

第三章 词法分析

Yinliang Zhao
Xi'an Jiaotong University
2009

本章内容

- 有限自动机
- 词法分析器的设计与实现
- 词法分析器的自动生成

2

3.1 有限自动机

- 确定有限自动机
- 非确定有限自动机
- 正规文法与确定自动有限自动机的等价性
- 正规式与确定自动有限自动机的等价性
- 确定自动有限自动机的化简

3

3.1.1 正规式与正规集

- Regular expressions represents patterns of strings of characters
- The language generated by the regular expression r is the set of strings that is matches, and is written as $L(r)$
- 字母表 Σ ,
- 元字符（元符号）, metacharacters, metasymbols
- 托字符, escape character

4

基本正规式

- 任何 $a \in \Sigma$, a 是 Σ 上的一个正规式, 它所表示的正规集为 $L(a) = \{a\}$;
- ϵ 和 ϕ 都是字母表 Σ 上的正规式, 它们所表示的正规集分别为 $\{\epsilon\}$ 和 ϕ ;

5

正规式的运算

- r 和 s 是 Σ 上的正规式, 那么:
 - ✦ 选择运算 $|$: $r|s$ 是正规式且 $L(r|s) = L(r) \cup L(s)$;
 - ✦ 连接运算 \bullet : rs 是正规式且 $L(r\bullet s) = L(r)\bullet L(s)$;
 - ✦ 重复运算 $*$: r^* 是正规式且 $L(r^*) = L(r)^*$;
- 运算的优先级和括号使用

• 通过有限次使用上述三个步骤而得到的表达式是 Σ 上的正规式, 由这些正规式所表示的符号串的集合是 Σ 上的正规集.

6

运算的优先级

- $a|b^*$
 - ◆ $(a|b)^*$
 - ◆ $a|(b^*)$
- 按惯例*的优先级较高
- 可用括号改变运算次序
 - ◆ $(a|b)c$
 - ◆ $(a|bb)^*$

- $a|bc^*$
 - ◆ $a|(b(c^*))$
- $ab|c^*d$
 - ◆ $(ab)|((c^*)d)$
- 按惯例*的优先级最高, 连接次之, |最低

7

举例

- 已知 $\Sigma=\{a, b\}$, 那么 ba^* , $a(a|b)^*$ 和 $(a|b)^*(aa|bb)(a|b)^*$ 都是 Σ 上的正规式.

a 是(2), $p(4)$ $\Rightarrow a^*$ 是(3闭包), b 是(2) $\Rightarrow ba^*$ 是(3连接)

正规集: Σ 上所有以 b 为先后跟0到多个 a 的字符串.

a 是(2), b 是(2) $\Rightarrow a|b$ 是(3或) $\Rightarrow (a|b)^*$ (3闭包)是: a 是(2) $\Rightarrow a(a|b)^*$ 是(3连接)

正规集: Σ 上所有以 a 为首的字符串.

a 是(2), b 是(2) $\Rightarrow a|b$ 是(3或) $\Rightarrow (a|b)^*$ (3闭包)是: a 是(2) $\Rightarrow aa$ 是(3连接); b 是(2) $\Rightarrow bb$ 是(3连接) $\Rightarrow (a|b)^*(aa|bb)(a|b)^*$ 是(3连接)

正规集: Σ 上所有含有两个相继 a 或两个相继 b 的字符串.

8

举例

令字母表 $\Sigma=\{a,b\}$, 则:

- 正规式 $a|b$ 表示集合 $\{a, b\}$
- 正规式 $(a|b)(a|b)$ 表示集合 $\{aa, ab, ba, bb\}$
- 正规式 a^+ 表示由1到多个 a 构成的字符串的集合
- 正规式 $(a|b)^*$ 表示由0到多个 a 或 b 构成的字符串的集合
- 正规式 $a|a^*b$ 表示字符串 a 以及由0到多个 a 后跟一个 b 构成的字符串的集合

9

举例

对于定义在字母表 $\{a,b\}$ 上的语言, 分别给出下述正规集的正规式:

- a) 所有以 a 为开头和结尾的字符串 $a|a(a|b)^*a$
- b) 第奇数位都是 a 的所有字符串 $(a(a|b))^*a|(a|b)a^*$
- c) 不具有任何两个连续 a 的所有字符串

若两个正规式 U 和 V 所表示的正规集相同, 则称为二者等价, 记为 $U=V$.

例如, $b(ab)^*=(ba)^*b$
 $(a|b)^*=(a^*b^*)^*$

正规式的命名

■ $(0|1|2|\dots|9)(0|1|2|\dots|9)^*$

↓

■ $\langle digit \rangle \langle digit \rangle^*$

■ 其中, $\langle digit \rangle ::= 0|1|2|\dots|9$

- 称, $\langle digit \rangle ::= 0|1|2|\dots|9$
- 为 $\langle digit \rangle$ 的正规定义,
- **regular definition of the name $\langle digit \rangle$**

11

正规定义

字母表 Σ 上的正规定义形如:

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

各个 d_i 的名字不同, 每个 r_i 是字母表 $\Sigma \cup \{d_1, d_2, \dots, d_{i-1}\}$ 上的正规式, 其中 $i=1, 2, \dots, n$.

$\langle digit \rangle \rightarrow 0|1|2|3|4|5|6|7|8|9$

$\langle digits \rangle \rightarrow \langle digit \rangle^*$

$\langle num \rangle \rightarrow [+|-]\langle digits \rangle \langle digits \rangle [E|+|-]\langle digits \rangle$

12

正规式的代数性质规定义

公理	描述
$U V = V U$	是可交换的
$U (V W) = (U V) W$	是可结合的
$(UV)W = U(VW)$	•是可结合的
$U(V W) = UV UW$ $(U V)W = UW VW$	•对 可分配
$\epsilon U = U$ $U \epsilon = U$	ϵ 是•的衡等元素
$U^* = U^*$	*是幂等的

* 大写字母为正规式

* | 正规式的或运算

* • 正规式的连接运算

* * 正规式的闭包运算

3.1.2 Finite-State Automata

- Alphabet Σ
- Set of states with initial and accept states
- Transitions between states, labeled with letters

$(0|1)^*. (0|1)^*$

初态

终态

Automaton Accepting String

- Conceptually, run string through automaton
- Have current state and current letter in string
- Start with start state and first letter in string
- At each step, match current letter against a transition whose label is same as letter
- Continue until reach end of string or match fails
- If end in accept state, automaton accepts string
- Language of automaton is set of strings it accepts

例

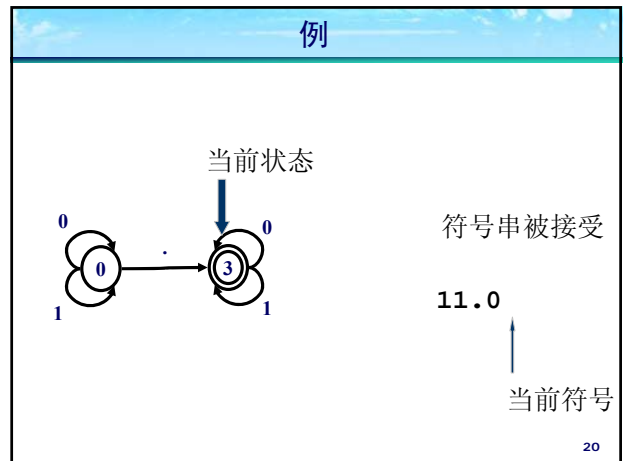
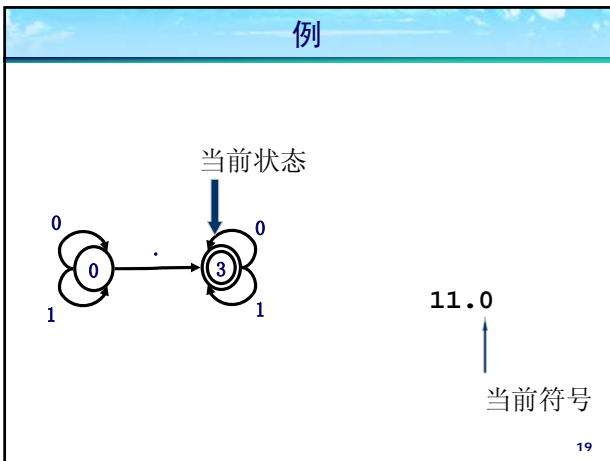
11.0
↑
当前符号

例

11.0
↑
当前符号

例

11.0
↑
当前符号



确定有限自动机(DFA)

一个**确定有限自动机M**是一个五元式 $M = (S, \Sigma, \delta, s_0, F)$, 其中,

- ✦ S是一个有限的**状态集合**;
- ✦ Σ 是有**穷字母表**, 它的每个元素称为一个**输入字符**;
- ✦ δ 是一个从 $S \times \Sigma$ 至 S 的单值部分映射。
 $\delta(s, a) = s'$ 意味着: 给定状态s和输入字符a返回状态s';
- ✦ $s_0 \in S$ 是唯一的**初态**;
- ✦ $F \subseteq S$ 是一个**终态集**(可空).

$M = (\{0, 1, 2, 3\}, \{a, b\}, \delta, 0, \{3\})$

21

状态矩阵

一个DFA可用一个矩阵表示, 该矩阵的行表示状态, 列表示输入字符, 矩阵元素表示 $\delta(s, a)$ 的值(转换的状态). 这个矩阵称为**状态转换矩阵**.

例: 有DFA $M = (\{0, 1, 2, 3\}, \{a, b\}, \delta, 0, \{3\})$

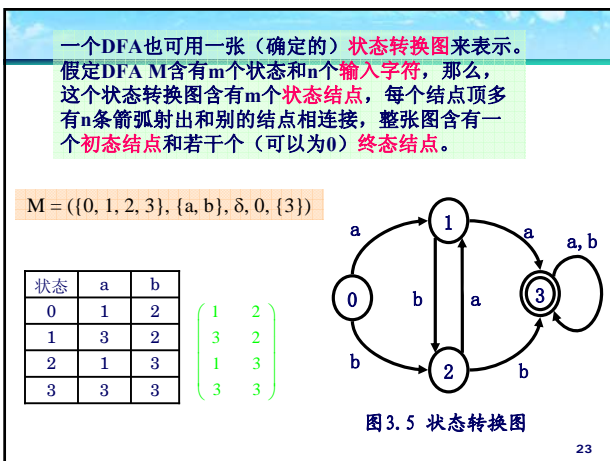
其中 δ 为:

$\delta(0, a) = 1$	$\delta(0, b) = 2$
$\delta(1, a) = 3$	$\delta(1, b) = 2$
$\delta(2, a) = 1$	$\delta(2, b) = 3$
$\delta(3, a) = 3$	$\delta(3, b) = 3$

相应的**状态转换矩阵**如下表: 其中

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

22



给定DFA $M = (S, \Sigma, \delta, s_0, F)$, 对于 Σ^* 中的任意符号串 α , 若存在一条从初态结点到终态结点的路径, 且这条路径上的所有弧的标记依次连接成一个符号串, 同时这个符号串等于 α 则称 α 可为DFA M所**识别**(或**接受**)。

给定 $M = (\{0, 1, 2, 3\}, \{a, b\}, \delta, 0, \{3\})$ 有:

- $S = \{0, 1, 2, 3\}$
- $\Sigma = \{a, b\}$
- $\Sigma^* = \{ \epsilon, a, b, aa, ab, ba, bb, aaa, aab, \dots \}$

例如: 对于 Σ^* 上的符号串:

abaa 识别

所有为M所识别的 Σ^* 中的符号串组成的集合称为M的语言, 记为L(M).

baba 不识别

24

正规式和DFA

如果一个DFA M 的输入字母表为 Σ (M 称为 Σ 上的一个DFA), 则 Σ 上的一个子集 $V \subseteq \Sigma^*$ 是正规的, 当且仅当存在 Σ 上的DFA M 使得 $V=L(M)$.

$(0|1|2|3|4|5|6|7|8|9) (0|1|2|3|4|5|6|7|8|9)^*$

$M = (\{s_0, int\}, \{0,1,2,3,4,5,6,7,8,9\}, \delta, s_0, \{int\})$

δ :

	int	int	int	int	int	int	int	int	int	int
int	int	int	int	int	int	int	int	int	int	int

25

$\langle digit \rangle \rightarrow 0|1|2|3|4|5|6|7|8|9$

$\langle digits \rangle \rightarrow \langle digit \rangle^+$

$\langle num \rangle \rightarrow [+|-]\langle digits \rangle \cdot \langle digits \rangle [E][+|-]\langle digits \rangle$

课堂练习: 写出该状态转换图所对应的DFA

123E45 123.0 -123.0 +123.45 -123.45E-67
123.0 -123.0 +123.45 -123.45E-67

26

接受正规式 $ab^+|ab^*|b^*$ 的正规集的正则集的DFA如下所示:

27

例3-14: 串中只有一个b被如下所示的DFA接受:

例3-15: 包含最多一个b的串被如下所示的DFA接受:

28

3.1.3 非确定有限自动机(NFA)

DFA $M = (S, \Sigma, \delta, s_0, F)$ 的确定性表现在 δ 映射是一个单值函数. 即对于任何状态 $s \in S$, 和输入符号 $a \in \Sigma$, $\delta(s, a)$ 唯一地确定了下一个状态.

当 $\delta(s, a)$ 的值不唯一时...

29

3.1.3 非确定有限自动机(NFA)

一个非确定有限自动机 M 是一个五元式

$M = (S, \Sigma, \delta, S_0, F)$, 其中,

- S 是一个有限的状态集合;
- Σ 是有穷字母表, 它的每个元素称为一个输入字符;
- δ 是一个从 $S \times \Sigma^*$ 至 2^S 的映射. $\delta(s, \alpha) = T$ 意味着: 给定状态 s 和输入符号串 α 返回状态集合 T , 其中 $T \subseteq S$;
- $S_0 \subseteq S$ 是非空初态集;
- $F \subseteq S$ 是一个终态集(可空).

同DFA

每个箭弧上的输入是一个符号串(含 ϵ)

状态的迁移不再对于一个输入符号串是唯一的;

开始状态可以多个

30

NFA接受输入串 α , 当且仅当转换图中存在从开始状态到终态的一条路径,使得该路径中每条边的输入符号依次连接成为 α

31

一个NFA也可用一张(确定的)状态转换图来表示。
假定NFA M 含有 n 个状态, 每个结点可射出若干个箭弧与别的结点相连, 每个弧用 Σ^* 中的一个字作为输入字。整张图至少含有一个初态结点以及0到多个终态结点。

$M = (\{0, 1, 2, 3\}, \{a, b\}, \delta, \{0\}, \{3\})$

状态	a	b
0	{0,1}	{0}
1	ϕ	{2}
2	ϕ	{3}
3	ϕ	ϕ

图 状态转换图

对应的正规式为 $(a|b)^*abb$

32

NFA接受输入串 α , 当且仅当转换图中存在从开始状态到终态的一条路径,使得该路径中每条边的输入字依次连接成为 α 。

对应的正规式为 $aa^*|bb^*$

图 状态转换图

33

练习: 考虑以下NFA通过怎样的转换接受串 $acab$:

→1 \xrightarrow{a} 2 \xrightarrow{a} 3 \xrightarrow{a} 4 \xrightarrow{a} 7 \xrightarrow{a} 2 \xrightarrow{c} 5 \xrightarrow{c} 6 \xrightarrow{c} 7 \xrightarrow{a} 8 \xrightarrow{a} 9 \xrightarrow{b} 10

34

NFA接受输入串 α , 当且仅当转换图中存在从开始状态到终态的一条路径,使得该路径中每条边的输入字依次连接成为 α

可接受的输入串为任意包含连续两个a或连续两个b的字符串
 $\Sigma = \{a, b\}$

35

定理: 对于每个NFA M 存在一个DFA M' 使得 $L(M) = L(M')$

证明: 假定NFA $M = (S, \Sigma, \delta, S_0, F)$, 对 M 的状态转换图进行改变, 构造等价的DFA M'

第一步: 由NFA M 构造等价的NFA M'

第二步: 由NFA M' 构造等价的DFA M''

36

① 引进新的初态结点 x 和终态结点 y , 从 x 到 S_0 中每一状态结点连一条输入符号为 ϵ 的箭弧; 从 F 中每一状态结点连一条输入符号为 ϵ 的箭弧;

② 重复进行如下图所示转换直到每条箭弧上的输入符号串的长度不大于 1 为止.

将最终得到的NFA记为 M' , 且有 $L(M)=L(M')$.

$\epsilon_CLOSURE(I)$ 的定义

DFA是NFA的特例, 可以采用子集法将NFA确定化.

假定 I 是NFA M 的状态集的一个子集, 我们定义 $\epsilon_CLOSURE(I)$ 为:

- 若 $s \in I$, 则 $s \in \epsilon_CLOSURE(I)$;
- 若 $s \in I$, 那么从 s 出发经过任意条 ϵ 弧而能到达的任何状态 s' 都属于 $\epsilon_CLOSURE(I)$.

状态集 $\epsilon_CLOSURE(I)$ 称为 I 的 ϵ -闭包.

求 $\epsilon_CLOSURE(I)$ 的例子

$\epsilon_CLOSURE(\{x\}) = \{x, 0, 1, 3\}$ $\epsilon_CLOSURE(\{0\}) = \{0, 1, 3\}$
 $\epsilon_CLOSURE(\{2\}) = \{2, y\}$ $\epsilon_CLOSURE(\{4\}) = \{4, y\}$
 $\epsilon_CLOSURE(\{x, 0, 2\}) = \{x, 0, 1, 3, 2, y\}$
 $\epsilon_CLOSURE(\{1, 3, 4\}) = \{1, 3, 4, y\}$

I_a 定义

给定NFA $M = (S, \Sigma, \delta, S_0, F)$, 假定 $I \subseteq S$, $a \in \Sigma$, 定义 $I_a = \epsilon_CLOSURE(J)$, 其中, J 是那些可从 I 中的某一状态结点出发经过一条 a 弧而到达的状态结点的全体.

$\{1\}_a = \epsilon_CLOSURE(\{2\}) = \{2, y\}$ $\{4\}_b = \epsilon_CLOSURE(\{4\}) = \{4, y\}$
 $\{x, 0\}_a = \epsilon_CLOSURE(\phi) = \phi$ $\{0, 3\}_b = \epsilon_CLOSURE(\{4\}) = \{4, y\}$

将NFA确定化为DFA的子集法

设 $\Sigma = \{a_1, a_2, \dots, a_k\}$. 先构造一张表, 该表的行数待定, 列数为 $k+1$ 列. 然后置该表的首行首列为 $\epsilon_CLOSURE(\{x\})$.

- 表的初始化构造
- 处理表的一行
- 重复处理

如果某一行的第一列的状态子集已经确定, 例如记为 I , 那么, 依次求出 $I_{a_1}, I_{a_2}, \dots, I_{a_k}$. 同时看它们是否已在表的第一列出现过, 将未出现的填入到后面空行的第一列.

重复上述过程, 直至所有第 2 列和第 $k+1$ 列的子集均已在第一列上出现了为止.

确定化: 由NFA M' 构造DFA M''

I	I_a	I_b
$\{x, 0, 1\}$	$\{0, 2, 1\}$	$\{0, 3, 1\}$
$\{0, 2, 1\}$	$\{0, 2, 1, 4, 5, y\}$	$\{0, 3, 1\}$
$\{0, 3, 1\}$	$\{0, 2, 1\}$	$\{0, 3, 1, 4, 5, y\}$
$\{0, 2, 1, 4, 5, y\}$	$\{0, 2, 1, 4, 5, y\}$	$\{0, 3, 1, 5, y\}$
$\{0, 3, 1, 5, y\}$	$\{0, 2, 1, 5, y\}$	$\{0, 3, 1, 4, 5, y\}$
$\{0, 3, 1, 4, 5, y\}$	$\{0, 2, 1, 5, y\}$	$\{0, 3, 1, 4, 5, y\}$
$\{0, 2, 1, 5, y\}$	$\{0, 2, 1, 4, 5, y\}$	$\{0, 3, 1, 5, y\}$

$I = \epsilon_CLOSURE(\{x\})$ 为 $\{x, 0, 1\}$. 从状态 I 出发经过一条 a 弧而能到达的状态全体 J 为 $\{0, 2\}$, 而 $\epsilon_CLOSURE(J) = \{0, 2, 1\}$. 从而 $I_a = \{0, 2, 1\}$.

把子集作为状态并命名

I	I _a	I _b
{x,0,1} ↔ 0	{0,2,1} ↔ 1	{0,3,1} ↔ 2
{0,2,1} ↔ 1	{0,2,1,4,5,y} ↔ 3	{0,3,1} ↔ 2
{0,3,1} ↔ 2	{0,2,1} ↔ 1	{0,3,1,4,5,y} ↔ 5
{0,2,1,4,5,y} ↔ 3	{0,2,1,4,5,y} ↔ 3	{0,3,1,5,y} ↔ 4
{0,3,1,5,y} ↔ 4	{0,2,1,5,y} ↔ 6	{0,3,1,4,5,y} ↔ 5
{0,3,1,4,5,y} ↔ 5	{0,2,1,5,y} ↔ 6	{0,3,1,4,5,y} ↔ 5
{0,2,1,5,y} ↔ 6	{0,2,1,4,5,y} ↔ 3	{0,3,1,5,y} ↔ 4

该DFA接受的输入串同样为 (a|b)*(aa|bb)(a|b)*

到此我们将这个表作为状态转换矩阵就得到

DFA $M'' = (S', \Sigma', \delta'', s_0, F')$

其中 δ'' 为通过子集算法构造的状态转换矩阵所表示的映射, 而 s_0 为 $\epsilon_CLOSURE(x)$ 命名后的状态名

44

例: 考虑如下NFA转换为DFA

{1}	{2,3,4,5,7,10}	ϕ
{2,3,4,5,7,10}	{4,5,6,7,9,10}	{4,5,7,8,9,10}
{4,5,6,7,9,10}	{4,5,6,7,9,10}	{4,5,7,8,9,10}
{4,5,7,8,9,10}	{4,5,6,7,9,10}	{4,5,7,8,9,10}

3.1.4 3型文法与有限自动机的等价性

Chomsky文法

$G = (V_T, V_N, S, P)$

2型文法 $A \rightarrow \beta, A \in V_N, \beta \in (V_N \cup V_T)^*$

3型文法 $A \rightarrow \alpha B, \text{或}, A \rightarrow \alpha, \text{且}, A, B \in V_N, \alpha \in (V_T)^*$

正规文法

46

3型文法有两种表示形式

$G = (V_T, V_N, S, P)$

左线性文法:

$A \rightarrow B\alpha, \text{或}, A \rightarrow \alpha, \text{且}, A, B \in V_N, \alpha \in V_T^*$

右线性文法:

$A \rightarrow \alpha B, \text{或}, A \rightarrow \alpha, \text{且}, A, B \in V_N, \alpha \in V_T^*$

47

3.1.5 正规文法与有限自动机的等价性

对于3型文法 (正规文法) G和有限自动机M, 如果 $L(G)=L(M)$, 则称G和M是等价的。

关于3型文法和有限自动机的等价性, 有以下结论:

- ①对于每一个3型文法G, 都存在一个有限自动机M, 使得 $L(M) = L(G)$.
- ②对于每一个有限自动机M, 都存在一个3型文法G, 使得 $L(M) = L(G)$.

48

定理：对于每一个右线性文法G, 都存在一个有限自动机M, 使得 $L(M) = L(G)$.

$G = (V_T, V_N, S, P)$ 做如下构造得到的M为一个NFA

令 $M = (V_N \cup \{f\}, V_T, \delta, S, \{f\})$, 其中 $f \notin V_N$, 且 δ 如下:

- 1) 若对于某个 $A \in V_N$ 及 $a \in V_T \cup \{\epsilon\}$ 有 $(A, a) \in P$ 则令 $\delta(A, a) = f$;
- 2) 对任意 $A \in V_N$ 及 $a \in V_T \cup \{\epsilon\}$, 设P中左端为A右端第一符号为a的产生式形如 $A \rightarrow aA_1 | aA_2 | \dots | aA_k$ 则令 $\delta(A, a) = \{A_1, \dots, A_k\}$.

49

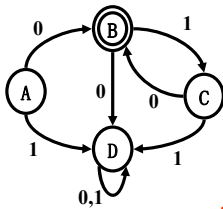
定理：对于每一个有限自动机M, 都存在一个右线性文法G, 使得 $L(M) = L(G)$.

DFA $M = (S, \Sigma, \delta, s_0, F)$ 做如下构造得到右线性文法G

- 1) 若 $s_0 \notin F$, 令 $G = (\Sigma, S, s_0, P)$, 其中P构造如下: 对于任何 $a \in \Sigma$ 及 $A, B \in S$, 若有 $\delta(A, a) = B$, 则:
 - i) 当 $B \notin F$ 时, 令 $A \rightarrow aB$
 - ii) 当 $B \in F$ 时, 令 $A \rightarrow a | aB$
- 2) $s_0 \in F$, 因为 $\delta(s_0, \epsilon) = s_0$, 所以 $\epsilon \in L(M)$, 从而 $L(G) = L(M) - \{\epsilon\}$. 因此令 $G' = (\Sigma, S \cup \{s'_0\}, s'_0, P \cup \{(s'_0, s_0), (s'_0, \epsilon)\})$ 其中 $s'_0 \notin S$. 最后令 $G = G'$

50

DFA $M = (\{A, B, C, D\}, \{0, 1\}, \delta, A, \{B\})$



$G_L = (\{0, 1\}, \{B, C, D, f\}, f, \emptyset)$
 $f \rightarrow 0C0 \quad B \rightarrow 0C0$
 $C \rightarrow B1 \quad D \rightarrow 1C1 | D0 | D1 | B0$

$G_R = (\{0, 1\}, \{A, B, C, D\}, A, \emptyset)$
 $A \rightarrow 0B | 1D \quad B \rightarrow 0D | 1C$
 $C \rightarrow 0B | 1D \quad D \rightarrow 0D | 1D$

NFA $M' = (\{A, B, C, D, f\}, \{0, 1\}, \delta, A, \{f\})$

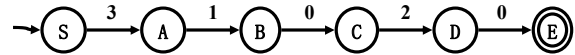
51

举例：自动机与右线性文法

■ 符号串 31020

■ 部分规则

$S \rightarrow 3A$
 $A \rightarrow 1B$
 $B \rightarrow 0C$
 $C \rightarrow 2D$
 $D \rightarrow 0$



■ $S \Rightarrow 3A \Rightarrow 31B \Rightarrow 310C \Rightarrow 3102D \Rightarrow 31020$

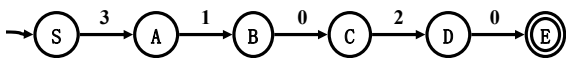
52

举例：自动机与左线性文法

■ 符号串 31020

■ 部分规则

$A \rightarrow 3$
 $B \rightarrow A1$
 $C \rightarrow B0$
 $D \rightarrow C2$
 $E \rightarrow D0$



■ $E \Rightarrow D0 \Rightarrow C20 \Rightarrow B020 \Rightarrow A1020 \Rightarrow 31020$

53

3.1.5 正规式与有限自动机的等价性

关于正规式和有限自动机的等价性, 有以下定理:

定理：对于字母表 Σ 上的任何有限自动机M, 都存在一个 Σ 上的正规式r, 使得 $L(r) = L(M)$. 反之亦然。

构造性证明:

1. 对 Σ 上的NFA M构造 Σ 上的正规式 r 使得 $L(M) = L(r)$.

2. 对 Σ 上的正规式 r, 构造 Σ 上的NFA M, 使得 $L(r) = L(M)$.

54

令每条弧以正规式作标记；在图中加进X和Y两个结点，从X用 ϵ 弧连接到M的所有初态结点，从M的所有终态结点用 ϵ 弧分别连接到Y；将X作为初态，Y作为终态构成新的一个NFA M' 且 $L(M)=L(M')$ 。

采用图示替换规则消去 M' 中所有结点，只剩下X和Y，则X到Y的弧的标记就是R

55

例：从NFA到正规式

56

例：从NFA到正规式(续)

57

2. 对 Σ 上的正规式r构造 Σ 上的NFA M使得 $L(M)=L(r)$

- $r = \epsilon$:
- $r = \phi$:
- $r = a$:

58

2续. 对 Σ 上的正规式r构造 Σ 上的NFA M使得 $L(M)=L(r)$

$r = r_1 | r_2$

$L(r) = L(M_1) \cup L(M_2)$

59

2续. 对 Σ 上的正规式r构造 Σ 上的NFA M使得 $L(M)=L(r)$

$r = r_1 r_2$

$L(r) = L(M_1) L(M_2)$

60

2续. 对 Σ 上的正规式 r 构造 Σ 上的NFA M 使得 $L(M)=L(r)$

■ $r=r_1^*$

$L(r)=L(M_1)^*$

61

62

例：从规式到 NFA正

■ $ab+|ab^*|b^*$

63

例：从规式到 NFA正(续)

■ $ab+|ab^*|b^*$

64

结论

■ 正规文法、正规式、确定有限自动机、非确定有限自动机在接受语言能力上等价。

65

3.1.6 确定有限自动机的化简

一个DFA M 的化简是指：
 寻找一个状态数比 M 少的DFA M' ，使得 $L(M)=L(M')$ 。

一个DFA M 的状态最少化过程旨在将 M 的状态集分割成一些不相交的子集，使得任何不同的两个子集中的状态都是可区分的，而同一子集中的任何两个状态都是等价的。最后，在每个子集中选出一个代表，同时消去其它等价状态。

M 的两个状态 s 和 t 是等价的：当且仅当分别从二者出发至终态均能够读出同一字。

M 的两个状态 s 和 t 是可区分的：当且仅当二者不等价。

66

状态集S的划分步骤:

- (1)基本划分{终结状态集, 其他状态集}是一个划分 Π ;
- (2)对于当前划分 $\Pi=\{I^{(1)}, \dots, I^{(m)}\}$, $m \geq 1$; 进行如下处理:
 对于每个 $a \in \Sigma$, 若 $I_a^{(i)}$ 分别与 Π 的 n 个不同块相交不为空 (其中 $1 < n \leq m$), 则将 $I^{(i)}$ 划分成 n 个不相交的子集, 使得每个子集 J 的 J_a 包含于 Π 的某个块. 然后将每个 J 加入 Π 并删除 $I^{(i)}$
- (3)重复上述过程(2)直至 Π 不再变化为止.

$$\Pi = \{X_1, \dots, X_m\}, \quad X_i \subseteq X, i=1 \dots m, m \geq 1$$

$$X_i \cap X_j = \emptyset, \bigcup_{i=1}^m X_i = X$$

67

例: DFA的化简:

把M的状态分为2组: 终态组{3,4,5,6}, 非终态组{0,1,2}

考察{3,4,5,6}, 由于{3,4,5,6}和{3,4,5,6}都包含于{3,4,5,6}, 所以它不能再划分

考察{0,1,2}, 由于{0,1,2} = {1,3}, 分别与{3,4,5,6}和{0,1,2}相交不为空, 故进行划分. 因 $\delta(1,a)=3$ 且 $\delta(0,a)=\delta(2,a)=1$, 故将{0,1,2}分成{1}, {0,2}

考察{0,2}, 由于{0,2} = {2,5}, 与上述3块中的2块相交不为空, 故划分成{0}, {2}

当前 Π 不能再划分, 最后重新命名状态, 其中新状态3代表{3,4,5,6}, 把原来射入3,4,5,6的弧改为射入新状态3

Step1: $\Pi = \{\{3,4,5,6\}, \{0,1,2\}\}$

Step2: $\Pi = \{\{3,4,5,6\}, \{1\}, \{0,2\}\}$

Step3: $\Pi = \{\{3,4,5,6\}, \{1\}, \{0\}, \{2\}\}$

68

3.2 词法分析器的设计与实现

- Introduction
- Design Issues
- Examples
- Generators

69

3.2.1 Introduction to Lexical Analyzer

70

Character Stream

- Source Code
 - Programming Languages
 - Script Languages (HTML, XML)
 - Others
- Grammar
 - Alphabet
 - Production rules

Token Stream

- Category
- Value
- Symbol Table
- Other Tables

71

Lexical Analyzer Generator

72

3.2.2 Design Issues on Lexical Analyzer

- Token 的表示
 - ✦ 全体一种
 - ✦ 一字一种
- 词法分析器与语法分析器的关系
 - ✦ 单独一遍
 - ✦ 作为语法分析器的子程序
- 输入缓冲区的控制
 - ✦ 预处理
 - ✦ 超前搜索
- DFA 的表示
 - ✦ Matrix
 - ✦ Case 语句

73

(1) 单词符号的分类

- **关键字**: 由程序语言定义的具有固定意义的标识符。也可称为保留字或基本字。例如: Pascal 中的 begin, end, if 等。它是确定的。
- **标识符**: 用来表示各种名字, 如变量名、数组名、过程名等。它是无限的。
- **常数**: 常数的类型一般有整型、实型、布尔型、文字型等。它是无限的。
- **运算符**: 如 +、-、*、/ 等。它是确定的。
- **界符**: 如逗号、分号、括号、/*、*/ 等。它是确定的。

74

(1) Token 的表示

- 用二元组表示: <种别, 属性值>
- **种别**: 采用整数表示、常量表示或者特殊标识符表示
- **属性值**: 对于常数采用内部表示; 标识符(变量、常量、数组名、函数名等)采用符号表入口, 等等。

种别的确定: 主要取决于实现上的方便, 例如: 标识符一般同归为一种。常数宜按类型(整、实、布尔)分。关键字可以将其全体视为一种, 也可一字一种。运算符可采用一符一种, 但也可把具有一定共性的视为一种。界符则一般采用一符一种。若是一种分种, 单词自身值就不需要了。

75

例

■ 例: FORTRAN 编译程序的词法分析器在扫描输入串 **IF (5=EQ-M) GOTO 100** 后, 它输出的**单词符号流**是:

- 逻辑IF (34, _)
- 左括号 (2, _)
- 整常数 (20, '5'的二进制表示)
- 等号 (6, _)
- 标识符 (26, 'M')
- 右括号 (16, _)
- GOTO (30, _)
- 标号 (19, '100'的二进制表示)

76

例

■ 例: C++ 代码段: **while (i >= j) i--;** 经词法分析器处理以后, 它将被转换为如下的**单词符号流**:

- (while, _)
- ((, _)
- (id, 指向 i 的符号表指针)
- (>=, _)
- (id, 指向 j 的符号表指针)
- (, _)
- (id, 指向 i 的符号表指针)
- (--, _)
- (;, _)

77

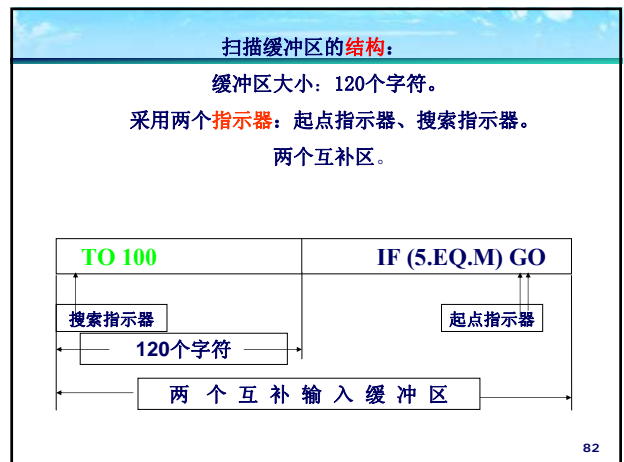
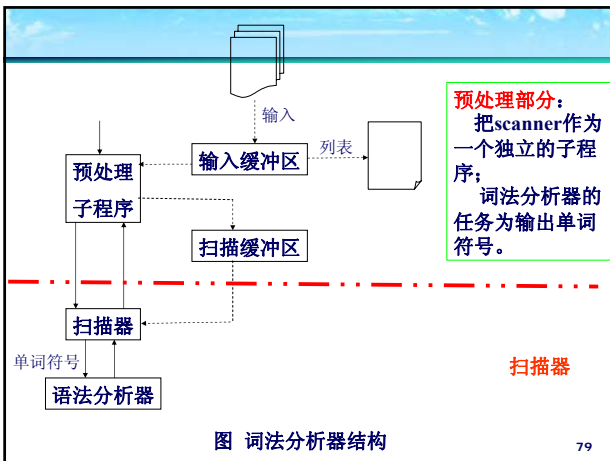
(2) 词法分析器与语法分析器的关系

```

graph TD
    A[源程序] --> B[词法分析器]
    B --> C[Tokens]
    C --> D[语法分析器]
    D --> E[源程序]
    E --> F[词法分析器]
    F --> G[语法分析器]
  
```

① 作为不同的通
② 作为子程序调用

78



例: 关键字识别中的超前搜索:

例如: 在标准FORTRAN中

1、 DO99K = 1,10	其中的DO、IF为关键字
2、 IF(5.EQ.M)I = 10	
3、 DO99K = 1.10	其中的DO、IF为标识符的一部分
4、 IF(5) = 55	

83

- 标识符的识别
 - ✦ 多数语言的标识符是字母开头的“字母/数字”串, 而且在程序中标识符的出现后都跟着算符或界符。因此, 不难识别。
 - 常数的识别
 - ✦ 对于某些语言的常数的识别也需要使用超前搜索。
 - 算符和界符的识别
 - ✦ 对于诸如C++语言中的“++”、“-.-”, 这种复合成的算符, 需要超前搜索。
- 84

(4) 状态转换图的实现

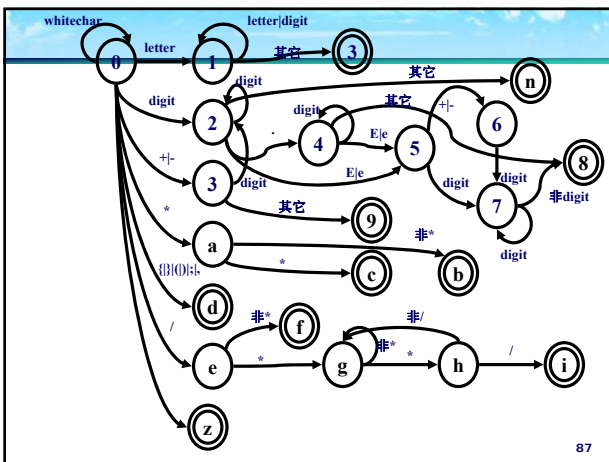
1. ch 字符变量, 存放最新读进的源程序字符。
2. TOKEN 字符数组, 存放构成单词的字符串。
3. GETCHAR 过程, 将下一输入字符读入ch, 搜索指示器前移一个字符。
4. GETBC 过程, 检查ch中的字符是否为空白。若是, 则调用GETCHAR 直至CHAR中进入一个非空白字符。
5. CONCAT 过程, 把ch中的字符连接到TOKEN之后。
6. IsLetter 布尔函数过程, 它们分别判断ch中的字符是数字或是字母, IsDigit 从而给出真假值TRUE、FALSE。
7. Reserve 整型函数过程, 用TOKEN中的字符串查保留字表, 若是一个保留字则给予编码, 否则回送0值(假定0不是保留字的编码)。
8. Retract 过程, 把搜索指示器回调一个字节, 把ch中的字符置为空白。
9. InsertID 插入符号表, 返回入口(指针)
10. InsertConst 插入常数表, 返回入口(指针)

85

3.2.3 Examples

■ C程序 (作为子程序)

86



87

符号表

```
struct entry
{
    struct entry *next; /* linked list implementation */
    char *name; /* the user-defined name of the identifier */
    int type; /* its type */
    int blockno; /* scoping information */
    int addr; /* address assigned by code generator */
};
```

88

```
/* constants for our scanner */
#define LPAREN '(' /* single char tokens */
#define RPAREN ')'
#define COMMA ','
#define GREATER '>'
#define LESS '<'
#define ASSIGN '='
...
#define WHILE 257 /* reserved words */
#define PRINT 258
#define RETURN 259
#define IF 260
...
#define VARIABLE 268 /* other tokens */
#define INTEGER 269
#define TEXT 270
#define EOF 271 /* end of file */
#define UNKNOWN 272 /* don't recognize */
```

```
/* some globals */

int ch; /* next char */
int intval; /* integer lexeme */
char textval[255]; /* text string lexeme */

void init()
{
    /* set up a lookup table for reserved words */
    create_reserved_table();
    insert_reserved(WHILE, "WHILE")
    insert_reserved(PRINT, "PRINT")
    insert_reserved(RETURN, "RETURN")
    ....
    /* get the first character */
    ch = getchar();
}
```

90

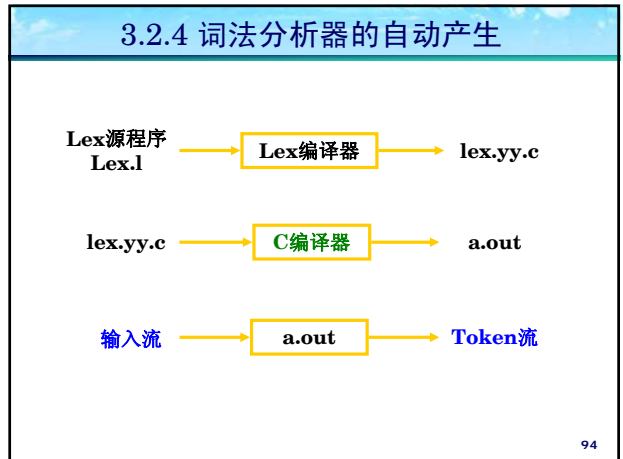
```
int scanner(){
switch(ch){
case ' ': /* white space */
case '\n':
case '\t':
    while (isspace(ch = getchar())); /* get rid of the white */
    return scanner();
case '/': /* check for a comment or DIVIDE operator */
    ch = getchar();
    if (ch != '/')
        return DIVIDE;
    ... /* code to skip over a comment */
case '(':
case ')':
case '!',... /* and any other single char tokens */
    intval = ch;
    ch = getchar();
    return intval;
}
```

91

```
case 'A': /*assume reserved words are caps, variables lower case*/
case 'B':
case 'C': ... /* reserved words */
    textval[0] = ch;
    for (i = 0; isupper(ch = getchar()); textval[i++] = ch)
        ;
    textval[i++] = '\0';
    return lookup_reserved(textval);
case 'a':
case 'b':
case 'c':... /* variables */
    textval[0] = ch;
    for (i = 0; islower(ch = getchar()); textval[i++] = ch);
    textval[i++] = '\0';
    if (lookup_syntab(textval) == UNKNOWN)
        add_syntab(textval);
    return VARIABLE;
```

```
case '0':
case '1':... /* integer */
    intval = ch - '0'; /* convert to a number */
    while (isdigit(ch = getchar()))
        intval = intval * 10 + ch - '0';
    return INTEGER;
case EOF:
    return EOF;
default:
    intval = ch;
    ch = getchar();
    return UNKNOWN;
}
```

93



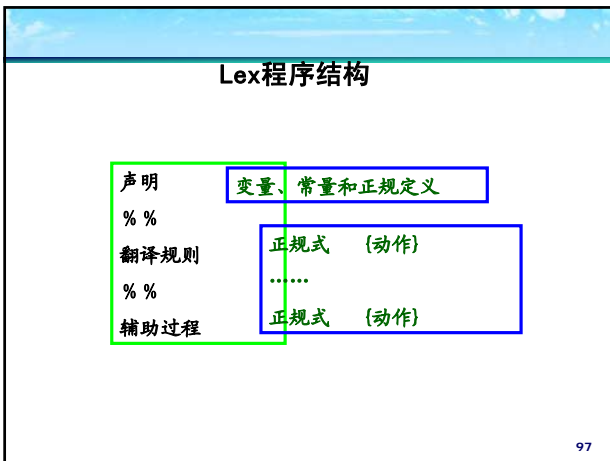
94



95

```

<程序> ::= <程序首部>; <分程序>.
<程序首部> ::= program <标识符>
<分程序> ::= <复合语句>
<复合语句> ::= begin <语句序列> end
<语句序列> ::= <语句> {, <语句>}
<语句> ::= <赋值语句> | <复合语句> | <条件语句>
<赋值语句> ::= <标识符> := <表达式>
<条件语句> ::= if <布尔表达式> then <语句> else <语句>
<表达式> ::= <项> { (+) | (-) <项>}
<项> ::= <因式> { (*) | (/) <因式>}
<因式> ::= <标识符> | <无正负号常量> | ' (' <表达式> ' ) '
<布尔表达式> ::= <表达式> <关系运算符> <表达式>
<关系运算符> ::= = | <| <= | > | <
<标识符> ::= <字母> { <字母> | <数字>}
<无正负号常量> ::= <数字> { <数字> } [ . <数字> { <数字> } ]
<字母> ::= a|b|c|d|e|f|g|...|u|v|w|x|y|z
<数字> ::= 0|1|2|3|4|5|6|7|8|9 输入以“#”为结束符
    
```

A-Z, 0-9, a-z	构成了部分模式的字符和数字。	98
.	匹配任意字符, 除了 \n。	
-	用来指定范围。例如: A-Z 指从 A 到 Z 之间的所有字符。	
[]	一个字符集合。匹配括号内的任意字符。如果第一个字符是 ^ 那么它表示否定模式。例如: [abc] 匹配 a, b, 和 C 中的任何一个。	
*	匹配0个或者多个上述的模式。	
+	匹配1个或者多个上述模式。	
?	匹配0个或1个上述模式。	
\$	作为模式的最后一个字符匹配一行的结尾。	
{ }	指出一个模式可能出现的次数。例如: A{1,3} 表示 A 可能出现1次或3次。	
\	用来转义元字符。同样用来覆盖字符在此表中定义的特殊意义, 只取字符的本意。	
~	否定。	
	表达式间的逻辑或。	
"<一些符号">	字符的字面含义。元字符具有。	
/	向前匹配。如果在匹配的模板中的 "/" 后跟有后续表达式, 只匹配模板中 "/" 前的部分。如: 如果输入 A01, 那么在模板 A0/1 中的 A0 是匹配的	
()	将一系列常规表达式分组。	

```

%{常量定义... %}
delim      [ \t\n]
ws         {delim}+
letter     [A-Za-z]
digit      [0-9]
id         {letter}({letter}|{digit})*
number     {digit}+(\.{digit}+)?(E[+-]?{digit}+)?

%%
{ws} {}
if      {return(IF);}
then    {return(THEN);}
else    {return(ELSE);}
{id}    {yyval=install_id();return(ID);}
{number} {yyval=install_num();return(NUMBER);}
"<"    {yyval=LT;return(RELOP);}
...
%%
install_id() { 单词装入符号表并返回指针; }
    
```

99

上机实验I

- 见论坛上的课程上机题目, 根据给定的C₀语言的文法, 采用Lex或者编写程序构造词法分析器。
- 上机时间第四周、第五周: 三个班安排在同一时间。由班级负责同学到计算机系实验中心登记, 定了以后通知辅导老师。
- 实验部分以当时上机情况和提交的报告进行评分, 占本课程成绩10%。

100

实验题目

实验I: 构造词法分析器, 采用Lex生成, 或者编程实现。
 实验II: 构造语法分析器, 采用Yacc生成, 或者编程实现。
 实验III*: 产生中间代码, 具体内容另给。

在论坛上公布上机文档:

- 实验I: 借助Lex和Yacc进行词法语法分析
 - ✦ 实验一lex与yacc的使用.doc
- 实验II: 用C或Java编写词法分析器
 - ✦ 实验二词法分析器.doc
- 实验III*: 用C或Java编写语法分析器
 - ✦ 实验三语法分析.doc

101

C₀语法

- ✦ program → { var-declaration | fun-declaration }
- ✦ var-declaration → int ID { , ID }
- ✦ fun-declaration → (int | void) ID (params) compound-stmt
- ✦ params → int ID { , int ID } | void | empty
- ✦ compound-stmt → { { var-declaration } { statement } }
- ✦ statement → expression-stmt | compound-stmt | if-stmt | while-stmt | return-stmt
- ✦ expression-stmt → [expression] ;

102

C₀语法 (续)

- ✦ if-stmt → if(expression) statement [else statement]
- ✦ while-stmt → while(expression) statement
- ✦ return-stmt → return [expression] ;
- ✦ expression → ID = expression | simple-expression
- ✦ simple-expression → additive-expression [relop additive-expression]

103

C₀语法 (续)

- ✦ relop → < | <= | > | >= | == | !=
- ✦ additive-expression → term [(+ | -) term]
- ✦ term → factor [(* | /) factor]
- ✦ factor → (expression) | ID | call | NUM
- ✦ call → ID(args)
- ✦ args → expression { , expression } | empty
- ✦ ID → ... ;参见C语言标识符定义
- ✦ NUM → ... ;参见C语言数的定义

104

报告提交

- 实验报告应该按照格式填写。
- 报告中写明程序设计思想及简要设计方案，代码简要描述，运行测试情况，体会。
- 每人一共提交2个报告，对应实验I和II（含III可选）
- 提交方式：作为附件email到指定邮箱，若得到回复确认表示提交成功。
- 指定邮箱为：**yinliang_zhao@163.com**
- 邮件标题格式：姓名-学号-班级-实验报告n
- 附件命名格式：姓名-学号-班级-实验报告n.rar
- 附件(.rar)中含有：源程序、测试数据结运行结果、报告、其他辅助文件（若有）

105

报告提交

- 例子
 - ✦ 赵银亮- 95434-软件91- 实验报告I.rar
 - ✦ 该文件中包括：实验报告，代码，数据等

106

本章练习

- 作业：P63-65:2,6-9,12,14,15
- 上机：完成上机题I并写好报告

107