第 6 卷第 8 期
1995 年 8 月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 6, No. 8
Aug. 1995

基于信息流分析的部分求值技术*

刘 磊 (吉林大学计算机科学系,长春 130023)

郑红军 (北京大学计算机科学技术系,北京 100080)

金成植 (吉林大学计算机科学系,长春 130023)

摘要 循环展开是过程式语言部分求值中的难题之一.本文提出一种基于信息流分析的部分求值技术,解决了部分求值中循环展开问题.本文利用这一技术,实现了一个过程式语言的部分求值器.

关键词 部分求值,信息流分析.

部分求值技术对程序优化及软件自动生成,特别是编译程序的自动生成,有着极为重要的作用[1-3],已引起人们的极大重视.过程式语言由于其自身的特点,使得对过程式语言程序部分求值相对较难,其中,循环展开就是一个至今未能很好解决的问题.目前,在此问题的解决方法中,存在着循环展开过多,因而产生大量冗余代码的缺点[4].本文提出一种基于信息流分析的过程式语言的部分求值技术,利用这一技术,我们实现了一个过程式语言的部分求值器,实践表明,这种技术可有效地解决过程式语言部分求值中的循环展开问题.

1 信息流分析技术

信息流分析技术在程序优化、程序静态分析、程序测试等许多方面都有重要的应用.应用领域的不同,分析数据的属性也不同.我们的方法是通过对程序中变量值的获取与传播的分析,着重描述变量与表达式、表达式与变量、变量与变量之间的关系.

为讨论方便,我们用 V 表示程序中所有变量的集合;用 E 表示程序中所有表达式的集合;设 $e\in E$,则 Ve 表示出现在表达式 e 中所有变量的集合.

定义 1. 若一个语句 S 含有一个对变量 v 的赋值语句,则说 S 定义了 v.用 Ds 表示 S 定义的所有变量之集.

定义 2. 在语句 S 的控制结构中,存在一条从入口到出口的路径,并在此路径上,没有对变量 v 的赋值,则说 S 保护了 v.用 Ps 表示 S 保护的所有变量之集.

* 本文 1994-05-31 收到,1994-08-30 定稿

作者刘磊,1960 年生,副教授,主要研究领域为软件新技术与软件自动生成.郑红军,1969 年生,博士生,主要研究领域为程序设计方法与新型语言.金成植,1935 年生,教授,主要研究领域为软件新技术与软件自动生成.

本文通讯联系人:刘磊,长春 130023,吉林大学计算机科学系

下面,我们定义3种关系,以描述变量与表达式、表达式与变量、变量与变量之间的关系.

(1) $\lambda s: V\times E$ 表示变量与表达式之间的关系.若$(v,e)\in\lambda s$表示在S入口处的变量v的值直接或间接地被用于表达式e的值的计算.

(2) $\mu s: E\times V$ 表示表达式与变量之间的关系.若$(e,v)\in\mu s$,表示变量v在S出口处的值依赖于表达式e的值.

(3) $\rho s: V\times V$ 表示变量与变量之间的关系.$\rho s=\lambda s\cdot\mu s\cup\{(v,v)\in V\times V\mid v\in Ps\}$.

$(v_1,v_2)\in\rho s$,表示S中存在表达式e,使得$(v_1,e)\in\lambda s$且$(e,v_2)\in\mu s$或$v_1=v_2$且$v_1\in Ps$.

例:设S为BEGIN $x:=y+1$; $w:=x+1$ END则:

$\lambda s:\{(y,y+1),(y,x+1)\}$,$\mu s:\{(y+1,x),(y+1,w),(x+1,w)\}$,$\rho s:\{(y,w),(y,x),(y,y)\}$

按照上述定义及过程式语言的语义,不难计算出程序中的上述3种关系.其中,关于循环语句信息流关系的计算用到了传递闭包.限于篇幅,在此不一一叙述.

2 循环语句的信息流分析

本文以过程式语言的基本语句为基础,重点讨论循环语句的部分求值技术,因此,有必要对循环语句作详细的分析.设S为循环语句[WHILE e DO A],$RG(A)$是关系ρ_A的图,其结点集为V,弧集为ρ_A,基于上述的3种关系,我们有:

定义 3. 关系子图$SRG(A)$是$RG(A)$的子图,其结点集为D_A,弧集为$\rho_A\cap(D_A\times D_A)$.

定义 4. 设$v\in D_A$,且在图$SRG(A)$中不存在环路到v的路径,则说v是关于S的稳定型变量.

定义 5. 若变量v是关于S的稳定型变量,则在图$SRG(A)$中,终止于变量v的最长路径长度被称为v的稳定长度,记为$\alpha s(v)$.

定义 6. 设e是S中的一个表达式,且对任一$(v,e)\in\lambda s$,其中v或是稳定型的,或是$v\overline{\in}D_A$,则称表达式e关于S的稳定型表达式.

定义 7. 设表达式e是关于S的稳定型表达式,则其稳定长度定义为:

$$\begin{cases}\beta s(e)=0, & \text{当}\{v\in D_A\mid(v,e)\in\lambda s\}=\Phi\\ \beta s(e)=\max(\alpha s(v)\mid v\in D_A\wedge(v,e)\in\lambda s)+1, & \text{否则.}\end{cases}$$

显然,容易求出稳定型变量和稳定型表达式以及它们的稳定长度.若用$v^{(k)}$表示S执行第k次后变量v的值,且v是关于S的稳定型变量,则对任意$k>\alpha s(v)$有:$v^{(k+1)}=v^{(k)}$.我们有如下结论:对于循环语句中的稳定型变量和稳定型表达式,当循环次数达到其稳定长度时,其值也达到了"稳定".

3 部分求值

过程式语言与函数式语言和逻辑式语言有很大的区别.其中,重要区别之一是:过程式语言程序中变量的值是随着程序的执行而动态变化的.这使我们不能采用其它类语言的部分求值技术.

基于过程式语言自身的特点,采用动态的部分求值技术更为实际和方便.为了对过程式

程序进行部分求值,我们必须清楚地知道一个语句是否应被执行和一个表达式是否可计算等问题.下面,我们给出部分求值技术的描述.

首先,我们引入状态 STA 和环境 ENV:　　$STA:V\rightarrow\{k,u\}$　　$ENV:V\rightarrow VAL$

STA 是变量到状态的映射,k 表示变量值已知,u 表示变量值未知(初始状态所有变量为 u);ENV 是变量到其值的映射.

我们引入一个表达式求值函数:　　$M:EXP\times STA\times ENV\rightarrow(EXP+VAL)\times\{k,u\}$

具体定义如下:

$$M(E,s,env)=\begin{cases}(n,k) & \text{若 } E=n,\text{且 } n \text{ 为常数或 } E=V,\text{且 } s(v)=k,env(v)=n\\(v,u) & \text{若 } E=V,\text{且 } s(v)=u\\M'(op,(x_1,w_1),(x_2,w_2)) & \\ & \text{若 } E=E_1\ op\ E_2,\text{且 } M(E_i,s,env)=(x_i,w_i),i=1,2\end{cases}$$

$$M'(op,(x_1,w_1),(x_2,w_2))=\begin{cases}(n,k) & \text{若 } w_1=w_2=k,\text{且 } x_1\ op\ x_2=n\\(x_1\ op\ x_2,u) & \text{若 } w_1=u \text{ 或 } w_2=u\end{cases}$$

语句求值函数定义如下:

$$TRANS:Stm\times STA\times ENV\rightarrow Stm\times STA\times ENV$$

在下面的语句求值函数中,(x/y)表示用 x 代替 y 的值.语句求值函数具体定义如下:

(1) $TRANS(skip,s,env)$

$=(skip,s,env)$

(2) $TRANS(read(x_1,x_2,\cdots,x_n,y_1,y_2,\cdots,y_m),s,env)$

$=(read(y_1,y_2,\cdots,y_m),s(k/x_i),env(val(x_i)/x_i))$

其中 x_i 为部分输入,$val(x_i)$为 x_i 的输入值.$i=1,2,\cdots,n$

(3) $TRANS(v:=E,s,env)=(v:=E',s[u/v],env)$　若 $M(E,s,env)=(E',u)$

$(skip,s[k/v],env[n/v])$ 若 $M(E,s,env)=(n,k)$

(4) $TRANS(S_1;S_2,s,env)=(S_1';S_2'',s'',env'')$

其中 $TRANS(S_1,s,env)=(S_1',s',env')$　$TRANS(S_2,s',env')=(S_2'',s'',env'')$

(5) $TRANS(\underline{\text{IF}}\ E\ \underline{\text{THEN}}\ S_1\ \ \underline{\text{ELSE}}\ S_2,s,env)$

$$=\begin{cases}TRANS(S_1,s,env) & \text{若 } M(E,s,env)=(\underline{\text{true}},k)\\TRANS(S_2,s,env) & \text{若 } M(E,s,env)=(\underline{\text{false}},k)\\(\text{IF } E'\ \text{THEN } S_1';S_1''\ \text{ELSE } S_2';S_2'',s',env') & \text{若 } M(E,s,env)=(E',u)\end{cases}$$

其中 $TRANS(S_1,s,env)=(S_1',s'',env'')$　$TRANS(S_2,s,env)=(S_2',s''',env''')$

$S''_1=v_1:=n_1;\cdots;v_m:=n_m\quad(1\leqslant i\leqslant m)$

若 $v_i\in Ds_1\wedge env''(v_i)=n_i\wedge((s''(v_i)=k\wedge s'''(v_i)=u)\vee(s''(v_i)=s'''(v_i)=k\wedge env''(v_i)\langle\rangle env'''(v_i)))$

$S_2''=v_1:=n_1;\cdots;v_m:=n_m\quad(1\leqslant i\leqslant m)$

若 $v_i\in Ds_2\wedge env'''(v_i)=n_i\wedge((s'''(v_i)=k\wedge s''(v_i)=u)\vee(s''(v_i)=s'''(v_i)=k\wedge env''(v_i)\langle\rangle env'''(v_i)))$

$$s'(v)=\begin{cases}k & \text{若 } s''(v)=s'''(v)=k\\u & \text{若 } s''(v)=u \text{ 或 } s'''(v)=u\end{cases}\qquad env'(v)=\begin{cases}n & \text{若 } s'(v)=k\wedge n=env''(v)\\\cdot & \text{若 } s'(v)=u\end{cases}$$

(6)设循环语句为WHILE E DO A，一般地，它的处理可分为下面几种情况：E 的值已知，且为false；E 的值已知，且为true；E 的值未知. 第一种情况容易处理；对于第二种情况，一般将WHILE E DO A 按如下的等价语句处理：

$$A;\ \text{WHILE}\ E\ \text{DO}\ A$$

但有时会产生过多的冗余代码，例如，对于下面程序段：

$i:=1;$ WHILE $i<100$ DO BEGIN $x:=x+1;$ $a[i]:=i;$ $i:=i+1$ END

若采用上述方法，循环将被展开100次. 显然，这是不理想的. 我们可利用循环语句的特性来解决它的部分求值问题. 根据循环语句中变量的最大稳定长度来控制部分求值时的循环展开层数；对于第三种情况，我们为了有效地进行部分求值，把循环展开一层，即将WHILE E DO A 按如下的等价语句处理：

IF E THEN A; WHILE E DO A ELSE $SKIP$;

基于上面的分析，我们把循环语句视为：

(WHILE E DO A, N)　其中，$N=\max(as(v)\mid\in D_A)+1$

循环语句的求值函数具体定义如下：

(1)若 $M(E,s,env)=(\text{false},k)$，则

$TRANS((\text{WHILE}\ E\ \text{DO}\ A,N),s,env)=(skip,s,env)$

(2)若 $M(E,s,env)=(\text{true},k)$，且 $N>=0$，则

$TRANS((\text{WHILE}\ E\ \text{DO}\ A,N),s,env)=TRANS(A;(\text{WHILE}\ E\ \text{DO}\ A,N-1),s,env)$

$TRANS((\text{WHILE}\ E\ DO\ A,0),s,env)=(\text{WHILE}\ E\ \text{DO}\ A,s,env)$

(3)若 $M(E,s,env)=(E',u)$，则

$TRANS((\text{WHILE}\ E\ \text{DO}\ A,N),s,env)$

$=TRANS'(\text{IF}\ E\ \text{THEN}\ A\ ;\text{WHILE}\ E\ \text{DO}\ A\ \text{ELSE}\ SKIP\ ;\ ,s,env)$

其中，函数 $TRANS'$ 为一新函数，它的定义与函数 $TRANS$ 基本相同，其不同之处在于赋值语句和循环语句的处理.

$TRANS'(v:=E,s,env)=(v:=E',s[u/v],env)$，其中 $M(E,s,env)=(E',k/u)$

$TRANS'(\text{WHILE}\ E\ \text{DO}\ A\ ,s,env)=(\text{WHILE}\ E\ \text{DO}\ A',s',env')$，其中 $TRANS(A,s,env)=(A',s',env')$

例如，设有程序段(a)，初始状态为 $s+(x.y.z.x_1\rightarrow u\}$，按上述方法可得剩余程序(b).

```
z:=2;
y:=z+3;
i:=1;
WHILE i<100 DO
BEGINx:=y+1;
     a[i]:=i+x1;
     y:=100;
     i:=i+1
END;
```

(a)

```
a[1]:=1+x1; a[2]:=2+x1; i:=3;
WHILE i<100 DO
BEGIN
  x:=101;
  a[i]:=i+x1;
  y:=100;
  i:=i+1
END;
```

(b)

其中 x,y 均为稳定型变量，x 的稳定长度 $as(x)=1$，y 的稳定长度 $as(y)=0$. 因此，程序段(a)中的循环展开二层，最后得到的状态：$s=\{i\rightarrow u, x\rightarrow k, y\rightarrow k, z\rightarrow k, x1\rightarrow u\}$，和环境：$env=\{x\rightarrow 101, y\rightarrow 100, z\rightarrow 2\}$. (有关复杂变量的部分求值问题，另有文章).

4　结束语

本文提出的基于信息流分析的部分求值技术较好地解决了过程式语言部分求值技术中一直存在的循环展开问题，从而取得了较满意的结果. 这种部分求值技术在一定的范围内可应用于递归过程(或函数)的部分求值中. 在某些情况下，对递归过程出口表达式及体内各变量的信息流分析，可使递归过程例化取得较好的结果[5].

我们实现了一个类 PASCAL 语言的部分求值器，本文提出的技术，在该部分求值器中得到了很好的应用. 实践表明，该技术是行之有效的.

参考文献

1　Ershov, Andrei P. Mixed computation: potential applications and problems for study. Theoretical Computer Science, 1982, **18**(1): 41—67.

2　Jones Neil D, Sestoft Peter. A self—applicable partial evaluator for experiments in compiler generation. LISP and Symbolic Computation, 1989, **1**(2): 9—50.

3　Sestoft Peter. The structure of a self—applicable partial evaluator. Program as Data Objects, 1985. 236—256.

4　Meyer, U. Techniques for partial evaluation of imperative language. ACM SIGPLAN NOTICES, 1991, **26**(9): 94—105.

5　Futamura Y. Essence of generalized partial computation. Theoretical Computer Science, 1991, **90**(1): 61—79.

THE PARTIAL EVALUATION TECHNIQUE BASED ON INFORMATION FLOW ANALYSIS

Liu Lei

(*Department of Computer Science, Jilin University, Changchun* 130023)

Zheng Hongjun

(*Department of Computer Science, Beijing University, Beijing* 100080)

Jin Chengzhi

(*Department of Computer Science, Jilin University, Changchun* 130023)

Abstract　The loop unfolding is one of difficult problems in the procedural language partial evaluation. A partial evaluation technique based on information analysis that solves the problem is presented in this paper. It has implemented a partial evaluator using this technique.

Key words　Partial evaluation, information analysis.