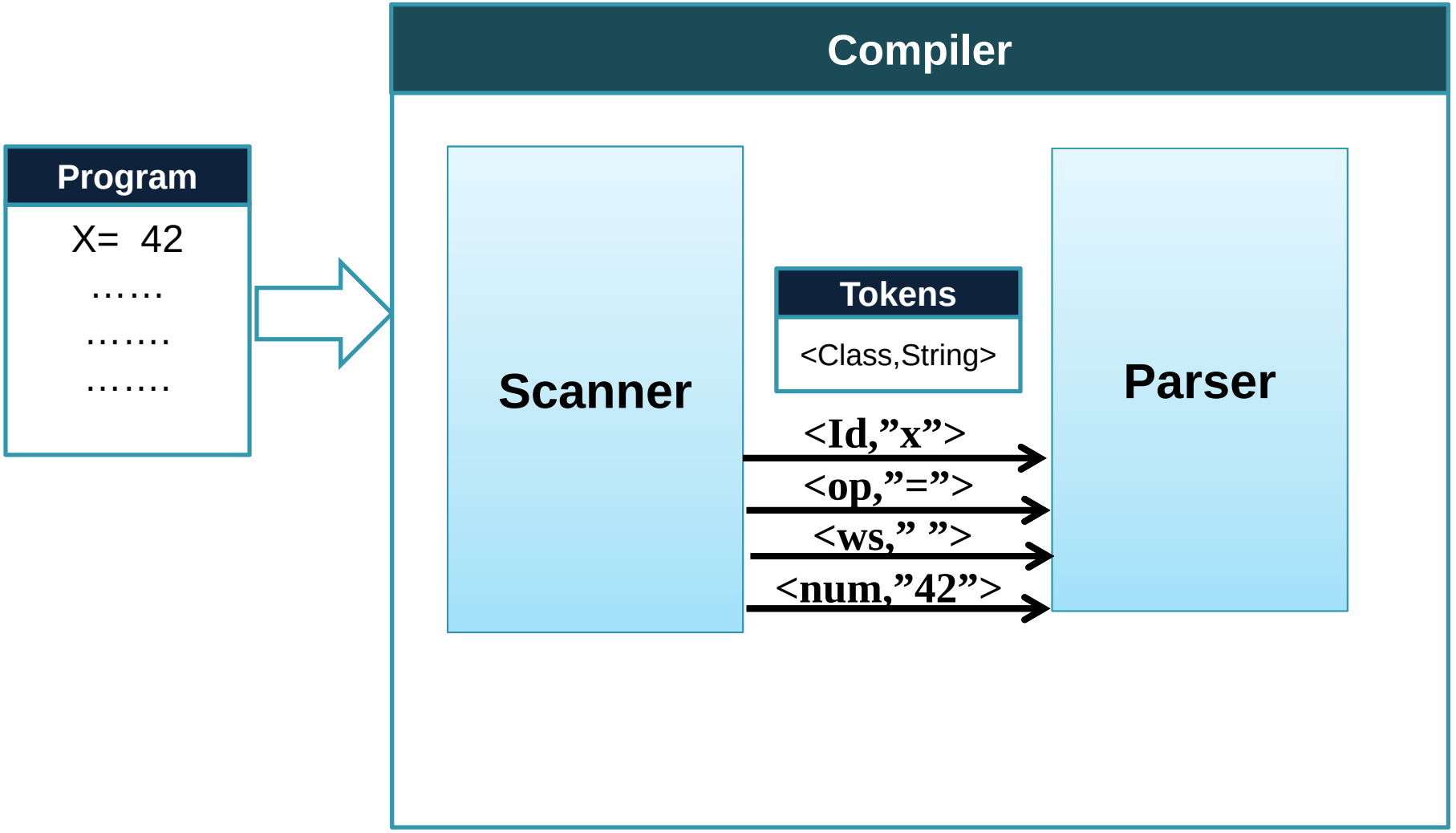
- In English:
  - Noun, verb, adjective, …
- In a programming language:
  - Identifier, Integer, Keyword, White-space, …

The Scanning Process (Continued)

- Input: source code as a stream of characters
- Output: *Tokens*



Scanning Process (Continued)



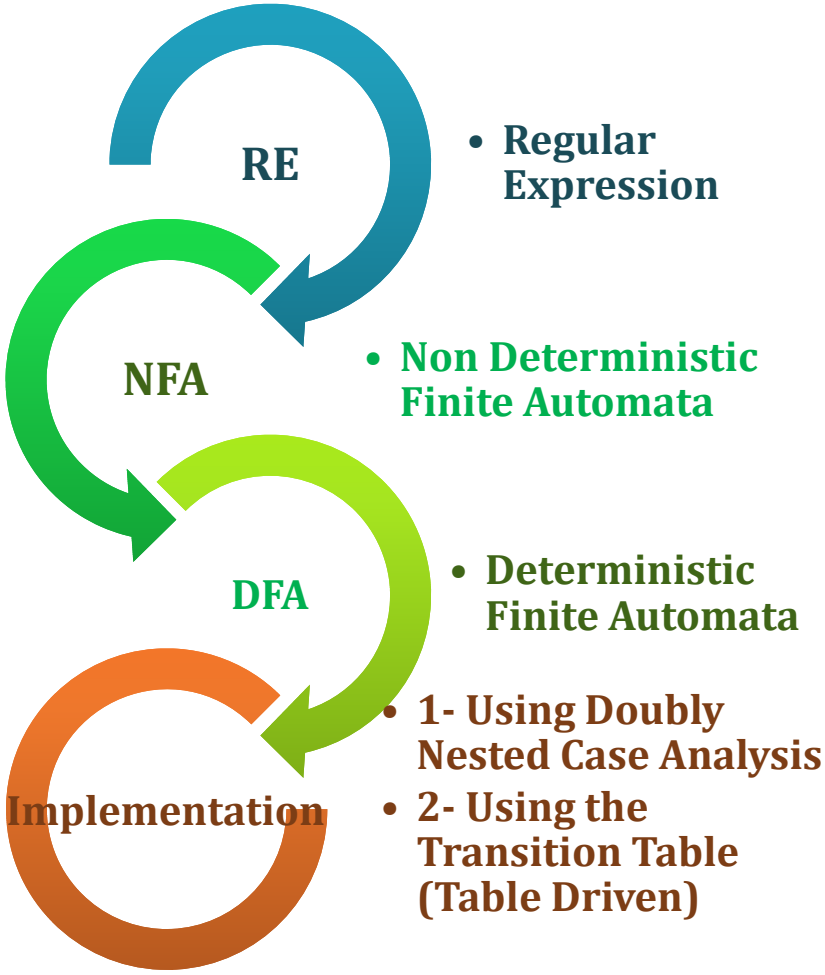
Tokens ,Lexemes and Patterns

- *Patterns* identify set of lexemes accepted by the token.

Tokens Class	Lexeme	Tokens	Pattern
id	Avg,balance, v12	<id,Avg>	Letter followed by letters and digits
kw Key words	If,while,for…	<kw,if>	“else” or “if” or “begin” or …
relop	<, <=,=,<>,>,>=	<relop,=>	< ,<= ,= ,<> ,> ,>=
num	31 , 28	<num,31>	a non-empty string of digits
op	+ , *, - ,/	<op,+>	Any arithmetic operator + or * or – or /
ws white spaces	\t ,\n	<ws,\t>	a non-empty sequence of spaces, newlines, and tabs

Scanner Phases

- The scanning process is a pattern matching process an it can be divided in two main phases:
  - Pattern specification using regular expressions
  - Pattern recognition using finite automata for recognizing patterns



Regular Expressions (RE)

Languages

- alphabet Σ :Finite, nonempty set of symbols.
  - A machine language $\Sigma = \{0,1\}$.
  - set off all English lower case character {a,b,c,d,….,z}.
- string :is a finite sequence of symbols over alphabet.
  - e.g. 0011001 is a string from machine language.
  - Empty string ε (not white space such as spaces or tabs).
- Language : set of strings over Σ
- Powers of an Alphabet Σ^k : the set of strings of length k from alphabet Σ. e.g.
  - $\Sigma = \{0,1\}$.
  - $\Sigma^1 = \{0,1\}$.
  - $\Sigma^2 = \{00,01,10,11\}$.
  - Σ^* = set of all strings over alphabet Σ.
  - $\Sigma^0 = \{\varepsilon\}$ empty string.

Operations on Languages

- Union

  $L \mid M = \{s \mid s \in L \text{ or } s \in M\}$

  e.g. (0|1) = 0 or 1.

- Concatenation

  $LM = \{xy \mid x \in L \text{ and } y \in M\}$

  e.g. ab = ab

  “hello” ”there” = “hellothere”

- Kleene closure

  $L^* = \cup_{i=0,\ldots,\infty} L^i$

  e.g. $1^* = \{\varepsilon,1,11,111,1111,\ldots\ldots\}$

Regular expressions

- A *regular expression* is an expression that matches sets of strings.
- Regular expressions is used for pattern specification.
- the "*language*" of the regular expressions is called Regular Languages.
- Parentheses for grouping (to change precedence, just as in arithmetic)
- Precedence of Operators **: () ,*,** concatenation, |

Regular Expressions (Continued)

Which of the following matches RE

a(ab)*a

1) abababa
2) aaba
3) aabbaa
4) aba
5) aabababa

Solution : 2,5

Operations on Languages

Let Σ = {a, b}.

1. RE =(a|b) .

   language = {a or b}.

2. RE=(a|b)(a|b)

   language ={aa, ab, ba, bb}.

3.RE=a*

   language ={ε, a, aa, aaa, . . . }. all strings of zero or more a's

4. RE= (a|b)*

   language= {ε, a, b, aa, ab, ba, bb, aaa, . . .}. zero or more instances of a or b

5. RE= a|a*b

   language = {a, b, ab, aab,aaab,. . .).all strings consisting of zero or more a's and ending in b.

Regular Expressions (Continued)

- Examples of Regular Expressions
  - letter → A |B| ….|Z | a | b |...|z
  - digit → 1 |2| 3|….|9
  - digits → digit digit*
  - id → letter (letter | digit)*
  - strings beginning with a, followed by any number of b's and c's. a(b|c)*
  - the string ab, or the strings ε, c, cc, …. ab|c*

Regular Expressions (Continued)

- Common Extensions of R.E. operations:
  - `+` : one or more repetitions of
    (*r*`+` is equivalent to *rr*`*`)
  - `[]` : range of characters
    (`[abcd]` = `[a-d]` = `a|b|c|d`)
  - `.` : any character
    (`.*b.*`: strings that contain at least one b)
  - `^` : negate a set
    (`[^abc]` = any character except `a`, `b`, or `c`)
  - `?` : optional sub-expression
    (`((+|-)?[0-9]` = `[0-9] | +[0-9] | -[0-9]`)
  - `\` : "escape" an operator or meta-symbol e.g. `\* \+ \? \| \.`

Regular Expressions (Continued)

- *letter* $\rightarrow$ [A-Za-z]
- *digit* $\rightarrow$ [0-9]
- *digits* $\rightarrow$*digit*+
- *id* $\rightarrow$ *letter* (*letter* | *digit*)*

Regular Expressions (Continued)

Which of the following matches RE

a[bc]+

1) abc
2) abbbbbbbb
3) azc
4) abcbcbcbbc
5) ac
6) asccbbbbcbcccc

Solution : 1

Finite Automata (FA)

Finite Automata (FA)

- Finite automata is used to recognize strings generated by regular expressions.
- Singular form (automaton).
- It also called Finite-State Automaton (FSA).
- Finite Automata used for modeling digital circuits, verifying communication protocols, and in the scanner of compiler.
- FA has two types:
  - Nondeterministic Finite Automata (NFA).
  - Deterministic Finite Automata (DFA).

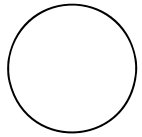
Finite Automata

An DFA is a 5-tuple $(S, \Sigma, \delta, s_0, F)$ where:

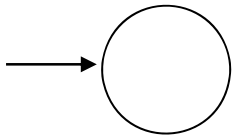
- S is a finite set of *states.*
- Σ a finite set of symbols, the input *alphabet.*
- δ transition function is a *mapping* from $S \times \Sigma$ to a set of states
- $s_0 \in S$ is the *start state*
- $F \subseteq S$ is the set of *accepting* (or *final) states*

Finite Automata

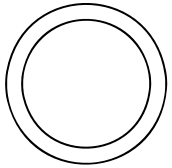
- A state



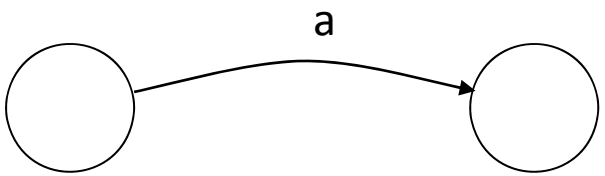
- The start state



- An accepting state



- A transition



Finite Automata

Transition $s_1 \xrightarrow{a} s_2$ is read

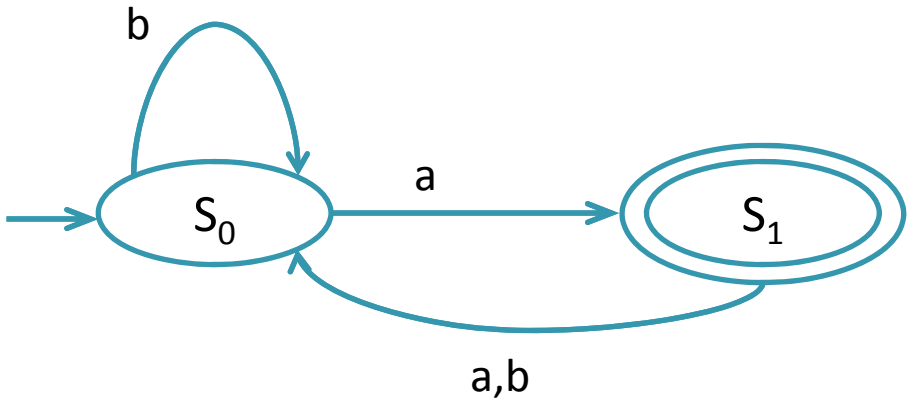
In state s_1 on input "a" go to state s_2

- If end of input
  - If in accepting state => *accept*
  - Otherwise => *reject*
- If no transition possible (got stuck) => reject

DFA Example(1)

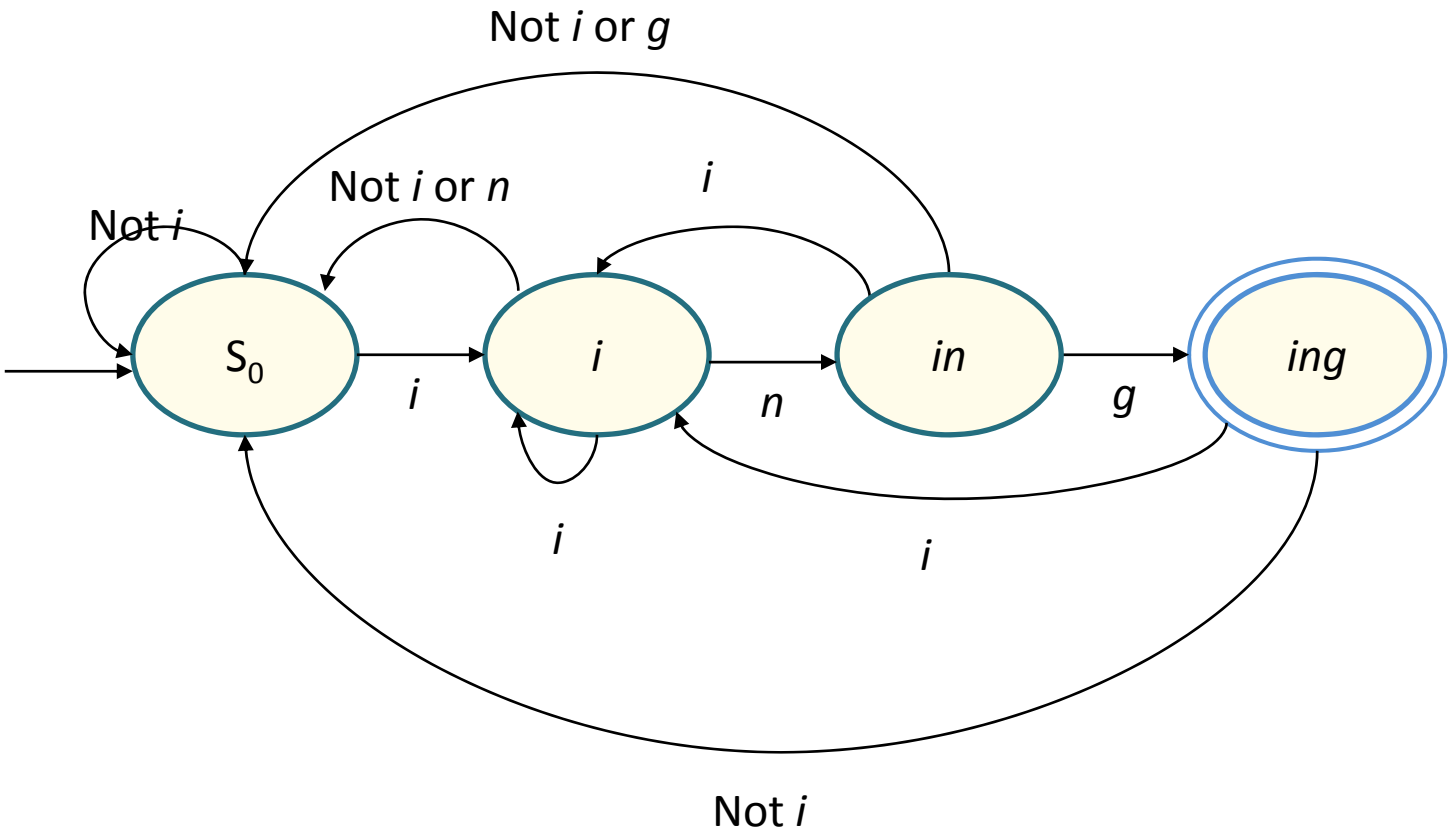
- Transition graph to represent DFA

Transition graph (accept 1 a)



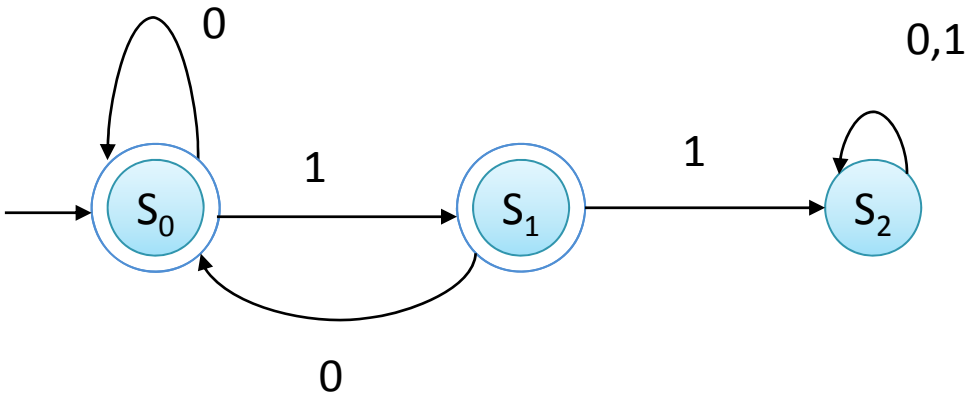
DFA Example (2)

Accepting Strings Ending in “ing”.



DFA Example (3)

Rejecting Strings containing 11

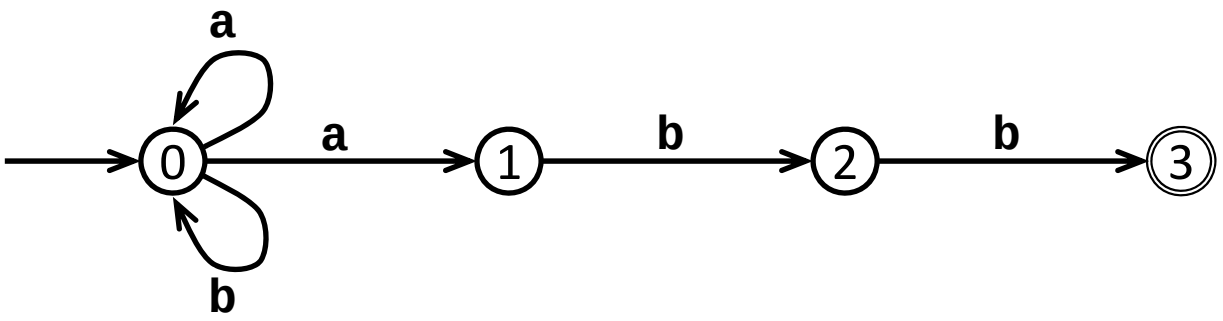


Finite Automata

```
While (! End of input String)
   {
        Read input character by character
        If  (no transition) break;
        Go to the next state
         }
    If (accept state)
        string accepted
    Else
        string rejected
```

Transition Graph

- Use a transition diagram to describe a FA
  - states are nodes, transitions are directed, labeled edges, some states are marked as *final*, one state is marked as *starting*



$S = \{0,1,2,3\}$
$\Sigma = \{\mathbf{a},\mathbf{b}\}$
$s_0 = 0$
$F = \{3\}$

Transition Table

- The mapping δ of an FA can be represented in a *transition table*

$\delta(0,\mathbf{a}) = \{0,1\}$
$\delta(0,\mathbf{b}) = \{0\}$
$\delta(1,\mathbf{b}) = \{2\}$
$\delta(2,\mathbf{b}) = \{3\}$

State	*Input* **a**	*Input* **b**
0	{0, 1}	{0}
1		{2}
2		{3}
*3		

NFA vs DFA

- multiple transitions on a single input character
- ε-move