



中山大學

Principles of Compiler Construction

Lecture 4 Lexical Analysis (III)

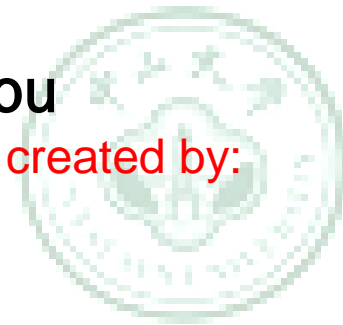
Lecturer: Chang Huiyou

Note that most of these slides were created by:

Prof. Wen-jun LI (School of Software)

Dr. Zhong-mei SHU (Department of Computer Science)

Dr. Han LIN (Department of Computer Science)



中山大學



中山大學

Regular Expression \rightarrow DFA (Directly)

Construct a DFA directly from a regular expression, **without constructing an intermediate NFA**

The resulting DFA may have **fewer states than the DFA constructed via an NFA**

Commonly used

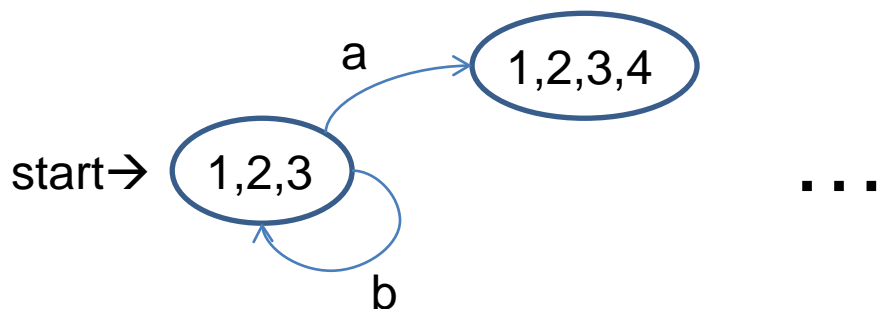



中山大學

用位置的集合来表示状态

- 例如 $(a|b)^*abb$

位置: 1 2 3 4 5





Syntax Tree

用'#'表示正则表达式的结束

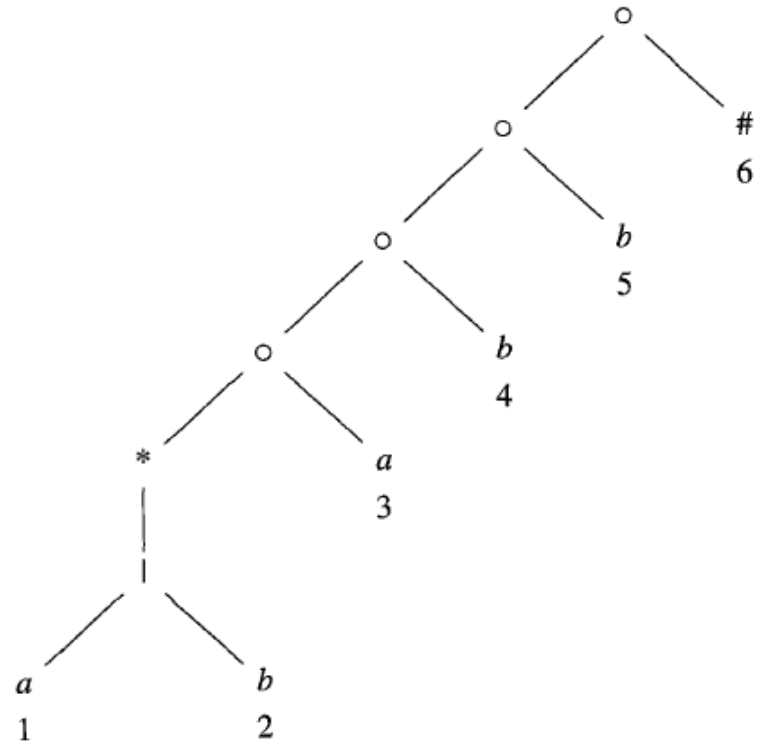
构造正则表达式的语法树

例如 $(a \mid b)^*abb\#$ 的语法树如右

叶节点下的数字表示位置

语法树的每个结点代表

一个子表达式



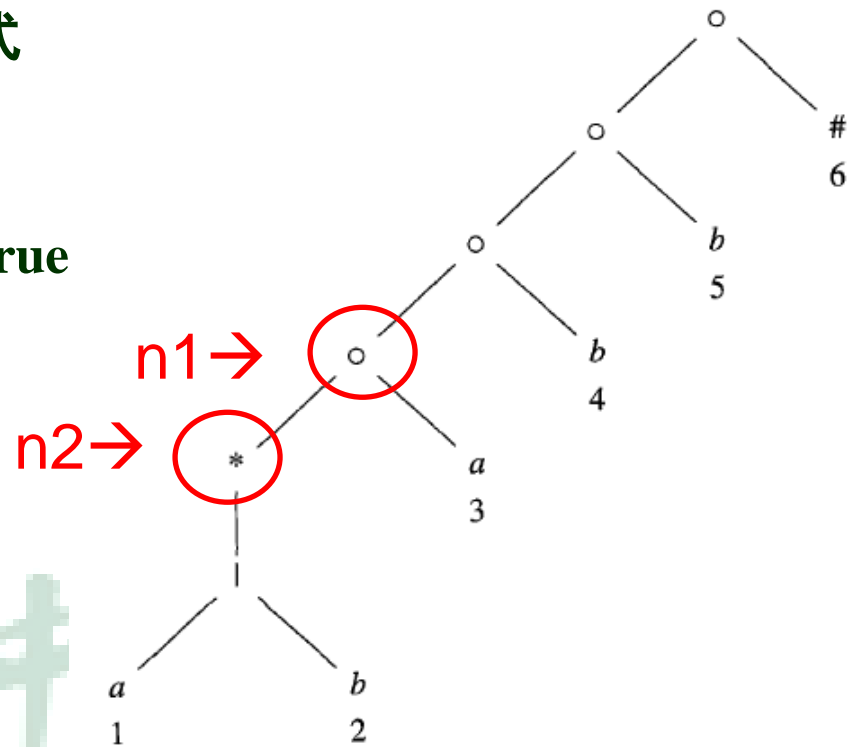
语法树的相关函数

1. *nullable(n)*: *nullable(n)* is true for a syntax-tree node *n* if and only if the subexpression represented by *n* has ϵ in its language.

例如：右图，结点n1代表子表达式

$(a|b)*a$, $nullable(n1) = \text{false}$, 而

结点n2代表 $(a|b)*$, $nullable(n2) = \text{true}$

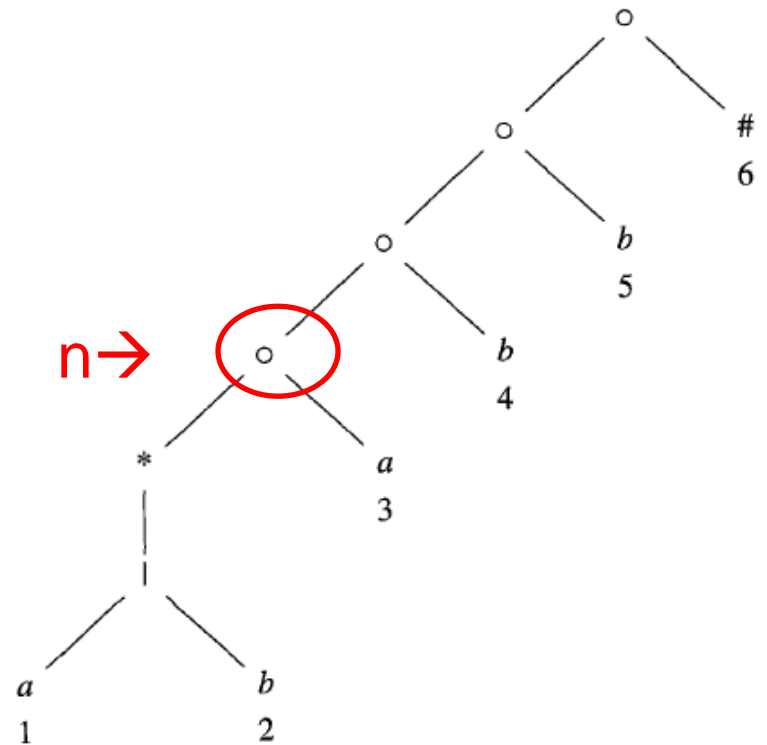


语法树的相关函数

2. $firstpos(n)$: $firstpos(n)$ is the set of positions in the subtree rooted at n that correspond to the first symbol of at least one string in the language of the subexpression rooted at n .

例如，右图中， n 代表 $(a|b)*a$,

$firstpos(n) = \{1,2,3\}$

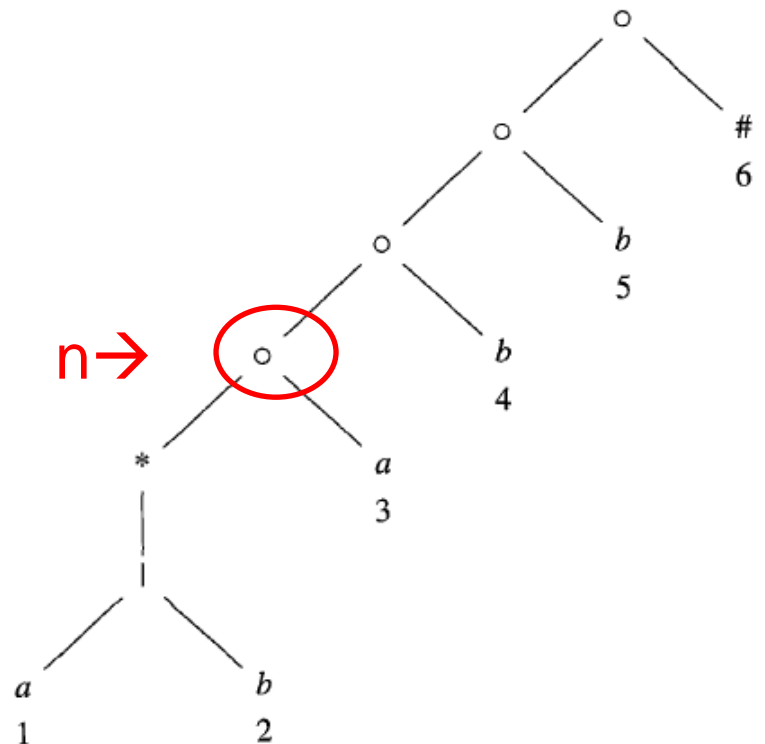


语法树的相关函数

3. $lastpos(n)$: $lastpos(n)$ is the set of positions in the subtree rooted at n that correspond to the last symbol of at least one string in the language of the subexpression rooted at n .

例如，右图中， n 代表 $(a|b)^*a$,

$lastpos(n) = \{3\}$





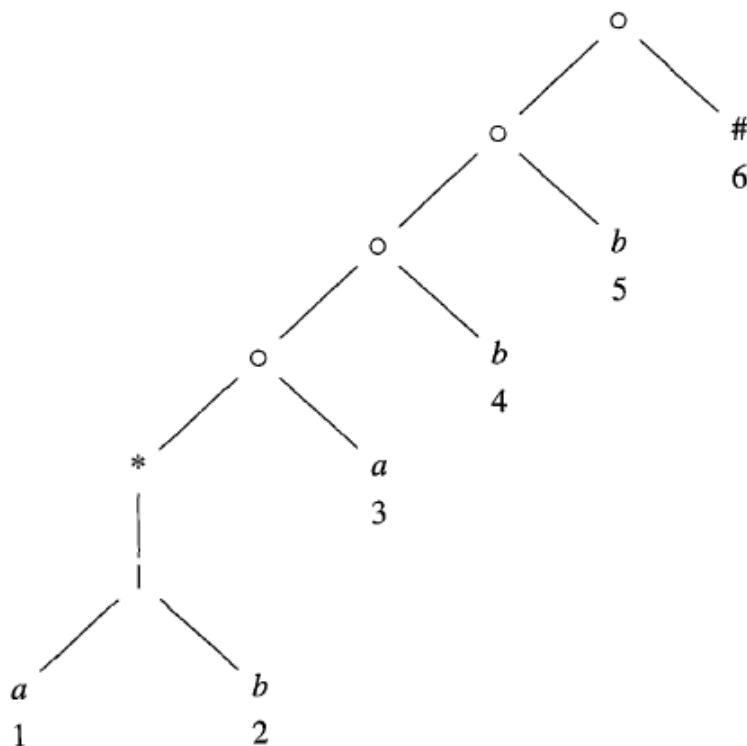
中山大學

语法树的相关函数

4. $followpos(p)$, for a position p , is the set of positions q in the entire syntax tree such that there is some string $x = a_1a_2 \cdots a_n$ in $L((r)\#)$ such that for some i , there is a way to explain the membership of x in $L((r)\#)$ by matching a_i to position p of the syntax tree and a_{i+1} to position q .

例如，右图中，

$$followpos(1) = \{1, 2, 3\}$$





中山大學

函数的计算方法

NODE n	$nullable(n)$	$firstpos(n)$
A leaf labeled ϵ	true	\emptyset
A leaf with position i	false	$\{i\}$
An or-node $n = c_1 c_2$	$nullable(c_1)$ or $nullable(c_2)$	$firstpos(c_1) \cup firstpos(c_2)$
A cat-node $n = c_1 c_2$	$nullable(c_1)$ and $nullable(c_2)$	if ($nullable(c_1)$) $firstpos(c_1) \cup firstpos(c_2)$ else $firstpos(c_1)$
A star-node $n = c_1^*$	true	$firstpos(c_1)$

中山大學



中山大學

計算 *followpos*

Two rules:

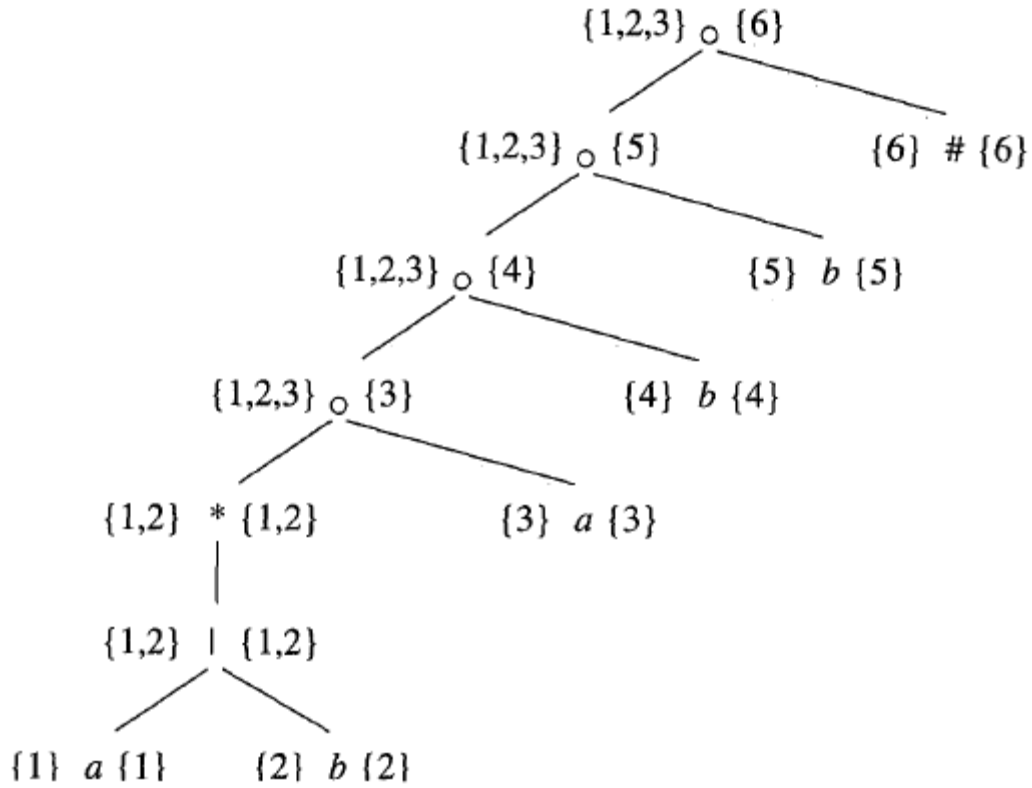
1. If n is a cat-node with left child c_1 and right child c_2 , then for every position i in $lastpos(c_1)$, all positions in $firstpos(c_2)$ are in $followpos(i)$.
2. If n is a star-node, and i is a position in $lastpos(n)$, then all positions in $firstpos(n)$ are in $followpos(i)$.



中山大學



计算followpos的例子



position i	$followpos(i)$
1	{1, 2, 3}
2	{1, 2, 3}
3	{4}
4	{5}
5	{6}
6	\emptyset

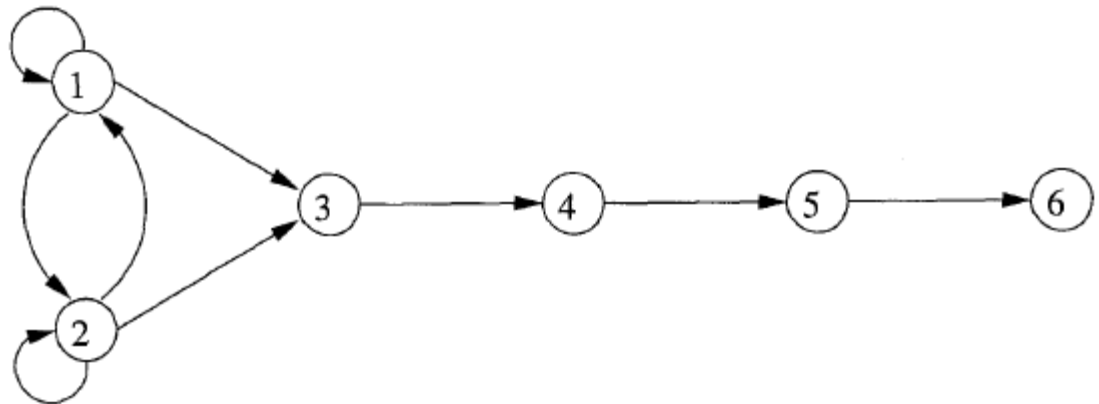


中山大學

用有向图表示 *followpos* 函数

position i	$followpos(i)$
1	$\{1, 2, 3\}$
2	$\{1, 2, 3\}$
3	$\{4\}$
4	$\{5\}$
5	$\{6\}$
6	\emptyset

可表示为



中山大學



中山大學

转换算法

INPUT: A regular expression r .

OUTPUT: A DFA D that recognizes $L(r)$.

METHOD:

initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
where n_0 is the root of syntax tree T for $(r)\#$;

while (there is an unmarked state S in $Dstates$) {
 mark S ;

for (each input symbol a) {
 let U be the union of $followpos(p)$ for all p
 in S that correspond to a ;

if (U is not in $Dstates$)
 add U as an unmarked state to $Dstates$;
 $Dtran[S, a] = U$;

 }

}

學



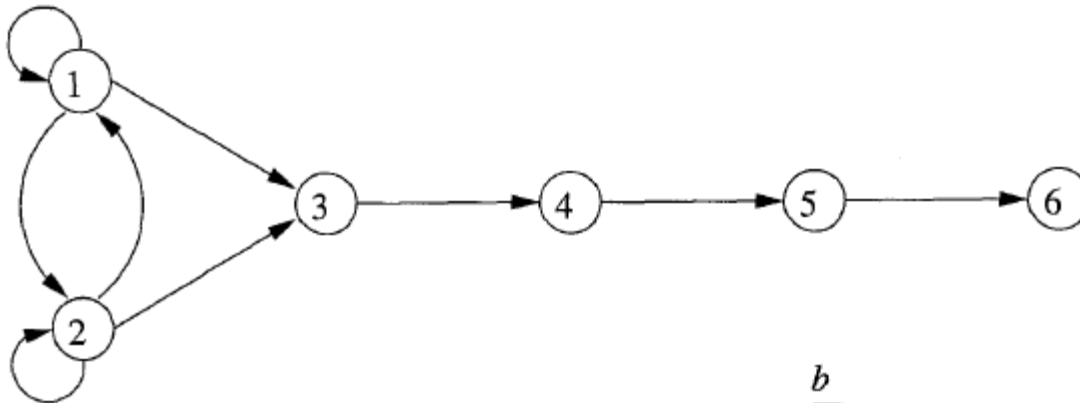
中山大學

• 例如 $(a|b)^*abb\#$

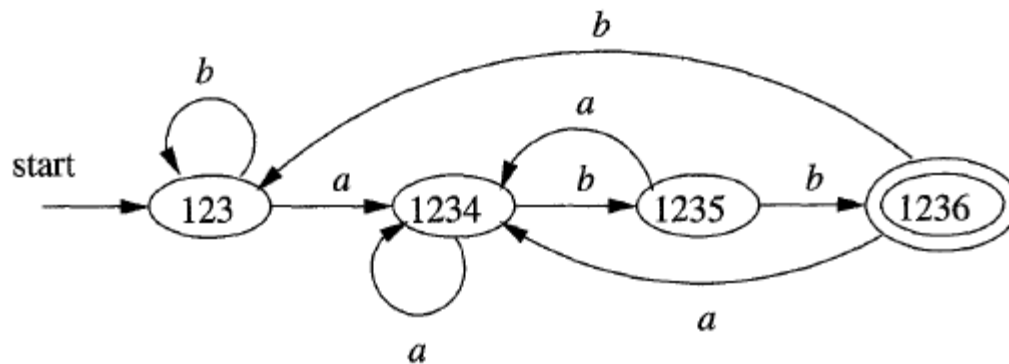


位置: 1 2 3 4 5 6

对应的followpos函数为



运用前述算法得

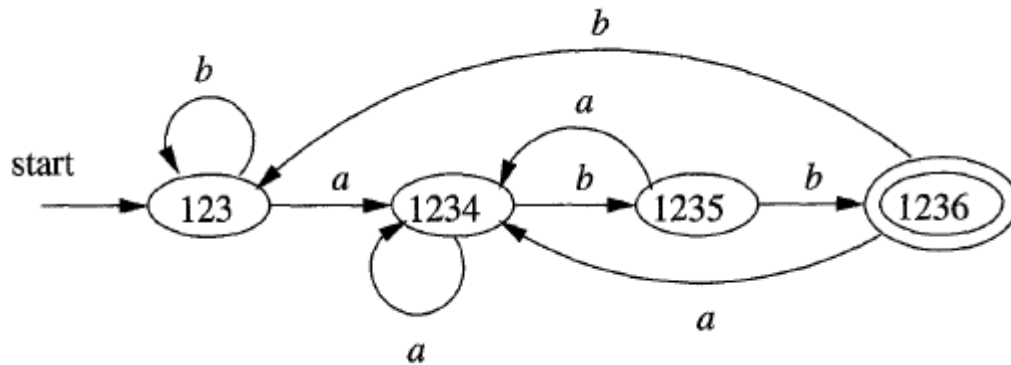


中山大學



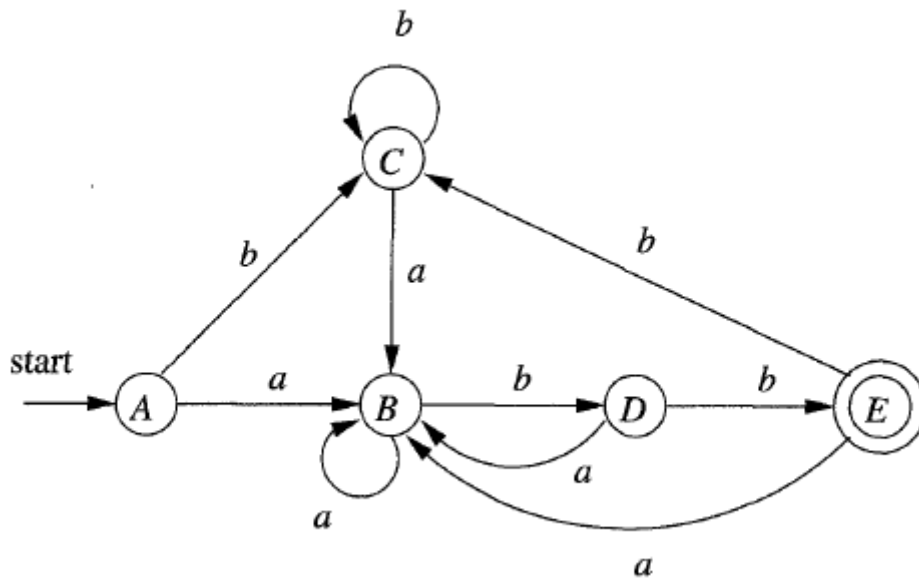
中山大學

比较



上图比下图少了一个状态

问题：如何使得
DFA的状态数最少？



山大學



中山大學

*DFA*的最小化

INPUT: A DFA D with set of states S , input alphabet Σ , state state s_0 , and set of accepting states F .

OUTPUT: A DFA D' accepting the same language as D and having as few states as possible.

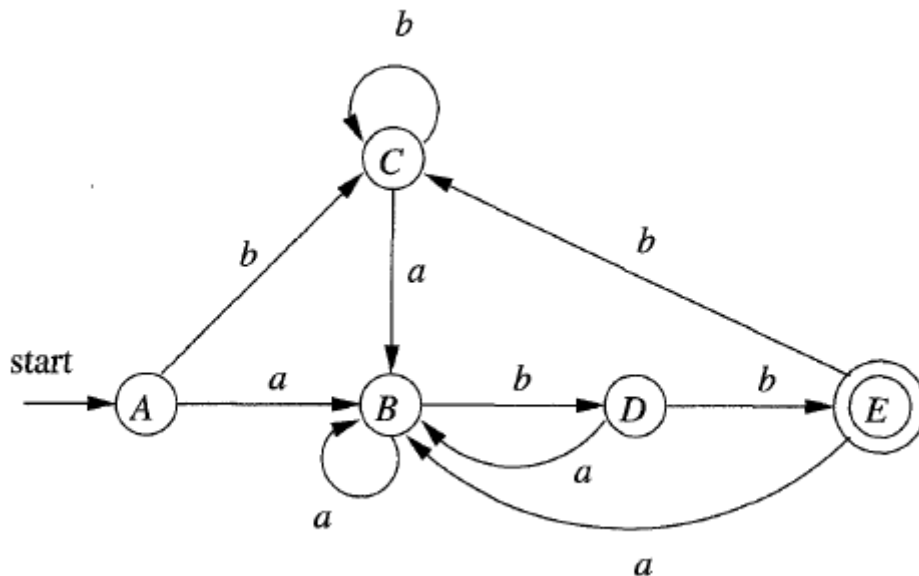


中山大學

Basic Idea

String x distinguishes state s from state t if **exactly one** of the states reached from s and t by following the path with label x is an accepting state.

State s is distinguishable from state t if there is some string that distinguishes them.



例如，左图中字符串bb区分状态B和C



中山大學

最小化 DFN 的算法

1. Start with an initial partition Π with two groups, F and $S - F$, the accepting and nonaccepting states of D .
2. Apply the procedure of Fig. 3.64 to construct a new partition Π_{new} .

initially, let $\Pi_{\text{new}} = \Pi$;

for (each group G of Π) {

 partition G into subgroups such that two states s and t
 are in the same subgroup if and only if for all
 input symbols a , states s and t have transitions on a
 to states in the same group of Π ;

 /* at worst, a state will be in a subgroup by itself */

 replace G in Π_{new} by the set of all subgroups formed;

}

Figure 3.64: Construction of Π_{new}

3. If $\Pi_{\text{new}} = \Pi$, let $\Pi_{\text{final}} = \Pi$ and continue with step (4). Otherwise, repeat step (2) with Π_{new} in place of Π .



中山大學

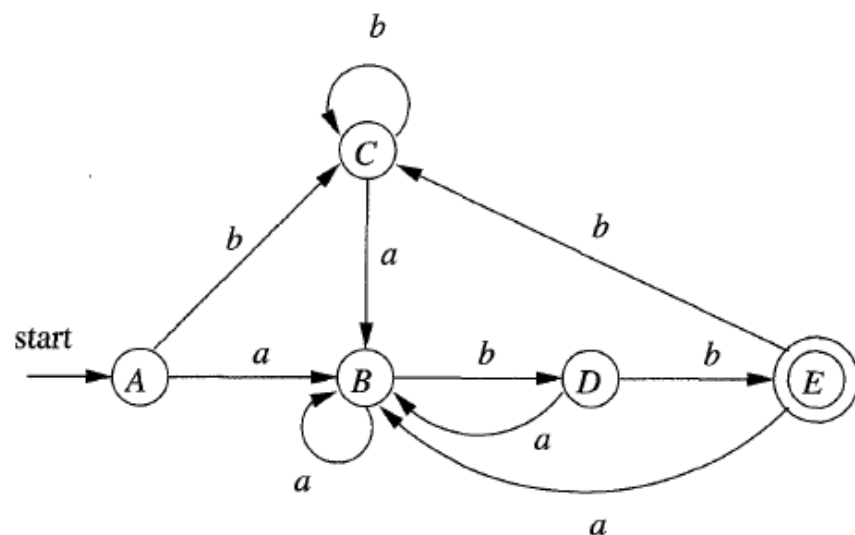
最小化 DFA 的算法 (续)

4. Choose one state in each group of Π_{final} as the *representative* for that group. The representatives will be the states of the minimum-state DFA D' . The other components of D' are constructed as follows:
 - (a) The state state of D' is the representative of the group containing the start state of D .
 - (b) The accepting states of D' are the representatives of those groups that contain an accepting state of D . Note that each group contains either only accepting states, or only nonaccepting states, because we started by separating those two classes of states, and the procedure of Fig. 3.64 always forms new groups that are subgroups of previously constructed groups.
 - (c) Let s be the representative of some group G of Π_{final} , and let the transition of D from s on input a be to state t . Let r be the representative of t 's group H . Then in D' , there is a transition from s to r on input a . Note that in D , every state in group G must go to some state of group H on input a , or else, group G would have been split according to Fig. 3.64.



中山大學

例子



最小化得

STATE	<i>a</i>	<i>b</i>
<i>A</i>	<i>B</i>	<i>A</i>
<i>B</i>	<i>B</i>	<i>D</i>
<i>D</i>	<i>B</i>	<i>E</i>
<i>E</i>	<i>B</i>	<i>A</i>

中山大學



中山大學

讨论

为什么上述最小化算法是正确的？

- 为什么合并不可区分状态依然得到等价的DFA？
- 为什么算法结束之后，在不同group中的状态一定是可区分的？而同一group中的状态一定是不可区分的？
- 有没可能重新构造一个新的等价DFA，其状态数更少？

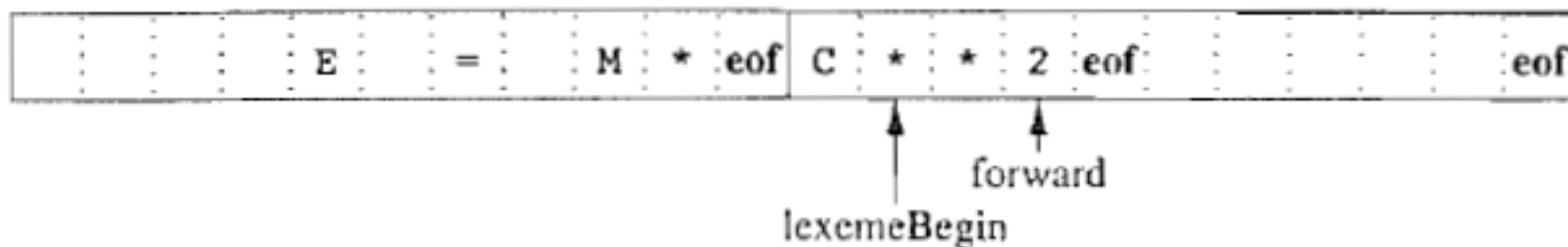
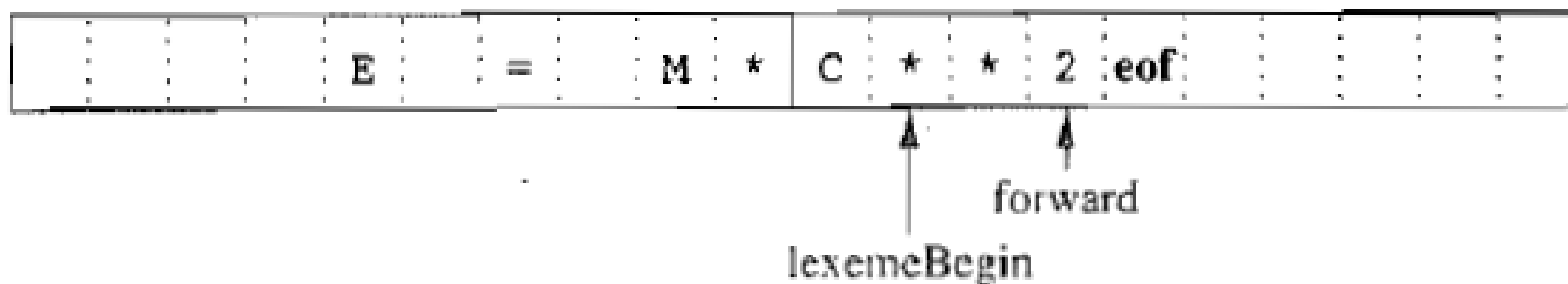
算法的复杂度多少（字母表大小视为常数）？

中山大學



中山大學

具体实现

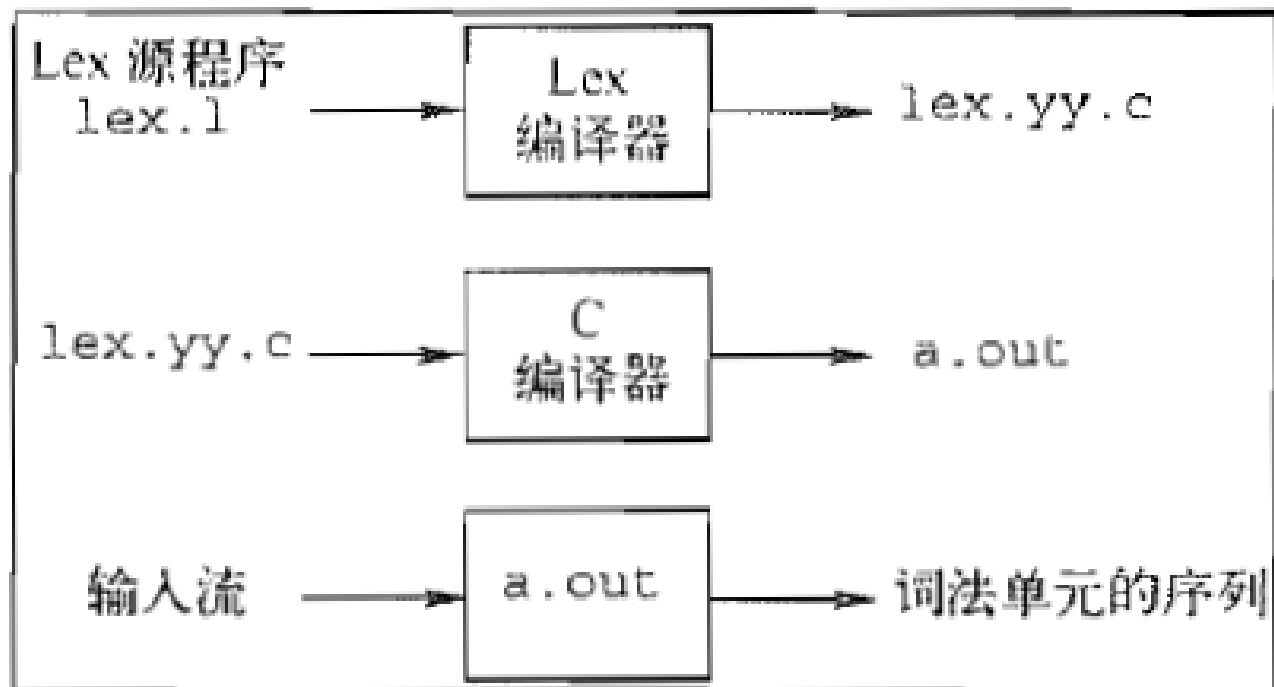


中山大學



中山大學

具体实现



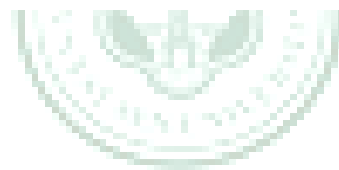
中山大學



中山大學

具体实现

```
/* regular definitions */  
delim    [ \t\n]  
ws       {delim}+  
letter   [A-Za-z]  
digit    [0-9]  
id       {letter}({letter}|{digit})*  
number   {digit}+(\.{digit}+)?(E[+-]?{digit}+)?
```



中山大學



中山大學

具体实现

```
{ws}      { /* no action and no return */ }
if        {return(IF);}
then      {return(THEN);}
else      {return(ELSE);}
{id}      {yyval = (int) installID(); return(ID);}
{number}  {yyval = (int) installNum(); return(NUMBER);}
"<"      {yyval = LT; return(RELOP);}
"<="     {yyval = LE; return(RELOP);}
"="       {yyval = EQ; return(RELOP);}
"<>"     {yyval = NE; return(RELOP);}
">"      {yyval = GT; return(RELOP);}
">="     {yyval = GE; return(RELOP);}
```

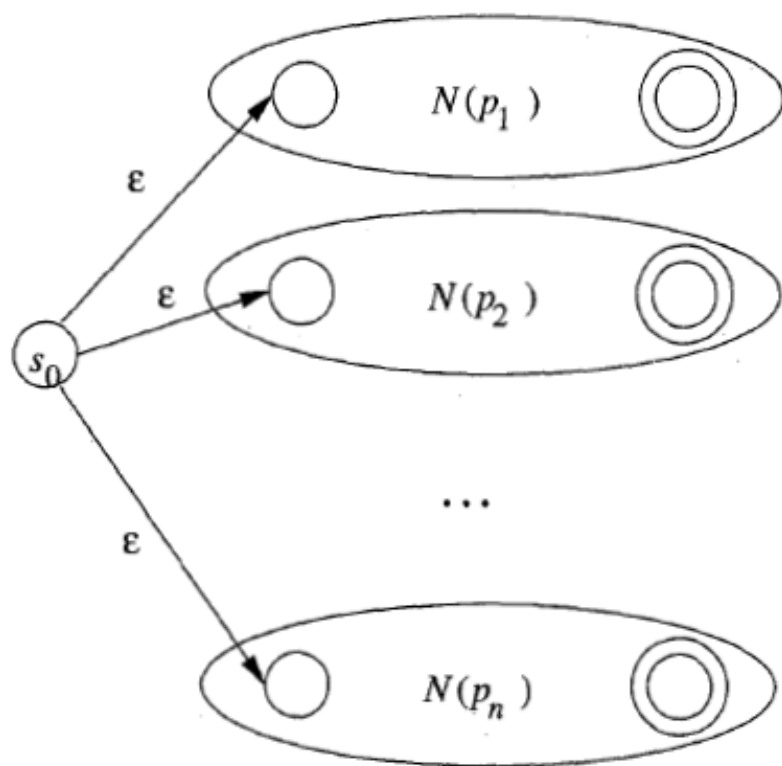
中山大學



中山大學

一些实际问题

多类token的识别

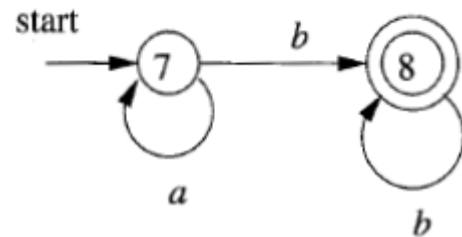
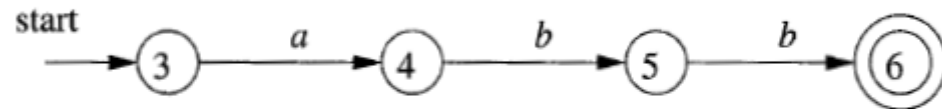
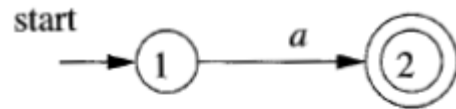


中山大學



中山大學

例子



NFA's for a, **abb**, and **a^{*}b⁺**

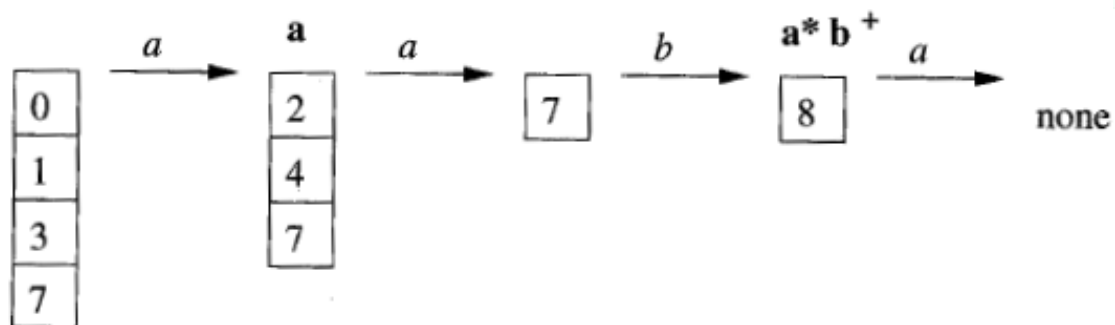
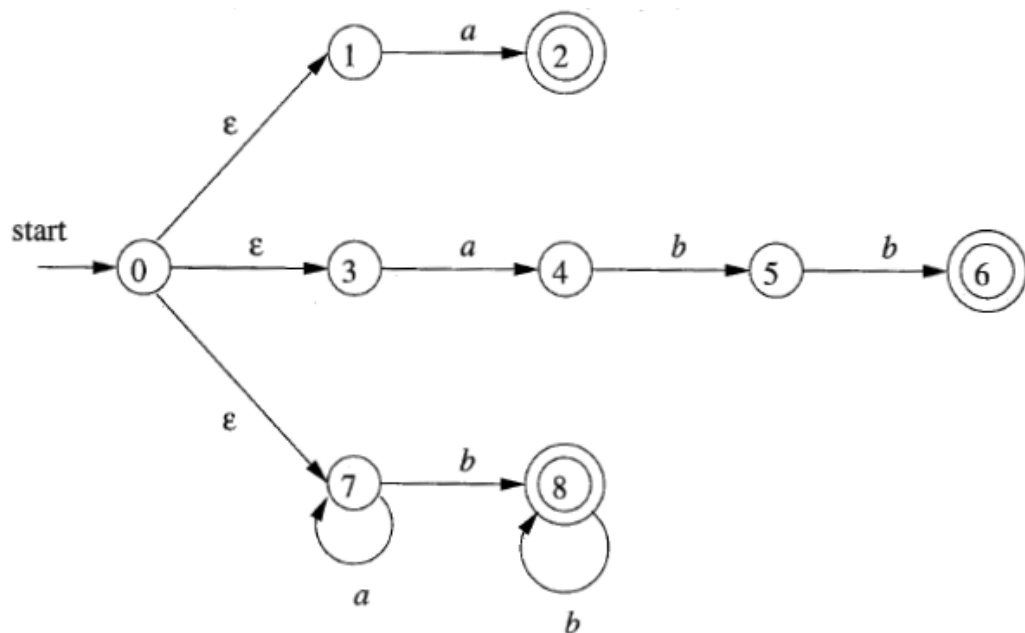


中山大學



中山大學

例子



山大學



中山大學

冲突的解决

取最长匹配串

规定接受状态的优先顺序



中山大學



中山大學

上次课的重要结论

DFA, NFA和正则表达式三者的描述能力是一样的.



中山大學



中山大學

正则表达式和有限自动机的局限性

“有限自动机无法计数”

- 例如，无法构造一个有限自动机接受语言

$$L = \{a^n b^n \mid n \geq 1\}$$

如何证明？



中山大學



中山大學

剩下的問題

如何构造（正则表达式的）语法树？

有没有新的形式化方法来克服有限自动机的
局限性？



中山大學



中山大學

See you next time!



中山大學