

警示

《中山大学授予学士学位工作细则》第
八条：“考试作弊者，不授予学士学位。”

中山大学数据科学与计算机学院
2017 年秋季学期

本科生《编译原理》课程期中考试

班级： 2015 级 专业： 软件工程专业（嵌软、通软方向） 任课教师： 李文军

考试时间： 120 分钟 考试形式： 闭卷（仅允许自带 1 张记有任何信息的 A4 纸）

注意：所有答案必须写在答题纸上！考试后请自己保存试卷留作纪念。

Part I、导论（共 2 题，20 分）

1、（10 分）给出以下文法：

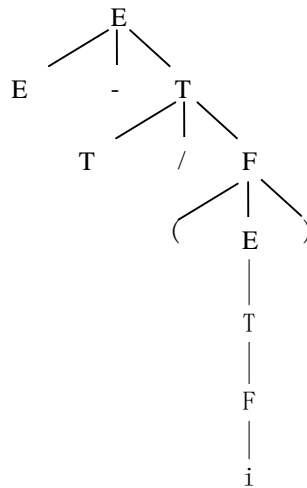
$$\begin{aligned} E &\rightarrow T \mid E+T \mid E-T \\ T &\rightarrow F \mid T * F \mid T / F \\ F &\rightarrow (E) \mid i \end{aligned}$$

(1)（5 分）证明 $E-T/(i)$ 是它的一个句型。

(2)（5 分）指出这个句型的所有短语（Phrase）、直接短语（Simple Phrase）和句柄（Handle, i.e. Left-Most Simple Phrase）。

【参考答案】

(1) 构造该句子的一个推导或画出该句子的分析树如下：



(2) 这个句型的短语有：i, (i), T/(i), E-T/(i)；直接短语:i；句柄：i。

2、（10 分）考虑如下文法：

$E \rightarrow T \mid T + E$
 $T \rightarrow \text{int} \mid \text{int} * E$

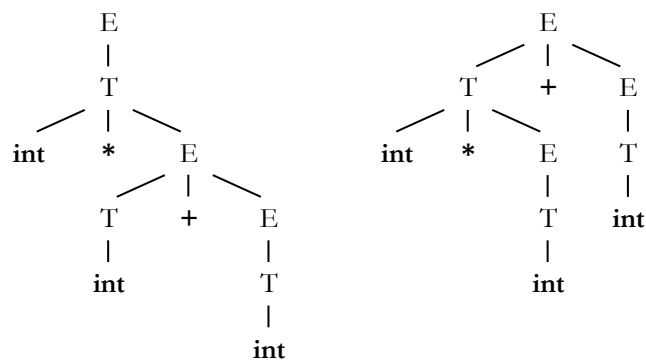
注意该文法是二义的。

(1) （5 分）试找出一个引起分析时产生二义性的输入串，并画出该输入串的两棵分析树。

(2) （5 分）写出一个与上述文法等价的无二义性文法，即两个文法产生的语言是相同的。

【参考答案】

(1) 可有多种答案。例如，输入串 **int * int + int** 可画出两棵不同的分析树：



(2) 等价的无二义性文法如下：

$E \rightarrow T \mid T + E$
 $T \rightarrow \text{int} \mid \text{int} * T$

Part II、词法分析（共 2 题，40 分）

3、（10 分）某程序员拟采用类似 URL（Uniform Resource Locator）风格的对象定位方式：

[<class>:] [//<address>/] <path>

其中，方括号[]表示任选部分；尖括号<>表示一种模式，如下所述：

- 标识符<id>是 1 个或多个小写字母、大写字母、数字、“_”（下划线）、“\$”和“-”（减号）组成的序列，并且不允许以数字或“-”（减号）开头。
- 类<class>是 1 个标识符。
- 地址<address>是 1 个或多个标识符的列表，标识符之间以“.”（小数点）分隔。
- 路径<path>是 1 个或多个标识符的列表，标识符之间以“/”分隔。

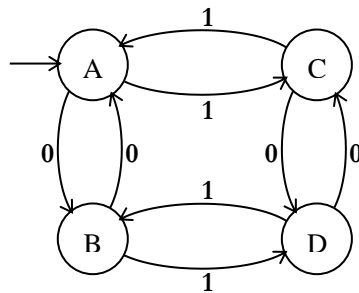
请写出描述上述 URL 的正则定义式（Regular Definition），正则定义式中可使用 lex 风格的简写记号。

【参考答案】

正则定义式如下：

```
id = [a-zA-Z_$][-a-zA-Z0-9_$]*  
class = { id }  
address = { id }(. { id })*  
path = { id }(/ { id })*  
url = (( { class } :) + epsilon) ((// { address } /) + epsilon) { path }
```

4、（30 分）设字母表{0,1}上的有限自动机（Finite Automaton）如下：



其中，A 是初始状态，但目前暂时还没有任何终结状态。

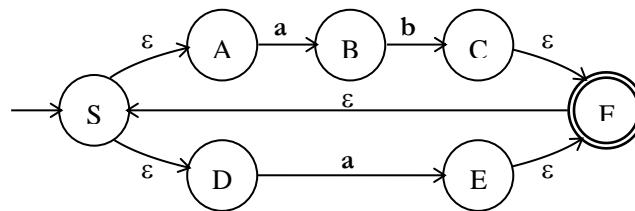
(1) （2 分）该自动机是否一个确定的有限自动机（DFA）？为什么？

(2) （5 分）为让该自动机识别含有偶数个 1（含零个 1）的所有串，应将该自动机中的哪些状态改为终结状态？

(3) （6 分）为让该自动机识别长度为奇数的所有串，应将该自动机中的哪些状态改为终结状态？

(4) （7 分）构造一个识别字母表{0,1}上至少含一个 0 且至少含有一个 1 的 DFA，并解释该 DFA 的每一个状态的直观含义。注意不要忘记标注 DFA 的初始状态！

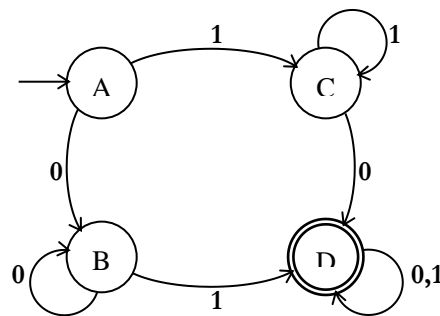
(5) (10 分) 设有字母表{a, b}上的 NFA 如下:



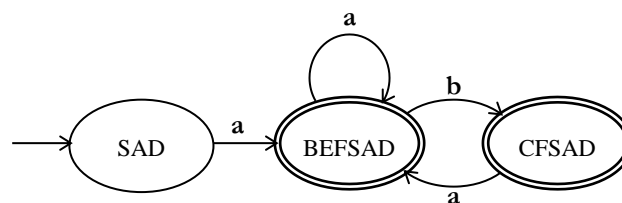
试将该 NFA 转换为等价的 DFA，并在 DFA 状态中标明它对应的原 NFA 状态的子集。注意不要忘记标识 DFA 的初始状态！

【参考答案】

- (1) 是一个 DFA，因为它既没有 ϵ 转移，所有状态也不存在相同符号的多个射出弧。
- (2) 将状态 A 和状态 B 均改为终结状态。
- (3) 将状态 B 和状态 C 均改为终结状态。
- (4) 状态 A 表示当前尚未看到任何的 0 或 1；状态 B 表示至少已经有一个 0，但还没有 1 出现；状态 C 表示至少已经有了一个 1，但还没有 0 出现；状态 D 表示 0 和 1 都有了。



(5) 确定化结果如下:



Part III、语法分析 (共 3 题, 40 分)

5、(15 分) 设有如下定义格言的文法:

$S \rightarrow F V$
 $F \rightarrow P \text{ makes } | L \text{ and } P \text{ make}$
 $L \rightarrow P | L, P$
 $P \rightarrow \text{truth} | \text{love} | \text{courage}$
 $V \rightarrow \text{honesty} | \text{compassion} | \text{valor} | \text{justice} | \text{sacrifice}$
 $\quad | \text{honor} | \text{spirituality} | \text{humility}$

(1) (4 分) 求该文法所有非终结符号的 FIRST 和 FOLLOW 集。

(2) (5分) 该文法是 LL(1) 的吗? 如果试图构造其 LL(1) 分析表, 会出现什么问题?

(3) (6分) 消除该文法中的左递归, 然后再为之构造 LL(1) 分析表, 此时是一个合法的 LL(1) 分析表吗?

【参考答案】

(1) 该文法所有非终结符号的 FIRST 和 FOLLOW 集为:

$\text{FIRST}(S) = \text{FIRST}(F) = \text{FIRST}(L) = \text{FIRST}(P) = \{ \text{truth}, \text{love}, \text{courage} \}$

$\text{FIRST}(V) = \{ \text{honesty}, \text{compassion}, \text{valor}, \text{justice}, \text{sacrifice}, \text{honor}, \text{spirituality}, \text{humility} \}$

$\text{FOLLOW}(S) = \text{FOLLOW}(V) = \{ \$ \}$

$\text{FOLLOW}(F) = \text{FIRST}(V)$

$\text{FOLLOW}(P) = \{ \text{makes}, \text{make}, , \}$

$\text{FOLLOW}(L) = \{ \text{and}, , \}$

(2) 不是, 因为 L 的产生式存在直接左递归。构造其 LL(1) 分析表结果如下:

	truth	honesty	makes	make	and	,	...
S	F	V					
F	P makes L and P make						
V		honesty					
P	truth						
L	P L, P						

由于上述分析表中存在一个单元有多个候选产生式的情况, 因而不是一个真正的 LL(1) 分析表。

(3) 消除左递归后, 文法改写为:

$S \rightarrow F V$

$F \rightarrow P \text{ makes} \mid L \text{ and } P \text{ make}$

$L \rightarrow P L'$

$L' \rightarrow , P L' \mid \varepsilon$

$P \rightarrow \text{truth} \mid \text{love} \mid \text{courage}$

$V \rightarrow \text{honesty} \mid \text{compassion} \mid \text{valor} \mid \text{justice} \mid \text{sacrifice}$

$\mid \text{honor} \mid \text{spirituality} \mid \text{humility}$

相应地, 文法非终结符号的 FIRST 和 FOLLOW 集为 (其余同旧):

$\text{FIRST}(L') = \{ , , \varepsilon \}$

$\text{FOLLOW}(L') = \text{FOLLOW}(L) = \{ \text{and} \}$

构造其 LL(1) 分析表结果如下:

	truth	honesty	makes	make	and	,	...
S	F	V					
F	P makes L and P make						

V		honesty					
P	truth						
L	P L'						
L'	P L'				ϵ	, P L'	

由于上述分析表中存在一个单元有多个候选产生式的情况，因而仍然不是一个真正的 LL(1) 分析表。

6、（10 分）某文法所有符号的 FIRST 集和 FOLLOW 集如下（终结符号的 FOLLOW 集含义与非终结符号的 FOLLOW 集相同）：

符号	FIRST 集	FOLLOW 集
X	b d f	\$
Y	b d	c e
Z	c e	a
a	a	\$
b	b	b d
c	c	c e
d	d	c e
e	e	a
f	f	\$

已知该文法的终结符号集为{ a, b, c, d, e, f }，非终结符号集为{ X, Y, Z }；该文法的每个非终结符号有且仅有两个产生式，并且其中不包含 ϵ 产生式。试给出该文法的所有产生式。

【参考答案】

可采用多种不同方式解决这一问题。譬如下面首先基于 FIRST 集构造明显符号首个终结符号的产生式：

$X \rightarrow b \mid d \mid f$
 $Y \rightarrow b \mid d$
 $Z \rightarrow c \mid e$

由于每个非终结符号仅有两个产生式，故需要减少 X 的产生式。考虑到 b 和 d 都在 Y 的 FIRST 集中，可尝试将 X 的产生式修改为：

$X \rightarrow Y \mid f$

此时非终结符号的 FIRST 集都正确了，转而尝试修改产生式以满足 FOLLOW 集。例如观察符号 c，注意到 c 和 e 都在 FOLLOW(Y)和 FIRST(Z)中，可将 Z 加到某一产生式中 c 的后面，譬如：

$X \rightarrow Y \mid f$
 $Y \rightarrow b \mid d$
 $Z \rightarrow cZ \mid e$

以同样的方式处理 FOLLOW(b)，可得：

$X \rightarrow Y \mid f$
 $Y \rightarrow bY \mid d$

$Z \rightarrow cZ \mid e$

又由于\$不是 FOLLOW(Y)中，故必须在产生式 $X \rightarrow Y$ 后面加上些符号。由于 FIRST(Z)都在 FOLLOW(Y)中，尝试：

$X \rightarrow YZ \mid f$

$Y \rightarrow bY \mid d$

$Z \rightarrow cZ \mid e$

此时还未使用符号 **a**。由于 $\text{Follow}(\mathbf{a}) = \{ \$ \}$ 和 $\text{Follow}(\mathbf{Z}) = \{ \mathbf{a} \}$ ，可得：

$X \rightarrow YZ\mathbf{a} \mid f$

$Y \rightarrow bY \mid d$

$Z \rightarrow cZ \mid e$

此时所有要求均已满足，故上述产生式即最终的文法。

7、（15 分）考虑两个 LL(1)文法，每个都只有一个非终结符号（设第一个文法的非终结符号为 **A**，第二个文法的非终结符号为 **B**），且不含 ϵ 产生式。考察第三个文法，该文法的开始符号为 **S**，并且产生式通过合并上述两个文法的所有产生式得到，然后再添加以下产生式：

$S \rightarrow A \mid B$

请回答以下问题并解释理由：

- (1) （2 分）在第三个文法中，是否可能从文法的开始符号 **S** 推导出空串？为什么？
- (2) （6 分）为保证第三个文法是 LL(1)的，原先的两个文法必须满足的充分必要条件是什么？请不要给出 **A** 或 **B** 的具体例子，你需要描述的是针对上述形式的所有文法均须满足的充要条件。
- (3) （7 分）为保证第三个文法是无二义的，原先的两个文法必须满足的充分必要条件是什么？请不要给出 **A** 或 **B** 的具体例子，你需要描述的是针对上述形式的所有文法均须满足的充要条件。

【参考答案】

- (1) 不可能，因为从 **A** 或 **B** 均不可能推导出空串。
- (2) 充要条件：**A** 和 **B** 能够推出的首字符不可相同，即 $\text{FIRST}(\mathbf{A}) \cap \text{FIRST}(\mathbf{B}) = \emptyset$ 。注意，题目已指出这两个文法是 LL(1)的，且 **A** 或 **B** 均不可能推导出空串。
- (3) 充要条件：这两个文法产生的语言的交集为空，即 $L(\mathbf{A}) \cap L(\mathbf{B}) = \emptyset$ ，其中 $L(\mathbf{X})$ 表示由文法 **X** 产生的语言。注意，由于题目已指出这两个文法是 LL(1)的，故它们一定是无二义的。

Part III、附加题（共 1 题，10 分）

8、（10 分）考察语言 $L = \{ \alpha c \alpha \mid \alpha \in (\mathbf{a} \mid \mathbf{b})^* \}$ ，该语言的每一个合法串的前后是相同的 **a** 和 **b** 组成的子串，中间由一个 **c** 分隔，例如 **aabcaab**、**abaacabaa**。有学者证明了该语言不是一个上下文无关语言。

- (1) （5 分）语言 L 是程序设计语言中什么问题的抽象？

(2) (5 分) 该语言不是上下文无关的, 这对于我们使用 BNF 定义程序设计语言的语法规则有什么启示?

【参考答案】

(1) 该语言是关于强类型程序设计语言的源程序中标识符“先声明、后使用”规则的抽象, $\alpha c \alpha$ 的前一个 α 代表标识符 α 的声明, 后一个 α 代表该标识符的使用。

(2) 由于 BNF 等价于上下文无关文法, 这意味着我们使用 BNF 定义一门语言的语法规则时, 无法表达标识符的“先声明、后使用”规则, 故对这一约束的检查必须推迟到语义分析阶段执行。

□