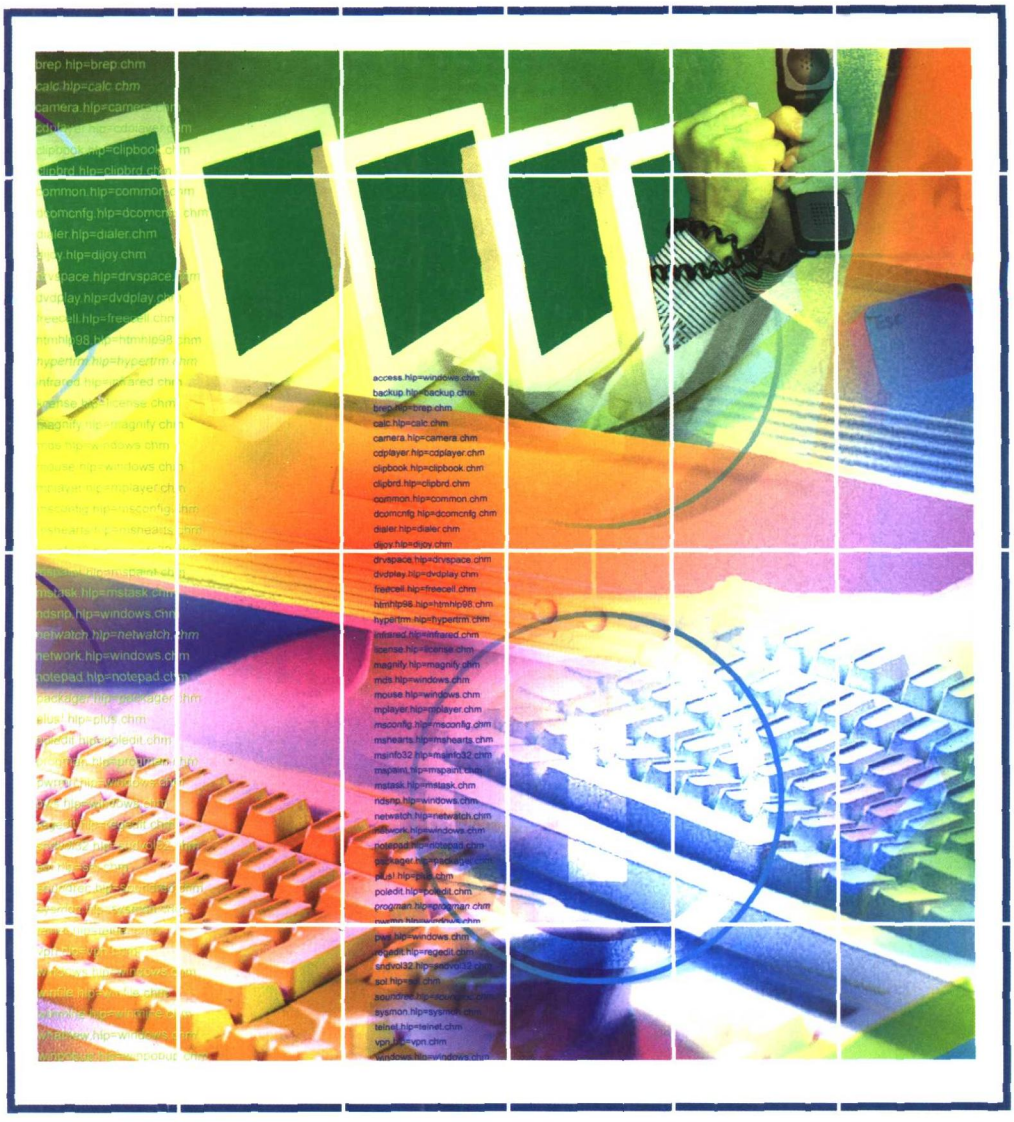
新世纪计算机类本科系列教材



《编译原理基础》习题与上机题解答

刘坚　郭强　胡圣明　编著

西安电子科技大学出版社
http://www.xduph.com

新世纪计算机类本科系列教材

《编译原理基础》习题与上机题解答

刘 坚 郭 强 胡圣明 编著

西安电子科技大学出版社

2003

内 容 简 介

本书是《编译原理基础》(2002 年 2 月出版，刘坚编著)的教学辅导书，内容包括两部分：习题解答，上机题与参考解决方案，并在附录中给出了源程序清单。

本书可以作为工科院校计算机专业或非计算机专业编译原理程课的辅助教材，也可作为软件工程技术人员或程序设计爱好者的参考书。

图书在版编目（CIP）数据

《编译原理基础》习题与上机题解答 / 刘坚，郭强，胡圣明编著.

—西安：西安电子科技大学出版社，2003.2

(新世纪计算机类本科系列教材)

ISBN 7-5606-1203-2

Ⅰ. 编… Ⅱ. ①刘… ②…郭 ③胡… Ⅲ. 编译程序-程序设计-高等学校-教学参考资料

Ⅳ. TP314

中国版本图书馆 CIP 数据核字(2002)第 110080 号

策　　划　陈宇光

责任编辑　戚文艳

出版发行　西安电子科技大学出版社(西安市太白南路 2 号)

电　　话　(029)8227828　邮　编　710071

http: //www.xduph.com　　E-mail: xdupfxb@pub.xaonline.com

经　　销　新华书店

印　　刷　西安兰翔印刷厂

版　　次　2003 年 2 月第 1 版　2003 年 6 月第 2 次印刷

开　　本　787 毫米×1092 毫米　1/16　印张 9.25

字　　数　214 千字

印　　数　4001～10 000 册

定　　价　10.00 元

ISBN 7-5606-1203-2/TP・0627

XDUP 1474001-2

如有印装问题可调换

前　言

本书是为西安电子科技大学出版社出版的教材《编译原理基础》(2002年2月出版，刘坚编著)配套的参考书，内容包括习题解答、上机题和参考解决方案，并在附录中给出了源程序清单。

习题解答部分给出了教材第1～6章中必做题的参考答案；上机题部分设计了一个函数绘图语言并要求为此语言编写一个解释器，解决方案以递归子程序方法为例详细讨论了文法的设计和词法分析器、语法分析器、语义支撑函数的编写与测试，同时也简单介绍了LEX/YACC方法的设计思想和程序设计特点；附录中分别给出了递归子程序和LEX/YACC两种实现方法的源程序清单以及在不同编译器条件下程序开发和运行环境的设置方法。

“编译原理”普遍被认为是比较难学的一门课程，而独立完成作业是理解、消化理论知识最好的辅助手段，同时也有助于培养读者理论与实践结合、思考与创新的能力。编写此书的目的是帮助读者而不是替读者学习，因此习题解答中仅给出了必做题的一般解答，上机题中仅构造了函数绘图语言最基本语句的解释器，选做题和上机题目的改进部分留给读者，以便为读者留下一定的思维空间。

最后附上大学生自己的一段话，真诚地希望读者能以正确的方式利用此书并从中真正受益。

> 大学里有三种“自杀”的方法：抄作业、copy程序和考试作弊。抄作业消磨的是你的精神和意志，copy上机程序剥夺的则是你以后赖以生存的专业技能，而考试作弊则马上能让你从哪里来回哪里去。……
>
> 王凯，西安电子科技大学学生科协《星火》2002年10月第2期

作者 刘坚

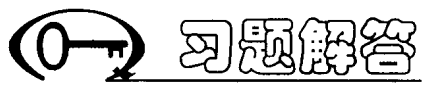
2003年1月

目　录

习题解答

上机题及参考解决方案

附　录



第1章 “引言”习题解答

1.1 列举出你所使用过的所有计算机语言和所有的“翻译”程序(编译、解释、汇编等)。

解:

(1) 汇编语言总是与具体的机器相关,早期的有 PDP11 和 VAX 的汇编语言及其汇编器,还有 Intel 系列微机上的 Z80 等汇编语言及其汇编器。

(2) 采用解释方式翻译语言的典型代表是早期的 Basic 和 Basic 的解释器。另外操作系统的命令也是解释执行,如 UNIX 的 shell 文件(也有人称其为 shell 程序)。

(3) 程序设计语言 Pascal; Ada83、Ada95; C/C++,编译器分别有 Turbo Pascal; VAXAda、ActiveAda、ObjectAda; Turbo C、C++ Builder、VC++等。

(4) 数据库查询语言 SQL,它一般有两种使用方式:交互式和嵌入式。交互式的翻译一般采用解释,而嵌入式的翻译一般采用编译(实质上是预编译,将 SQL 翻译为宿主语言,如 C、Ada 等)。SQL 的翻译一般由数据库管理系统提供,并且同时提供解释器和编译器,它们包括 informix 的 isql 和 esql、Oracle 的 sqlplus 和 proc 等。

(5) 语言识别器描述语言 Lex 和 Yacc,对应的编译器有 xdcfle 和 xdyacc。

1.2 如果在 Pascal 源程序中出现这样一些情况:12x 2*/3 3.5+"end" x/y(运行时 y=0),请指出它们分别是什么类型的错误。

解:

12x 是一个词法错误(但是编译器会认为是一个语法错误,为什么?); 2*/3 是一个语法错误,因为两个运算符*和/之间没有操作数; 3.5+"end"是一个静态语义错误,因为子表达式的类型不匹配; x/y 是一个动态语义错误(运行错误或逻辑错误),运行时可能会因为溢出而造成停机。

1.3 从你使用过的编译器中选择一个最熟悉的,写出从编写到运行一个应用程序的全过程。

解:

当前普遍使用的是 VC 6.0。它是一个集成环境,包括了源程序的编写、编译和运行测试等全过程。使用 VC 6.0 进行程序设计的基本步骤包括:首先建立一个工程(project workspace)并将程序所涉及的文件加入到工程中;然后反复进行下述操作,直到结束:打开工程中的文件并且编写(或修改)源程序,编译并连接(单步或合并进行均可)源程序,若有错误,则修改;否则运行测试。

第2章 “词法分析”习题解答

2.1 分别给出下述 Pascal 和 C 程序段的记号流形式。其中每个记号以有序对(记号类别，记号属性)的形式表示。例如：left + right 的记号流应该是：(id, left)(op,+) (id, right)。程序段中的注释可以忽略。

(1) Pascal

```
function max (i, j : integer ) : integer ;   { return maximum of i and j }
begin  if i > j  then  max := i  else  max := j  end;
```

(2) C

```
int max (i , j )   int i, j ;    /* return maximum of i and j */
{  return  i > j ? i : j ;  }
```

解：

(1) Pascal

类别	属性	类别	属性
(keyword,	function)	(id,	max)
(keysymbol,	( )	(id,	i)
(keysymbol,	,)	(id,	j)
(keysymbol,	:)	(keyword,	integer)
(keysymbol,	))	(keysymbol,	:)
(keyword,	integer)	(keysymbol,	;)
(keyword,	begin)	(keyword,	if)
(id,	i)	(relation,	>)
(id,	j)	(keyword,	then)
(id,	max)	(assign,	:=)
(id,	i)	(keyword,	else)
(id,	max)	(assign,	:=)
(id,	j)	(keyword,	end)
(keysymbol,	;)		

(2) C

类别	属性	类别	属性
(keyword,	int)	(id,	max)
(keysymbol,	( )	(id,	i)
(keysymbol,	,)	(id,	j)

(keysymbol,	)	)	(keyword,	int	)
(id,	i	)	(keysymbol,	,	)
(id,	j	)	(keysymbol,	;	)
(keysymbol,	{	)	(keyword,	return	)
(id,	i	)	(relation,	>	)
(id,	j	)	(keysymbol,	?	)
(id,	i	)	(keysymbol,	:	)
(id,	j	)	(keysymbol,	;	)
(keysymbol,	}	)			

2.2　用正规式描述习题 2.1 中的记号。

解：

(1) Pascal

keyword　= function | integer | begin | if | then | else | end

id　= char (char | digit)*　(其中，char=[a - zA - Z],digit=[0 - 9])

assign　= :=

keysymbol　= (|) | , | : | ;

relation　= >

(2) C

keyword　= int | return

id　= (_ | char) (_ | char | digit)*　(其中 char=[a - zA - Z],digit=[0 - 9])

keysymbol　= (|) | { | } | : | ? | ; | ,

relation　= >

2.3　令 A、B、C 是任意的正规式，证明下述关系成立：

(1) A|A = A

(2) $(A^*)^* = A^*$

(3) $A^* = \varepsilon | AA^*$

(4) $(AB)^*A = A(BA)^*$

证：

(1) 设正规式 A 表示的正规集为 L(A)。

因为　$L(A|A) = L(A) \cup L(A) = L(A)$

所以　$A|A = A$

(2) 因为　$(A^*)^* = A^{**}$　$A^{**} = A^*$

所以　$(A^*)^* = A^*$

(3) 因为　$AA^* = A^+$　$A^* = A^+ | \varepsilon = \varepsilon | A^+$

所以　$A^* = \varepsilon | AA^*$

(4) 因为　$(AB)^* = \varepsilon | AB | ABAB | ABABAB | \cdots$

$(BA)^* = \varepsilon | BA | BABA | BABABA | \cdots$

所以 $(AB)^*A = (\varepsilon|AB|ABAB|ABABAB|\cdots)A$

$= A|ABA|ABABA|ABABABA|\cdots$

$= A(\varepsilon|BA|BABA|BABABA|\cdots)$

$= A(BA)^*$

2.4 写出下述语言的正规式描述。

(1) 由偶数个 0 和奇数个 1 构成的所有 01 串。

(2) 所有不含子串 011 的 01 串。

(3) 每个 a 后边至少紧随两个 b 的 ab 串。

(4) C 的形如 /* … */ 的注释。其中…代表不含*/的字符串。

解:

(1) A1A|A0A1A0A

其中，A = ((00|11)|(10|01)(00|11)* (10|01))*。

(2) 1*(01|0)*

(3) (b|abb)*

(4) /*([^*]|*[^*/])**/

2.5 合法的日期表示有如下三种形式，请给出描述日期的正规式。

年.月.日，如 1992.08.12

日 月 年，如 12 08 1992

月/日/年，如 08/12/1992

解:

digit = [0 - 9]

year = (digit)(digit)(digit)(digit)

month = 0[1 - 9]|1[0 - 2]

day = 0[1 - 9]|[1 - 2][0 - 9]|3[0 - 1]

date1 = year.month.day

date2 = day month year

date3 = month/day/year

date = date1|date2|date3

2.6 有 NFA 定义如下:

N = (S={0,1}, Σ={a,b}, s0=0, F={0},

move={move(0,a)=0, move(0,a)=1, move(0,b)=1, move(1,a)=0})

(1) 画出 N 的状态转换图;

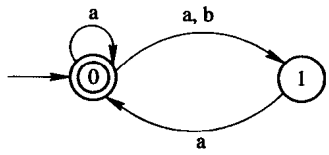
(2) 构造 N 的最小 DFA D;

(3) 给出 D 所接受语言的正规式描述;

(4) 举出语言中的三个串，并给出 D 识别它们的过程。

解:

(1) N 的状态转换图如题 2.6 图 1 所示。



题 2.6 图 1

(2) 由子集法构造 DFA：

① 确定化：

ε _闭包({0})　　= {0}　　A (初态，终态)

ε _闭包(smove(A, a)) = {0,1}　　B(终态)

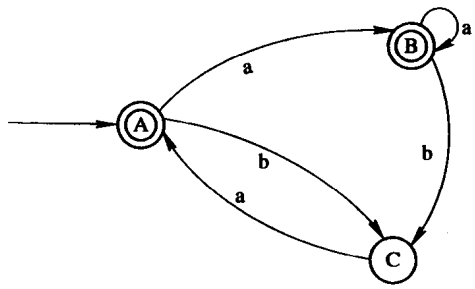
ε _闭包(smove(A, b)) = {1}　　C

ε _闭包(smove(B, a)) = {0,1}　　B

ε _闭包(smove(B, b)) = {1}　　C

ε _闭包(smove(C, a)) = {0}　　A

DFA 的图形表示如题 2.6 图 2 所示。



题 2.6 图 2

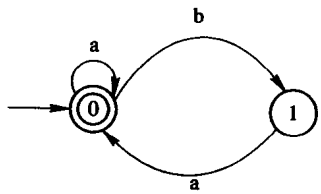
② 最小化 DFA：

初始划分：Π0={{A, B}, {C}};其中 C 自身一组，A 和 B 分在一组。查看状态转移：

move(A, a) = B　　move(A, b) = C

move(B, a) = B　　move(B, b) = C

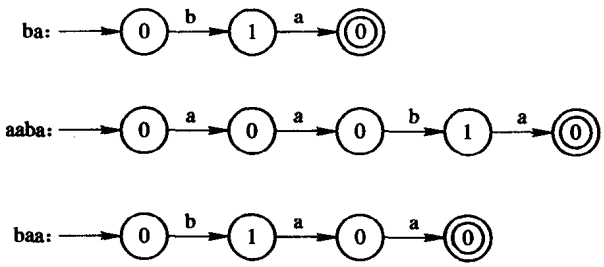
显然 A 和 B 不可区分，故合并为一个状态，选 A 为代表，并分别将 A 和 C 重新编号为 0 和 1，得最小 DFA 如题 2.6 图 3 所示。



题 2.6 图 3

(3) 所接受的语言为：$(alba)^*$。

(4) 对于串 ba、aaba、baa，最小 DFA 识别过程如题 2.6 图 4 所示。

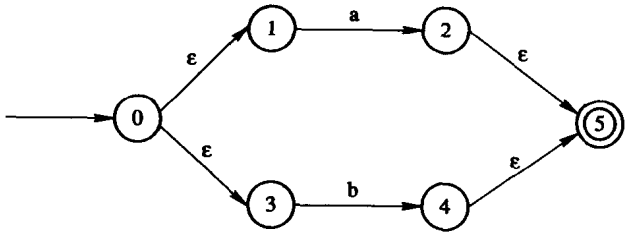


题 2.6 图 4

2.7 对于图 2.11(a)中的正规式 r3 和 r5，请分别构造它们的不确定有限自动机 N(r3)和 N(r5)。

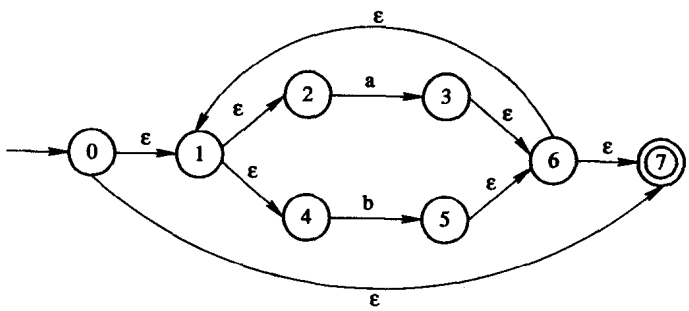
解：

r3 = alb，它的 NFA N(r3)如题 2.7 图 1 所示。



题 2.7 图 1

r5 = $(alb)^*$，它的 NFA N(r5)如题 2.7 图 2 所示。



题 2.7 图 2

2.8 将教材中图 2.16 所示的状态转换图表示的 FA 分别确定化和最小化。

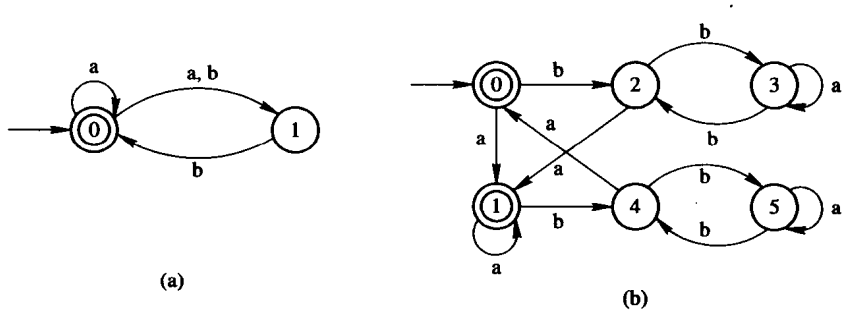


图 2.16　状态转换图

解：

(1) 对于图 2.16 (a)：

① 确定化 NFA：

ε _闭包({0})　　= {0}　　A(初态，终态)

ε _闭包(smove(A, a)) = {0,1}　　B(终态)

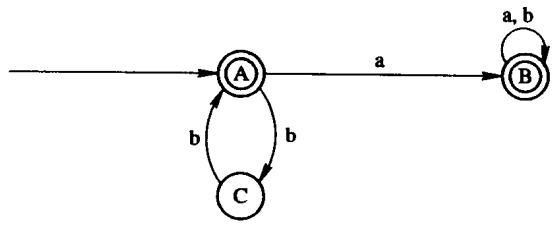
ε _闭包(smove(A, b)) = {1}　　C

ε _闭包(smove(B, a)) = {0,1}　　B

ε _闭包(smove(B, b)) = {0,1}　　B

ε _闭包(smove(C, b)) = {0}　　A

DFA 的图形表示如题 2.8 图 1 所示。



题 2.8 图 1

② 最小化 DFA：

初始划分：Π0={{A, B}, {C}};其中 C 自身一组，A 和 B 分在一组，考察 A 和 B 的下一状态转移：

move(A, a) = B　　move(A, b) = C

move(B, a) = B　　move(B, b) = B

由于从 A 和 B 出发经 b 的下一状态转移到达不同的组，故 A 和 B 是可区分的。于是得新的划分Π1={{A},{B}, {C}}。上述 DFA 已经是最小化的 DFA。

(2) 对于图 2.16 (b)：

① 经观察，原图已经是 DFA。

② 最小化 DFA：

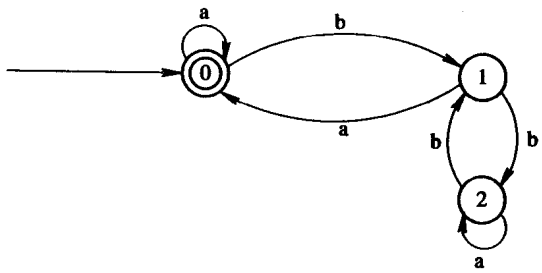
初始划分：Π0 = {{0,1}, {2,3,4,5}}。{0,1}为终态组，{2,3,4,5}为非终态组。考察Π0 中的所有下一状态转移：

move(0, a) = 1	move(0, b) = 2
move(1, a) = 1	move(1, b) = 4
move(2, a) = 1	move(2, b) = 3
move(3, a) = 3	move(3, b) = 2
move(4, a) = 0	move(4, b) = 5
move(5, a) = 5	move(5, b) = 4

move(2,a)和 move(4,a)与 move(3,a)和 move(5,a)分别落在Π0 的不同组中，因此，2 和 4 与 3 和 5 是可区分的，应分在不同的组中以形成新的划分：

Π1 = {{0,1}，{2,4}，{3,5}}。

再考察{0,1}，{2,4}，{3,5}中的所有下一状态转移，均落在相同的组中，固Π1 已经是最终划分。令 1 代表{0, 1}消去 0, 令 2 代表{2, 4}消去 4, 令 3 代表{3, 5}消去 5，最后得到最小化的 DFA 如题 2.8 图 2 所示。



题 2.8 图 2

2.9 用自然语言给出下述正规式所描述的语言，并构造它们的最小 DFA。

(1) 10^*1

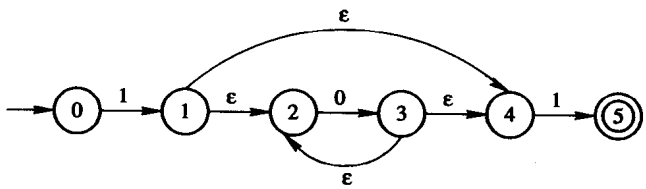
(2) $(0|1)^*011(0|1)^*$

解：

(1) 10^*1

自然语言描述为：两个 1 之间含有 0 或多个 0 的 01 串。

根据 Thompson 算法构造正规式的 NFA 如题 2.9 图 1 所示。



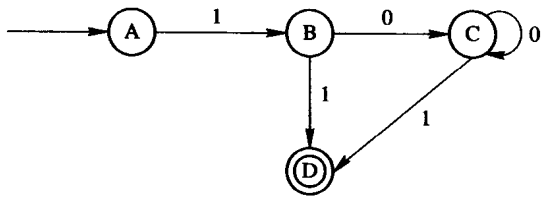
题 2.9 图 1

将 NFA 转化为 DFA：

ε _闭包({0}) = {0} A(初态)

ε _闭包(smove(A,1))= {1,2,4}　　B
ε _闭包(smove(B,0))= {2,3,4}　　C
ε _闭包(smove(B,1))= {5}　　D(终态)
ε _闭包(smove(C,0))= {2,3,4}　　C
ε _闭包(smove(C,1))= {5}　　D

DFA 的图形表示如题 2.9 图 2 所示。



题 2.9 图 2

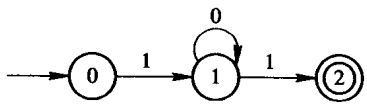
将 DFA 最小化：

① 初始划分：Π0 = {{A,B,C},{D}}

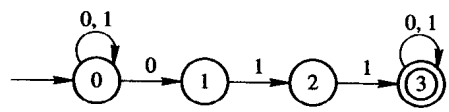
② 考察Π0 的组{A,B,C}中的所有下一状态转移：

move(A,1) = B　　move(A,0) =
move(B,1) = D　　move(B,0) = C
move(C,1) = D　　move(C,0) = C

A 的下一状态转移与 B 和 C 的下一状态转移落在不同的组中，故 A 应从{A,B,C}中分离，形成新的划分Π1= {{A},{B,C},{D}}。由于 B 和 C 的下一状态转移均相同，故Π1 成为最终划分。将Π1 中的三个组重新编号为 0、1、2，得最小 DFA 如题 2.9 图 3 所示。



题 2.9 图 3



题 2.9 图 4

(2) $(0|1)^*011(0|1)^*$

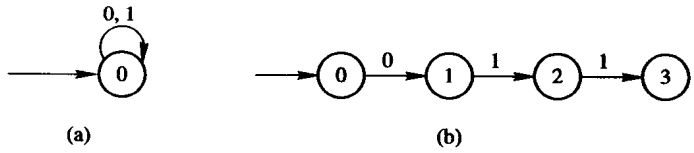
自然语言描述为：含有至少一个 011 子串的 01 串。

构造正规式的 NFA 如题 2.9 图 4 所示。

由于采用 Thompson 算法构造 NFA 是一件很繁琐的事情，因此上边采用直接根据正规式构造状态转换图的方法，其中的一种思想是：首先将正规式分解，然后分别构造子正规式的最简状态转换图，最后根据子正规式之间的运算关系和 Thompson 算法将所有子图组合成一个完整的状态转换图。一般情况下，所构造的状态转换图是一个 FA，如果已经是一个最小 DFA，构造就已经完成。否则视情况进行确定化和最小化。

根据这一思想，将 $R=(0|1)^*011(0|1)^*$分解为 $R=R_1R_2R_3$，其中 $R_1=R_3=(0|1)^*$，$R_2=011$。则 $N(R_1)$和 $N(R_3)$如题 2.9 图 5(a)所示，$N(R_2)$如题 2.9 图 5(b)所示。根据 Thompson 算法对连接

运算的处理(首尾相接)，将题 2.9 图 5(a)分别叠加到图(b)的状态 0 和状态 3 上，即得到上述的 NFA。

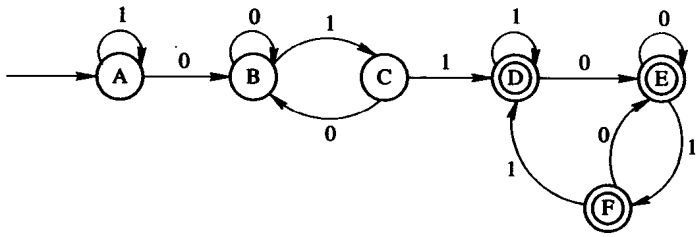


题 2.9 图 5

将 NFA 转化为 DFA：

ε _闭包({0}) = {0} A(初态)
ε _闭包(smove(A,0)) = {0,1} B
ε _闭包(smove(A,1)) = {0} A
ε _闭包(smove(B,0)) = {0,1} B
ε _闭包(smove(B,1)) = {0,2} C
ε _闭包(smove(C,0)) = {0,1} B
ε _闭包(smove(C,1)) = {0,3} D(终态)
ε _闭包(smove(D,0)) = {0,1,3} E(终态)
ε _闭包(smove(D,1)) = {0,3} D
ε _闭包(smove(E,0)) = {0,1,3} E
ε _闭包(smove(E,1)) = {0,2,3} F(终态)
ε _闭包(smove(F,0)) = {0,1,3} E
ε _闭包(smove(F,1)) = {0,3} D

DFA 的图形表示如题 2.9 图 6 所示。



题 2.9 图 6

将此 DFA 最小化：

① 初始划分：Π0 = {{A,B,C},{D,E,F}}，其中{A,B,C}是非终态组，{D,E,F}是终态组。

② 考察Π0 中的组{D,E,F}中的任何下一状态转移均落在{D,E,F}中，故{D,E,F}已不可再分。再考察{A,B,C}中的下一状态转移：

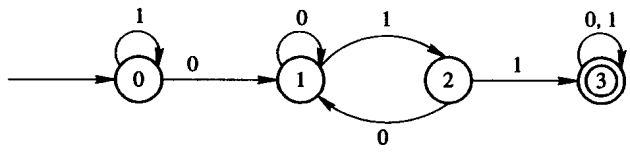
move(A,0) = B move(A,1) = A
move(B,0) = B move(B,1) = C

move(C,0) = B　　move(C,1) = D

由于 move(C, 1)=D，与 move(A, 1)和 move(B, 1)落在Π0 不同的组中，使得 C 从{A, B, C}中被分离，形成新的划分Π1= {{A, B}, {C}, {D, E, F}}。

同理再考察Π1 的组{A, B}的所有下一状态转移，move(A, 1)和 move(B,1)落在Π1 不同的组中，故再分离 A 和 B 形成新的划分Π2 = {{A}, {B}, {C}, {D, E, F}}。

Π2 是最终划分，将Π2 中的组重新编号为 0、1、2、3，得最小 DFA 如题 2.9 图 7 所示。



题 2.9 图 7

2.10　有一 NFA 的状态转换矩阵如表 2.9 所示，其中 S 为初态，D 为终态。

表 2.9　状态转换矩阵

	a	b	c	ε
S	A,B	C,D	D	A,B,C
A	A		C	B
B	A	D		C
C	B	A		A
D	C	B		S

(1) 求出它的最小 DFA；

(2) 用正规式描述 DFA 所接受的语言。

解：

事实上，从状态转换矩阵表示的 NFA 构造 DFA 更直观、简单。ε_闭包和 smove 的算法可以描述如下。

设 X = { x_i|i=1, 2, ..., n}是终态集，Trans 是终态转移矩阵，则

```
function  ε_闭包(X) is
begin     T := X;
          for each t_i ∈ T
          loop u = Trans[t_i, ε];
          if u 不空且不在 T 中 then 加入 u 到 T 中; end if;
          exit when 再无 u 可以加入到 T 中;
     end loop;
     return T;
end  ε_闭包;
```

```
function smove(X, a) is
begin
  T := {};
  for each t_i ∈ X
  loop   u := Trans[t_i, a];
    if u 不空且不在 T 中 then 加入 u 到 T 中; end if;
  end loop;
  return T;
end smove;
```

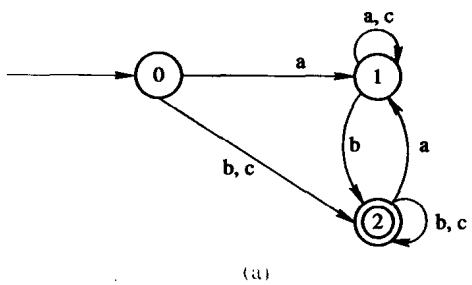
直观上讲，ε _闭包(X)就是对所有在 X 中的元素 t_i，将矩阵中第 t_i 行第 ε 列的元素加入到 X 中，直到 X 不再扩大为止，最终的 X 即为所求；而 smove(X, a)就是对所有在 X 中的元素 t_i，将矩阵中第 t_i 行第 a 列的元素加入到 T 中，最终的 T 是所求结果。

(1) 计算最小 DFA。

首先根据上述算法计算 DFA：

ε _闭包({S}) ={S, A, B, C} 0(初态)
ε _闭包(smove(0,a))={A, B, C} 1
ε _闭包(smove(0,b))={S, A, B, C, D} 2(终态)
ε _闭包(smove(0,c))={S, A, B, C, D} 2
ε _闭包(smove(1,a))={A, B, C} 1
ε _闭包(smove(1,b))={ S, A, B, C, D} 2
ε _闭包(smove(1,c))={A, B, C} 1
ε _闭包(smove(2,a))={A, B, C} 1
ε _闭包(smove(2,b))={ S, A, B, C, D} 2
ε _闭包(smove(2,b))={ S, A, B, C, D} 2

DFA 的图形和矩阵分别如题 2.10 图的(a)、(b)所示。图(b)中的 0 为初态，2 为终态。



(a)

	a	b	c
0	1	2	2
1	1	2	1
2	1	2	2

(b)

题 2.10 图

(a) 图形；(b) 矩阵

然后最小化 DFA：

从上述矩阵中马上可以看出：此 DFA 已经是最小化的，因为三个状态都是可区分的。首先 2 是终态，与 0 和 1 可区分；然后考察 0 和 1 两行对应的列，马上可以看出：在 c 列上两个状态转移一个是终态，而另一个是非终态，故 0 和 1 也是可区分的。