中山大學

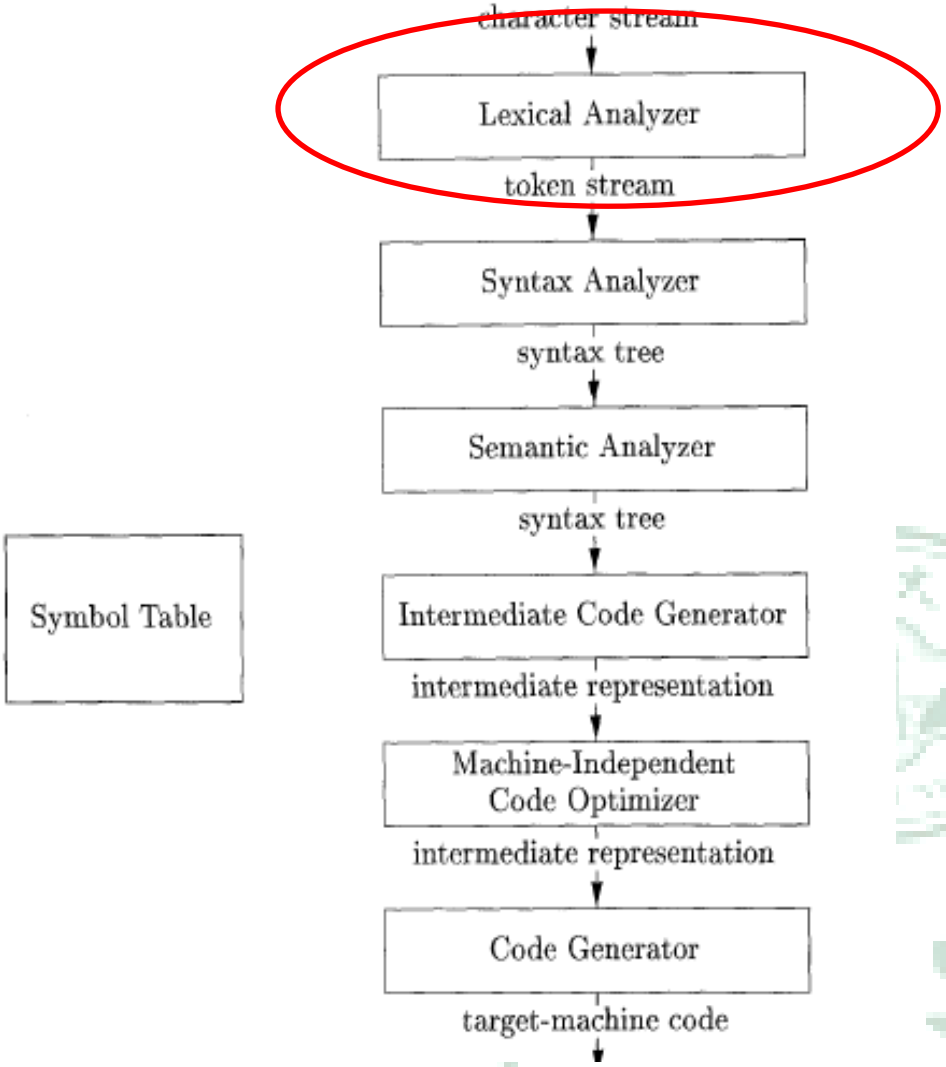
Principles of Compiler Construction

Lecture 2 Lexical Analysis (I)

中山大學

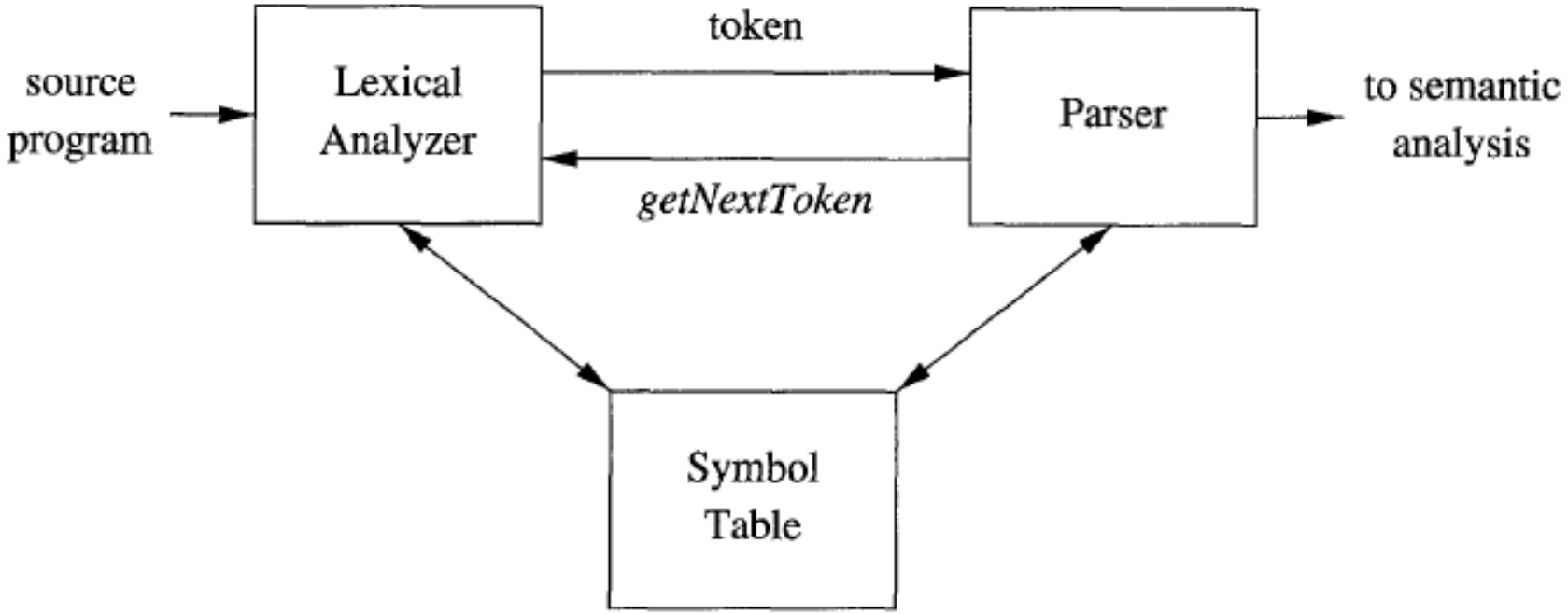
中山大學

回顾：编译器的结构



中山大學

Lexical Analyzer vs. Syntax Analyzer (Parser)



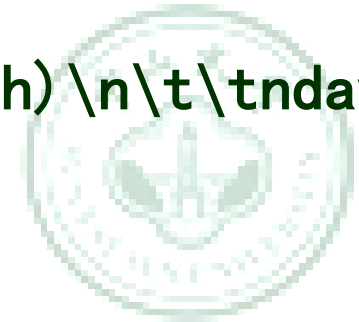
词法分析的任务

源程序在经词法分析之前是字符的序列

例如：

```
if (month == March)
 nday = 31;
```

是字符序列：\tif (month == March)\n\t\tnday = 31;



中山大學

词法分析的任务

词法分析器将其转化为token的序列

- token是指带有附加信息的字符串

  例如<“month”, id>

  单纯的字符串我们称为lexeme

- 在自然语言中：名词，动词，形容词……
- 程序语言中：标识符，关键字，常量，运算符，分界符
- 类别属性是包括在token中的基本信息

- 语法分析器每次从词法分析器获得一个token，对不同类别token的处理截然不同

中山大學

关键问题

- 如何识别出一个token?
  - 如何识别出一个lexeme?
  - 如何知道lexeme的类别属性?

中山大學

类别定义

- 关键字或保留字：预先指定的单词，如if, for, else等
- 标识符：
  - 字母开头的字母或数字序列
  - 非保留字
- 数字：
  - 正整数：首字符非0的数字序列
  - 负数：’-’和正整数的连接
  - ……

中山大學

类别定义

- 自然语言定义方式的缺陷:
  - 繁琐
  - 不精确
  - 难以被计算机处理
- 怎么办?
  - 借助形式化的语言!

3.3.1 Strings and Languages

字母表：An alphabet is any finite set of symbols.

语言：字母表∑上的语言，是由∑中字符组成的字符串的集合

例如：

∑={0，1}

则{001，100100}，{}，{1，11，111，1111，…}都是定义在字母表∑的语言

语言（*Language*）

字母表：字符的集合

语言：字母表∑上的语言，是由∑中字符组成的字符串的集合

例如：

∑={0，1}

则{001，100100}，{}，{1，11，111，1111，…}都是定义在字母表∑的语言

程序设计语言

程序：　告诉计算机“做什么”和“怎么做”

程序设计：　编写程序的过程

程序设计语言：　编写程序使用的语言

语句：　定义在一个字符集合上的按一定的语法规则连接的有意义的字符串

语言：　由语句组成的全体

作用：　交换信息的工具

自然语言与形式语言

- 自然语言：

人与人交换信息的工具。不精确，有二义性

汉语、英语

手提包

- 形式语言：

数学语言，精确语言，无二义性。

计算机语言为人与计算机交换信息的工具

计算机语言

文法、自动机、表达式

中山大學

语言的运算

语言是一种集合，所以集合的运算也适用于语言

OPERATION	DEFINITION AND NOTATION
Union of L and M	$L \cup M = \{s \mid s \text{ is in } L \text{ or } s \text{ is in } M\}$
Concatenation of L and M	$LM = \{st \mid s \text{ is in } L \text{ and } t \text{ is in } M\}$
Kleene closure of L	$L^* = \cup_{i=0}^{\infty} L^i$
Positive closure of L	$L^+ = \cup_{i=1}^{\infty} L^i$

中山大學

语言运算的例子

L={A,B,...,Z,a,b,...,z}, D={0,1,...,9}

1. $L \cup D$ is the set of letters and digits — strictly speaking the language with 62 strings of length one, each of which strings is either one letter or one digit.

2. LD is the set of 520 strings of length two, each consisting of one letter followed by one digit.

3. L^4 is the set of all 4-letter strings.

4. L^* is the set of all strings of letters, including ϵ, the empty string.

5. $L(L \cup D)^*$ is the set of all strings of letters and digits beginning with a letter.

6. D^+ is the set of all strings of one or more digits.

正则语言（*Regular Language*）

3.3.3 Regular Expressions

正则语言的优点：

- 容易理解
- 能高效实现
- 具有坚实的理论基础

中山大學

正则表达式及其描述的语言

递归定义：

- **Basis:**

  ε is a regular expr; L(**ε**) = {ε}.

  a∈Σ is a regular expr; L(**a**) = {a}.

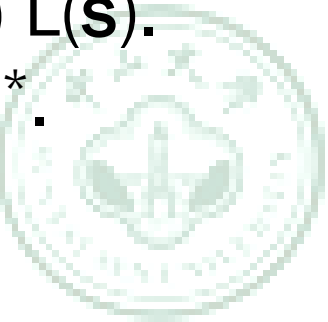
- **Induction: if r and s are regular exprs,**

  r | s is a regular expr; L(**r | s**) = L(**r**) ⋃ L(**s**).

  r s is a regular expr; L(**r s**) = L(**r**) L(**s**).

  r* is a regular expr; L(**r***) = (L(**r**))*.

  (r) is a regular expr; L(**(r)**) = L(**r**).



一些规定

为了去除不必要的符号，我们规定:

- 一元运算符*具有最高的优先级
- 连接运算符具有次高优先级
- |运算符的优先级最低
- 上述运算符都是左结合的（left associative）

中山大學

例子

令字母表为{a, b}

1. The regular expression **a**|**b** denotes the language $\{a, b\}$.

2. (**a**|**b**)(**a**|**b**) denotes $\{aa, ab, ba, bb\}$, the language of all strings of length two over the alphabet Σ. Another regular expression for the same language is **aa**|**ab**|**ba**|**bb**.

3. **a*** denotes the language consisting of all strings of zero or more a's, that is, $\{\epsilon, a, aa, aaa, \ldots\}$.

4. (**a**|**b**)* denotes the set of all strings consisting of zero or more instances of a or b, that is, all strings of a's and b's: $\{\epsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}$. Another regular expression for the same language is (**a*****b***)*.

5. **a**|**a*****b** denotes the language $\{a, b, ab, aab, aaab, \ldots\}$, that is, the string a and all strings consisting of zero or more a's and ending in b.

中山大學

正则语言

- **A language that can be defined by a regular expression is called a *regular set*.**
- **If two regular expressions r and s denote the same regular set, we say they are *equivalent* and write *r* = *s*.**
  - 例如：(a|b)=(b|a)

中山大學

一些正則語言的等价规则

LAW	DESCRIPTION
$r\|s = s\|r$	$\|$ is commutative
$r\|(s\|t) = (r\|s)\|t$	$\|$ is associative
$r(st) = (rs)t$	Concatenation is associative
$r(s\|t) = rs\|rt;\ (s\|t)r = sr\|tr$	Concatenation distributes over $\|$
$\epsilon r = r\epsilon = r$	ϵ is the identity for concatenation
$r^* = (r\|\epsilon)^*$	ϵ is guaranteed in a closure
$r^{**} = r^*$	* is idempotent

中山大學

例子：

C语言的标识符：

(a | b | ... | z | A | B | ... | Z|_) ((a | b | ... | z | A | B | ... | Z|_) | (0 | 1 | ... | 9))*

还是太繁琐了，有没更简洁的表示?

中山大學

例子：

C语言的标识符：

(a | b | ... | z | A | B | ... | Z|_) ((a | b | ... | z | A | B | ... | Z|_)
| (0 | 1 | ... | 9))*

还是太繁琐了，有没更简洁的表示?

letter → A | B | · · · | Z | a | b | · · · | z

digit → 0 | 1 | · · · | 9

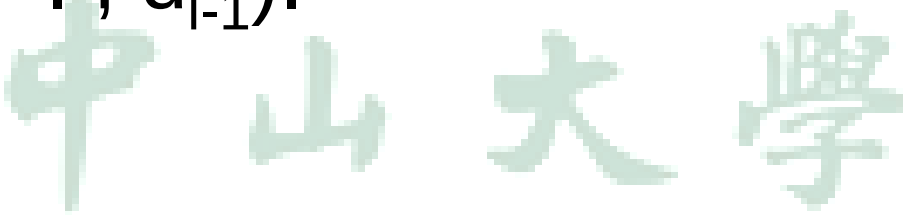
id → letter (letter | digit)*

正则定义（ *Regular Definition* ）

$$d_1 \rightarrow r_1$$
$$d_2 \rightarrow r_2$$
$$\ldots$$
$$d_n \rightarrow r_n$$

where:

1. Each d_i is a new symbol, not in C and not the same as any other of the d'_s, and
2. Each ri is a regular expression over the alphabet C U {d_1, d_2 , .. . , d_{i-1}).



中山大學

例子

例1：C语言的标识符

$$
\begin{aligned}
letter\_ &\rightarrow \texttt{A} \mid \texttt{B} \mid \cdots \mid \texttt{Z} \mid \texttt{a} \mid \texttt{b} \mid \cdots \mid \texttt{z} \mid \texttt{\_} \\
digit &\rightarrow \texttt{0} \mid \texttt{1} \mid \cdots \mid \texttt{9} \\
id &\rightarrow letter\_ \; (\; letter\_ \mid digit \;)^*
\end{aligned}
$$

例子

例2：无符号浮点数

digit → 0 | 1 | … |9
digits → digit digit*
optionalFraction → . digits | ε
optionalExponent → (E (+ | - | ε) digits) | ε
number → digits optionalFraction optionalExponent

正則表达式的扩展

- $r^+=rr^*$
- r?=r|ϵ
- [abc]=a|b|c,
- [a-z]=a|b|...|z

中山大學

例子

C语言标识符:

$$
\begin{aligned}
letter\_ &\rightarrow [\texttt{A-Za-z\_}] \\
digit &\rightarrow [\texttt{0-9}] \\
id &\rightarrow letter\_\ (\ letter \mid digit\ )^*
\end{aligned}
$$

无符号浮点数:

$$
\begin{aligned}
digit &\rightarrow [\texttt{0-9}] \\
digits &\rightarrow digit^+ \\
number &\rightarrow digits\ (\ .\ digits)?\ (\ \texttt{E}\ [\texttt{+-}]?\ digits\ )?
\end{aligned}
$$

中山大學

练习

下列正则表达式描述什么语言?

- a(a|b)*a
- (a|b)*a(a|b)(a|b)
- a*ba*ba*ba*
- ((ε|a)b*)*

中山大學

练习

用正则表达式描述下列语言:

- **所有由按词典递增序排列的小写字母组成的字符串**

  例如：add, low都符合要求，而zzg则不符合

  a*b*c*d*e*f*g*h*i*....

- **不以ab开头的所有只含有字母a和b的字符串.**

正则表达式的其它应用

- 文本搜索和模糊匹配
- 合法性检查
- 文本的自动更正和编辑
- 信息提取
- ……

Chomsky hierarchy

Grammar	Languages	Automaton	Production rules (constraints)
Type-0	Recursively enumerable	Turing machine	$\alpha \rightarrow \beta$ (no restrictions)
Type-1	Context-sensitive	Linear-bounded non-deterministic Turing machine	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Type-2	Context-free	Non-deterministic pushdown automaton	$A \rightarrow \gamma$
Type-3	Regular	Finite state automaton	$A \rightarrow a$ and $A \rightarrow aB$

中山大學

A Big QUESTION

如何让计算机识别用正则表达式描述的语言?

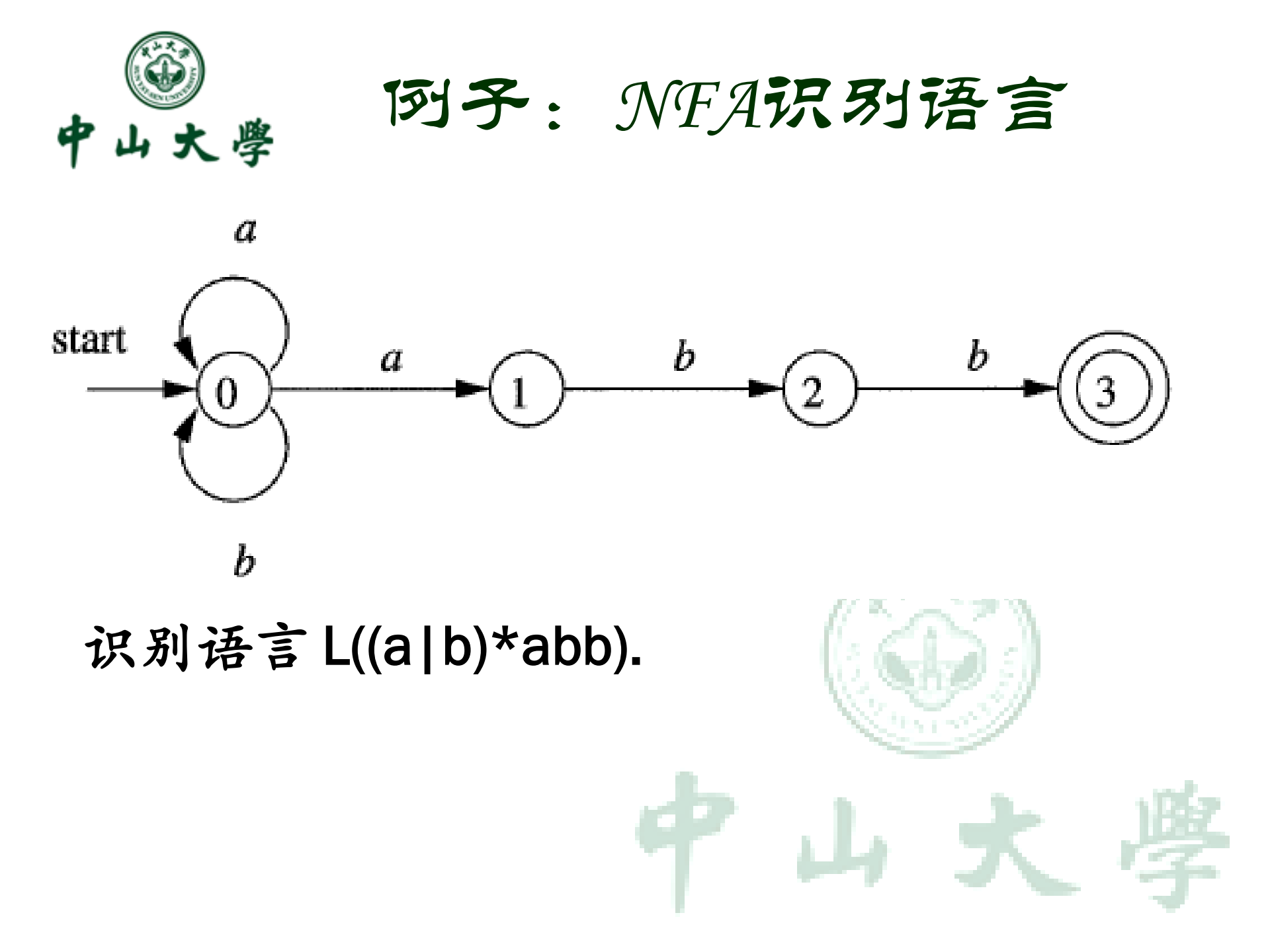
中山大學

非确定有限自动机（*NFA*）

A *nondeterministic finite automaton* (NFA) consists of:

1. A finite set of states S.
2. A set of input symbols Σ, the *input alphabet*. We assume that ϵ, which stands for the empty string, is never a member of Σ.
3. A *transition function* that gives, for each state, and for each symbol in $\Sigma \cup \{\epsilon\}$ a set of *next states*.
4. A state s_0 from S that is distinguished as the *start state* (or *initial state*).
5. A set of states F, a subset of S, that is distinguished as the *accepting states* (or *final states*).

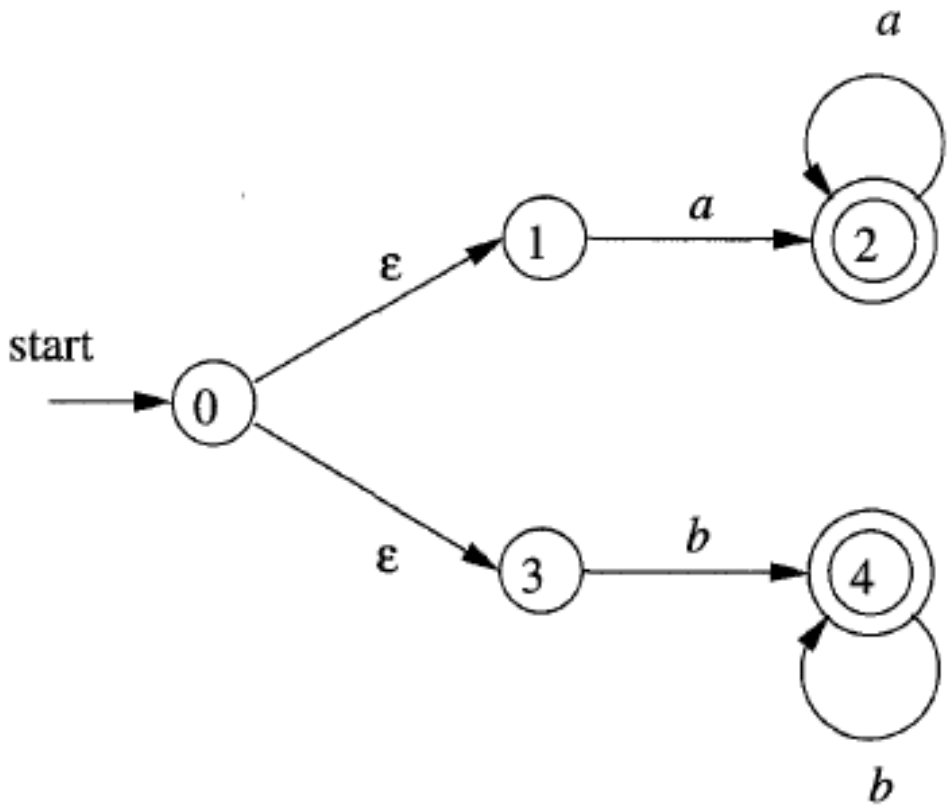
The *language defined (or accepted) by an NFA is the set of strings labeling* some path from the start to an accepting state.



例子：NFA识别语言

识别语言 L((a|b)*abb).

例子：带ϵ的NFA



识别语言L(aa*|bb*).

中山大學

确定有限自动机（DFA）

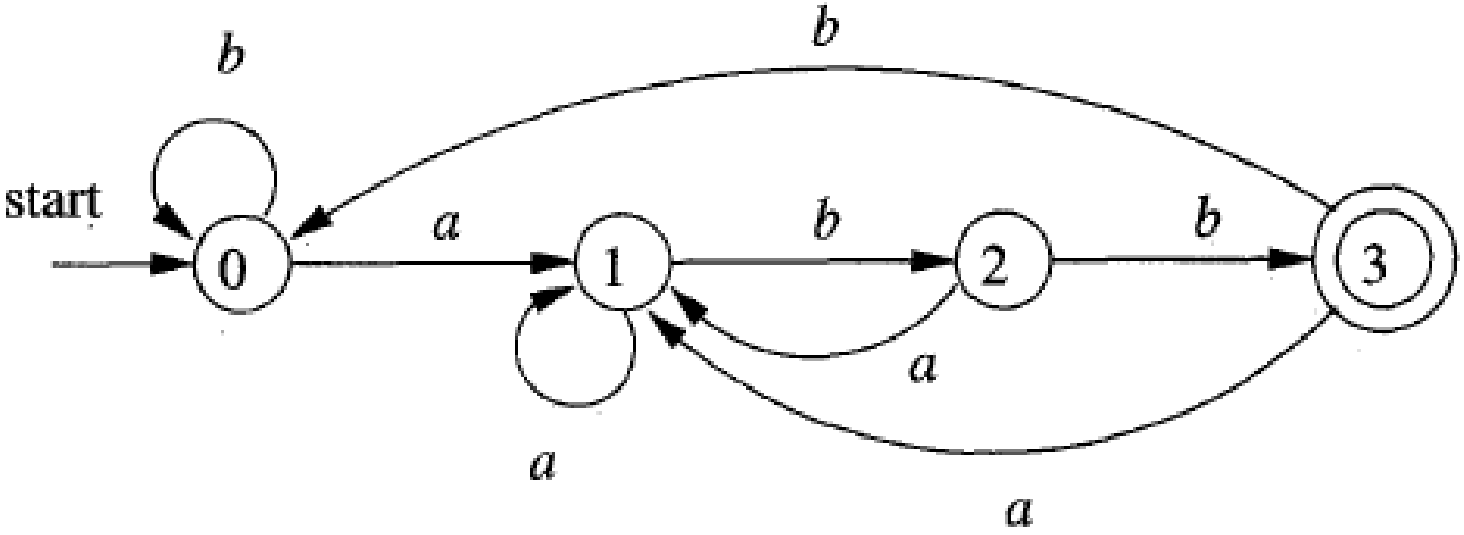
A *deterministic finite automaton (DFA) is a special case of an NFA where:*

1. There are no moves on input ϵ, and
2. For each state s and input symbol a, there is exactly one edge out of s labeled a.

中山大學

例子

识别(a|b)*abb的DFA



中山大學

基于DFA的识别算法

```
s = s0;
c = nextChar();
while ( c != eof ) {
        s = move(s, c);
        c = nextChar();
}
if ( s is in F ) return "yes";
else return "no";
```

See you next time!