第二章：词法分析

自动机与正则表达式
词法分析器的设计

内容介绍

- 自动机与正则表达式的相互转换
- 词法分析器的具体设计

1. 自动机与正则表达式的相互转换

- 正则表达式向自动机的转换
- 自动机向正则表达式的转换
- 相关的例子

1.1 正则表达式向自动机的转换

- **定理 1**

  对∑上的每一个正则表达R，存在一个∑上的非确定有限自动机M，使得 L(M) = L(R) .

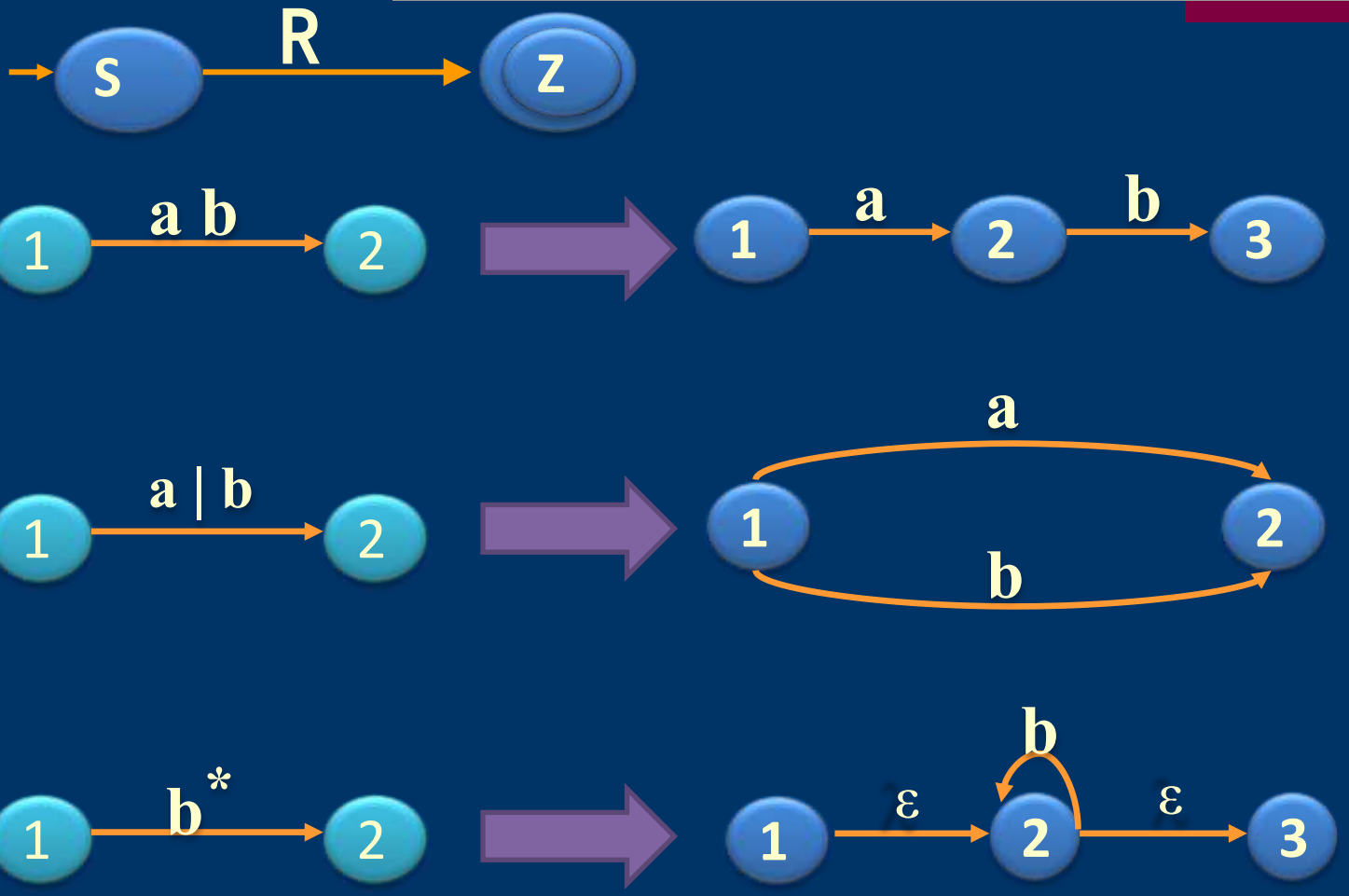
1.1 正则表达式向自动机的转换

- 构造方法

➢ step1：首先构造此非确定有限自动机M的初始状态S和终止状态Z，由S发出指向Z的有向弧并标上标记正则表达式R.

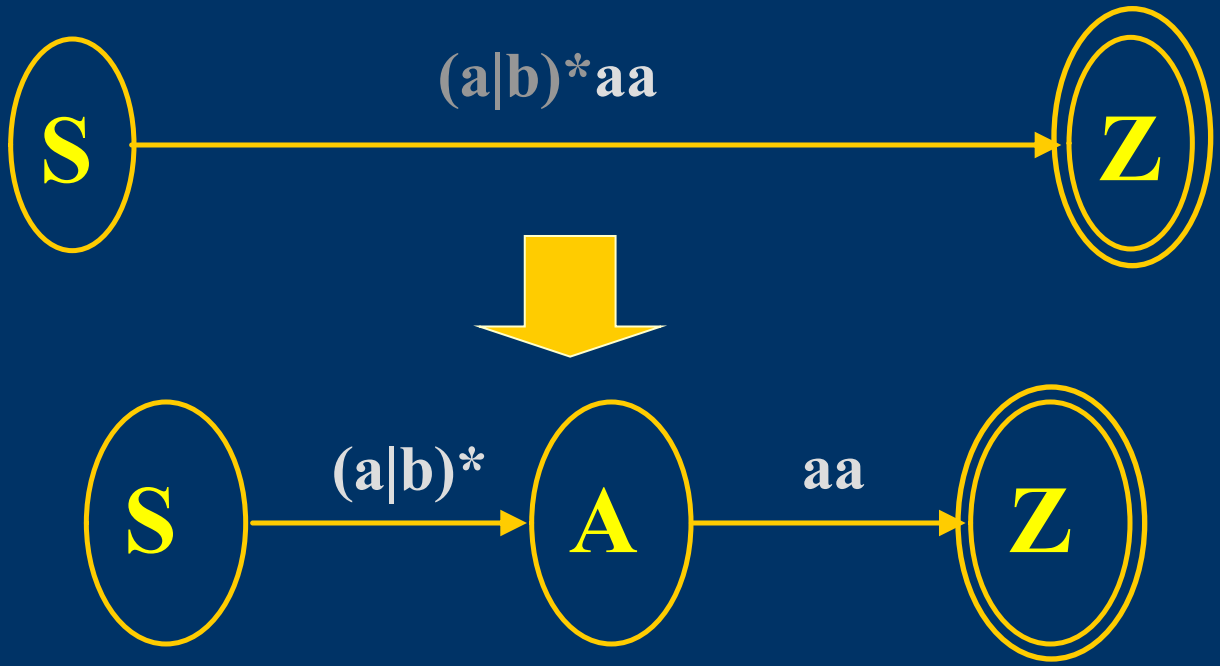
➢ step2：反复利用下述的替换规则对正则表达式R依次进行分解，直至状态转换图中所有有向弧上标记的符号都是字母表Σ上的元素或ε为止.

1.1 正则表达式向自动机的转换

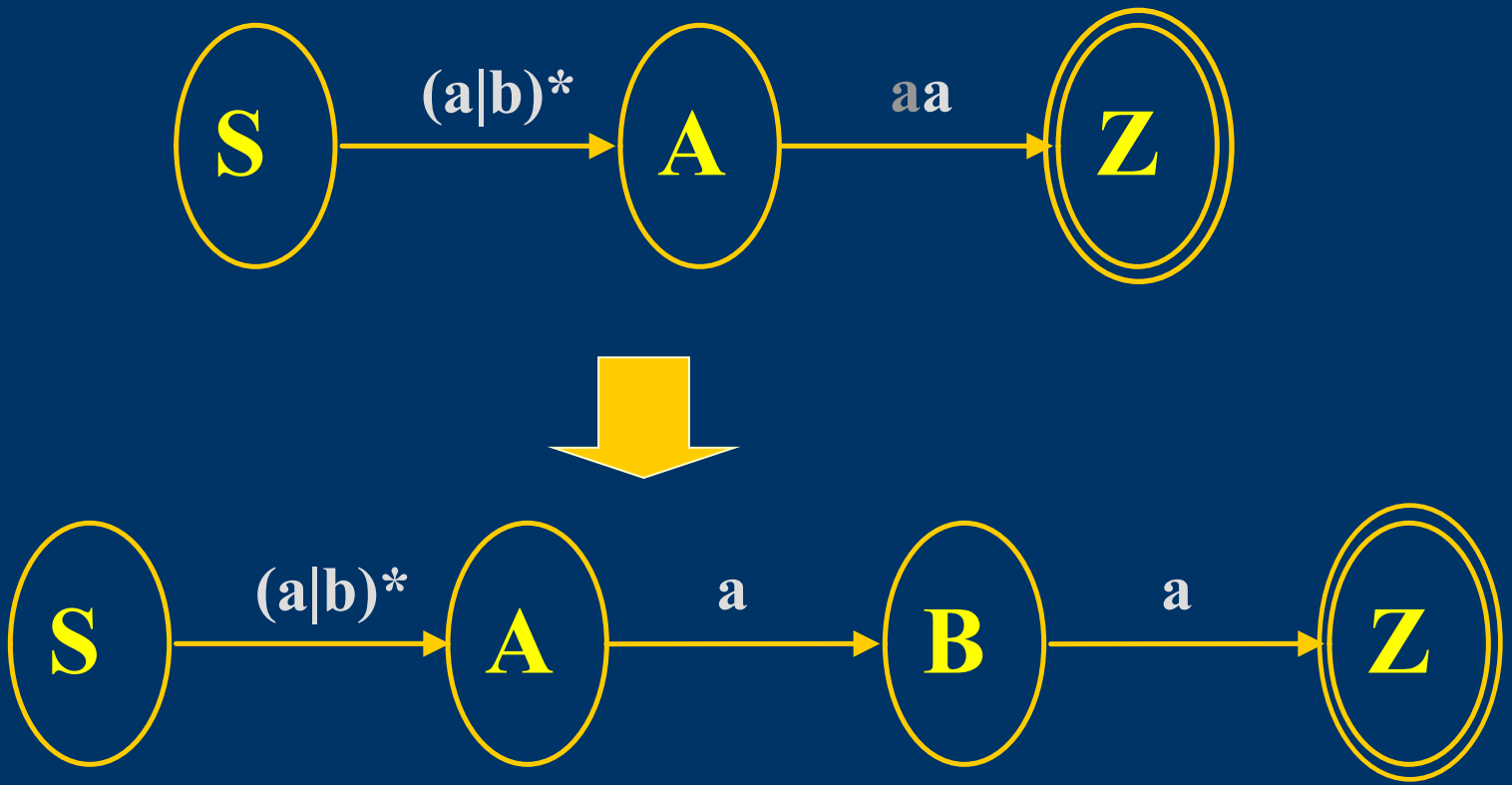


1.1 正则表达式向自动机的转换

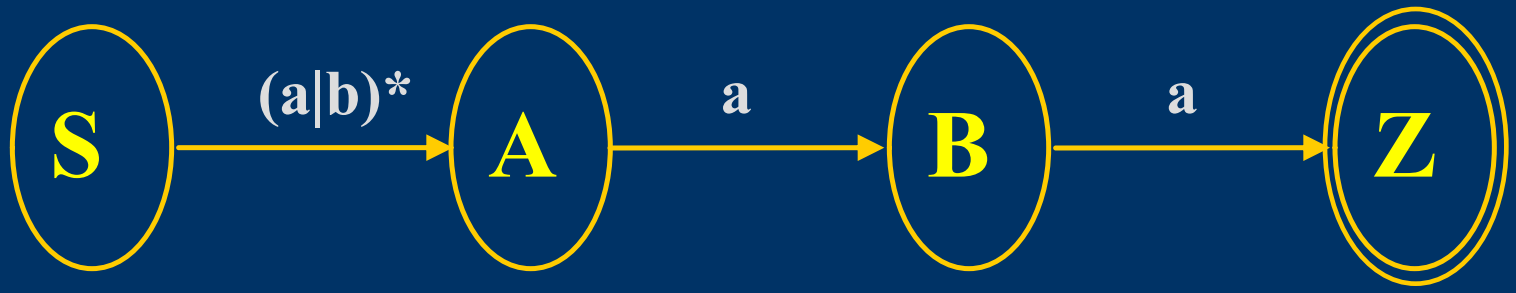
- 例：有正规表达式(a|b)*aa,为之构造等价的NFA.

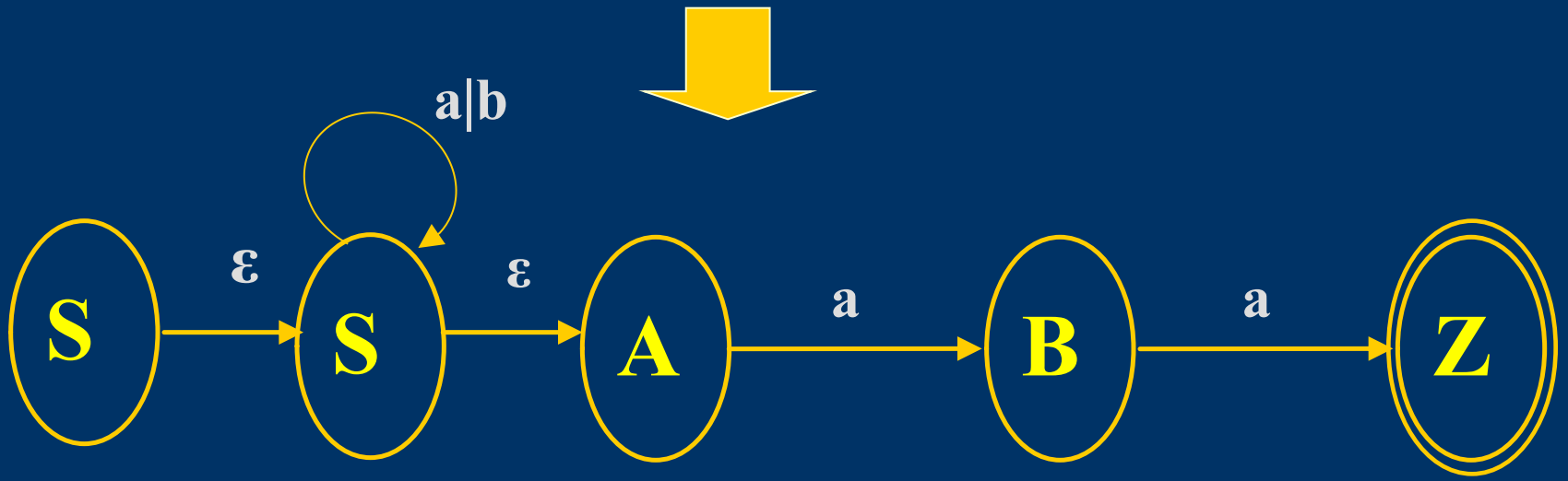


1.1 正则表达式向自动机的转换



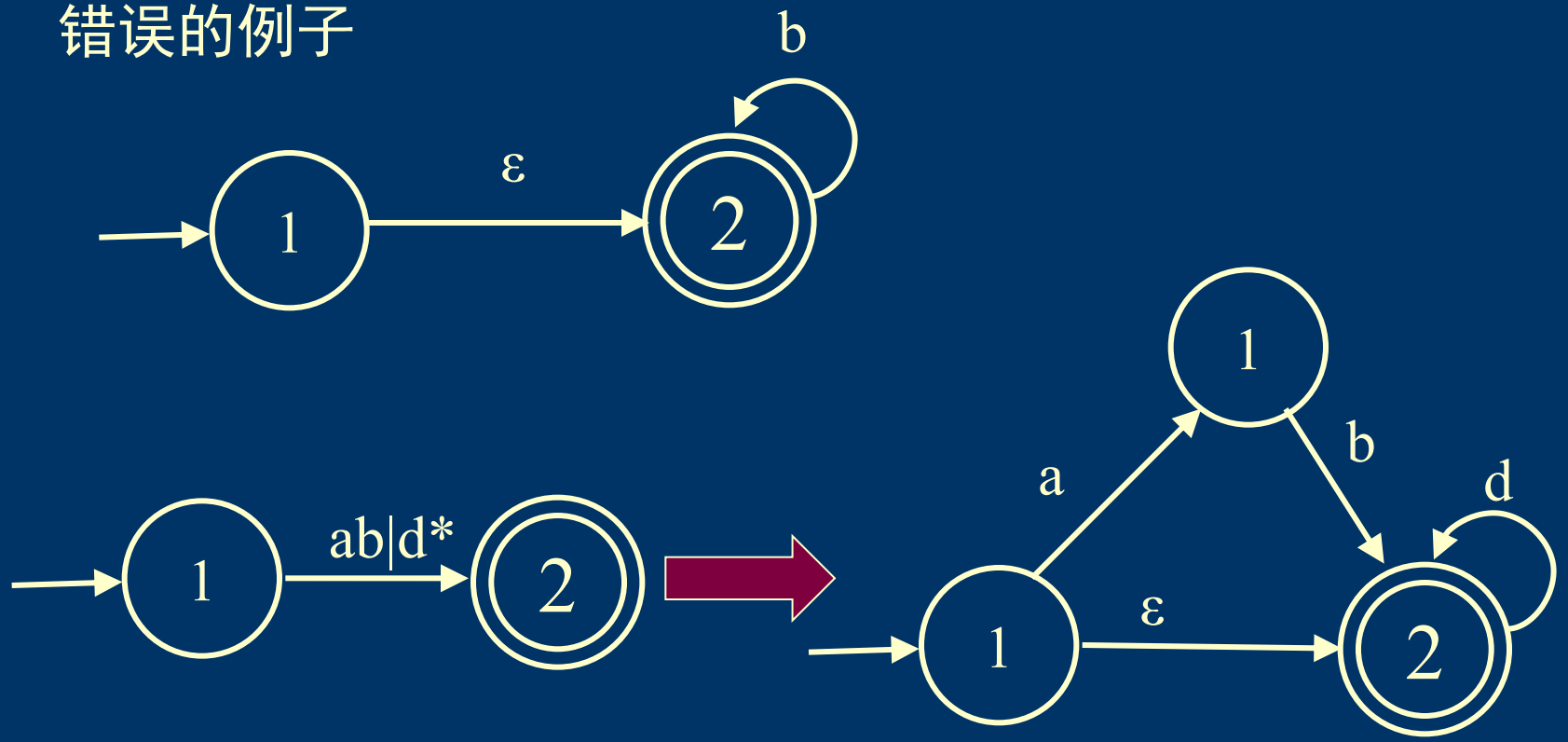
1.1 正则表达式向自动机的转换





1.1 正则表达式向自动机的转换

错误的例子



1.2 自动机向正则表达式的转换

- **定理2：对于字母表∑上的非确定有限自动机M，存在∑上的正则表达式R，使得L(R) = L(M).**
- **注意：这里M可以是DFA，DFA是NFA的一种特例。**

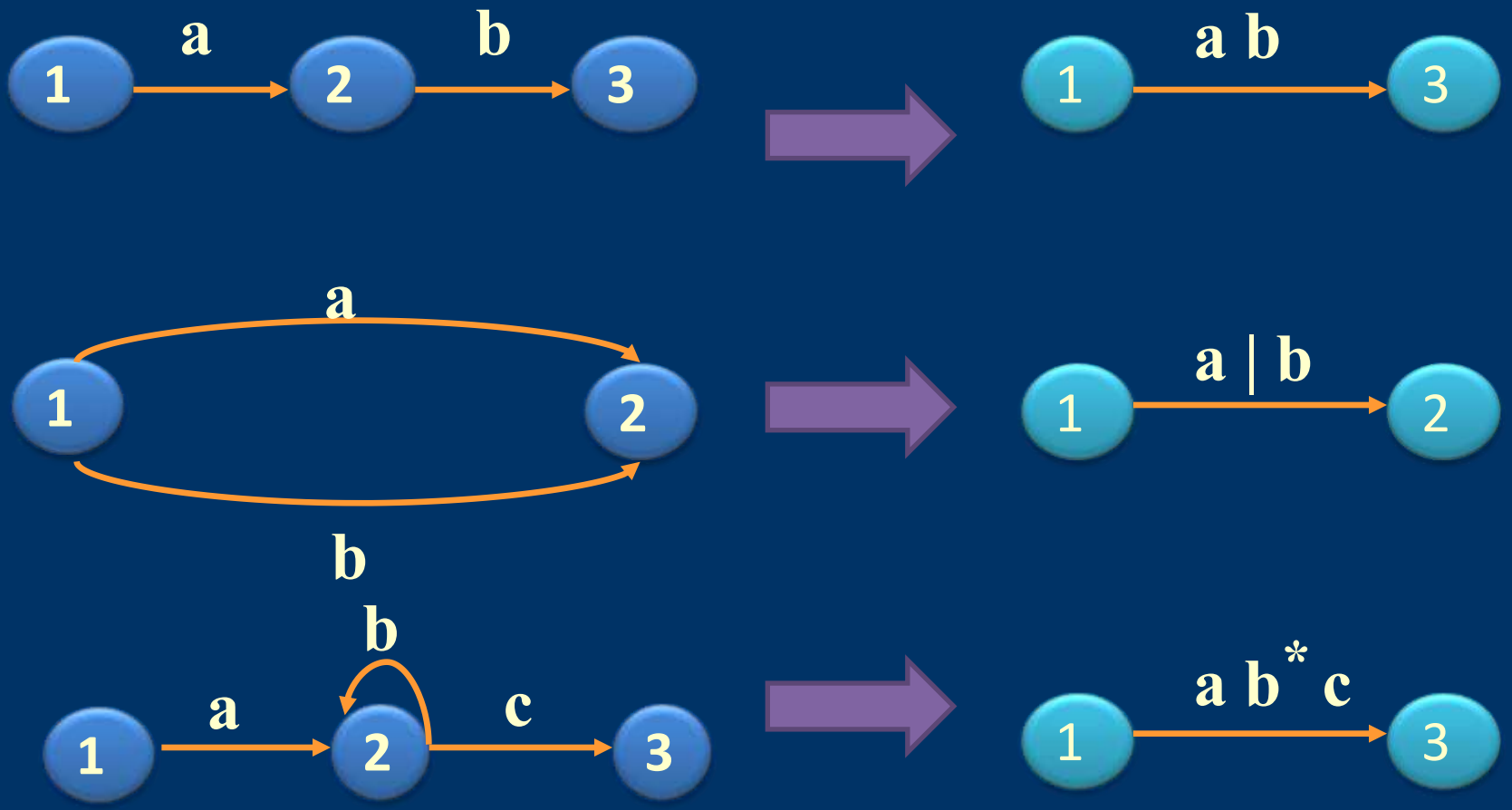
1.2 自动机向正则表达式的转换

- 构造方法：

Step1：在M的状态转换图中加入两个节点，一个是唯一的开始状态节点S，另一个是唯一的终止状态节点Z .从S出发用标有 ε 的有向弧连接到M 的所有初始状态节点上，从M 的所有终止状态节点用标有 ε 的有向弧连接到Z节点.

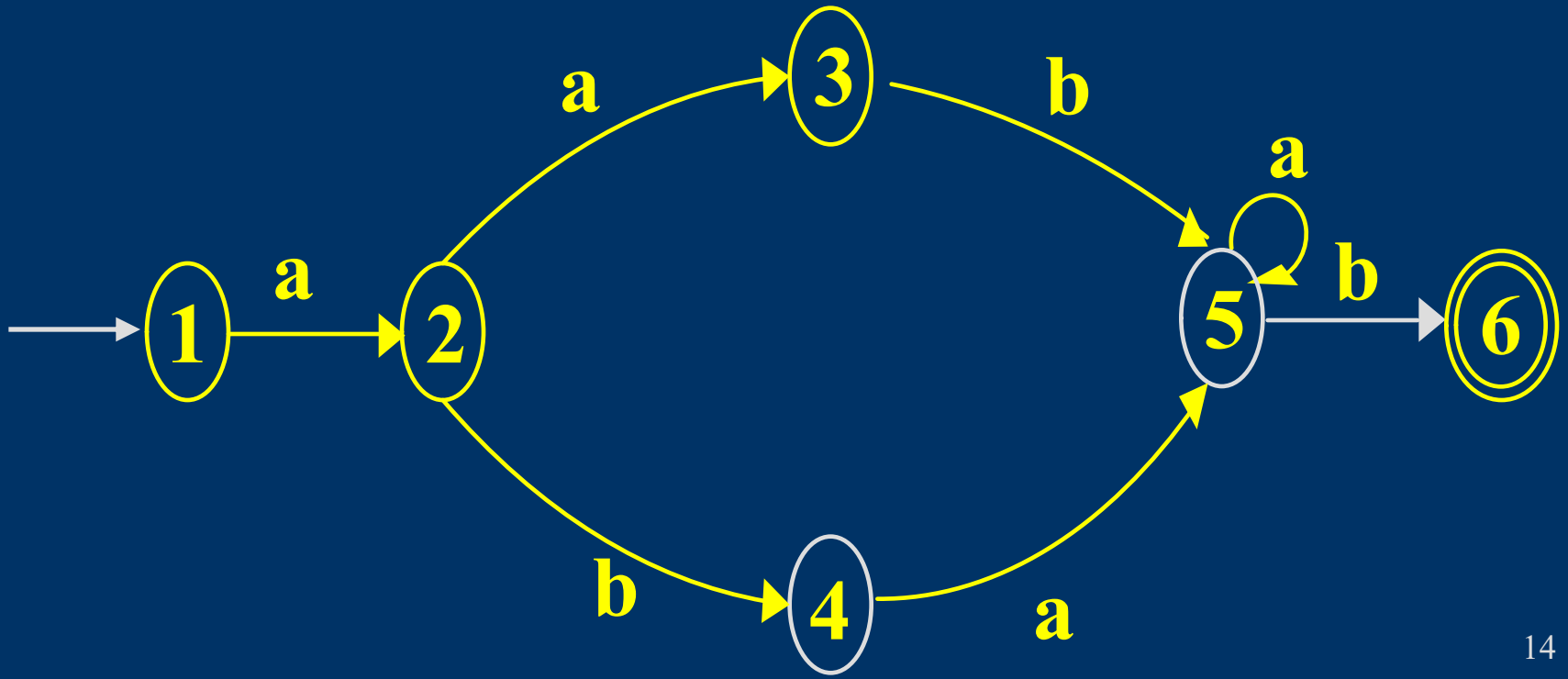
Step2：反复利用下述的替换规则进行替换，直到状态转换图中只剩下节点S和Z，在S指向Z的有向弧上所标记的正则表达式就是所求的结果.

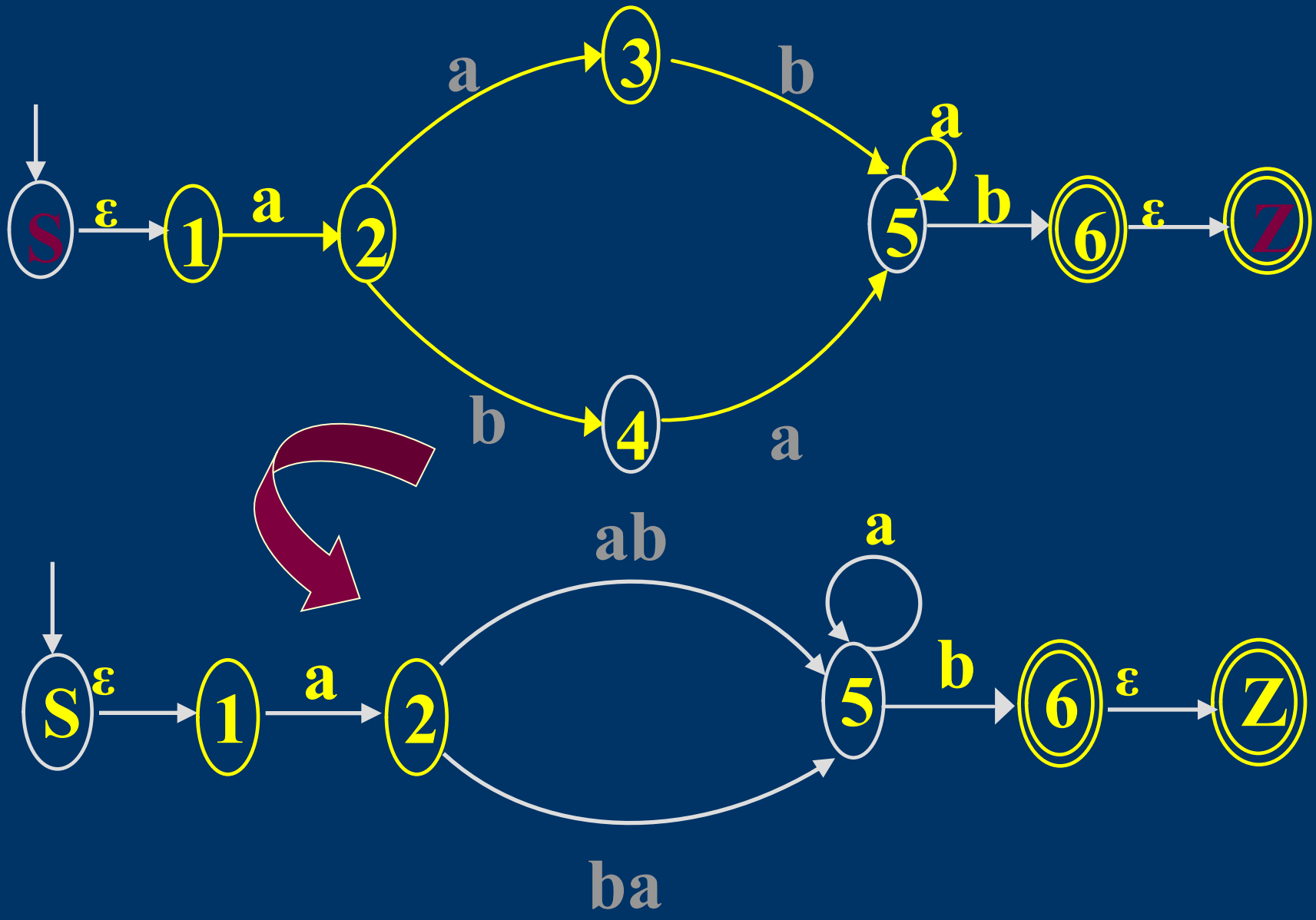
1.2 自动机向正则表达式的转换

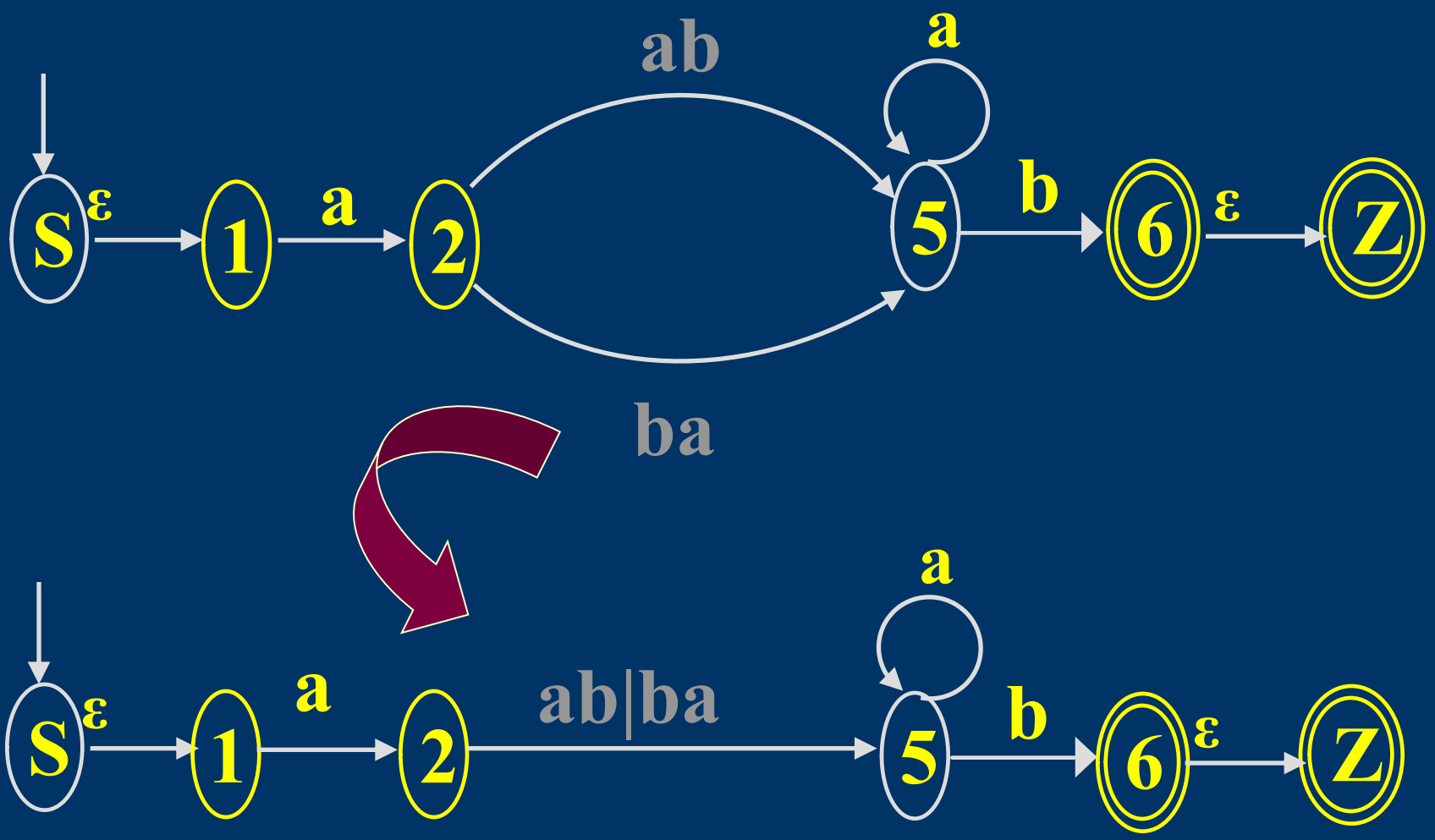


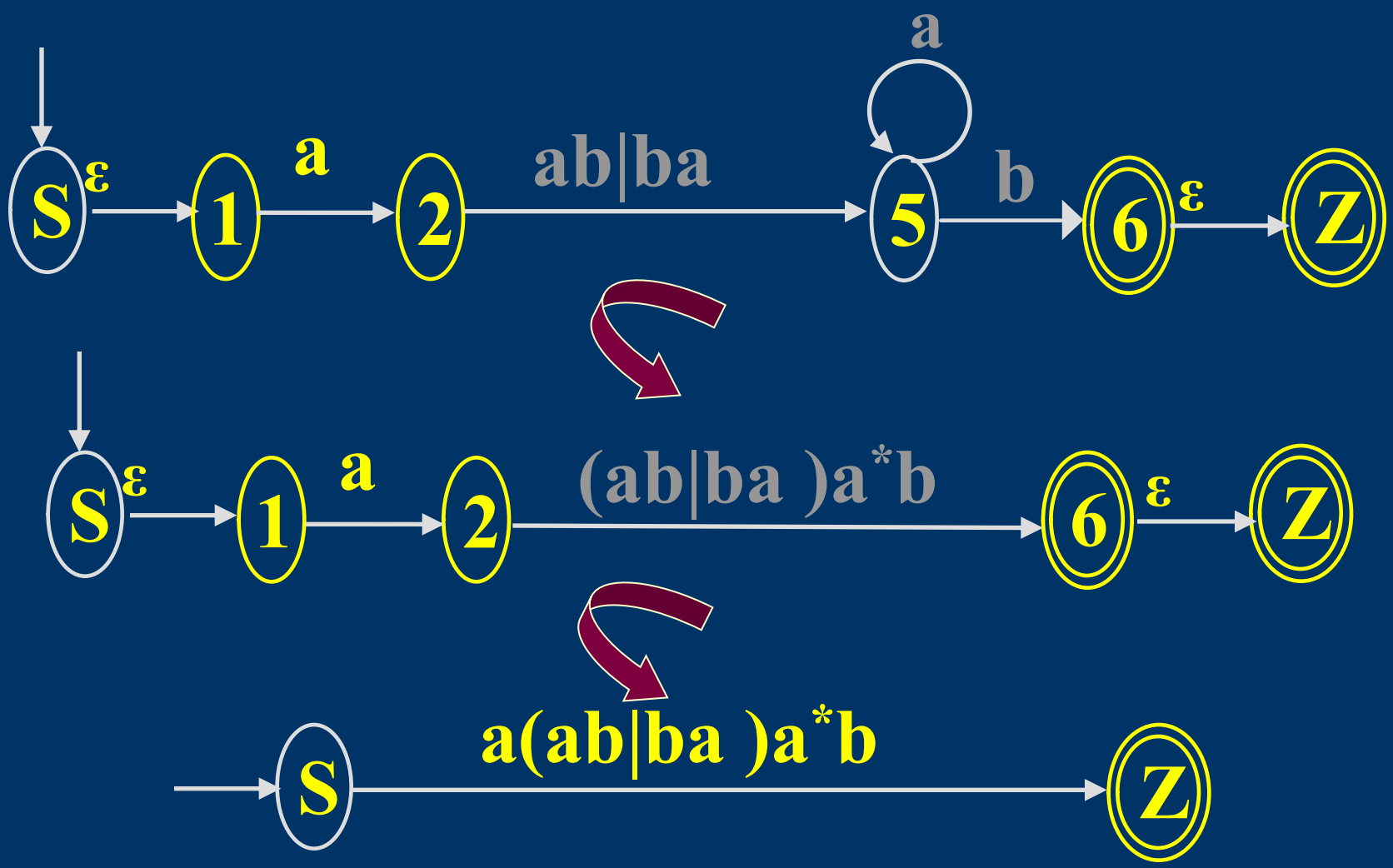
1.2 自动机向正则表达式的转换

- 例　有如下图所示的NFA M,试构造与之等价的正则表达式r



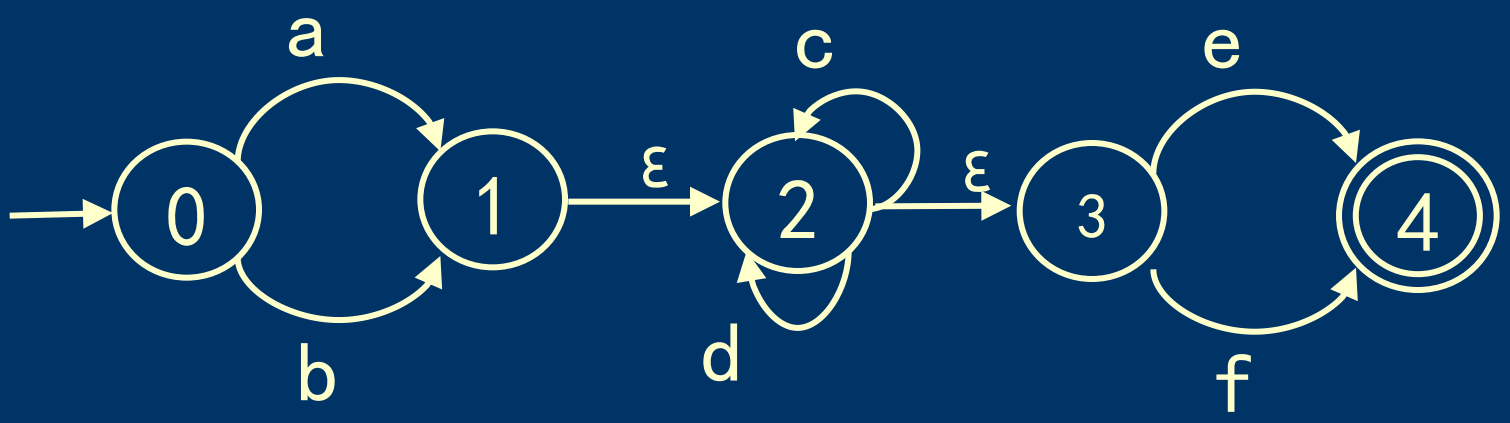

S
ε
1
a
2
a
3
b
b
4
a
5
a
b
6
ε
Z
ab
ba


ab
a
S
ε
1
a
2
5
b
6
ε
Z
ba
a
S
ε
1
a
2
ab|ba
5
b
6
ε
Z


S
ε
1
a
2
ab|ba
a
5
b
6
ε
Z
S
ε
1
a
2
(ab|ba)a*b
6
ε
Z
S
a(ab|ba)a*b
Z

1.3 相关的例子

- 给出与正则表达式(a|b)(c|d)*(e|f)等价的最简DFA



1.3 相关的例子

- 转化成DFA

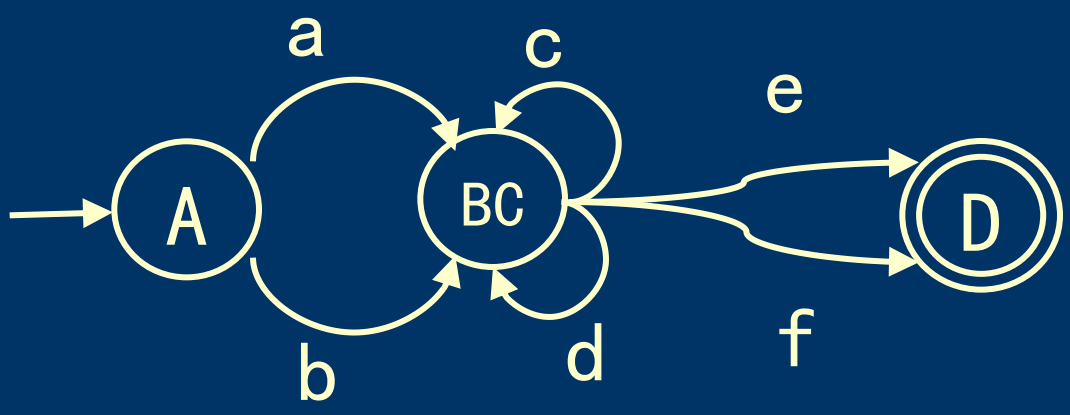
		a	b	c	d	e	f
A	{0}+	{123}	{123}				
B	{123}			{23}	{23}	{4}	{4}
C	{23}			{23}	{23}	{4}	{4}
D	{4}*						

1.3 相关的例子

- 最小化

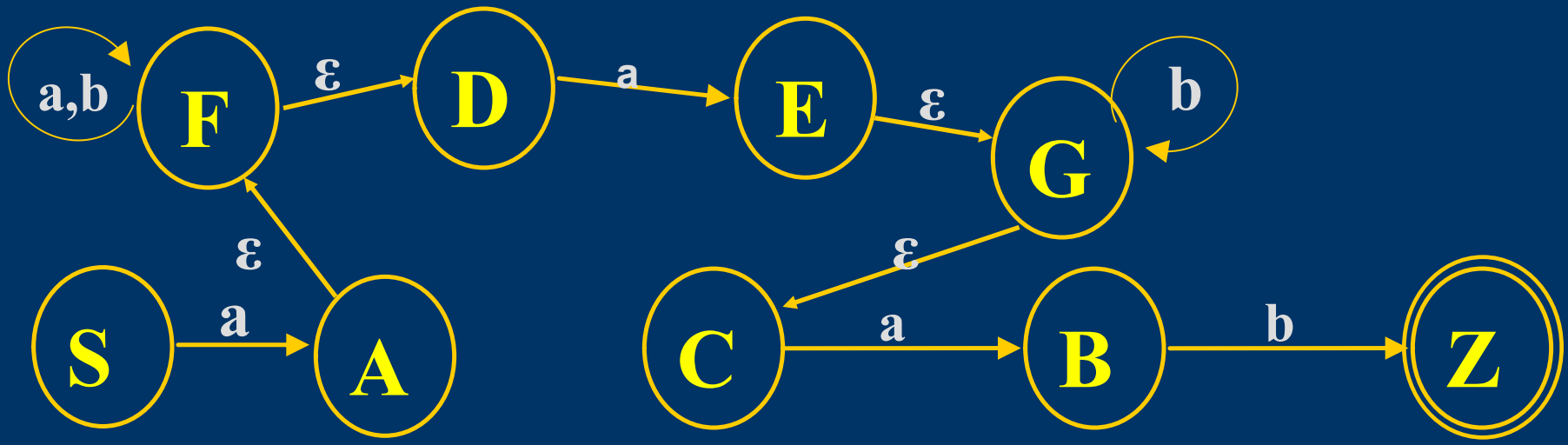
{A, B, C}　{D}

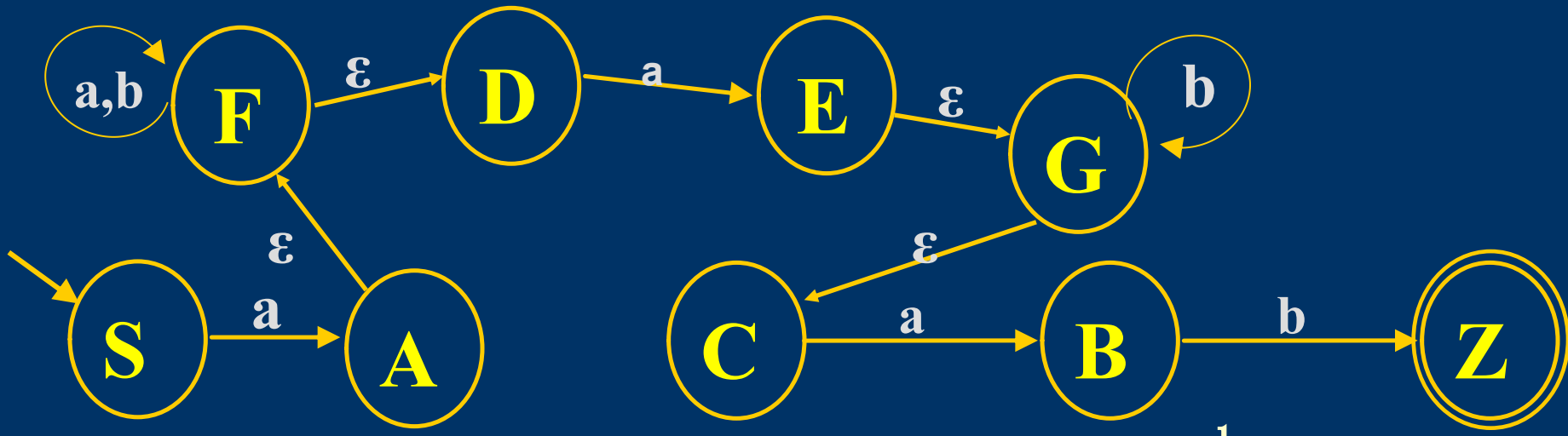
{A}　{B, C}　{D}



类型题

- 构造与正则表达式a((a|b)*ab*a)b等价的最简DFA，要求给出中间转换步骤和结果

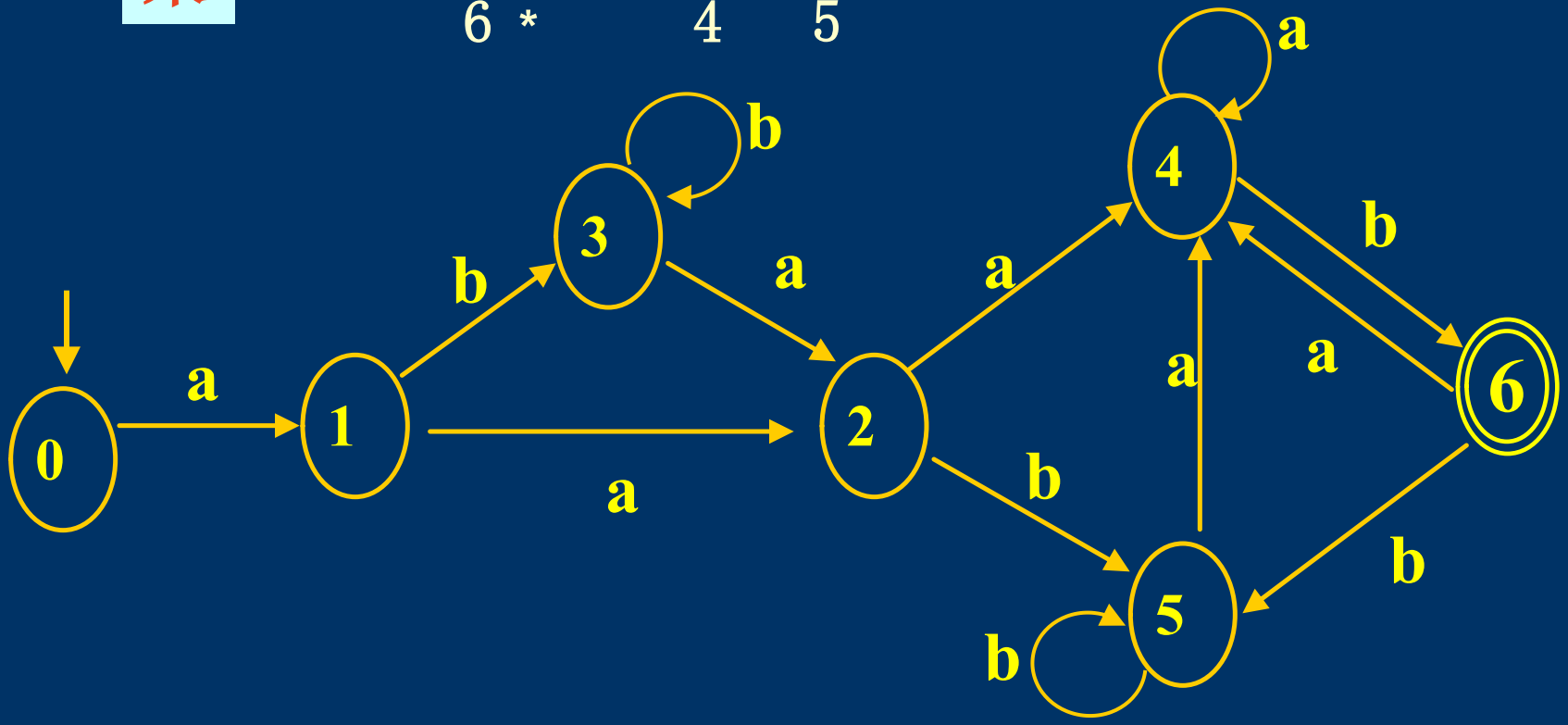




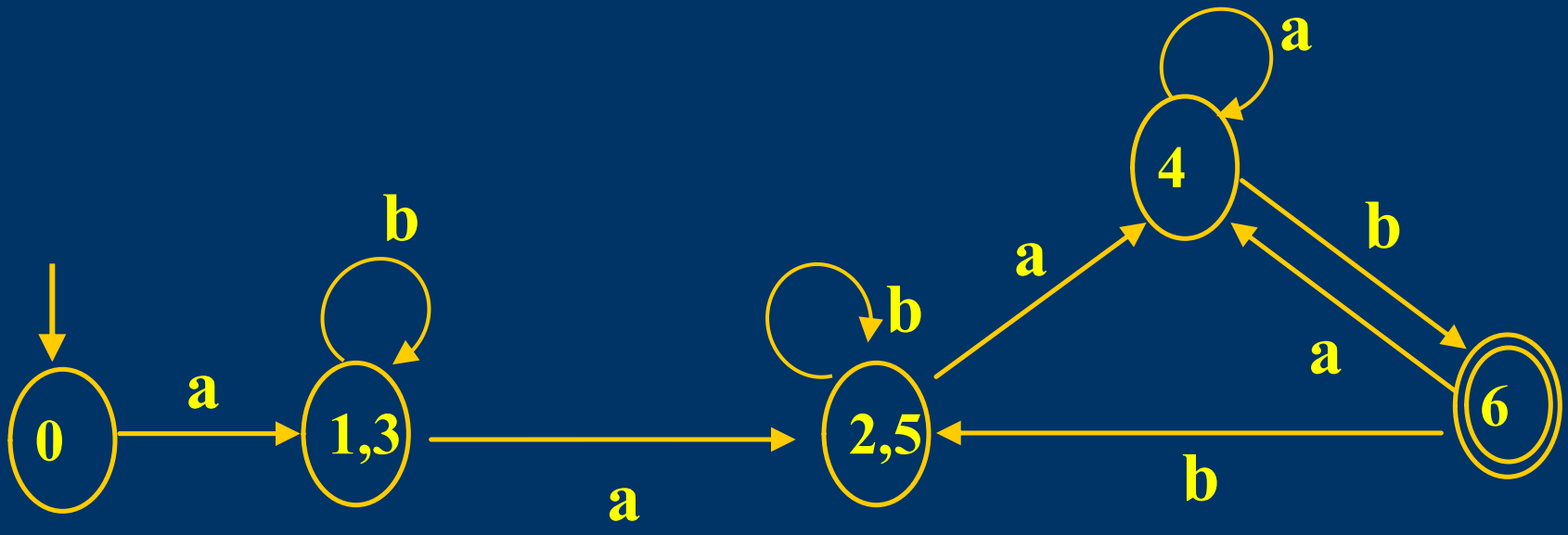
	a	b
{S} +	{A, F, D}	
{A, F, D}	{F, E, D, G, C}	{F, D}
{F, E, D, G, C}	{F, E, B, D, G, C}	{F, G, D, C}
{F, D}	{F, E, D, G, C}	{F, D}
{F, E, B, D, G, C}	{F, E, B, D, G, C}	{F, G, Z, D, C}
{F, G, D, C}	{F, E, B, D, G, C}	{F, G, D, C}
{F, G, Z, D, C} *	{F, E, B, D, G, C}	{F, G, D, C}

确定化结果

	a	b
0^+	1	
1	2	3
2	4	5
3	2	3
4	4	6
5	4	5
6 *	4	5



最小化结果



2. 词法分析器的设计

- 设计步骤
- 确定词法分析器的接口
- 确定单词的结构
- 给出单词的描述
- 设计算法

2.1 词法分析器的接口


CharList
附 属
词法分析器
call
语法分析
Token
CharList
独 立
词法分析器
TokenList
语法分析

2.2 单词的结构

- 源程序的处理过程是，首先从源程序文件一个字符一个字符进行读取，并逐个分离出单词，然后构造它们的机内表示Token。
- 关于Token的结构没有统一的规定，但至少包括两部分内容：

  单词的类型（语法信息）

  单词的内容 （语义信息）

一种可行的Token样例

Token结构

类型	内容

- ✓ 如果单词为标识符，则类型填1
- ✓ 如果单词为常量，则类型填2
  内容填其在相应索引表中的位置
- ✓ 如果单词为保留字或特殊符号，则类型填该保留字或特殊符号所对应的数字
  内容填‘空’

标识符索引表	
1	
…	…
n	

常量索引表	
1	
…	…
n	

保留字、特殊符号表	
3	while
4	if
…	…
56	}

例子

保留字、特殊符号表	
3	while
4	if
5	(
6	)
7	>
8	=
9	;
…	…
56	}

有程序：if (x>0) y=x;

Token序列为：

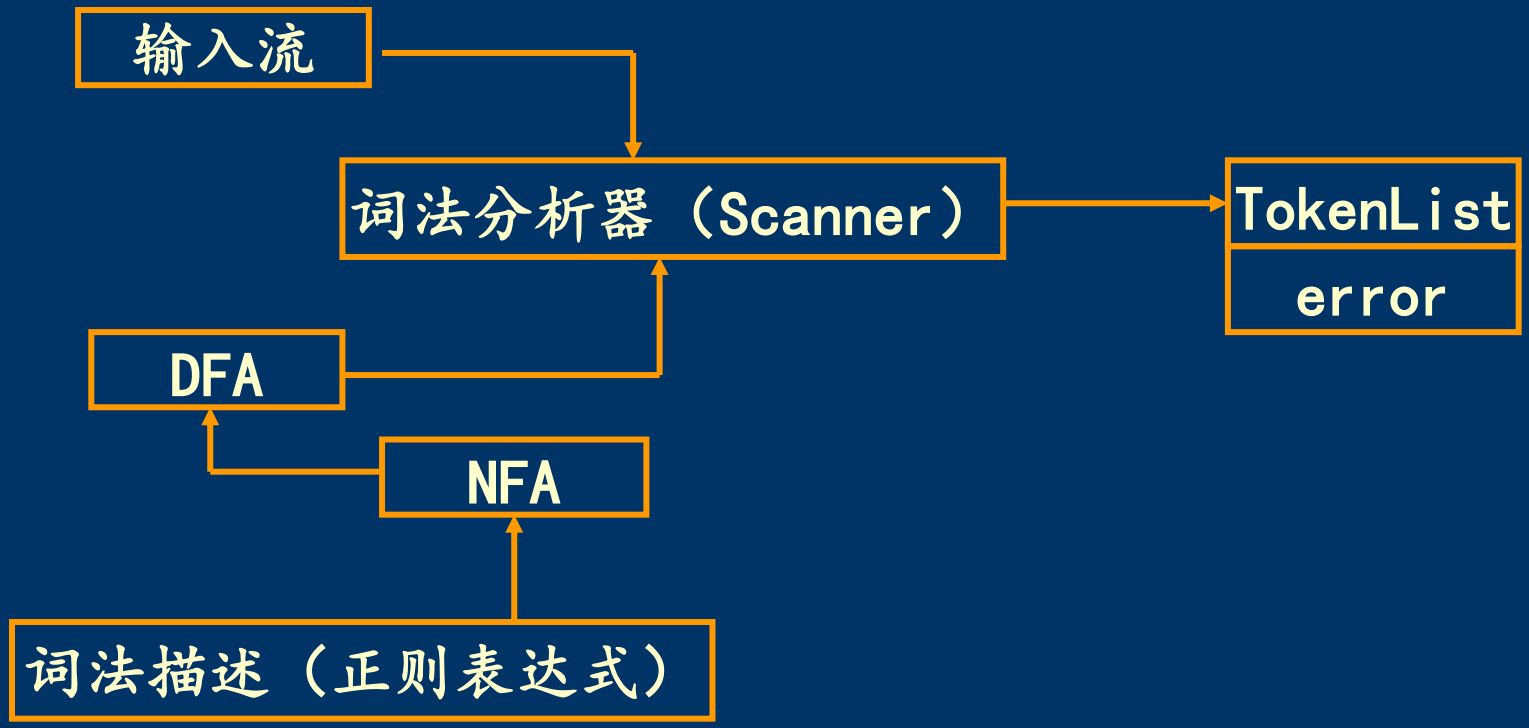
<4,->,<5,->,<1,1>,<7,-><2,1>

<6,->,<1,2>,<8,->,<1,1>,<9,->

标识符索引表	
1	x
2	y
…	…

常量索引表	
1	0
…	…

2.3 词法分析器的工作过程



根据词法分析器的工作流程可以将词法分析器的实现分为两部分：

- 单词的DFA描述及实现
- 针对不同的单词生成对应的Token

2.4 针对不同的单词生成对应的Token

- 标识符的Token生成

  首先查找保留字、特殊符号表，判断这个字符串是否为保留字，若是则生成保留字对应Token，不是则继续查找标识符索引表，确定其在标识符索引表中的位置，生成该标识符对应的Token。

- 常量的Token生成

  查找常量索引表，确定其在常量索引表中的位置，生成该常量对应的Token。

- 特殊符号的Token生成

查保留字、特殊符号表，确定这个特殊符号在表中的类型编码，生成该特殊符号对应的Token。

特殊符号Token的生成过程和常量Token的生成过程比较相似，但需要注意的是查保留字、特殊符号表的作用是为了确定单词的种类，这个信息填写在单词对应Token的“类别”这一项，当单词为保留字或特殊符号时，对应Token的“内容”这一项不填任何信息；而查标识符索引表和常量索引表的作用为了确定单词的内容，这个信息填写在单词对应Token的“内容”这一项，此时单词的类别已经确定了。

例 某程序片段如下：

```
VAR  sum, first, count: real;
 BEGIN
        sum:=first + count * 10
 END.
```

- 设置标识符表和常量表,词法分析程序扫描该程序段的字符序列，生成下列TOKEN序列：

1.($var,)　2. ($id, p_1)
3. ($comma,)　4. ($id, p_2)
5. ($comma,)　6. ($id, p_3)
7. ($colon,)　8. ($real,)
9. ($semi,)　10. ($return,)
11. ($begin,)　12. ($return,)
13. ($id, p_1)　14. ($assi,)
15. ($id, p_2)　16. ($plus,)
17. ($id, p_3)　18. ($mult,)
19. ($int, p_4)　20. ($return,)
21 . ($end,)　22. ($stop,)

常量表

p_4	10

标识符表

p_1	sum
p_2	first
p_3	count

2.5 注意的问题

- 名字的的实现方式
- 复合单词的识别
- 数的转换
- 向前看若干个字符
- 控制字符的处理
- 注释的处理

词法分析总结

- 这一章的内容需要掌握的是以下几点：
- 一个核心：词法分析器的设计
- 两个工具：正则表达式和自动机
- 三个转换算法：NFA到DFA，自动机的极小化，正则表达式和自动机的互相转换