§4.11 说明的翻译(Translation of Declarations)

1. 翻译说明的任务

将被说明的名字及其属性记入符号表.翻译时不产生中间代码.

1. 过程中的说明

1.语义变量和语义过程

offset: 变量名在内存中的相对地址, offset在处理第一个说明语句之前置为0.

enter(name, type, offset): 为name建立符号表条目, 它的类型是type, 它在数据区的相对地址是offset.

type: 表示非终结符的类型.

width: 表示非终结符的宽度(该类型的对象所需的存储单元数).

1. 语义动作:

P→MD

M→ε {offset := 0}

D→D; D

D→id : T {enter(id.name, T.type, offset);

offset := offset + T.width }

T→integer {T.type := integer;

T.width := 4 }

T→real {T.type := real;

T.width := 8 }

T→array [num] of T1

{T.type := array(num.val, T1.type);

T.width := num.val×T1.width }

T→↑T1 {T.type := pointer(T1.type);

T.width := 4 }

1. 作用域信息的保存

1.语法:

P→D

D→D; D | id : T | proc id; D; S

2. 过程说明的处理

当看见过程说明D→proc id; D1; S 时, 建立新的符号表, D1说明的变量在该新符号表中建立条目.新的表回指直接外围过程的符号表, 由id代表的过程名是局部于该外围过程的.

3. 符号表的例子

(见黑板)

1. 语义变量和语义过程

(1) mktable(previous): 建立新的符号表, 并返回新符号表指针. 变元previous指向先前建立的符号表, 即直接外围过程的符号表.

(2) enter(table, name, type, offset): 在table指向的符号表中为名字name建立新条目.

(3) addwidth(table, width): 把符号表table所有条目的累加宽度记录在该符号表的首部.

(4) enterproc(table, name, newtable): 为过程名name在table指向的符号表建立新条目.变元newtable指向这个过程name本身的符号表.

(5) tblptr: 栈tblptr用来保存指向所有外围过程符号表的指针.

1. 语义动作:

P→MD {addwidth(top(tblptr), top(offset));

pop(tblptr); pop(offset) }

M→ε {t := mktable(nil);

push(t, tblptr); push(0, offset) }

D→D1; D2

D→proc id; N D1

{ t := top(tblptr);

addwidth(t, top(offset));

pop(tblptr); pop(offset);

enterproc(top(tblptr), id.name, t) }

D→id : T {enter(top(tblptr), id.name, T.type, top(offset));

top(offset) := top(offset) + T.width }

N→ε {t := mktable(top(tblptr));

push(t, tblptr); push(0, offset) }

§4.12 S属性与L属性(S-Attributes and L-Attributes)

1. 综合属性(synthesized attributes)与继承属性(inherited attributes)

1.定义: 在语法制导定义中, 每个文法符号有一组属性, 每个文法产生式A→α有一组形式为b := f(c1, c2, …, ck)的语义规则, 其中f是函数, b和c1, c2, …, ck是该产生式的文法符号的属性, 并且

(1) 如果b是A的属性, c1, c2, …, ck是产生式右部文法符号的属性或A的其它属性, 那末b叫做文法符号A的综合属性.

(2) 如果b是产生式右部某个文法符号X的属性, c1, c2, …, ck是A的属性或右部文法符号的属性, 那么b叫做文法符号X的继承属性.

在这两种情况下, 我们都说属性b依赖于属性c1, c2, …, ck.每个文法符号的综合属性集和继承属性集的交应为空.

2. 例子:

1. 综合属性

产生式 语义规则

L→En {print(E.val) }

E→E1+T {E.val := E1.val + T.val }

E→T {E.val := T.val }

T→T1\*F {T.val := T1.val × F.val }

T→F {T.val := F.val }

F→(E) {F.val := E.val }

F→num {F.val := num.lexval }

1. 继承属性

产生式 语义规则

D→TL {L.in := T.type }

T→int {T.type := integer }

T→real {T.type := real }

L→L1, id {L1.in := L.in;

addtype(id.entry, L.in) }

L→id {addtype(id.entry, L.in) }

1. 综合属性与S属性定义

1. S属性定义(S-Attributed Definitions):

仅仅使用综合属性的1(SDD)叫做S属性定义.

2. 例子:

前面我们所讲的语法制导翻译的语义规则基本上都是S属性定义.

例如: 3\*5+4n的注释分析树(见书 P197)

3. S属性定义与自下而上翻译

如果我们的语法规则可以写成S属性定义, 则该语言可以方便地用自下而上分析途径翻译.

分析器可以保存与栈中文法符号有关的综合属性, 每当进行归约时, 新的属性值就由栈中正在归约的产生式右部符号的属性值计算出来.

阅读: 讲义和教材 6.3, 5.1, 5.2.3 节.