属性文法(Attribute Grammar)

- 属性

  对文法的每一个符号，引进一些属性，这些属性代表与文法符号相关的信息，如类型、值、存储位置等。

- 语义规则

  为文法的每一个产生式配备的计算属性的计算规则，称为语义规则。

- 属性文法是带属性的一种文法

  它的主要思想：

➢ 首先对于每个文法符号引进相关的属性符号；

➢ 其次对于每个产生式写出计算属性值的语义规则

- 属性文法的形式定义

  一个属性文法是一个三元组,A = (G, V, F)

  - G是一个上下文无关文法；
  - V是属性的有穷集；
  - F是关于属性的断言的有穷集。

说明：

1. 每个**属性**与文法符号相联，N.t表示文法符号N的属性t。**属性值**又称**语义值**。存储属性值的变量又称**语义变量**。
2. 每个断言与文法的某个产生式相联，写在{ }内。属性的断言又称**语义规则**，它所描述的工作可以包括属性计算、静态语义检查、符号表的操作、代码生成等，有时写成函数或过程段。

属性文法的例子

对于文法：

G[E]：

$E \rightarrow n$

$E \rightarrow E + n$

属性文法的例子

表达式值的计算的属性文法：

规则：$E \rightarrow n$　　语义动作 { E.val=n }

规则：$E \rightarrow E+n$ 语义动作 { E.val=E.val+n }

属性文法的例子

语法树生成的属性文法：

规则：$E \rightarrow n$ { p=new BTreeNode;p->data=n;
p->lchild=p->rchild=0;
E.root=p;
}

规则：$E_2 \rightarrow E_1 + n$ { p=new BTreeNode; p->data=n;
p->lchild=p->rchild=0;
nr=new BTreeNode; nr->data='+';
nr->lchild= E_1 .root;
nr->rchild=p;
E_2.root=nr;
}

属性文法的例子

汇编代码生成的属性文法：

规则：E → n　　{　Gen(Ldc　n)　}

规则：E → E + n　　{　Gen(Ldc　n)，　Gen(Adi)　}

练习：定义算术表达式计算的属性文法

（1）分析C语言算术表达式的书写格式：

3+4*5　　a*(b+c)

（2）按书写格式规律写出对应的C语言算术表达式文法

G[E′]

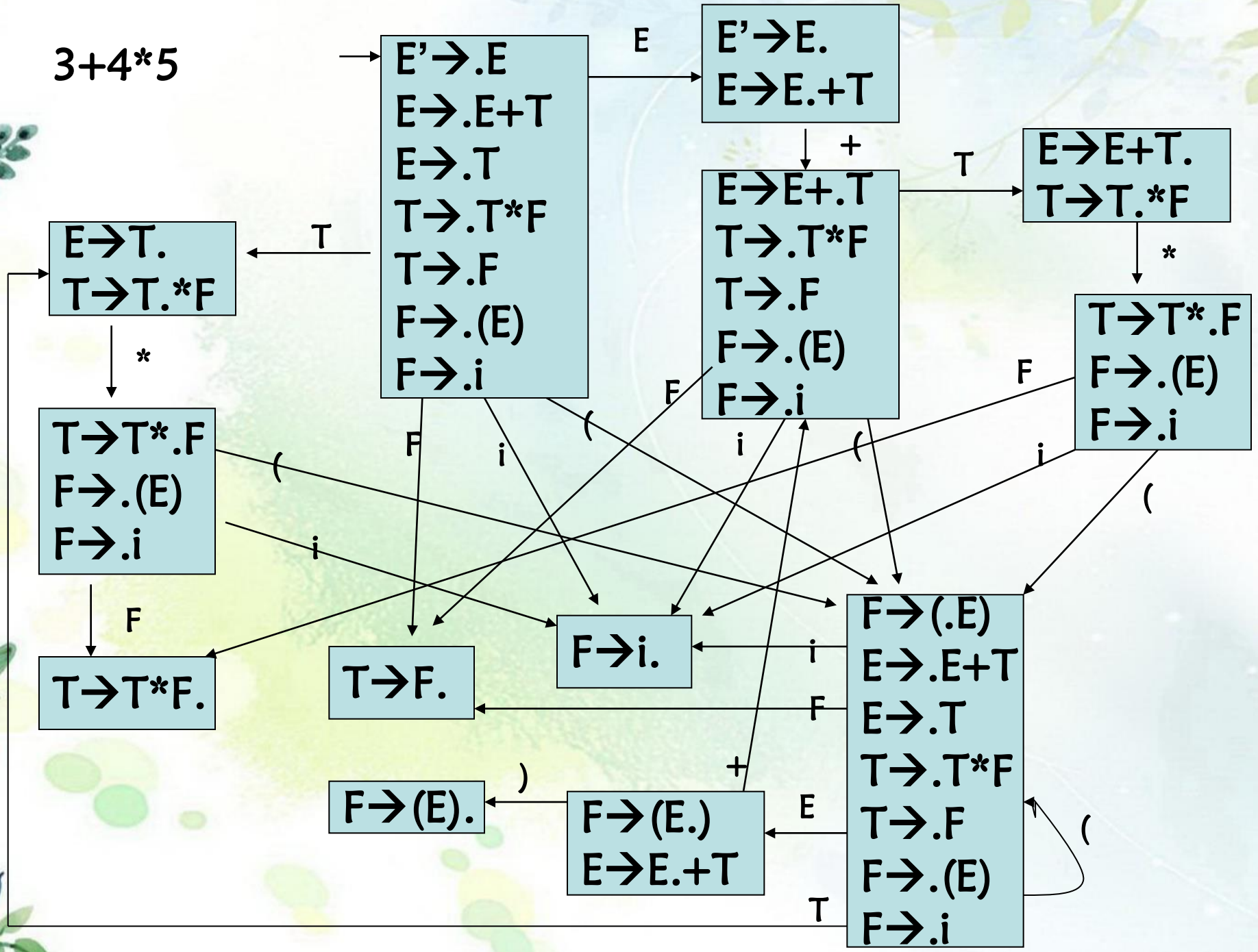
E′→ E
$E^{(1)}$→ $E^{(2)}$+T | $E^{(2)}$-T | T
$T^{(1)}$→ $T^{(2)}$*F | $T^{(2)}$/F | F
F→(E) | i

3+4*5

G[E']

E'→E
$E^{(1)}$→ $E^{(2)}$+T
$E^{(1)}$→ $E^{(2)}$-T
$E^{(1)}$→ T
$T^{(1)}$→ $T^{(2)}$*F
$T^{(1)}$→ $T^{(2)}$/F
$T^{(1)}$ → F
F→ (E)
F→ i

3+4*5


E'→.E
E→.E+T
E→.T
T→.T*F
T→.F
F→.(E)
F→.i
E
E'→E.
E→E.+T
+
E→E+.T
T→.T*F
T→.F
F→.(E)
F→.i
T
E→E+T.
T→T.*F
*
T→T*.F
F→.(E)
F→.i
T
E→T.
T→T.*F
*
T→T*.F
F→.(E)
F→.i
F
T→T*F.
T→F.
F→i.
F→(.E)
E→.E+T
E→.T
T→.T*F
T→.F
F→.(E)
F→.i
F→(E).
F→(E.)
E→E.+T
(
i
)
+
E
T

（3）定义属性文法：

规则	语义动作
1） E′→E	{ E′.val=E.val }
2） $E^{(1)}$→ $E^{(2)}$+T	{ $E^{(1)}$.val= $E^{(2)}$.val +T.val }
3） $E^{(1)}$→ $E^{(2)}$-T	{ $E^{(1)}$.val= $E^{(2)}$.val -T.val }
4） $E^{(1)}$→ T	{ $E^{(1)}$.val=T.val }
5） $T^{(1)}$→ $T^{(2)}$*F	{ $T^{(1)}$.val= $T^{(2)}$.val*F.val }
6） $T^{(1)}$→ $T^{(2)}$/F	{ $T^{(1)}$.val= $T^{(2)}$.val/F.val }
7） $T^{(1)}$→ F	{ $T^{(1)}$.val=F.val }
8） F→(E)	{ F.val=E.val }
9） F→ i	{ F.val=i }

简单例子

- 说明语句

int i,j,k;

name	type	kind	val	……
i	int	var		
j	int	var		
k	int	var		

符号表

- 可执行语句

i=2;

i=i+1;

1. (= , 2 , , i)

2. (+ , i , 1 , T)

3. (= , T , , i)

中间代码

符号表

标识符定义实体

实体属性保存在符号表

符号表的形式

每个名字对应一个表项

一个表项包括名字域和属性信息域

名字	属性信息

符号表

属性信息域

多个子域及标志位

类型、值、存储大小、相对地址

形参标志、说明标志、赋值标志

符号表的操作分析

- **说明语句**

  int i,j,k;

name	type	kind	val	……
i	int	var		
j	int	var		
k	int	var		

符号表

- **可执行语句**

  i=2;

  i=i+1;

符号表的存储结构

- 需要查找效率高
- 散列存储结构

符号表的散列存储方法

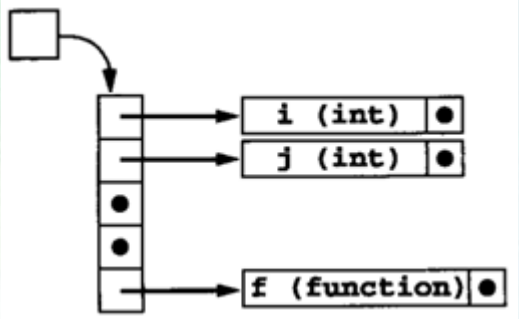
- 说明语句

int i,j,k;

name	type	kind	val	……
i	int	var		
j	int	var		
k	int	var		

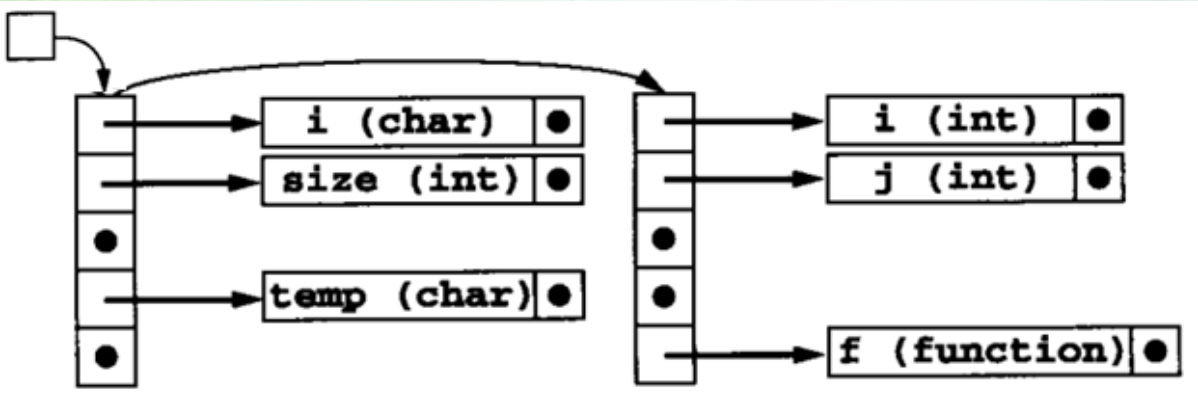
符号表如何解决作用域的问题

```
int  i,j;
int f()
{  char *j;    …   }
void main( )
{  int  size;
   char i,temp;
   f();
 }
```



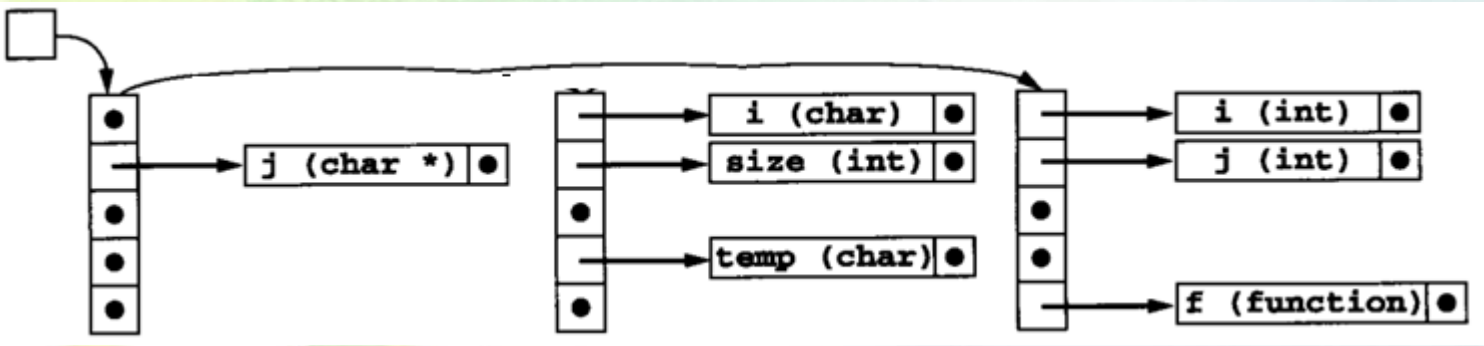
符号表如何解决作用域的问题

```
int  i,j;
int f()
{  char *j;    …   }
void main( )
{  int  size;
   char i,temp;
   f();
 }
```



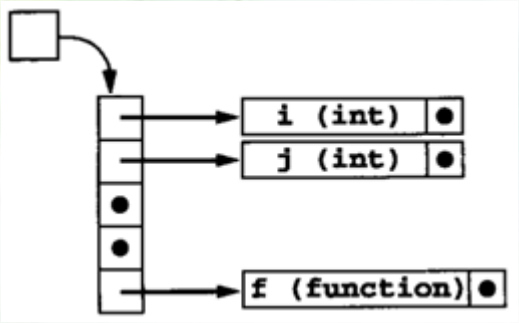
符号表如何解决作用域的问题

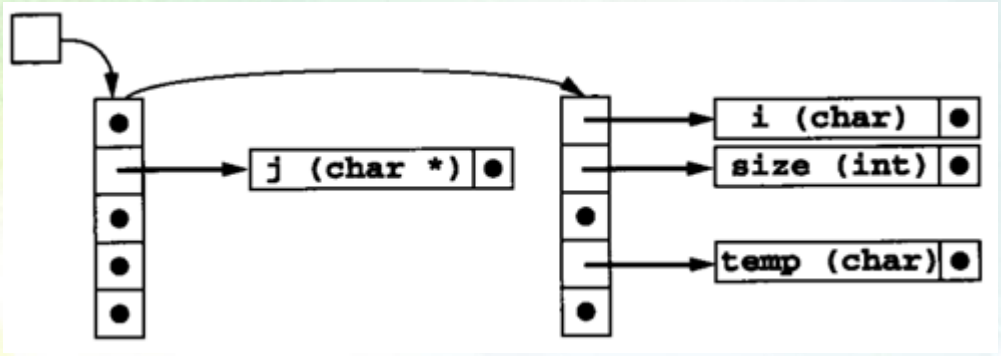
```
int  i,j;
int f()
{  char *j;    …   }
void main( )
{  int  size;
   char i,temp;
   f();
 }
```



符号表在程序运行时的作用

```
int  i,j;
int f()
{  char *j;    …   }
void main( )
{  int  size;
   char i,temp;
   f();
 }
```





符号表操作的函数

- 1)、LOOKUP(NAME)：以符号名NAME(标识符)查符号表，若表中已存在该标识符，则返回其在表中的位置(序号)，否则返回NULL。

- 2)、ENTER(NAME)：在符号表中新登记一名字为NAME的项，并返回该项在表中的位置(序号)。

- 3).ENTRY(NAME):查、填符号表的语义函数：

```
Pointer  ENTRY(NAME)
{
    ENTRYNo=LOOKUP(NAME);
    if(ENTRYNo==NULL)
        return(ENTER(NAME);
}
```

- 4）Fill(符号表位置，类型)属性填写函数