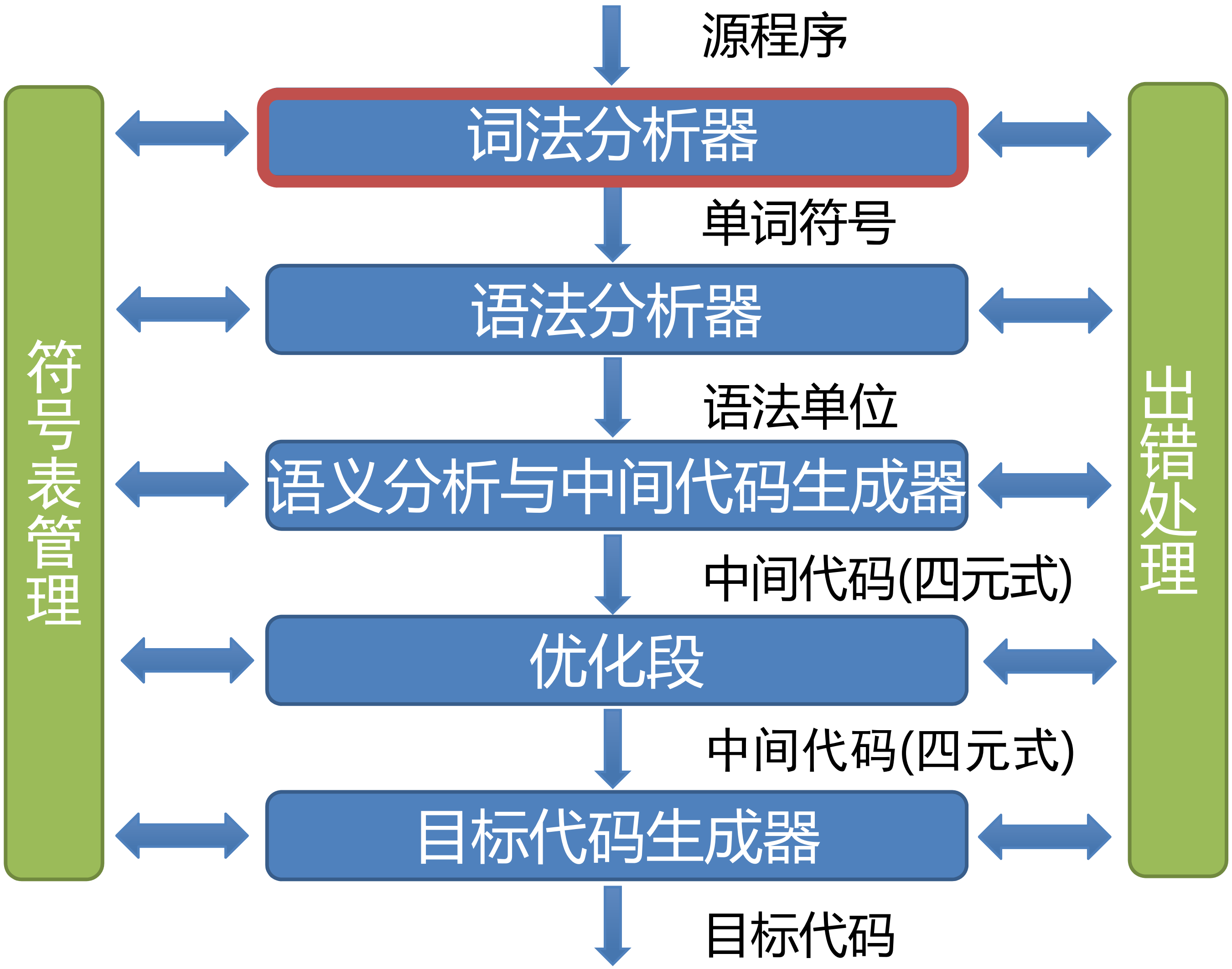
编译原理

词法规则的形式化

编译原理

词法规则的形式化
——回顾

编译程序总框



源程序
词法分析器
单词符号
语法分析器
语法单位
语义分析与中间代码生成器
中间代码(四元式)
优化段
中间代码(四元式)
目标代码生成器
目标代码
符号表管理
出错处理

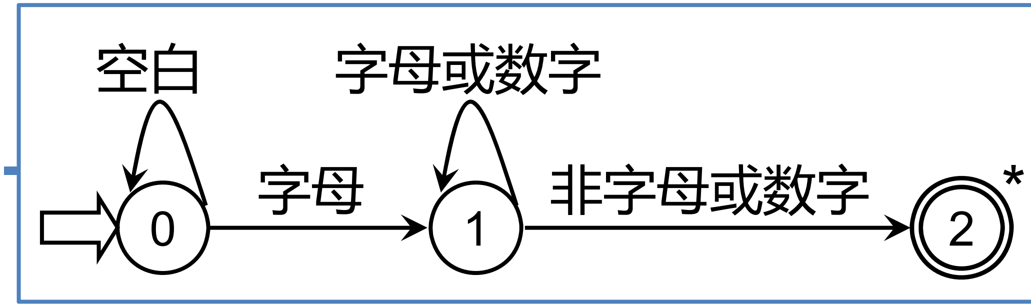
词法分析

- 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

词法分析器的设计

- 词法分析器的功能
  - 输入源程序、输出单词符号
- 词法分析器的设计
  - 给出程序设计语言的单词规范——单词表
  - 对照单词表设计识别该语言所有单词的状态转换图
  - 根据状态转换图编写词法分析程序

将状态图的代码



- 变量curState用于保存现有的状态
- 用二维数组表示状态图：stateTrans[state][ch]

```
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 curState是终态 then 返回strToken中的单词
    GetChar( );
}
```

词法分析

- 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

语法描述的几个基本概念

- 字母表：一个有穷字符集，记为∑
- 字母表中每个元素称为字符
- ∑上的字(也叫字符串) 是指由∑中的字符所构成的一个有穷序列
- 不包含任何字符的序列称为空字，记为ε
- 用∑*表示∑上的所有字的全体，包含空字ε
- 例如: 设 ∑={a， b}，则

$$\Sigma^*=\{\varepsilon,a,b,aa,ab,ba,bb,aaa,...\}$$

语法描述的几个基本概念

- Σ^*的子集U和V的连接（积）定义为

$$UV = \{\, \alpha\beta \mid \alpha \in U \;\&\; \beta \in V \,\}$$

- V自身的 n次积记为

$$V^n = \underbrace{V\,V \ldots V}_{n\text{个}}$$

- $V^0 = \{\varepsilon\}$
- V^*是V的闭包： $V^* = V^0 \cup V^1 \cup V^2 \cup V^3 \cup \ldots$
- V^+是V的正规闭包： $V^+ = V\,V^*$

编译原理

词法规则的形式化
——正规集和正规式

单词集合、正规集和正规式

- 程序设计语言的单词符号都是一些特殊的字符串
- 用正规集和正规表达式（简称正规式）来描述

单词符号	种别编码	助忆符	内码值
DIM	1	$DIM	-
IF	2	$IF	-
DO	3	$DO	-
STOP	4	$STOP	-
END	5	$END	-
标识符	6	$ID	内部字符串
常数(数)	7	$INT	标准二进制形式
=	8	$ASSIGN	-
+	9	$PLUS	-
*	10	$STAR	-
**	11	$POWER	-
,	12	$COMMA	-
(	13	$LPAR	-
)	14	$RPAR	-

正规式和正规集

- 正规集可以用正规式表示
- 正规式是表示正规集一种方法
- 一个字集合是正规集当且仅当它能用正规式表示

正规式和正规集的递归定义

- 对给定的字母表Σ

  1) ε 和$\varnothing$都是Σ上的正规式，它们所表示的正规集为$\{\varepsilon\}$和$\varnothing$;

测试：ε是什么?

A. 字符

B. 字 √

C. 正规式 √

测试：$\varnothing$是什么？

A. 集合 √

B. 字

C. 正规式 √

正规式和正规集的递归定义

- 对给定的字母表Σ

  1) ε 和∅都是Σ上的正规式，它们所表示的正规集为{ε}和∅;

  2) 任何a∈Σ， a是Σ上的正规式，它所表示的正规集为{a}；

测试：a ($a \in \Sigma$)是什么?

A.字符 √

B. 字 √

C. 正规式 √

正规式和正规集的递归定义

- 对给定的字母表Σ

  1) ε 和$\varnothing$都是Σ上的正规式，它们所表示的正规集为$\{\varepsilon\}$和$\varnothing$;

  2) 任何$a \in \Sigma$，a是Σ上的正规式，它所表示的正规集为$\{a\}$；

正规式和正规集的递归定义（续）

3) 假定e_1和e_2都是Σ上的正规式，它们所表示的正规集为$L(e_1)$和$L(e_2)$，则

i) $(e_1|e_2)$为正规式，它所表示的正规集为$L(e_1) \cup L(e_2)$

ii) $(e_1.e_2)$为正规式，它所表示的正规集为$L(e_1)L(e_2)$

iii) $(e_1)^*$为正规式，它所表示的正规集为$(L(e_1))^*$

仅由有限次使用上述三步骤而定义的表达式才是Σ上的正规式，仅由这些正规式表示的字集才是Σ上的正规集。

正规式的等价性

- 若两个正规式所表示的正规集相同，则称这两个正规式**等价**。如

$$b(ab)^{*}=(ba)^{*}b$$

$L(b(ab)^*)$	$L((ba)^*b)$
$= L(b)L((ab)^*)$	$= L((ba)^*) L(b)$
$= L(b) (L(ab))^*$	$= (L(ba))^*L(b)$
$= L(b) (L(a)L(b))^*$	$= (L(b)L(a))^* L(b)$
$=\{b\} \{ab\}^*$	$= \{ba\}^* \{b\}$
$= \{b\} \{\varepsilon, ab, abab, ababab, \ldots\}$	$= \{\varepsilon, ba, baba, bababa, \ldots\} \{b\}$
$= \{b, bab, babab, bababab, \ldots\}$	$= \{b, bab, babab, bababab, \ldots\}$

$$\because L(b(ab)^*)= L((ba)^*b)$$

$$\therefore b(ab)^*=(ba)^*b$$

作业

- 利用正规式与正规集的对应关系，证明 $(a^*b^*)^*=(a|b)^*$

正规式的性质

- 对正规式，下列等价成立
  - $e_1|e_2 = e_2|e_1$　　交换律
  - $e_1|(e_2|e_3) = (e_1|e_2)|e_3$　　结合律
  - $e_1(e_2e_3) = (e_1e_2)e_3$　　结合律
  - $e_1(e_2|e_3) = e_1e_2|e_1e_3$　　分配律
  - $(e_2|e_3)e_1 = e_2e_1|e_3e_1$ 分配律
  - $e\varepsilon = \varepsilon e = e$　　$e_1e_2 <> e_2e_1$

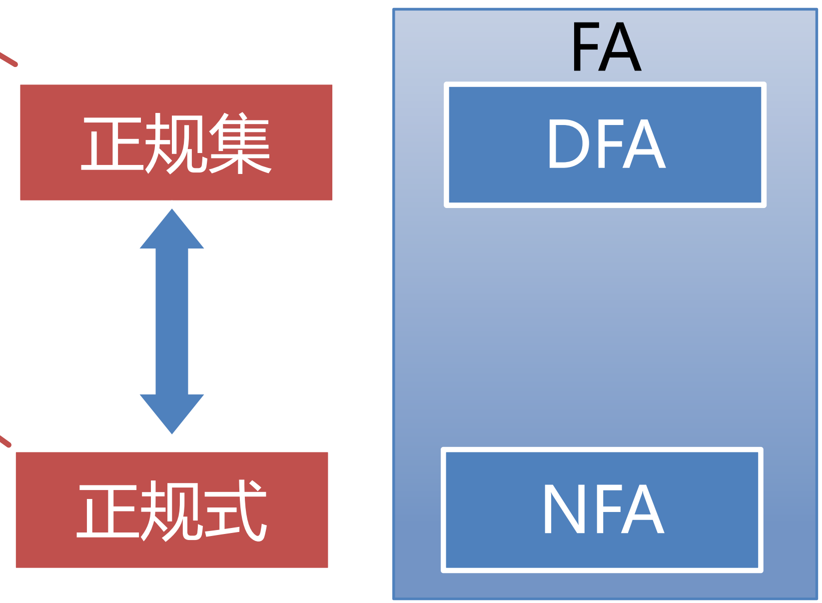
$L(e_1|e_2)$
$= L(e_1) \cup L(e_2)$
$= L(e_2) \cup L(e_1)$
$= L(e_2|e_1)$

小结：正规式和正规集

单词符号	种别编码	助忆符	内码值
DIM	1	$DIM	-
IF	2	$IF	-
DO	3	$DO	-
STOP	4	$STOP	-
END	5	$END	-
标识符	6	$ID	内部字符串
常数(数)	7	$INT	标准二进制形式
=	8	$ASSIGN	-
+	9	$PLUS	-
*	10	$STAR	-
**	11	$POWER	-
,	12	$COMMA	-
(	13	$LPAR	-
)	14	$RPAR	-

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

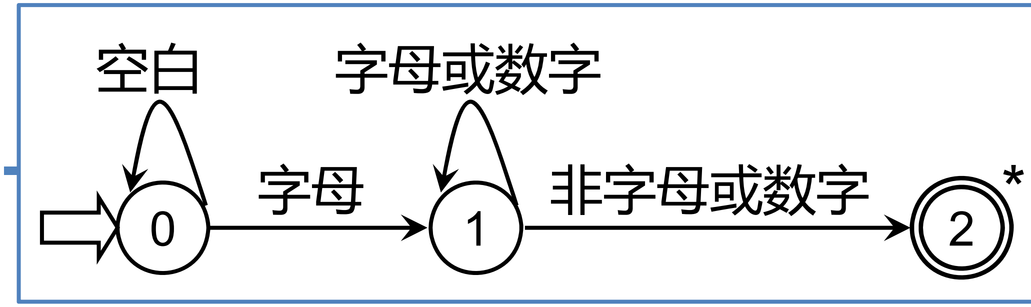
DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*



编译原理

词法规则的形式化
——确定有限自动机(DFA)

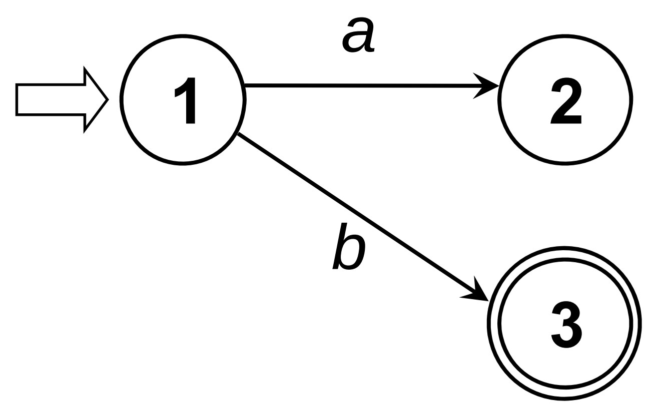
将状态图的代码



- 变量curState用于保存现有的状态
- 用二维数组表示状态图：stateTrans[state][ch]

```
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 curState是终态 then 返回strToken中的单词
    GetChar( );
}
```

确定有限自动机



- 对状态图进行形式化定义
- 确定有限自动机(Deterministic Finite Automata，DFA) M是一个五元式 M=(S, Σ, f, S_0, F)，其中:
  1. S: 有穷状态集
  2. Σ：输入字母表(有穷)
  3. f：状态转换函数，为S×Σ→S的单值部分映射，f(s，a)=s' 表示：当现行状态为s，输入字符为a时，将状态转换到下一状态s'，s'称为s的一个后继状态
  4. $S_0 \in S$是唯一的一个初态
  5. $F \subseteq S$：终态集(可空)

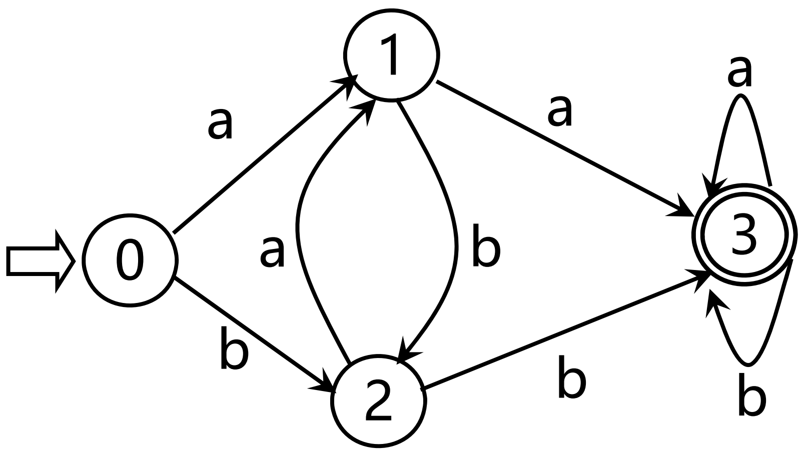
确定有限自动机

- DFA M=({0，1，2，3}，{a，b}，f，0，{3})，其中f定义如下：

f(0，a)=1　　f(0，b)=2
f(1，a)=3　　f(1，b)=2
f(2，a)=1　　f(2，b)=3
f(3，a)=3　　f(3，b)=3

状态转换矩阵

	a	b
0	1	2
1	3	2
2	1	3
3	3	3



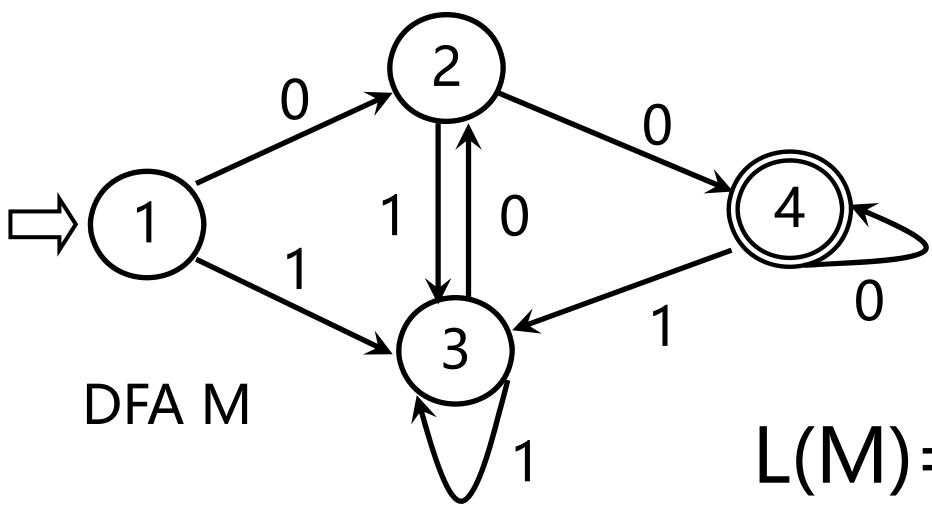
状态转换图

确定有限自动机

- DFA表示为状态转换图
  - 假定DFA M含有m个状态和n个输入字符
  - 对应的状态转换图含有m个状态结点，每个结点顶多含有n条箭弧射出，且每条箭弧用Σ上的不同的输入字符来作标记

确定有限自动机

- 对于Σ*中的任何字α，若存在一条从初态到某一终态的道路，且这条路上所有弧上的标记符连接成的字等于α，则称α为DFA M所识别(接收)
- DFA M所识别的字的全体记为L(M)



DFA M

L(M)＝{以00结尾的串}

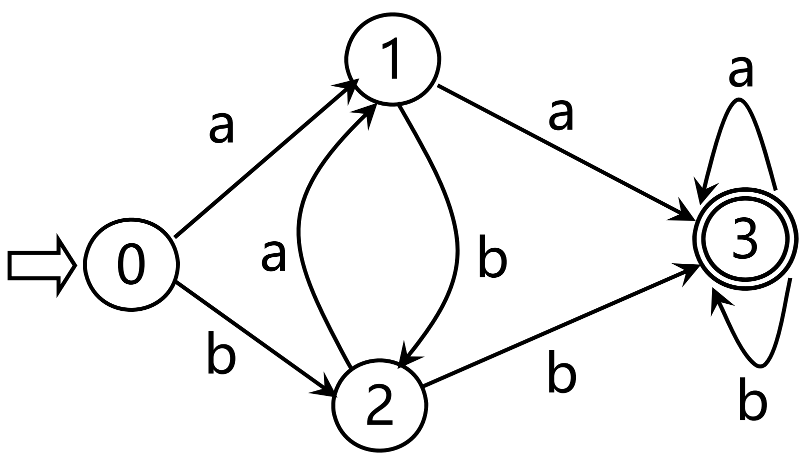
测试

- 图中DFA M识别的L(M) 是什么?

A. L(M)={以aa或bb开头的字}

B. L(M)={含aa或bb的字} √

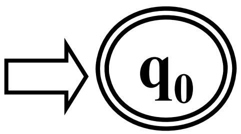
C. L(M)={以aa或bb结尾的字}



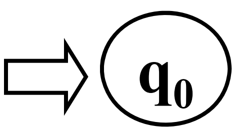
DFA M

测试

- 哪个DFA识别{ε}？

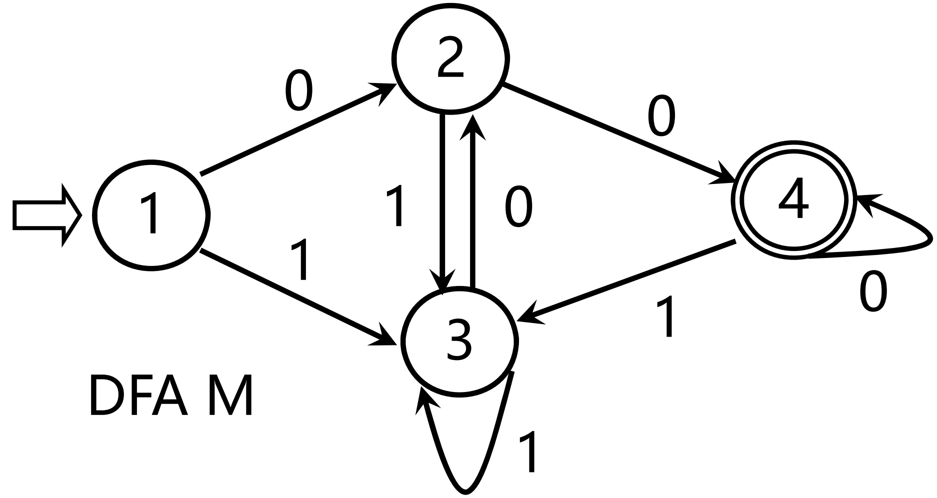


A



B

DFA的程序实现




```
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 curState是终态 then 返回strToken中的单词
    GetChar( );
}
```

单词符号	种别编码	助忆符	内码值
DIM	1	$DIM	-
IF	2	$IF	-
DO	3	$DO	

正规式、正

```
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 curState是终态 then 返回strToken中的单词
    GetChar( );
}
```

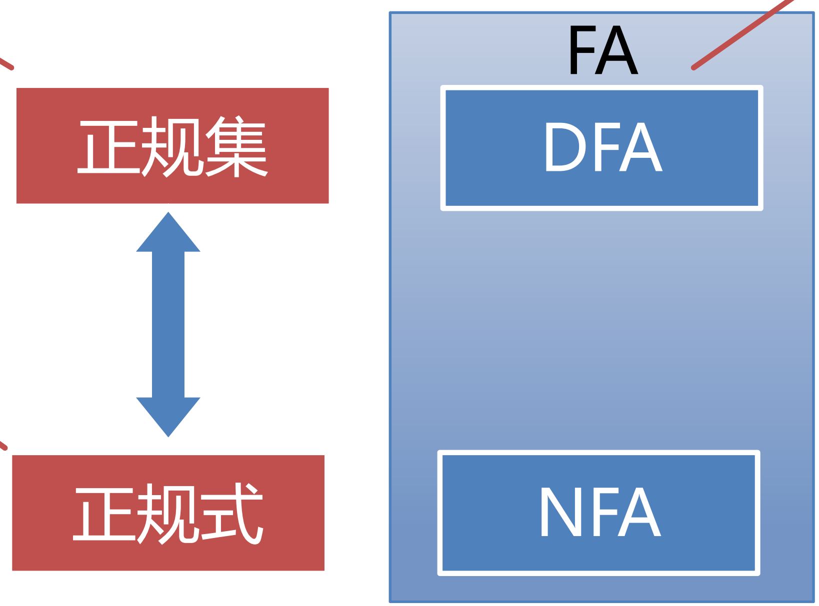
DIM,IF, DO,STOP,E

number, name, ag

125, 2169

…

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



编译原理

词法规则的形式化
——非确定有限自动机(NFA)

非确定有限自动机

- 1976年图灵奖：For their joint paper "Finite Automata and Their Decision Problem," which introduced the idea of nondeterministic machines, which has proved to be an enormously valuable concept. Their (Scott & Rabin) classic paper has been a continuous source of inspiration for subsequent work in this field.


M. O. Rabin*
D. Scott†

Finite Automata and Their Decision Problems‡

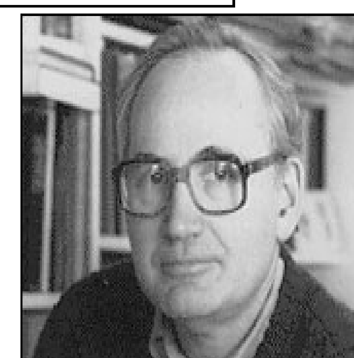
Abstract: Finite automata are considered in this paper as instruments for classifying finite tapes. Each one-tape automaton defines a set of tapes, a two-tape automaton defines a set of pairs of tapes, et cetera. The structure of the defined sets is studied. Various generalizations of the notion of an automaton are introduced and their relation to the classical automata is determined. Some decision problems concerning automata are shown to be solvable by effective algorithms; others turn out to be unsolvable by algorithms.

Introduction

114

IBM JOURNAL • APRIL 1959

Michael O. Rabin



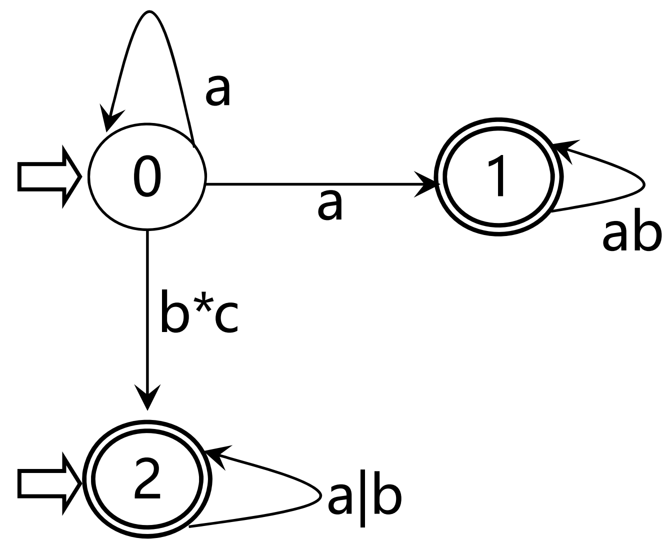
Dana S. Scott

非确定有限自动机

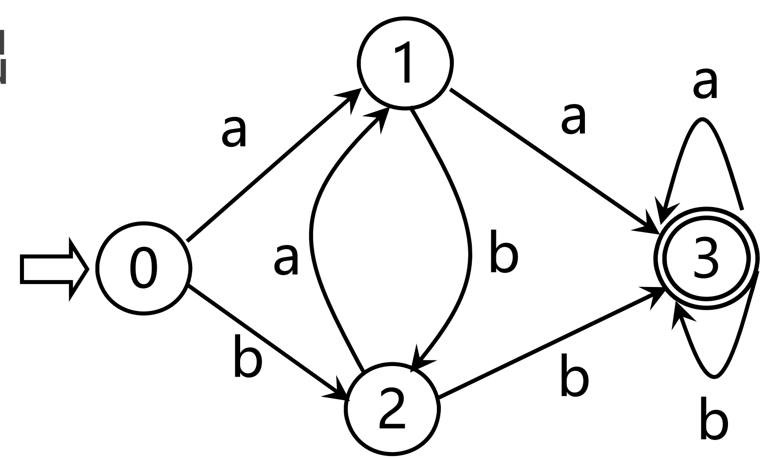
- 定义：一个非确定有限自动机 (Nondeterministic Finite Automata，NFA) M是一个五元式M=(S, Σ, f, S_0, F)，其中：
  1. S: 有穷状态集
  2. Σ： 输入字母表(有穷)
  3. f: 状态转换函数，为$S\times\Sigma^* \rightarrow 2^S$的部分映射
  4. $S_0 \subseteq S$是非空的初态集
  5. $F \subseteq S$： 终态集(可空)

非确定有限自动机

- 从状态图看NFA 和DFA的区别
  - NFA可以有多个初态
  - 弧上的标记可以是Σ*中的一个字（甚至可以是一个正规式），而不一定是单个字符
  - 同一个字可能出现在同状态射出的多条弧上
- DFA是NFA的特例



NFA M_1

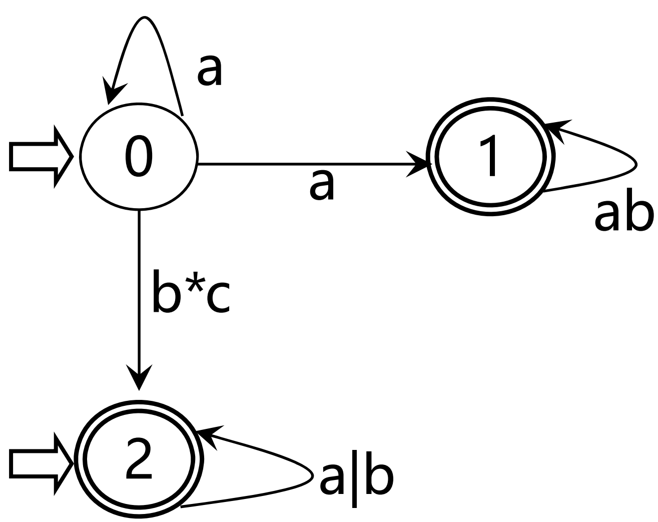


DFA M_2

非确定有限自动机

- 对于Σ*中的任何字α，若存在一条从初态到某一终态的道路，且这条路上所有弧上的标记字连接成的字等于α（忽略那些标记为ε的弧），则称α为NFA M所识别(接收)
- NFA M所识别的字的全体记为L(M)

NFA M



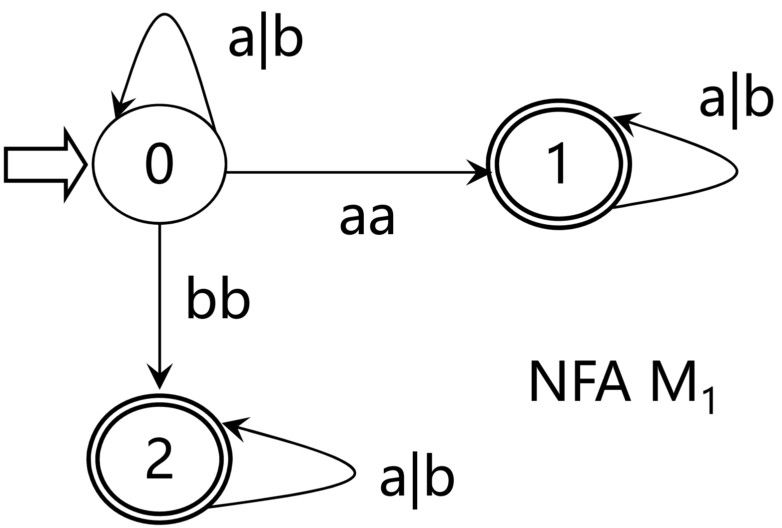
测试

- 图中NFA M_1识别的$L(M_1)$ 是什么?

A. $L(M_1)$={以aa或bb开头的字}

B. $L(M_1)$={含aa或bb的字}

C. $L(M_1)$={以aa或bb结尾的字}



NFA M_1

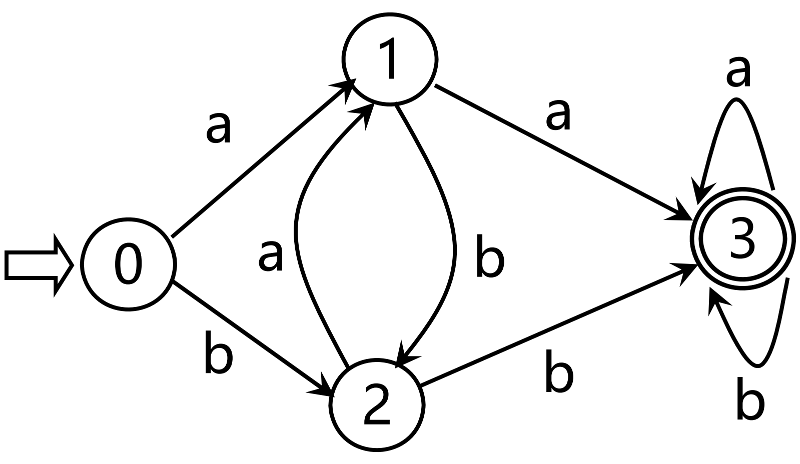
测试

- 图中NFA M_1识别的$L(M_1)$ 是什么?

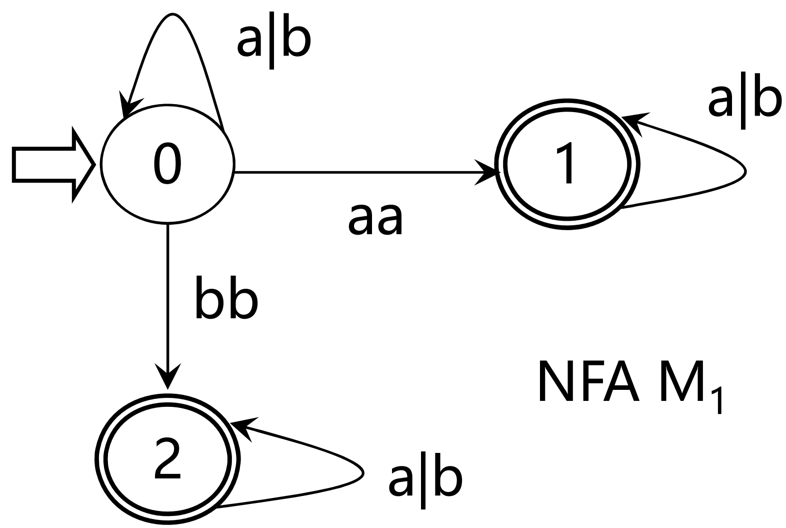
A. $L(M_1)$={以aa或bb开头的字}

B. $L(M_1)$={含aa或bb的字}

C. $L(M_1)$={以aa或bb结尾的字}



DFA M



NFA M_1

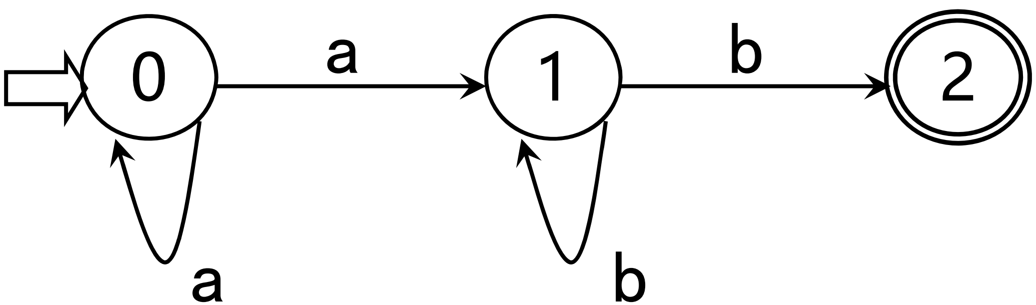
测试

- 图中NFA M_2识别的$L(M_2)$ 是什么?

A. $L(M_2)=\{ab^n \mid n \geq 1\}$

B. $L(M_2)=\{a^n b^n \mid n \geq 1\}$

C. $L(M_2)=\{a^m b^n \mid m，n \geq 1\}$



NFA M_2

DFA和NFA

- 定义：对于任何两个有限自动机M和M’，如果L(M)=L(M’)，则称M与M’等价
- 自动机理论中一个重要的结论：判定两个自动机等价性的算法是存在的
- 对于每个NFA M存在一个DFA M’，使得L(M)=L(M’)
- DFA与NFA识别能力相同!

正规式、正

单词符号	种别编码	助忆符	内码值
DIM	1	$DIM	-
IF	2	$IF	-

```
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 curState是终态 then 返回strToken中的单词
    GetChar( );
}
```

DIM,IF, DO,STOP,E
number, name, ag
125, 2169
…

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

正规集

正规式

FA

DFA

NFA

易于人工设计

小结

- 正规式和正规集
- 确定有限自动机和非确定有限自动机