第六章

1. 判断：使用产生规则可以定义一个程序的意义。（错）语义规则
2. 判断：终结符只有继承属性，它由词法分析器提供。（错）综合属性
3. 判断：非终结符可以有综合属性，但不能有继承属性。（错）都可以有
4. 判断：属性值在分析过程中可以进行计算，但不能传递。（错）可以传递，由语义规则决定
5. 判断：S-属性文法的翻译器只可借助于LR分析器实现。（错）通常可借助于LR分析器实现，还可以用其他分析器实现，例如还可以用LL分析器实现
6. 判断：语法制导翻译只可借助于自下而上分析法实现。（错）可以用其他分析法实现，例如还可以用自上而下分析法
7. 判断：使用语法制导翻译方案的编译程序能同时进行语法分析和语义分析。（对）
8. 判断：语法制导翻译中，在各个非终结符之前放置语义规则来计算它的继承属性，并在产生式后端放置语义规则计算综合属性。（对）
9. 在语法制导翻译中，非终结符可以具有综合和继承属性，终结符具有综合属性。
10. 已知文法G[S]和对应的语义规则

S’ → S {print(S.val)}

S → (L) {S.val = L.val +1}

S → a {S.val = 0}

L →L1, S {L.val = L1.val + S.val}

L →S {L.val = S.val}

若输入为( a,( a))，且采用自下而上的分析方法，则输出为( C )。

1. 0 B. 1 C. 2 D. 4

找到( a,( a))的最右推导：1.S’⇒2.S⇒3.(L)⇒4.(L1, S)⇒5.(L1, (L))⇒6.(L1, (S))⇒7.( L1, (a))⇒8.(S, (a))⇒9.(a, (a))。

规范规约是最右推导的逆过程，按语义规则计算综合属性：

8. S.val = 0; S.val = 0

7. L1.val = 0; L.val = S.val

6. L1.val = 0, S.val = 0; S.val = 0

5. L1.val = 0, L.val = 0; L.val = S.val

4. L1.val = 0, S.val = 1; S.val = L.val +1

3. L.val = 1; L.val = L1.val + S.val

2. S.val = 2; S.val = L.val +1

1. S.val = 2; print(S.val)

1. 有文法G[E]和对应的语义规则

E→E1∧ T {E.val = E1.val \* T.val}

E→T {E.val = T.val}

T→T1# id {T.val = T1.val + id.lexval}

T→id {T.val = id.lexval}

则分析句子3 ∧ 3 # 4其值为（B）。

1. 10 B. 21 C. 14 D.24

找到3 ∧ 3 # 4的最右推导：1.E⇒2. E1∧ T⇒3. E1∧ T1# id3⇒4. E1∧ id2 # id3⇒5. T ∧ id2 # id3⇒6. id1∧ id2 # id3⇒7. 3 ∧ 3 # 4。

规范规约是最右推导的逆过程，按语义规则计算综合属性：

6. id1.lexval = 3, id2.lexval = 3,id3.lexval = 4;

5. T.val = 3, id2.lexval = 3,id3.lexval = 4; T.val = id.lexval

4. E1.val = 3, id2.lexval = 3,id3.lexval = 4; E.val = T.val

3. E1.val = 3, T1.val = 3,id3.lexval = 4; T.val = id.lexval

2. E1.val = 3, T.val = 3+4 = 7; T.val = T1.val + id.lexval

1. E.val = 3\*7 = 21; E.val = E1.val \* T.val

1. 有文法G[E]和对应的语义规则

E→bAb {print “1”}

A→(B {print “2”}

A→a {print “3”}

B→aA) {print “4”}

若输入序列为b(a(a(aa)))b，采用自下而上的分析方法，则输出序列为( B)。

A. 32224441 B. 34242421 C. 12424243 D. 34442212

找到b(a(a(aa)))b的最右推导：1.E⇒2.bAb⇒3.b(B b⇒4.b(aA)b⇒5.b(a(B)b⇒6.b(a(aA))b⇒7.b(a(a(B))b⇒8.b(a(a(aA)))b⇒9.b(a(a(aa)))b。

8. 3; print “3”

7. 34; print “4”

6. 342; print “2”

5. 3424; print “4”

4. 34242; print “2”

3. 342424; print “4”

2. 3424242; print “2”

1. 34242421; print “1”

第七章

1. 判断：中间代码的生成是依据语法规则生成的。（错）语义规则
2. 判断：中间语言是独立于机器的语言。（对）
3. 判断：在编译程序中，中间语言是为了便于目标代码的优化和移植。（错）中间语言的优点：使编译程序的结构在逻辑上更为简单明确，便于进行与机器无关的代码优化工作，利于编译程序的移植。
4. 判断：对于任何后缀式，无论从哪一端进行扫描，都能对它无歧义的分解。（对）
5. 判断：后缀式和中缀式一样，都可以使用括弧。（错）后缀式不能用括弧
6. 判断：后缀式abc++的中缀表达式是a+b+c。（错）a+(b+c)
7. 判断：对于产生式E -> E1+E2的语义规则E.place := newtemp中E.place存放的是E值单元的名字或地址。（对）
8. 判断：赋值语句生成三地址代码的S-属性文法的code和place属性都是继承属性。（错）是综合属性
9. 判断：产生赋值语句的三地址代码的翻译模式适合用自上而下的一遍扫描。（错）是适合自下而上的一遍扫描
10. 判断：数组元素的地址计算与数组的存储方式有关。（对）
11. 判断：在编译时，只对可执行语句进行翻译。（错）说明语句也翻译。
12. 判断：数组元素的地址由两部分构成，一部分是基址，一部分是偏移量。（对）
13. 判断：数组元素的偏移地址的计算只取决于数组的下标。（错）还有基址和数组元素的宽度。
14. 后缀式abc-/对应的中缀式是a/(b-c) ，中缀式(a-b)\*c对应的后缀式是ab-c\* 。
15. 间接三元式是用了一张间接码表和三元式来表示中间代码。
16. 四元式之间的联系是通过（B）实现的。
17. 指示器 B. 临时变量 C. 符号表 D. 程序变量

29. 写出中缀式A+B\*(C-D)+E/(C-D) ↑N的(x↑2 = x2)

（1）四元式序列；（2）三元式序列；（3）间接三元式序列。

答：先转换成后缀式：

1. 设X1= C-D，则中缀式为A+B\*X1+E/X1↑N，

X1的后缀式为CD-；

1. 设X2 =B\*X1, X3 = X1↑N，则中缀式为A+ X2+E/ X3，

X2的后缀式为BX1\* = BCD-\*,

X3的后缀式为X1N↑= CD-N↑；

1. 设X4 = E/X3，则中缀式为A+ X2 + X4，

X4的后缀式为EX3/ = ECD-N↑/；

1. 设X5 =A+ X2，则中缀式为X5 + X4，

X5的后缀式为AX2+ = ABCD-\*+,

X5+X4的后缀式为X5X4+ = ABCD-\*+ECD-N↑/ +

根据后缀式中操作符从左到右出现的顺序，依次写出四元式序列，三元式序列和间接三元式序列：

* 表达式ABCD-\*+ECD-N↑/ +的四元式序列：

1. (-,C,D,T1)
2. (\*,B,T1,T2)
3. (+,A,T2,T3)
4. (-,C,D,T4)
5. (↑,T4,N,T5)
6. (/,E,T5,T6)
7. (+,T3,T6,T7)

* 表达式ABCD-\*+ECD-N↑/ +的三元式序列

1. (-,C,D)
2. (\*,B,(1))
3. (+,A,(2))
4. (-,C,D)
5. (↑,(4),N)
6. (/,E,(5))
7. (+,(3),(6))

* 表达式ABCD-\*+ECD-N↑/ +的间接三元式序列

(1) (1) (-,C,D)

(2) (2) (\*,B,(1))

(3) (3) (+,A,(2))

(1) (4) (↑,(1),N)

(4) (5) (/,E,(4))

(5) (6) (+,(3),(5))

(6)

30．写出下述语句的四元式序列：

（1） if (x>0) then x := (a-b/2)\*c;

（2） while (aVb) b := 2\*a/5;

答：（1）先转换成后缀式：

1）If判断优先，设X1 = x>0，则中缀式为if X1 then x := (a-b/2)\*c,

X1的后缀式为x0>;

2）括弧优先，设X3 = b/2，则中缀式为if X1 then x := (a-X3)\*c,

X3的后缀式为b2/;

3）括弧优先，设X4 = a-X3，则中缀式为if X1 then x := X4\*c,

X4的后缀式为aX3- = ab2/-;

4)设X5 = X4\*c，则中缀式为if X1 then x := X5,

X5的后缀式为X4c\* **=** ab2/-c\*;

5)设X6 = x := X5，则中缀式为if X1 then X6,

X6的后缀式为xX5:= **=** xab2/-c\*:=，

if X1 then X6的后缀式为if x0> then xab2/-c\*:=。

根据后缀式中操作符从左到右出现的顺序，写出四元式序列：

1. (>, x, 0, T1)
2. (if, T1, \_, \_)
3. (/, b, 2, T2)
4. (-, a, T2, T3)
5. (\*, T3, c, T4)
6. (=, T4, \_, x)
7. (ie, \_, \_, \_)

（2）先转换成后缀式：

1）while判断优先，设X1 = aVb，则中缀式为while X1 b := 2\*a/5,

X1的后缀式为abV;

2）设X3 = 2\*a，则中缀式为while X1 b := X3/5,

X3的后缀式为2a\*;

3)设X4 = X3/5，则中缀式为while X1 b := X4,

X4的后缀式为X35/ = 2a\*5/;

4)设X5 =b := X4，则中缀式为while X1 X5,

X5的后缀式为bX4:= = b2a\*5/:=,

while X1X5的后缀式为while abVb2a\*5/:=。

根据后缀式中操作符从左到右出现的顺序，写出四元式序列：

1. (while, \_, \_,\_)
2. (V, a, b, T1)
3. (do, T1, \_, \_)
4. (\*, 2, a, T2)
5. (/, T2, 5, T3)
6. (=, T3, \_, b)
7. (we, \_, \_, \_)