编译原理

第三章 词法分析

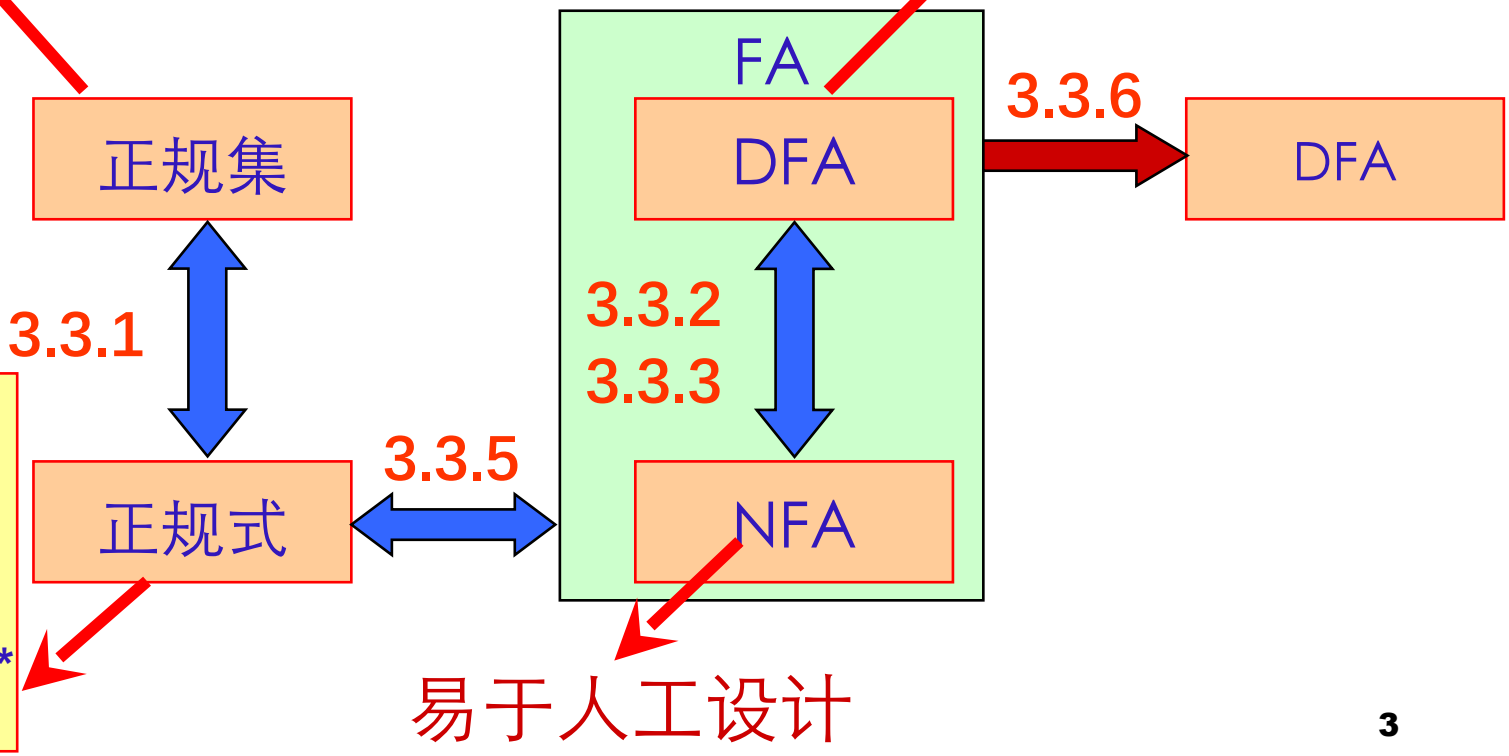
第三章 词法分析

- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

关系图

```
curState = 初态
GetChar();
while( stateTrans[curState][ch] 有定义 ){
    // 存在后继状态，读入、拼接
    Concat();
    // 转换入下一状态，读入下一字符
   curState= stateTrans[curState][ch];
    if cur_state 是终态 then 返回 strToken 中的单
    GetChar( );
}
```

DIM,IF, DO,STOP,END
number, name, age
125, 2169,
…



DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*

3.3.6 确定有限自动机的化简

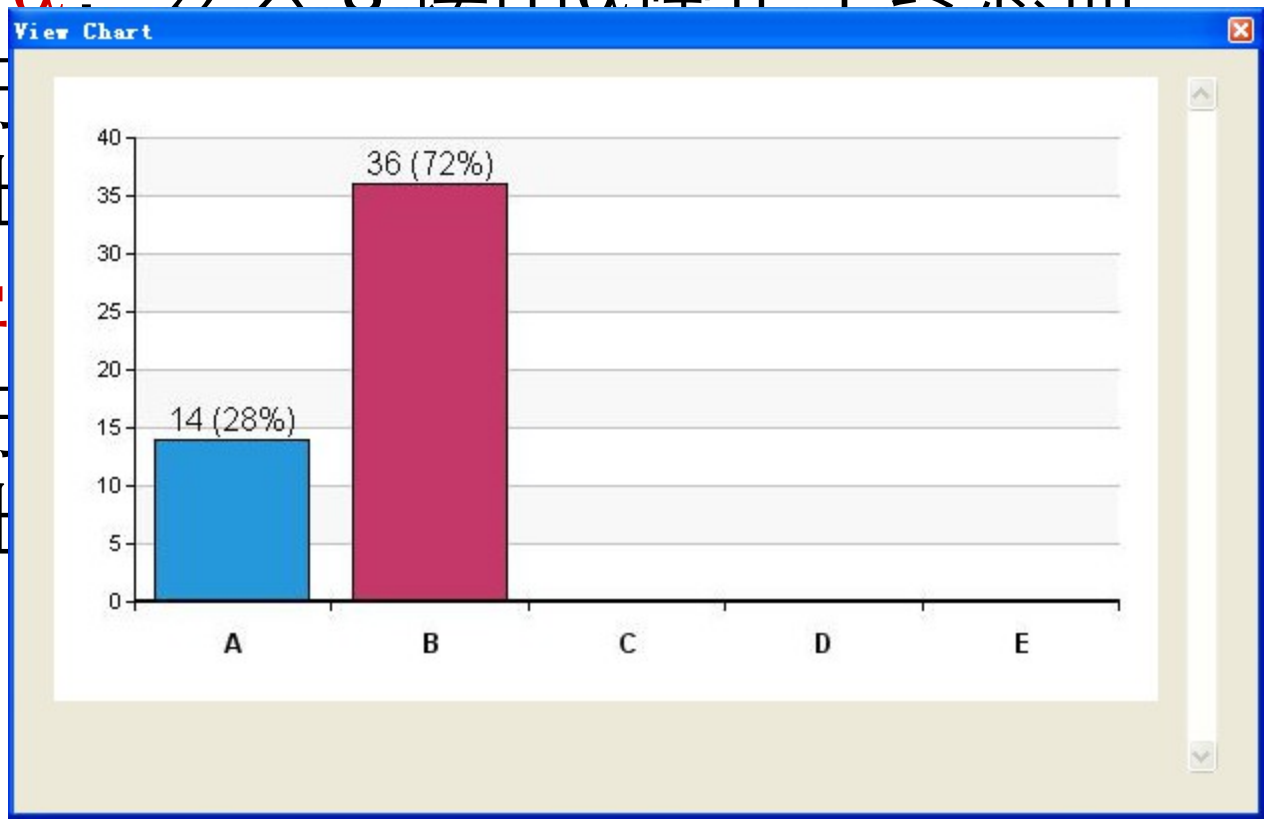
- 对 DFA M 的化简：寻找一个状态数比 M 少的 DFA M’，使得 L(M)=L(M’)
- 假设 s 和 t 为 M 的两个状态，称 s 和 t 等价：如果从状态 s 出发能读出某个字α而停止于终态，那么同样，从 t 出发也能读出α而停止于终态；反之亦然
- 两个状态不等价，则称它们是可区别的

测试：状态的可区分性

- 两个状态 s 和 t 是可区分的，是指 ( )

A. 对于任意字α，要么 s 读出α停止于终态而 t 读出α停止于
终态而 s 读出

B. 存在一个字
t 读出α停止于
终态而 s 读出



DFA M 最少化的基本思想

- 把 M 的状态集划分为一些不相交的子集，使得任何两个不同子集的状态是可区别的，而同一子集的任何两个状态是等价的。最后，让每个子集选出一个代表，同时消去其他状态

测试：初始划分

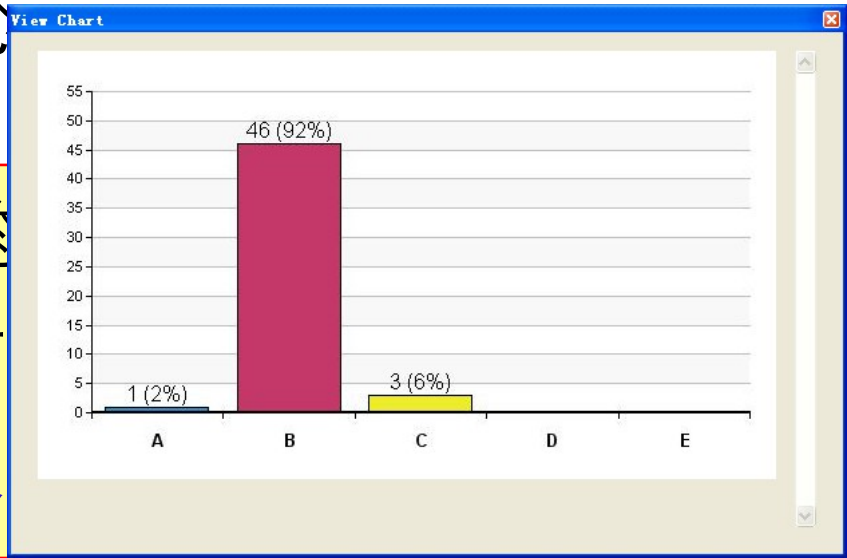
- 按照上述原则对 DFA 的状态集合 S 进行第一次划分，正确的分法是 ( )

A. 初态和非初态

B. 终态和非终态

C. 初态、终态、其他状态

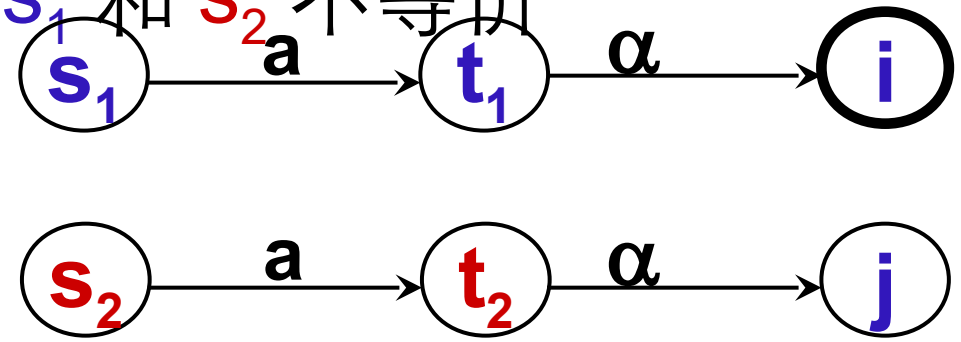
- 把 M 的状态
子集，使得
是可区别的
状态是等价

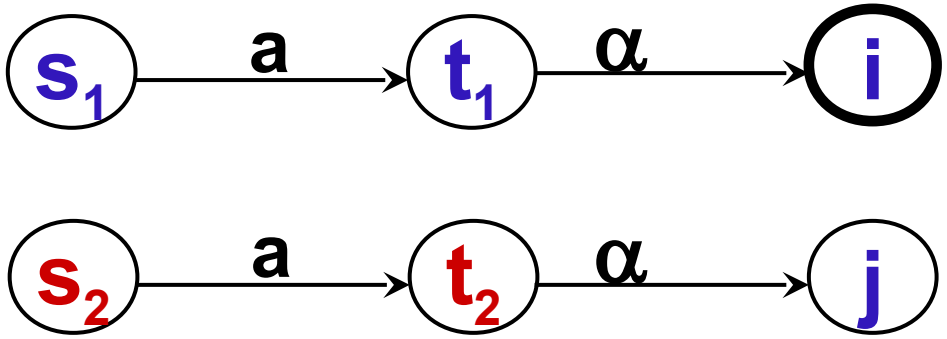


对 M 的状态集进行划分

- 首先，把 S 划分为终态和非终态两个子集，形成基本划分Π。
- 假定到某个时候，Π已含 m 个子集，记为 $\Pi=\{I^{(1)}, I^{(2)}, \ldots, I^{(m)}\}$，检查Π中的每个子集看是否能进一步划分：
  - 对某个 $I^{(i)}$，令 $I^{(i)}=\{s_1,s_2,\ldots,s_k\}$，若存在一个输入字符 a 使得 $I_a^{(i)}$ 不会包含在现行Π的某个子集 $I^{(j)}$ 中，则至少应把 $I^{(i)}$ 分为两个部分。

- 假定状态 s_1 和 s_2 经 a 弧分别到达 t_1 和 t_2
- t_1 和 t_2 属于现行Π中的两个不同子集
  - 说明有一个字α， t_1 读出α后到达终态，而 t_2 读出α后不能到达终态，或者反之
- 那么对于字 aα ， s_1 读出 aα 后到达终态，而 s_2 读出 aα 不能到达终态，或者反之
- 所以 s_1 和 s_2 不等价





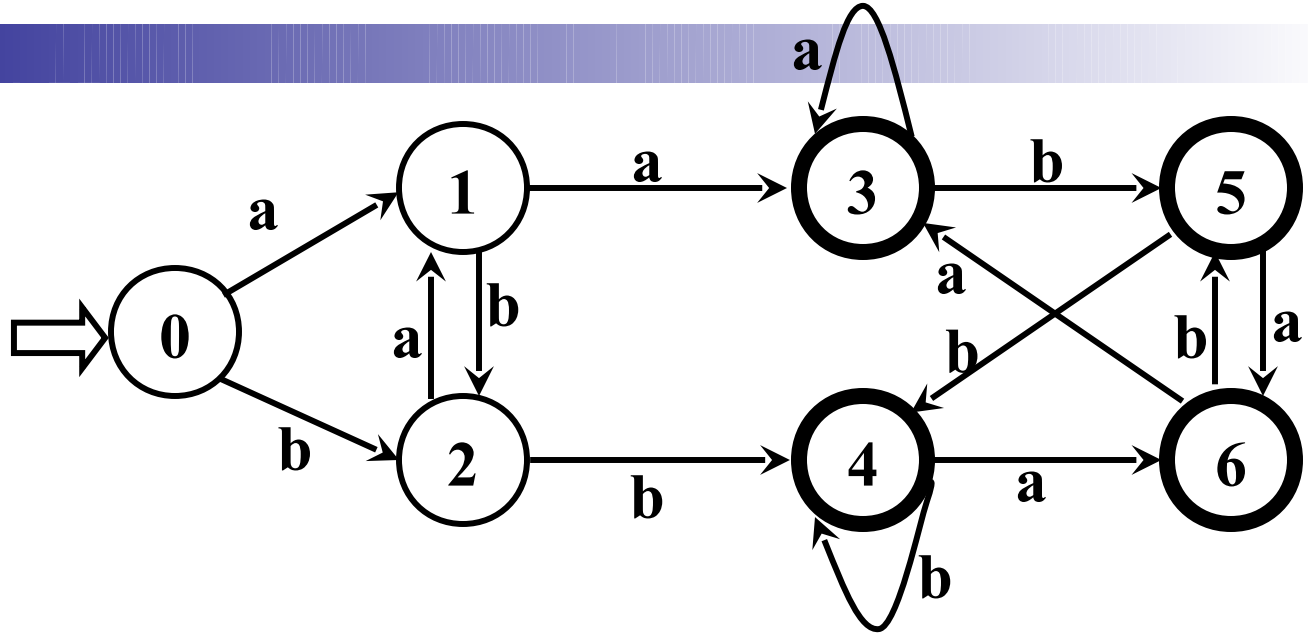
- 将 $I^{(i)}$ 分成两半，使得一半含有 s_1：

  $I^{(i1)}$={s|s∈$I^{(i)}$ 且 s 经 a 弧到达 t，

  且 t 与 t_1 属于现行Π中的同一子集 }

  另一半含有 s_2：$I^{(i2)}=I^{(i)}-I^{(i1)}$

- 一般地，对某个 a 和 $I^{(i)}$，若 $I_a^{(i)}$ 落入现行Π中 N 个不同子集，则应把 $I^{(i)}$ 划分成 N 个不相交的组，使得每个组 J 的 J_a 都落入的Π同一子集。这样构成新的划分。
- 重复上述过程，直到Π所含子集数不再增长。
- 对于上述最后划分Π中的每个子集，我们选取每个子集 I 中的一个状态代表其他状态，则可得到化简后的 DFA M’ 。
- 若 I 含有原来的初态，则其代表为新的初态，若 I 含有原来的终态，则其代表为新的终态。



$I^{(1)}=\{0, 1, 2\}$ $I^{(2)}=\{3, 4, 5, 6\}$

$I_a^{(1)}=\{1, 3\}$

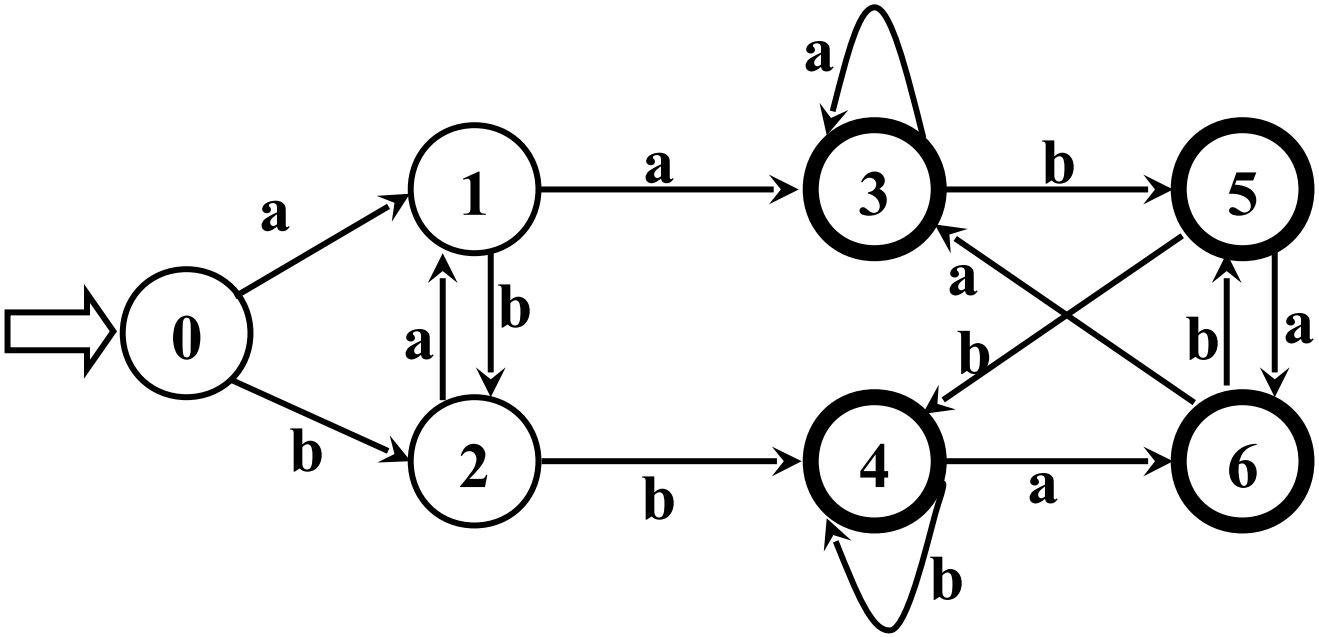
$I^{(11)}=\{0, 2\}$ $I^{(12)}=\{1\}$ $I^{(2)}=\{3, 4, 5, 6\}$

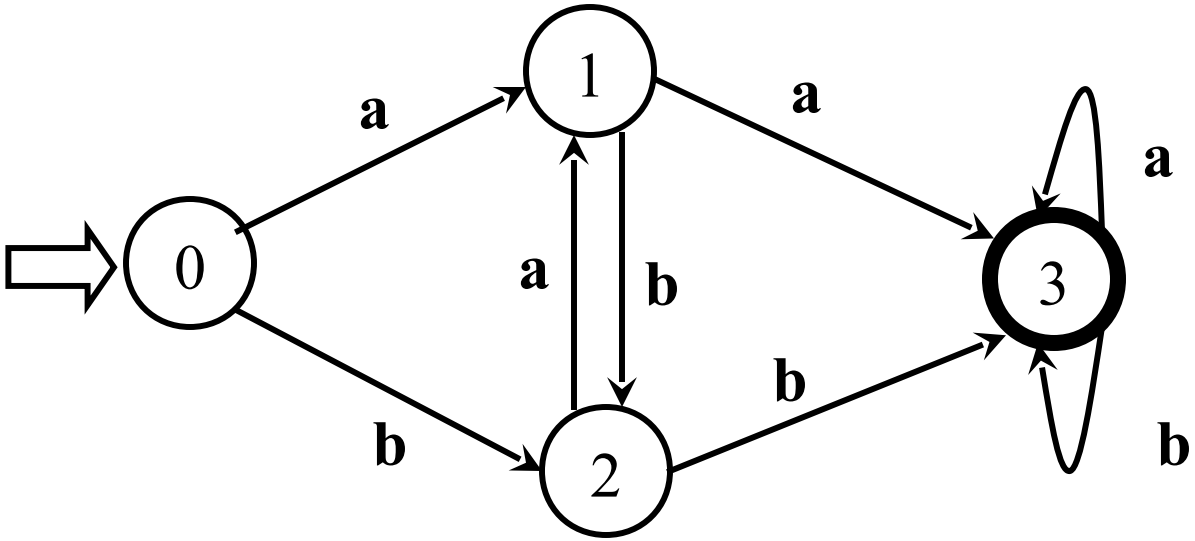
$I^{(11)}=\{0, 2\}$

$I_a^{(11)}=\{1\}$ $I_b^{(11)}=\{2, 4\}$

$I^{(111)}=\{0\}$ $I^{(112)}=\{2\}$ $I^{(12)}=\{1\}$ $I^{(2)}=\{3, 4, 5, 6\}$

$I_a^{(2)}=\{3, 6\}$ $I_b^{(2)}=\{4, 5\}$

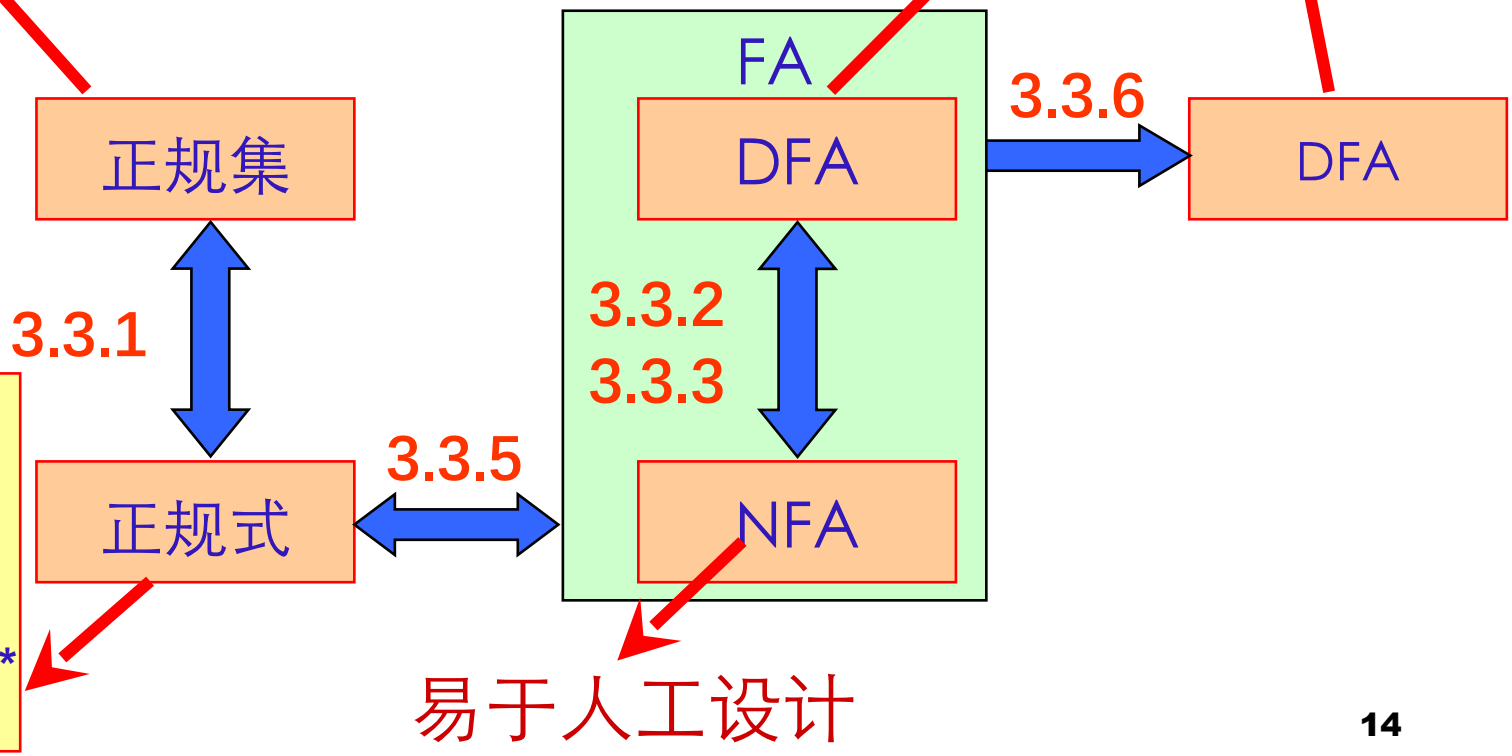

0
1
2
3
4
5
6
a
b


0
1
2
3
a
b

关系图

```
curState = 初态
GetChar();
while( stateTrans[curState][ch] 有定义 ){
    // 存在后继状态，读入、拼接
    Concat();
    // 转换入下一状态，读入下一字符
   curState= stateTrans[curState][ch];
    if cur_state 是终态 then 返回 strToken 中的单
    GetChar( );
}
```

DIM,IF, DO,STOP,END
number, name, age
125, 2169
…

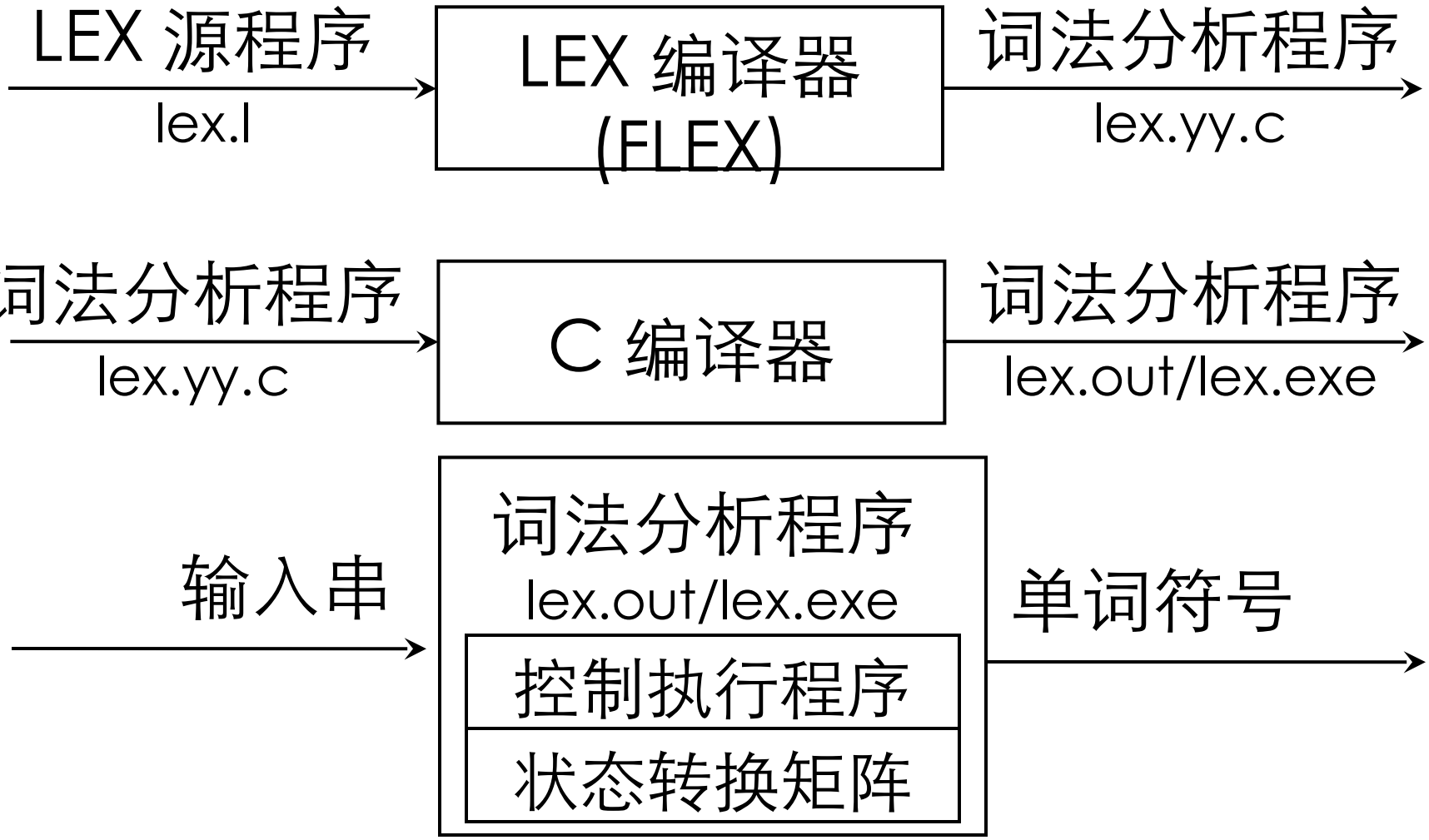


DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*

第三章 词法分析

- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

3.4 词法分析器的自动产生 --LEX



AUXILIARY DEFINITION

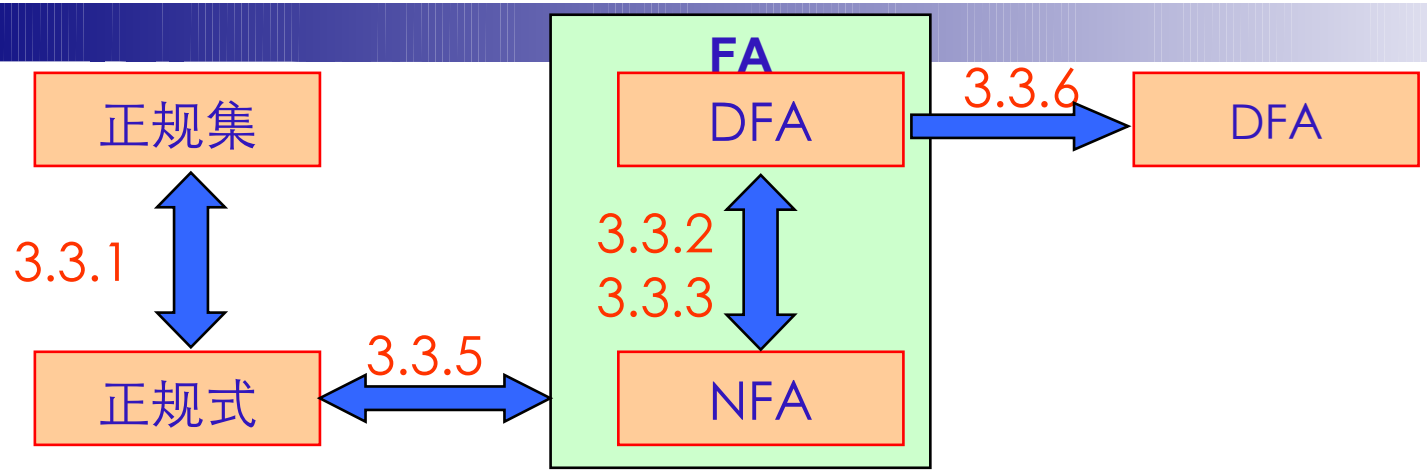
letter→A|B|...|Z

digit →0|1|...|9

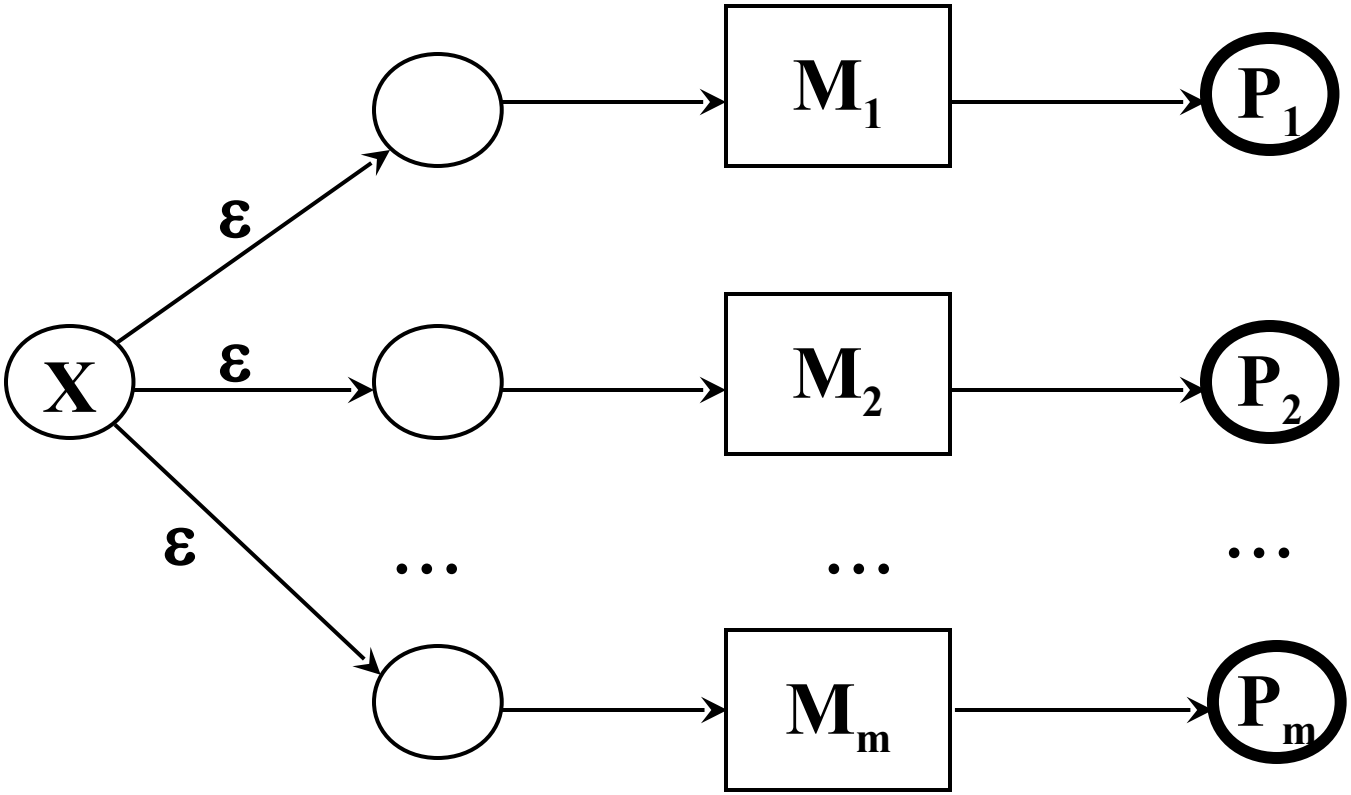
正规式

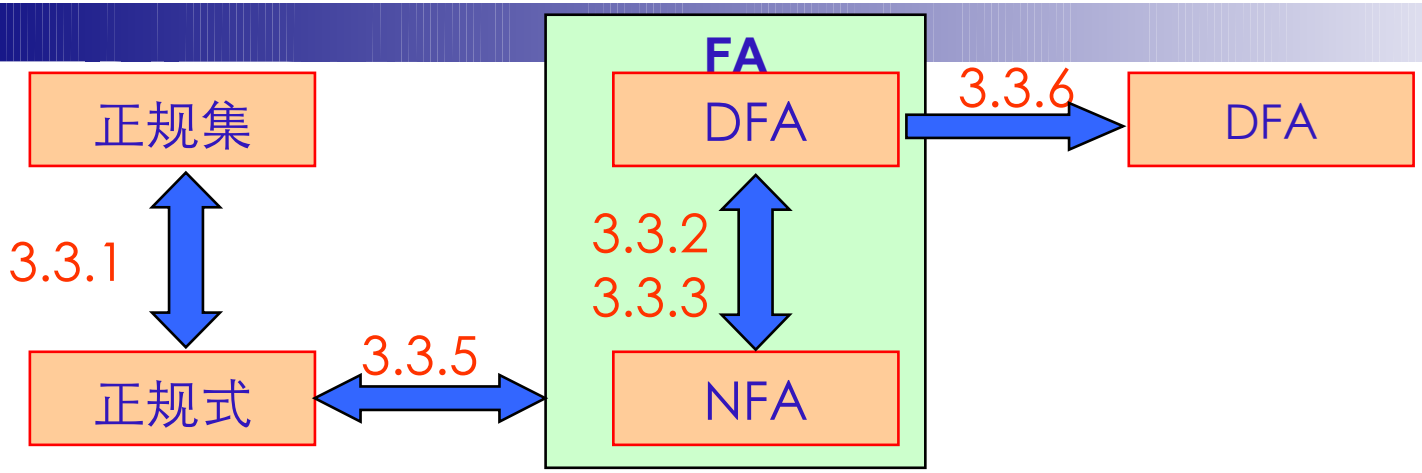
RECOGNITION RULES

1	DIM	{ RETURN (1,-) }
2	IF	{ RETURN (2,-) }
3	DO	{ RETURN (3,-) }
4	STOP	{ RETURN (4,-) }
5	END	{ RETURN (5,-) }
6	letter(letter|digit) *	{ RETURN (6, TOKEN) }
7	digit(digit)*	{ RETURN (7, DTB) }
8	=	{ RETURN (8, -) }
9	+	{ RETURN (9,-) }
10	*	{ RETURN (10,-) }
11	**	{ RETURN (11,-) }
12	,	{ RETURN (12,-) }
13	(	{ RETURN (13,-) }
14	)	{ RETURN (14,-) }



- LEX 的工作过程：
  - 首先，对每条识别规则 P_i 构造一个相应的非确定有限自动机 M_i；
  - 然后，引进一个新初态 X，通过ε弧，将这些自动机连接成一个新的 NFA；


X
ε
ε
ε
M1
M2
Mm
P1
P2
Pm
…
…
…



- LEX 的工作过程：
  - 首先，对每条识别规则 P_i 构造一个相应的非确定有限自动机 M_i；
  - 然后，引进一个新初态 X，通过ε弧，将这些自动机连接成一个新的 NFA；
  - 最后，把 M 确定化、最小化，生成该 DFA 的状态转换表和控制执行程序

LEX 参考资料

- Yacc 与 Lex 快速入门
  - http://www.ibm.com/developerworks/cn/linux/sdk/lex/index
  - UNIX, LINUX
- The Lex & Yacc Page
  - http://dinosaur.compilertools.net/
- Flex (The Fast Lexical Analyzer)
  - http://flex.sourceforge.net/
  - for Windows: http://gnuwin32.sourceforge.net/packages/flex.htm

实验 :LEX(FLEX) 的使用

- 用 LEX 生成 PL 语言的词法分析器
  - 词法规则
    - 编译实习教材，表 17.2.1 PL 语言单词符号及其种别值
  - 功能
    - 输入一个 PL 语言源程序文件 demo.pl
    - 输出一个文件 tokens.txt，该文件包括每一个单词及其种别枚举值，每行一个单词
  - 提交 5 个文件
    - PL 语言的 LEX 源程序： pl.lex
    - PL 语言词法分析程序 C 源程序： lex.yy.c
    - PL 语言词法分析程序的可执行文件： pl.out/pl.exe
    - PL 语言源程序文件： demo.pl
    - 词法分析及结果文件： tokens.txt

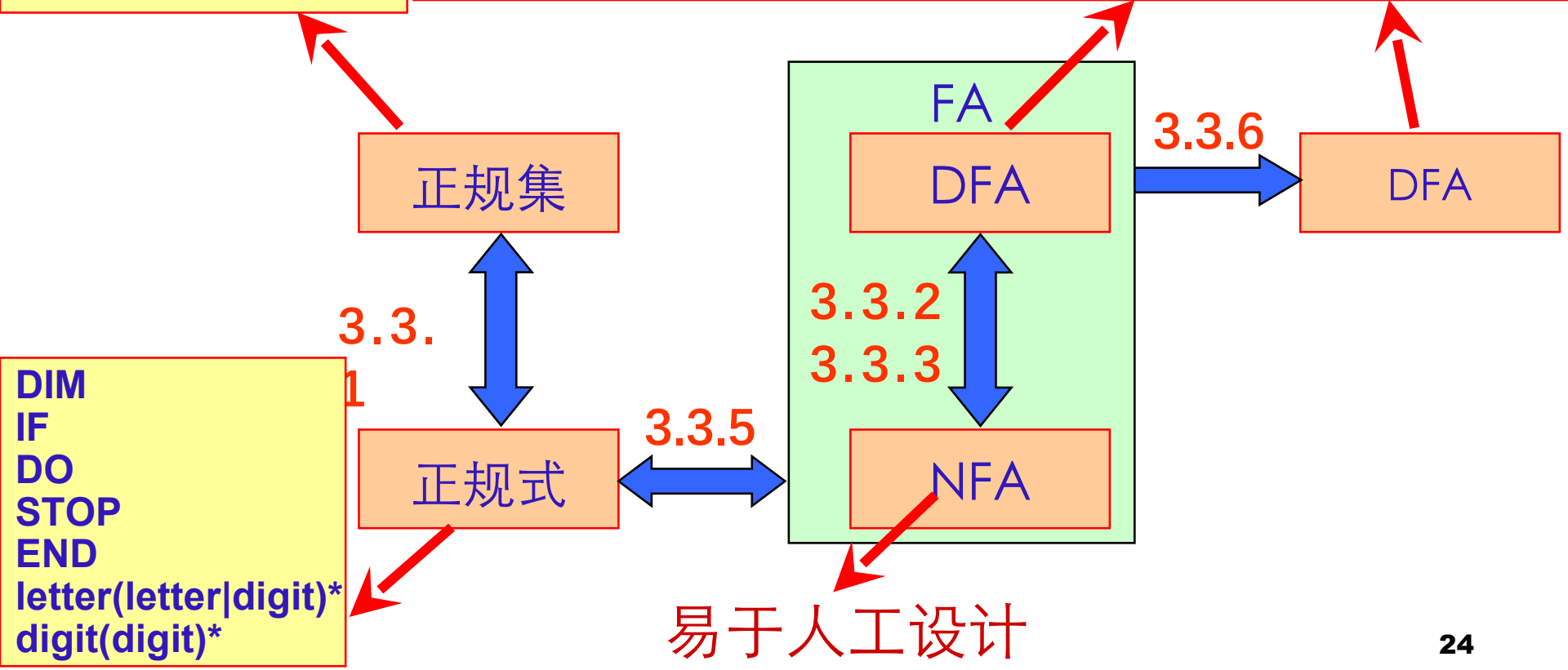
实验 :LEX(FLEX) 的使用

- 参考： Flex, version 2.5 文档
  - 阅读 (Flex for Windows 首页 .pdf)， 了解各压缩文件
  - 阅读 flex.pdf， 了解如何使用 Flex 及示例
    - 0.5 Some simple examples， scanner for a toy Pascal-like language

小结

```
curState = 初态
GetChar();
while( stateTrans[curState][ch] 有定义 ){
    // 存在后继状态，读入、拼接
    Concat();
    // 转换入下一状态，读入下一字符
   curState= stateTrans[curState][ch];
    if cur_state 是终态 then 返回 strToken 中的单
    GetChar( );
}
```

DIM,IF, DO,STOP,END
number, name, age
125, 2169
…



DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*