编译原理第一次作业

一、判断题

1. 编译程序是对高级语言的翻译。（对）
2. 设有表达式$a \times b - c$，将其中$a \times b$识别为表达式的编译阶段是语法分析。（对）
3. 一个有限状态自动机中，有且仅有一个唯一的终态。（错）
4. 由“非终结符 -> 符号串”形式的规则构成的文法是 1 型文法。（错）
5. 文法识别符号经过任意步推导得到的结果是句型。（对）
6. 最左短语一定是句柄。（错）
7. 在编译中产生语法树是为了语法分析。（对）
8. 语法分析时必须先消除文法中的左递归。（错）
9. 自下而上分析过程是对句子实施推导的过程。（错）
10. LR 分析法在自左至右扫描输入串时就能发现错误，但不能准确地指出出错地点。（对）

二、主观题

1. 描述下列正则表达式所表示的语言，或对于下列语言分别写出它们的正则表达式

(1) $a(a|b)*a$

(2) $((\epsilon|a)b*)*$

(3) $(a|b)*a(a|b)(a|b)$

(4) $a*ba*ba*ba*$

(5) $(aa|bb)*((ab|ba)(aa|bb)*(ab|ba)(aa|bb)*)*$

(6) 包含 5 个元音的所有小写字母串，这些串中的元音按顺序出现。

(7) 所有由按词典递增序排列的小写字母组成的串。

(8) 所有由 0 和 1 组成且包含偶数个 1 的串。

(9) 所有由 a 和 b 组成且不含子序列 abb 的串。

(10)所有由 a 和 b 组成且不含子串 abb 的串。

解析：

(1) 所有以 a 开始并以 a 结尾的由 a 和 b 组成的字符串。

(2) 所有由 a 和 b 组成的字符串。

(3) 所有由 a 和 b 组成且倒数第三个字符是 a 的串。

(4) 所有由 a 和 b 组成且只包含三个 b 的串。

(5) 所有由偶数个 a 和偶数个 b 构成的串。

(6)

```
Plain Text
1 want -> other* a (other|a)* e (other|e)* i (other|i)* o (other|o)* u (o
  ther|u)*
2 other -> [bcdfghjklmnpqrstvwxyz]
```

(7)

```
Plain Text
1 a* b* ... z*
```

(8)

```
Plain Text
1 (0|10*1)*
```

(9)

```
Plain Text
1 b* | b*a+ | b*a+ba*
```

(10)

```
Plain Text
1 b*(a+b?)*
```

2. 将下面的正则表达式转化成 DFA

$((\epsilon|a)b^*)^*$

(1) 使用 Thompson 构造法为其构造 NFA，写出每个 NFA 处理符号串 ababbab 过

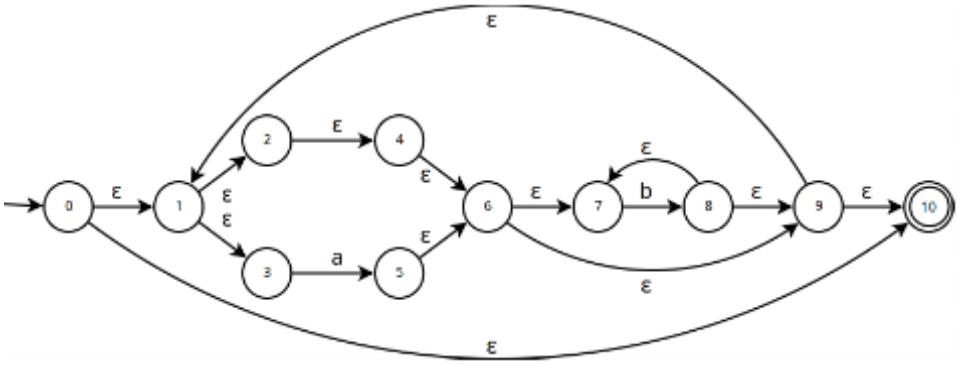
程中的状态转换序列。

(2) 利用子集构造法将（1）得到的 NFA 转换为 DFA，同样写出分析符号串 ababbab 过程中的状态转换。

(3) 最小化（2）得到的 DFA

解析：

(1)



ababbab 状态转换序列：

$0 \to 1 \to 3 \to 5 \to 6 \to 7 \to 8 \to 9 \to 1 \to 3 \to 5 \to 6 \to 7 \to 8 \to 7 \to 8 \to 9 \to 1 \to 3 \to 5 \to 6 \to 7 \to 8 \to 9 \to 10$

(2)

$\epsilon - closure(\{0\}) = \{0,1,2,3,4,6,7,9,10\} = A$

$\epsilon - closure(\delta(A,a)) = \{1,2,3,4,5,6,7,9,10\} = B$

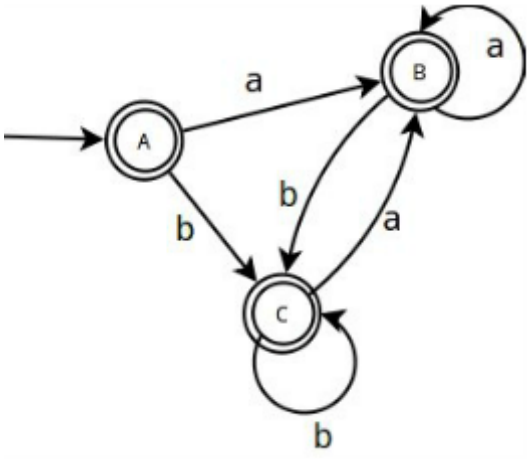
$\epsilon - closure(\delta(A,b)) = \{1,2,3,4,6,7,8,9,10\} = C$

$\epsilon - closure(\delta(B,a)) = B$

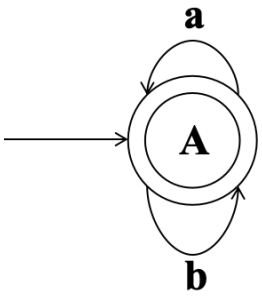
$\epsilon - closure(\delta(B,b)) = C$

$\epsilon - closure(\delta(C,a)) = B$

$\epsilon - closure(\delta(C,b)) = C$



(3)



3. 已知文法 G[A]

$E \to TE'$

$E' \to +E|\epsilon$

$T \to FT'$

$T' \to T|\epsilon$

$F \to PF'$

$F' \to *F'|\epsilon$

$P \to (E)|a|b|\wedge$

(1) 给出$(a*+b*)$的最左推导。

(2) 对每个非终结符写出不带回溯的递归子程序。

(3) 经改写后的文法是否是 LL(1)的？给出它的预测分析表。

(4) 给出输入串 a 的分析过程，并说明该串是否为 G 的句子。

解析：

(1)

$$E \Rightarrow TE' \Rightarrow FT'E' \Rightarrow PF'T'E' \Rightarrow (E)F'T'E' \Rightarrow (TE')F'T'E' \Rightarrow (FT'E')F'T'E' \Rightarrow (PF'T'E')F'T'E' \Rightarrow (aF'T'E')F'T'E' \Rightarrow (a*F'T'E')F'T'E' \Rightarrow (a*T'E')F'T'E' \Rightarrow (a*E')F'T'E' \Rightarrow (a*+E)F'T'E' \Rightarrow (a*+TE')F'T'E' \Rightarrow (a*+FT'E')F'T'E' \Rightarrow (a*+PF'T'E')F'T'E' \Rightarrow (a*+bF'T'E')F'T'E' \Rightarrow (a*+b*F'T'E')F'T'E' \Rightarrow (a*+b*T'E')F'T'E' \Rightarrow (a*+b*E')F'T'E' \Rightarrow (a*+b*)F'T'E' \Rightarrow (a*+b*)T'E' \Rightarrow (a*+b*)E' \Rightarrow (a*+b*)$$

(2)

Plain Text

```
char CH;
void P_E(){
    if(IsIn(CH,FIRST_TE')){ //产生式E->TE'
        P_T();
        P_E'();
    }
    else ERR;
}
void P_E'(){
    if(CH=='+'){ //产生式E'->+E
        READ(CH);
        P_E();
    }
    else{ //产生式E'->ε
        if(IsIn(CH,FOLLOW_E'))
            return;
        else ERR;
    }
}
void P_T(){
    if(IsIn(CH,FIRST_FT')){ //产生式T->FT'
        P_F();
        P_T'();
    }
    else ERR;
}
void P_T'(){
    if(IsIn(CH,FIRST_T)){ //产生式T'->T|ε
        READ(CH);
        P_T();
    }
    else{ //产生式T'->ε
        if(IsIn(CH,FOLLOW_T'))
            return;
        else ERR;
    }
}
void P_F(){
    if(IsIn(CH,FIRST_PF')){ //产生式E->PF'
        P_P();
        P_F'();
    }
```

```
43      else ERR;
44  }
45  void P_F'(){
46      if(CH=='*'){ //产生式F'->*F'
47          READ(CH);
48          P_F'();
49      }
50      else{ //产生式F'->ε
51          if(IsIn(CH,FOLLOW_F'))
52              return;
53          else ERR;
54      }
55  }
56  void P_P(){
57      if(CH=='a')READ(CH); //产生式P->a
58      else if(CH=='b')READ(CH); //产生式P->b
59      else if(CH=='^')READ(CH); //产生式P->^
60      else if(CH=='('){
61          READ(CH);
62          P_E();
63          if(CH==')')READ(CH);
64          else ERR;
65      }
66      else ERR;
67  }
```

(3) 计算 FIRST 集和 FOLLOW 集

$FIRST(E) = FIRST(T) = FIRST(F) = FIRST(P) = \{(,a,b,\wedge\}$

$FIRST(E') = \{+,\epsilon\}$

$FIRST(T') = \{(,a,b,\wedge,\epsilon\}$

$FIRST(F') = \{*,\epsilon\}$

$FOLLOW(E) = FOLLOW(E') = \{\$,)\}$

$FOLLOW(T) = FOLLOW(T') = \{+,\$,)\}$

$FOLLOW(F) = FOLLOW(F') = \{(,a,b,\wedge,+,\$,)\}$

$FOLLOW(P) = \{*,(,a,b,\wedge,+,\$,)\}$

预测分析表如下：

	+	*	(	)	a	b	^	#
E			E'→TE'		E→TE'	E→TE'	E→TE'	
E'	E'→+E			E'→ε				E'→ε
T			T'→FT'		T→FT'	T→FT'	T→FT'	
T'	T'→ε		T'→T	T'→ε	T'→T	T'→T	T'→T	T'→ε
F			F→PF'		F→PF'	F→PF'	F→PF'	
F'	F'→ε	F'→*F'	F'→ε	F'→ε	F'→ε	F'→ε	F'→ε	F'→ε
P			P→(E)		P→a	P→b	P→^	

(4) 给出串 a$ 的分析过程

步骤	分析栈	剩余输入串	所用产生式
1	$E	a$	E→TE'
2	$E'T	a$	T→FT'
3	$E'T'F	a$	F→PF'
4	$E'T'F'P	a$	P→a
5	$E'T'F'a	a$	a匹配
6	$E'T'F'	$	F'→ε
7	$E'T'	$	T'→ε
8	$E'	$	E'→ε
9	$	$	接受

4. 证明下面文法是SLR(1)文法，并构造其SLR分析表

$E \to E+T|T$

$T \to TF|F$

$F \to F^*|a|b$

解析：

该文法的拓广文法G'为：

(0)$E' \to E$

(1)$E \to E+T$

(2)$E \to T$

(3)$T \to TF$

(4)$T \to F$

(5)$F \to F^*$

(6)$F \to a$

(7)$F \to b$

由产生式求 First 集和 Follow 集：

$FIRST(E') = FIRST(E) = FIRST(T) = FIRST(F) = \{a,b\}$

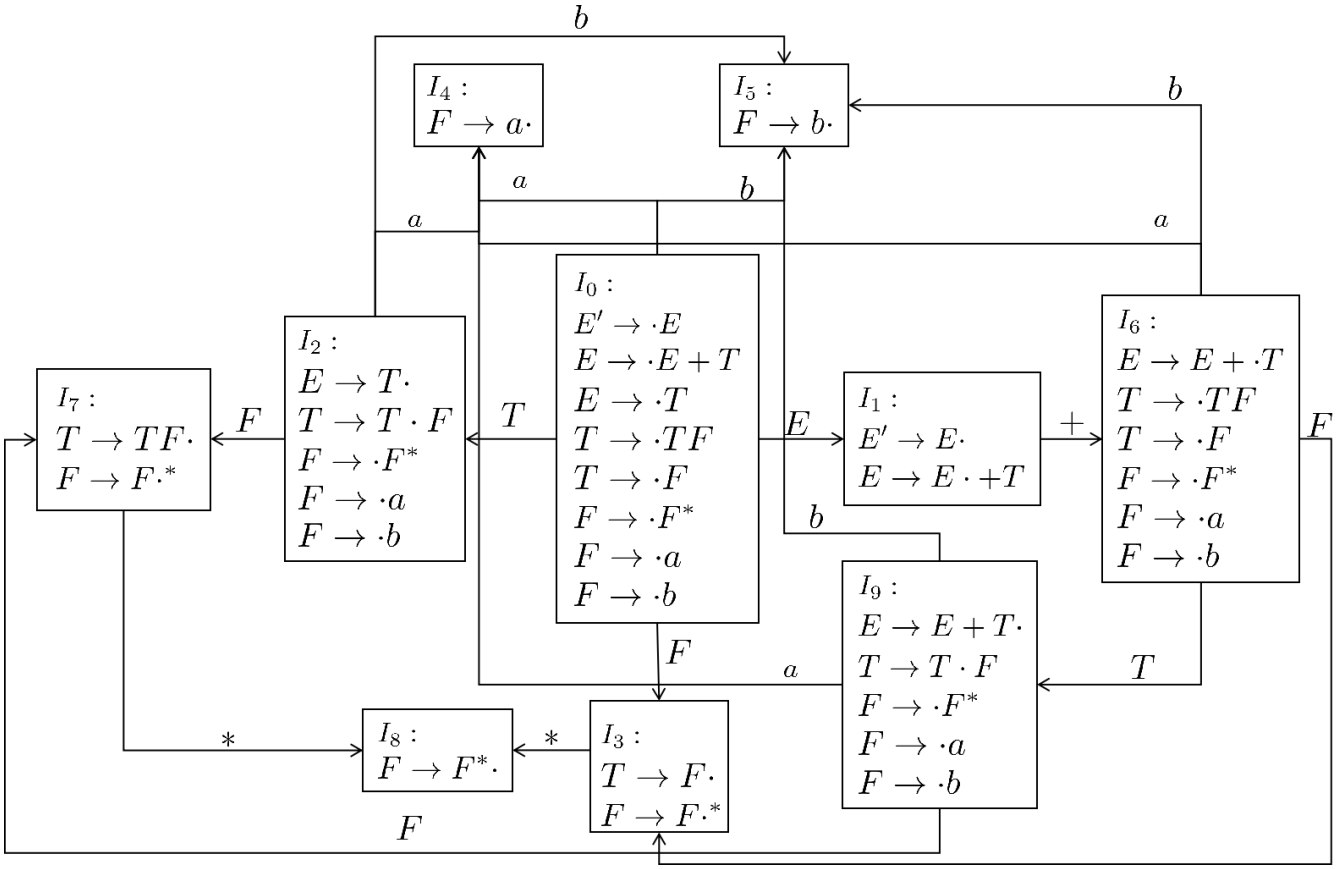
$FOLLOW(E') = \{\$\}$

$FOLLOW(E) = \{+,\$\}$

$FOLLOW(T) = \{+,\$,a,b\}$

$FOLLOW(F) = \{+,\$,a,b,*\}$

G'的 LR(0)项目集族及识别活前缀的 DFA 如下图所示：



构造的 SLR 分析表如下：

状态	action					goto		
	+	*	a	b	$	E	T	F
0			S4	S5		1	2	3
1	S6				acc			
2	r2		S4	S5	r2			7
3	r4	S8	r4	r4	r4			
4	r6	r6	r6	r6	r6			
5	r7	r7	r7	r7	r7			
6			S4	S5			9	3
7	r3	S8	r3	r3	r3			
8	r5	r5	r5	r5	r5			
9	r1		S4	S5	r1			7

显然，此分析表没有移入－规约冲突，也没有规约－规约冲突，所以该文法是SLR(1)文法。