**编译原理试题及答案**

一、对于文法 G[S] ：

　　S → 1A | 0B | ε 　　A → 0S | 1AA 　　B → 1S | 0BB

　　⑴ (3 分 ) 请写出三个关于 G[S] 的句子；

　　⑵ (4 分 ) 符号串 11A0S 是否为 G [S] 的句型？试证明你的结论。

　　⑶ (3 分 ) 试画出 001B 关于 G [S] 的语法树。

二、请构造一个文法，使其产生这样的表达式 E ：表达式中只含有双目运算符 + 、 \* ，且 + 的优先级高于 \* ， + 采用右结合， \* 采用左结合，运算对象只有标识符 i ，可以用括号改变运算符优先级。要求给出该文法的形式化描述。

三、设有语言 L={ α | α∈ {0,1} + ，且α不以 0 开头，但以 00 结尾 } 。

　　⑴试写出描述 L 的正规表达式；

　　⑵构造识别 L 的 DFA （要求给出详细过程，并画出构造过程中的 NDFA 、 DFA 的状态转换图，以及 DFA 的形式化描述 ) 。

四、给定文法 G[S] ：

　　S → AB

　　A → aB | bS | c

　　B → AS | d

　　⑴ (6 分 ) 请给出每一个产生式右部的 First 集；

　　⑵ (3 分 ) 请给出每一个非终结符号的 Follow 集；

　　⑶ (8 分 ) 请构造该文法的 LL(1) 分析表；

　　⑷ (8 分 ) 什么是 LL(1) 文法？该文法是 LL(1) 文法吗？为什么？

五、给定文法 G[S] ：

　　S → SaA|a

　　A → AbS|b

　　⑴请构造该文法的以 LR(0) 项目集为状态的识别规范句型活前缀的 DFA 。

　　⑵请构造该文法的 LR(0) 分析表。

　　⑶什么是 LR(0) 文法？该文法是 LR(0) 文法吗？为什么？

　　⑷什么是 SLR(1) 文法？该文法是 SLR(1) 文法吗？为什么？

六、给定下列语句：

　　if a+b>c

　　then x := a\*(b-c) + (b\*c-d)/e

　　⑴写出其等价的逆波兰表示；

　　⑵写出其等价的四元式序列。

七、已知下列 C 语言程序：

　　int \* f()

　　{ 　int a = 100; return &a;

　　}

　　main()

　　{ 　int \* i = f();

　　　　char a[] = “compiler”; printf(“the result is %d\n”, \*i);

　　}

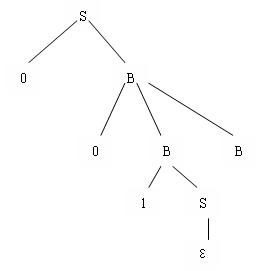
　　程序运行结果为： the result is 26157,

　　请解释为什么程序运行的结果不是期望的“ the result is 100 ”？

1.1 三个 0 和 1 数量相等的串

1.2 S => 1A => 11AA => 11A 0S

1.3



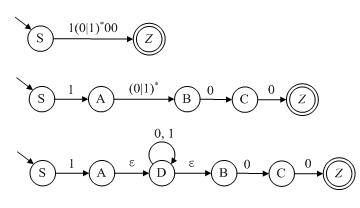
第二题 构造文法如下 :

G[E]=({+,\*,(,),i}, {E,F,T}, P, E) ， 其中 P 为：

E→E\*F|F F→T+F|T T→(E)|i

第三题 （ 1 ）正规表达式： 1(0|1) \* 00

（ 2 ）第一步：将正规表达式转换为 NDFA

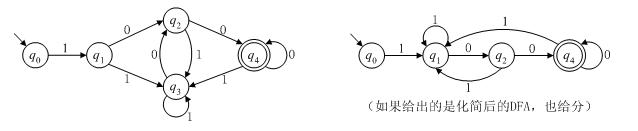


第二步：将 NDFA 确定化为 DFA ：

造表法确定化（ 3 分） 确定化后 DFA M 的状态转换表 (2 分 )

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 状态 输入 | I 0 | I 1 |  | **t** | 0 | 1 |
| [S] | — | [A,D,B] |  | *q* 0 | — | *q* 1 |
| [A,D,B] | [D,B,C] | [D,B] | 重新命名 | *q* 1 | *q* 2 | *q* 3 |
| [D,B,C] | [D,B,C,Z] | [D,B] |  | *q* 2 | *q* 4 | *q* 3 |
| [D,B] | [D,B,C] | [D,B] |  | *q* 3 | *q* 2 | *q* 3 |
| [D,B,C,Z] | [D,B,C,Z] | [D,B] |  | *q* 4 | *q* 4 | *q* 3 |

DFA 的状态转换图（ 3 分）



第三步：给出 DFA 的形式化描述

DFA M = （ { *q* 0 , *q* 1 , *q* 2 , *q* 3 , *q* 4 }, {0,1}, t, *q* 0 , { *q* 4 } ）

t 的定义见 M 的状态转换表。

第四题

（ 1 ） First(AB) = {a, b, c}

First(aB) = {a}

First(bS) = {b}

First(c) = {c}

First(AS) = {a, b, c}

First(d) = {d}

（ 2 ） Follow(S) = {#, a, b, c, d}

Follow(A) = {a, b, c, d}

Follow(B) = {#, a, b, c, d}

（ 3 ） LL(1) 分析表（ 8 分）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| V N | V T | | | | |
| a | b | c | d | # |
| S | S ? AB | S ? AB | S ? AB |  |  |
| A | A ? aB | A ? bS | A ? C |  |  |
| B | B ? AS | B ? AS | B ? AS | B ? d |  |

（ 4 ）对于文法 G 的每一个非终结符 U 的产生式 U ? α 1 |α 2 |…|α n ，

如果 SELECT(U ? α i ) ? SELECT(U ? α j ) = ? （ i≠j, i,j=1, 2, …, n ），

则文法 G 是一个 LL(1) 文法。 该文法是 LL(1) 文法。

因为 SELECT(A ? aB) ? SELECT(A ? bS) ? SELECT(A ? C) = ?

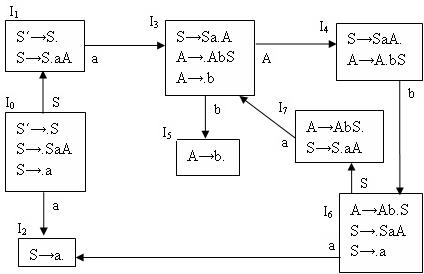
SELECT(B ? AS) ? SELECT(B ? d) = ?

第五题 ⑴拓广文法 1 分

G[S ′ ]: S ′→ S ⑴

S → SaA ⑵ S → a ⑶ A → AbS ⑷ A → b ⑸

该文法的以 LR(0) 项目集为状态的识别规范句型活前缀的 DFA ：



⑵ 该文法的 LR(0) 分析表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 状态 | ACTION | | | GOTO | |
| a | b | # | S | A |
| 0 | S 2 |  |  | 1 |  |
| 1 | S 3 |  | acc |  |  |
| 2 | r 3 | r 3 | r 3 |  |  |
| 3 |  | S 5 |  |  | 4 |
| 4 | r 2 | r 2 /S 6 | r 2 |  |  |
| 5 | r 5 | r 5 | r 5 |  |  |
| 6 | S 2 |  |  | 7 |  |
| 7 | r 4 /S 3 | r 4 | r 4 |  |  |

⑶ LR(0) 文法：该文法的以 LR(0) 项目集为状态的识别规范句型活前缀的 DFA 中没有冲突状态。

该文法不是 LR(0) 文法

因为存在冲突状态： I 4 和 I 7

⑷ SLR(1) 文法：该文法的以 LR(0) 项目集为状态的识别规范句型活前缀的 DFA 中有冲突状态，冲突可用 FOLLOW 集解决。

该文法不是 SLR(1) 文法。

因为 FOLLOW(S)={a,b,#} ，所以无法解决冲突

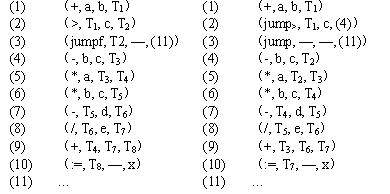
第六题

（ 1 ） (1) ab+c> (23) jumpf

(8) xabc-\*bc\*d-e/+:=

(23) ...

（ 2 ）



第七题

C 语言采用栈式存储分配方法作为其运行环境；

f() 返回的是指向其活动记录某一位置的指针；

f() 返回后，其活动记录被释放，并且，其对应的存储空间被数组 a 占用，

再次引用该指针时，其结果由于对回收的活动记录所占用的内存空间的再分配，其所指的值

发生了改变。

释放在前，引用在后的现象称 : Dangling Reference 。