编译作业参考答案 第二次作业

部分题目为了可读性，没把字面量用单引号包围。

P28 1

已知文法 $G[<\text{标识符}>]$：

```
<标识符> ::= 'a' | 'b' | 'c' | <标识符>'a' | <标识符>'c' | <标识符>'0' | <标识符>'1'
```

写出 V_t 和 V_n，并对符号串 `a`、`ab0`、`a0c01`、`0a`、`11`、`aaa` 给出可能的一些推导。

答案解析

$V_t = \{a, b, c, 0, 1\}$

$V_n = \{<\text{标识符}>\}$

终结符号中，根据规则可以发现，对于此文法的**合法句子**，`b` 只可能出现在句子的最左侧，`0` 和 `1` 不可能出现在最左侧。因此不满足这两个条件的符号串（`ab0` 、`0a` 、`11`），**都不是此文法的合法句子，因此不可能被推导出来**。

注意到原文法是左递归文法，比较适合进行最右推导。

推导过程略。

批改标准

- V_n 和 V_t 有一项写错，视为错误
- 将不是合法句子的符号串写出推导过程，视为错误
- 将是合法句子的符号串说明为不合法，视为错误
- 缺少合法句子的推导，视为错误
- 不管使用什么神奇的推导方式，推出来就行

P28 2

写一文法，使其语言是偶整数的集合

注：这道题下面的第 3 题，不允许有前导 0，因此这道题可以。

答案解析

使用 BNF，只写规则：

```
<even-number> ::=
    <optional-sign><optional-digit-seq><even-digit>
<optional-sign> ::=
    <sign> | ε
<sign> ::=
    '+' | '-'
<optional-digit-seq> ::=
    <digit-seq> | ε
<digit-seq> ::=
      <digit>
    | <digit-seq><digit>
<even-digit> ::=
    '0' | '2' | '4' | '6' | '8'
<digit> ::=
    '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```

批改标准

- 使用其他文法描述方式（四元式、EBNF），只要不影响正常的理解，不视为错误
- 有没有前导 0，不考虑
- 是否可以推导负偶数，不考虑（事实上 C 的字面量都不带符号，符号作为单目运算符，是表达式的组分）
- 文法可以推导出奇数，视为错误
- 文法不能推导出某些正偶数，视为错误

P28 4

设文法 G:

```
<A> ::= 'b'<A> | 'cc'
```

试证明：cc , bcc , bbcc , bbbcc $\in L[G]$

答案解析

如果文法能够推导出某符号串，那么这个符号串是该文法定义的合法句子，属于该文法定义的语言。

太简单了，推导过程省略（推荐使用最左推导）。

批改标准

- 不管使用什么神奇的推导方式，推出来就行

P28 5

试对如下语言构造相应的文法:

(1) $\{a(b^n)a \mid n = 0, 1, 2, 3, \cdots\}$

(2) $\{(a^n)(b^n) \mid n = 1, 2, 3, \cdots\}$

其中左右圆括号均为终结符。

答案解析

使用 BNF:

(1)

```
<S> ::=
    'a('<optional-b-seq>')a'
<optional-b-seq> ::=
    <b-seq> | ε
<b-seq> ::=
      'b'
    | <b-seq>'b'
```

(2)

```
<S> ::=
    '('<strange-seq>')'
<strange-seq> ::=
     'a)(b'
    |'a'<strange-seq>'b'
```

批改标准

- 由于题面明确指出了圆括号是终结符，因此没有将其视为终结符的都视为错误
- 使用 EBNF 作答时，必须把圆括号用单引号转义，否则算错
- 文法推导出的语言应当和题目要求一致
- 如果把第二问的两个上标 n 理解为了可以取不同的值，写出了如下文法，那也算对:

```
<S> ::=
    '('<a-seq>')('<b-seq>')'
<a-seq> ::=
      'a'
    | <a-seq>'a'
<b-seq> ::=
```

```
      'b'
    | <b-seq>'b'
```

P28 6

有文法 $G3[<\text{表达式}>]$:

```
<表达式> ::= <项> | <表达式>'+'<项> | <表达式>'-'<项>
<项> ::= <因子> | <项>'*'<因子> | <项>'/'<因子>
<因子> ::= '('<表达式>')' | 'i'
```

试给出下列符号串的推导：`i`、`(i)`、`i*i`、`i*i+i`、`i*(i+i)`

答案解析

最左最右推导都可以，推导过程略。

批改标准

- 不管使用什么神奇的推导方式，推出来就行

P28 7

对上一题的文法，列出句型<表达式>+<项>*<因子>的所有短语和简单短语

答案解析

根据短语的定义：

如果对于 $U \in V_n, u \in V^+, x \in V^*, y \in V^*$，有 $w = xuy \in L(G[Z]), Z \overset{*}{\Rightarrow} xUy, U \overset{+}{\Rightarrow} u$，则称 u 是一个相对于非终结符 U 的句型 w 的短语。如果 $U \Rightarrow u$，则是简单短语。

因此只要枚举 w 的子串，尝试对其进行规约即可（或是画出语法树，直接枚举所有子树的根节点，此方法的例子详见下面的 P36 5）。

需要注意的是，短语的定义中并没有明确指出必须是最右推导，因此如果存在多种不同的推导方式，得到的短语集合可能也不同，而所有的可能性都应该被列举出来。

推导出本题的句型的过程是唯一的：

```
    <表达式>
=> <表达式>+<项>
=> <表达式>+<项>*<因子>
```

根据短语的定义：

U	x	u（短语）	y	简单短语？（N/Y）
<表达式>	ε	<表达式>+<项>*<因子>	ε	N
<项>	<表达式>+	<项>*<因子>	ε	Y

注意：**简单短语也是短语**。

因此：

- 短语：<表达式>+<项>*<因子>、<项>*<因子>
- 简单短语：<项>*<因子>

P36 1

给定文法 $G[E]$：

```
E ::= RP | P
P ::= '('E')' | 'i'
R ::= RP'+' | RP'*' | P'+' | P'*'
```

（1）证明 `i+i*(i+i)` 是文法 G 的句子

（2）画出该句子的语法树

答案解析

（1）使用最右推导即可：

```
   E
=> RP
=> R(E)
=> R(RP)
=> R(Ri)
=> R(P+i)
=> R(i+i)
=> RP*(i+i)
=> Ri*(i+i)
=> P+i*(i+i)
=> i+i*(i+i)
```

从开始符号 `E` 能推出来 `i+i*(i+i)`，说明是此文法的句子

（2）根据推导直接构造树即可：

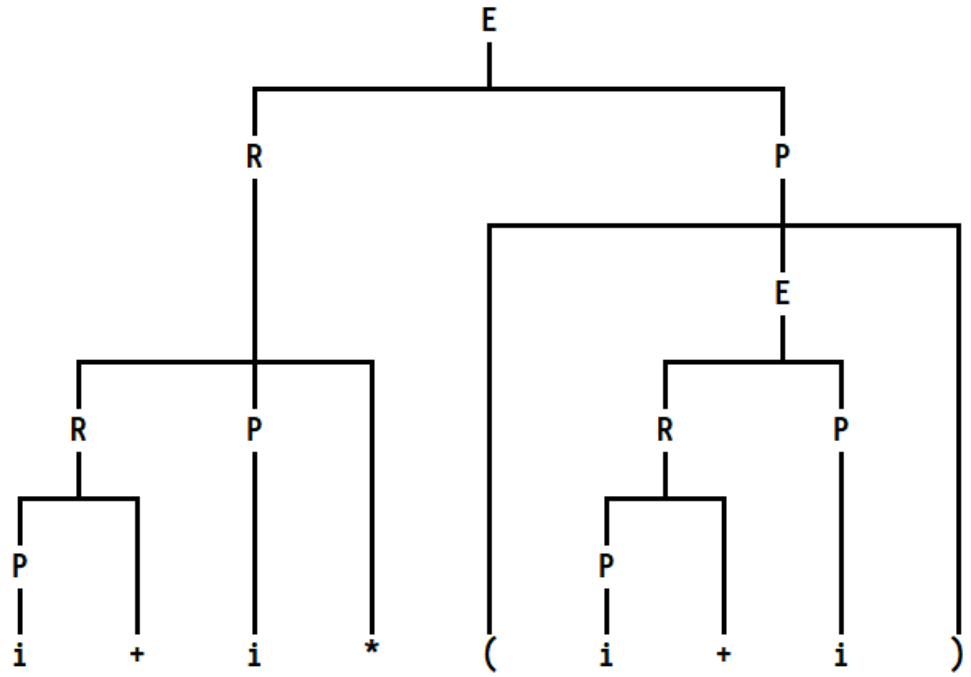

Figure 1: P36.1 语法树

批改标准

- 不管使用什么神奇的推导方式，推出来就行
- 不管使用什么神奇的推导方式，树画对就行

P36 5

已知文法 $G[E]$:

```
E ::= ET'+' | T
T ::= TF'*' | F
F ::= FP'↑' | P
P ::= '('E')' | 'i'
```

有句型 TF*PP↑+ ，问此句型的短语、简单短语和句柄是什么?

答案解析

注意：**句柄也是简单短语**

本题的句型比较复杂，因此直接通过推导得出答案比较难，建议采用枚举子串并规约得到语法树的方式。

注意到 *、↑、+ 这三个终结符出现的规则是唯一的，因此可以推测:

- TF*，由规则 `T ::= TF'*'`直接推导得到，将 TF*PP↑+ 规约为 TPP↑+
- PP↑，由规则 `F ::= FP'↑'`多步推导得到，将 TPP↑+ 规约为 TFP↑，规约为 TF+
- TF+，由规则 `E ::= ET+` 多步推导的到，将 TF+ 规约为 TT+（或 ET+），规约为 ET+，规约为 E

得到语法树:

根据语法树和推导的定义，容易得知句型 w 的语法树中:

- 对于树中的任意节点 U，若以其为根节点的子树的高度不为 0，将此子树的所有叶节点连接得到串 u:

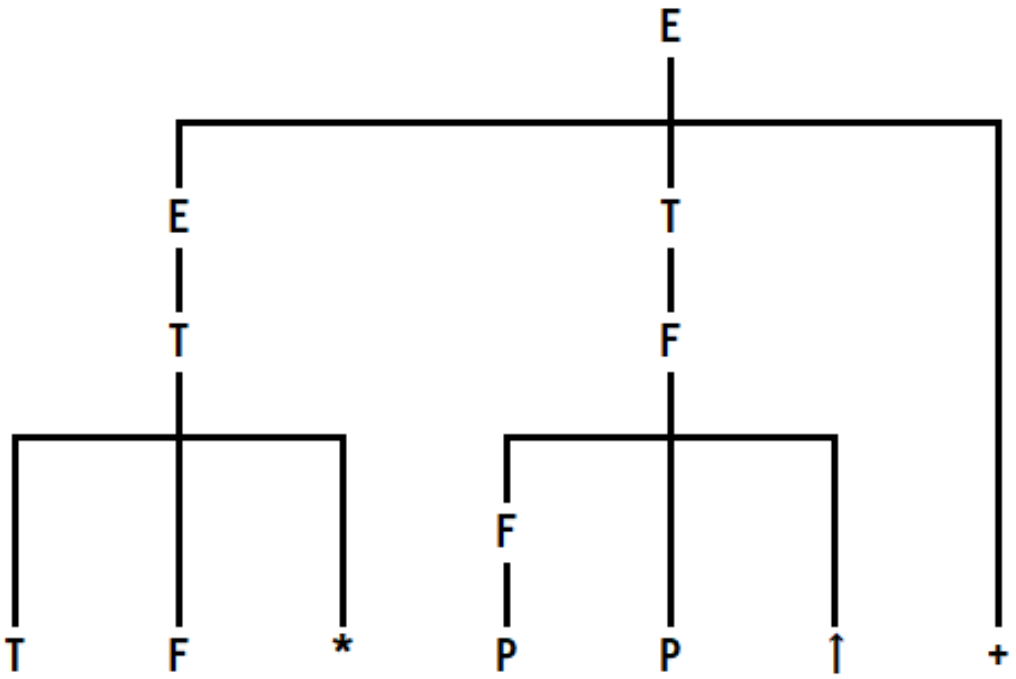

Figure 2: P36.5 语法树

- u 是相对于 U 的 w 的短语
- 如果此子树的高度为 1，那么 u 是相对于 U 的 w 的简单短语
- 如果在所有高度为 1 的子树中，此子树的最左叶节点在原符号串中的下标最小，那么 u 是 w 的句柄

U	x	u（短语）	y	短语/简单短语/句柄
E	ε	TF*PP↑+	ε	短语
E	ε	TF*	PP↑+	短语
T	ε	TF*	PP↑+	句柄/简单短语/短语
T	TF*	PP↑	+	短语
F	TF*	PP↑	+	短语
F	TF*	P	P↑+	简单短语/短语

需要注意的是：**短语不限制推导的具体过程，因此在一次推导过程中，你可能无法同时得到所有短语**。

另外注意：**简单短语也是短语，句柄也是简单短语，句柄如果有则一定是唯一的**。

简单来写的话：

- 短语：TF*PP↑+、TF*、PP↑、P
- 简单短语：TF*、P
- 句柄：TF*

批改标准

- 完整列举出所有短语、简单短语、句柄

P36 6

分别对 `i+i*i` 和 `i+i+i` 中的每一个句子构造两颗语法树，从而证明下述文法 $G[<\text{表达式}>]$ 是二义性的：

```
<表达式> ::= 'i' | '('<表达式>')' | <表达式><运算符><表达式>
<运算符> ::= '+' | '-' | '*' | '/'
```

答案解析

分别找出两种最右推导（有两种最右推导也是有二义性的判据），根据推导构造树即可，推导过程略：

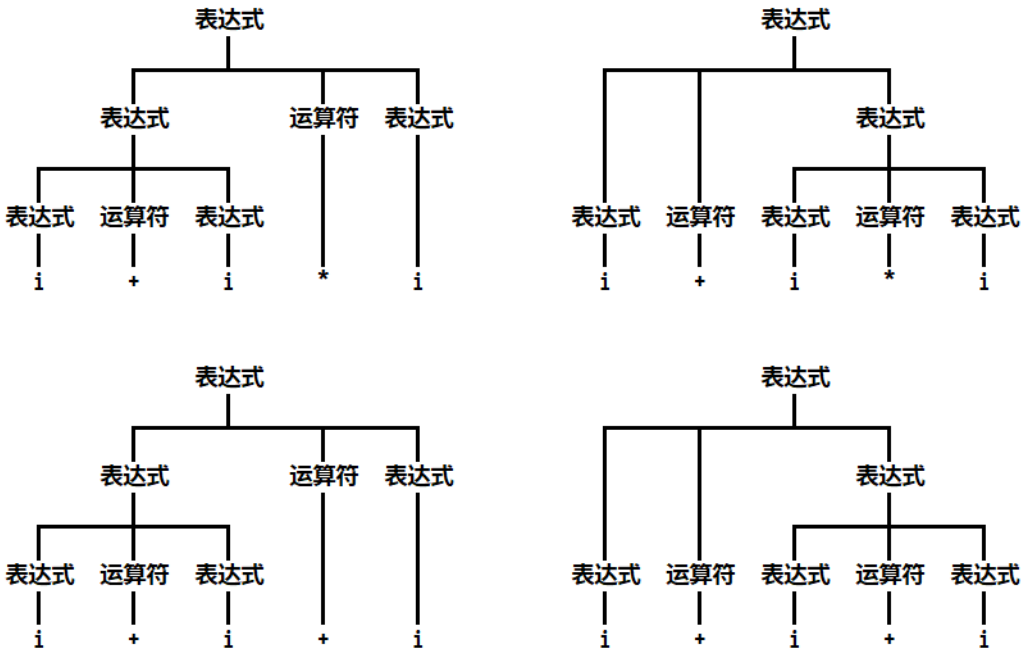

Figure 3: P36.6 语法树

题外话

其实这个是经典的算术表达式运算符优先级的问题，可以通过改写文法消除这种二义性：

```
<expr>   ::= <term>   | <expr><additive-operator><term>
<term>   ::= <factor> | <factor><multiplicative-operator><factor>
<factor> ::= 'i'      | '('<expr>')'

<additive-operator>       ::= '+' | '-'
<multiplicative-operator> ::= '*' | '/'
```

批改标准

- 此题答案是唯一的
- 可以不给出推导过程，树画对就行

P36 8

证明下面的文法 $G[S]$ 是二义的：

```
S ::= 'i'S'e'S | 'i'S | 'i'
```

答案解析

使用最右推导：

```
   S
=> iSeS
=> iSei
=> iiSei
=> iiiei

   S
=> iS
=> iiSeS
=> iiSei
=> iiiei
```

`iiiei` 存在两种最右推导，因此有二义性。

题外话

其实这个是经典的 if-else 语句匹配问题：

```
<statement> ::=
      'if' <condition> <statement> 'else' <statement>
    | 'if' <condition> <statement>
    | <other-statement>
```

在 C 语言中，解决这个二义性的办法是在推导过程中遵循**最大吞噬**规则，即总是尽可能匹配更长的候选项（比如这里的第一个候选项）；表现出来的是：else 总是匹配位置上更近的 if。

批改标准

- 如果使用**存在句子有多棵语法树**证明，那么语法树不能有错
- 如果使用**存在句子有多种最右推导（或最左推导）**证明，那么两个推导都必须严格遵循最右推导（或最左推导），且推导过程不能有错
- 如果使用**存在句子有多种最左规约（该句子的任意规范句型的句柄不唯一）**证明，那么两个推导都必须严格遵循最左规约，且规约过程不能有错（指出的句柄不能有错）

P36 9

有文法 $G[N]$:

```
N ::= SE | E
E ::= '0' | '2' | '4' | '6' | '8' | '10'
S ::= SD | D
D ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```

举例说明此文法有二义性。

此文法描述的是什么语言?

尝试写出另一文法 G'，其定义的语言和 G 相同，但是没有二义性。

答案解析

推导 `10`:

```
   N
=> E
=> '10'

```

```
   N
=> SE
=> S'0'
=> D'0'
=> '10'
```

两种最右推导，有二义性。

考虑使用 EBNF 改写规则：

- `D` 是数字
- `S::=D{D}`, 是长度大于 0 的数字串
- `E` 是偶数字和 `10`
- `N::=[S]E` => `N::=[D{D}]E` => `N::={D}E`, 是可以为空的数字串后跟一个偶数字

因此 `N` 定义的语言是**可以有前导 0 的偶自然数**。

考虑到二义性是 `E ::= '10'`引入的：

`10` 可以视为 `DE`，也是 `SE`，可以由 `N` 推导得出，因此 `E` 即使无法直接推导出 `10`，也不影响 $G[N]$ 定义的语言。删除该规则即可：

```
N ::= SE | E
E ::= '0' | '2' | '4' | '6' | '8'
S ::= SD | D
D ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```

很遗憾，**没有算法可以判定文法是否有二义性**，因此这里无法给出证明。

批改标准

- 举例要完整包括：句子、推导（或规约，或语法树）
- 写出的新文法和参考答案等价即可

P41 2

设文法 $G[<$ 目标 $>]$:

```
<目标> ::= V1
V1 ::= V2 | V1'i'V2
V2 ::= V3 | V2'+'V3 | 'i'V3
V3 ::= ')'V1'*' | '('
```

试分析句子 `(`、`)(*`、`i(`、`(+(`、`(+(i`、`(+)(i*i(` 。

答案解析

`(`、`)(*`、`i(`、`(+(` 是此文法的合法句子，推导过程略。

`(+(i`、`(+)(i*i(` 不是此文法的合法句子。

批改标准

- 能推导出来的，要给出正确的推导过程