ISSN1000-9825 Journal of Software 软件学报 2000,11(3):334~341

数据流分析中的区域覆盖技术*

胡世亮 臧斌宇 凌冰 朱传琪

(复旦大学并行处理研究所 上海 200433)

E-mail: byzang@fudan.edu.cn

摘要 用计算函数模型进行精确的数据流分析,条件谓词之间的逻辑关系被转化为空间区域之间的覆盖关系.该文讨论在各种常见的程序构造下如何表示、计算和传递 Ω 区域、Φ 区域,并在数据流分析过程中,利用 Ω 区域、Φ 区域之间的覆盖关系消除条件分支语句带来的不确定性,以得到更精确的数据流信息.

关键词 Ω 区域,Φ 区域,数组数据流分析,数组私有化,符号分析.

中图法分类号 TP311

区域覆盖的概念源自数组数据流分析.并行化编译器的编译对象主要是高性能计算领域中的科学计算程序.该类程序普遍的特点之一是,数据结构简单,数组和标量构成了数据集的主体.因此,数组数据流分析是并行化编译器的关键技术.数组区域(array region)的表示、运算及其相互之间的覆盖关系是数组数据流分析中的主要问题.

文献[1]提出的相关-覆盖方法就是利用数组区域的覆盖技术和相关性测试来解决数组私有化的判定问题.实验测试表明,相关-覆盖判定法是实效最好的数组私有化方法之一.但是该方法目前还不能处理条件分支语句所产生的流不确定性.鉴于数组私有化是众多并行化变换中最有效的方法之一[1,2],有必要扩充相关-覆盖方法,使之能够处理条件分支语句产生的流不确定性.

在计算函数模型[3]下,条件读写引用的语义可以表示为 $\Omega_p \xrightarrow{x} Set_p(x)$ 或 Φ 区域,形式上是数组区域在概念上的一个拓广.利用区域、区域的覆盖关系,可以在既有数组数据流分析的框架下处理条件分支语句产生的流不确定性,增强传统的数组数据流分析.此外,Ω 区域的覆盖技术还可以应用在符号分析等问题上.

本文第1节介绍在数据流分析的过程中如何表示、计算条件读写的 Ω 区域和 Φ 区域.第2节阐述利用 Ω 区域、Φ 区域的覆盖关系得到更精确的数据流信息,并给出典型的实例.第3节同国外的相关工作进行比较.最后得出结论.

1 Ω 区域、Φ 区域的表示与计算

在最一般的意义下,条件谓词的 Ω 区域在 $Dom(P)$ 空间中可能是非常复杂的空间区域,因而是难以计算的,判定其相互之间的覆盖关系在一般意义下是NP问题.令人却步的复杂度源于程序逻辑语义本身的潜在复杂性.

值得庆幸的是,编译器的分析优化并不需要处理所有可能的情况,而只需处理好实际情况中常见的、影响并行化变换的情形.对SPEC95fp,PERFECT等测试程序包的分析表明:条件谓词的Omega区域在其直接出现的变量集 $(t_1,t_2,\ldots,t_k)$ 上通常可以表示为线性凸区域.把 $\Omega_T(t_1,t_2,\ldots,t_k)$ 视为Omega区域的参数形式.当需要判

* 本文研究得到国家自然科学基金(No.69633030)、国家863高科技项目基金(No.863-306-ZT01-02-01)、教育部科学技术项目基金和国防科技重点实验室基金(No.97JS76.5.2.JW0701)资助.作者胡世亮,1972年生,硕士生,主要研究领域为并行与分布式计算.臧斌宇,1965年生,副教授,主要研究领域为并行处理,高性能计算.凌冰,1974年生,硕士生,主要研究领域为并行编译.朱传琪,1943年生,教授,博士生导师,主要研究领域为并行处理,高性能计算.

本文通讯联系人:臧斌宇,上海 200433,复旦大学并行处理研究所

本文1998-09-21收到原稿,1999-03-09收到修改稿

定覆盖关系的 Omega 区域处在不同的参数空间时,进行参数的反向替代,以使它们处于同一参数空间,并进行覆盖关系的比较,如果参数替换过程产生过于复杂的表达式或过于复杂的 Omega 区域,使得判定 Omega 区域之间的覆盖关系变得过于复杂,可以视为静态不可判定的问题.对于程序段 P,参数的替代最终只能进行到所有的参数全部替换为 $In(P)$中的变量,所以这个替换过程可行,并且一定结束.

并行化编译器主要分析结构化的程序段,它们一般由表达式、赋值语句、条件分支语句、各种循环语句以及顺序结构构成.并行化变换主要优化确定次数的循环,例如 FORTRAN 中的 DO 循环,因为它们是最具有并行性的程序构造.DOWHILE 和 REPEAT-UNTIL 类型的循环本质上是递归计算,难以有效地并行计算,因此这里不再考虑.

1.1 条件谓词的 Omega 区域表示

条件分支语句的逻辑谓词主要有这样几种常见的基本形式:数值关系的比较,如$=,\leqslant,\geqslant,\neq,<,>$;整型变量的 MOD 运算;绝对值运算以及在此基础上的逻辑运算.由文献[3]中的引理 1.1 可知,逻辑谓词可以直接表示为 Omega 区域的特征函数表达式.

(1) 数值关系的比较,整型变量的 MOD 运算直接化为空间区域的约束方程、不等式.例如,

IF (X>4) 化为 $\Omega=\{Dom(P)\mid X>4\}$,

IF (MOD(X,2)=0) 化为 $\Omega=\{Dom(P)\mid mod(X,2)=0\}$.

(2) 绝对值运算化为线性方程组.例如,

IF (|X|<1.0) 化为 $\Omega=\{Dom(P)\mid X<1.0;X>-1.0\}$.

(3) 逻辑运算化为区域之间的交并补运算:

IF (C1 .and. C2) 化为 $\Omega=\Omega_1\cap\Omega_2$,

IF (C1 .OR. C2) 化为 $\Omega=\Omega_1\cup\Omega_2$,

IF (.NOT. C1) 化为 $\Omega=\neg(\Omega_1)$.

例如:

IF (.NOT.(X>4)) 化为 $\Omega=\neg\Omega_1=\{Dom(P)\mid X\leqslant 4\}$,

IF ((X>2) .AND. (Y>1)) 化为 $\Omega=\Omega_1\cap\Omega_2=\{Dom(P)\mid X>2;Y>1\}$,

以下简记$\{Dom(P)\mid p(x_1,x_2,\ldots,x_n)\}$为$\{p(x_1,x_2,\ldots,x_n)\}$,";"表示与关系.

1.2 嵌套在条件分支语句下的情形

设嵌套在条件分支语句 if 内部的 Omega 区域为 Ω_i,则在 if 之外其相应的 Omega 区域 Ω_0 是 Ω_i 与其所在 if 分支之条件谓词 Omega 区域的交.

由 Φ 区域的定义:$\Phi_P=\Omega_{P_x}\times Set_P(x)$,$\Phi$ 区域的计算只是相应 Ω 区域的计算.

例 1:

```
Step 1.   IF (X>4) THEN
Step 2.     IF (Y>0) THEN
Step 3.       DO I=1,10
Step 4. w           A(I)=...
Step 5.       ENDDO
Step 6.     ENDIF
Step 7.   ENDIF
```

在程序段 Step 3~Step 5 中,w 的数组引用区域是 $A(1:10)$;在程序段 Step 2~Step 6 中,Step 2 的条件谓词的 Omega 区域是 $\Omega_2=\{Y>0\}$.数组定义点 w 在程序段 Step 2~Step 6 范围中的 Φ 区域为

$$\Phi_{2\sim6}(w)=\Omega_2\times Set_{2\sim6}(w)=\{Y>0\}\times\{A(\varphi)\mid\varphi=1,\ldots,10\}.$$

在程序段 Step 1~Step 7 下,嵌套的 Omega 区域是 $\Omega=\Omega_1\cap\Omega_2=\{X>4;Y>0\}$,数组定义点 w 的 Φ 区域为

$$\Phi_{1\sim7}(w)=\Omega\times Set_{1\sim7}(w)=\{X>4;Y>0\}\times\{A(\varphi)\mid\varphi=1,\ldots,10\}.$$

1.3 嵌套在DO语句中的情形

嵌套在DO语句中的IF语句,其计算语义可以非常复杂.这里只需考虑对数据流分析有利的几种常见情形.其他复杂情形可以用文献[3]中提出的忽略值不确定性IF语句的方法排除在分析范围之外.

嵌套在DO语句中的逻辑条件可以含有标量或数组变量.

如果嵌套在DO语句中的IF语句,其条件中仅含标量,而且是非递归标量,则该IF语句为循环不变条件分支语句,可以提到DO循环之外,用1.2节所述方式来处理.

如果含递归标量(induction variable),那么该逻辑条件是对循环的迭代空间进行约束的逻辑谓词.设条件引用 x 嵌套在程序段 P 的 n 重DO循环中,相应的循环变量为 $i_1,i_2,\ldots,i_n$, $t_1,t_2,\ldots,t_k$ 是直接出现在谓词 $P(t_1,t_2,\ldots,t_k)$ 中的变量,该谓词确定的 $D_T=D_{t_1}\times D_{t_2}\times\ldots\times D_{t_k}$ 中的区域 Ω_T,可以表示为 $i_1,i_2,\ldots,i_n$ 和 $In(P)$ 上的方程和不等式组.当需要考虑 X 在 n 重循环之外的数据流关系时,需要把Omega区域投射到循环外程序段 P 的定义变量集 $In(P)$ 上.投射过程把循环变量 $i_1,i_2,\ldots,i_n$ 从Omega区域表达式中消去,事实上,消去的过程与数组引用区域向DO循环外扩的过程对应.循环变量的语义作用从逻辑条件约束转变为数组区域的约束.

例2:

```
Step 1.    DO LL=1,31
Step 2.      N=LL-2
Step 3.      DO I=1,31
Step 4.        N=N+1
Step 5.        IF (N>=31) THEN
Step 6. w          Y(I,LL)=Y(I,LL)+S(I,LL)+TEM(I,LL)
Step 7.        ENDIF
Step 8.      ENDDO
Step 9.    ENDDO
```

在循环体中(语句Step 4~Step 7),w 的区域为

$$\begin{aligned}\Phi^w_{4\sim7}&=\Omega^w_{4\sim7}\times Set_{4\sim7}(w)=\{N\geqslant31\}\times\{Y(\varphi_1,\varphi_2)\mid\varphi_1=LL;\varphi_2=I\}\\&=\{1\leqslant LL,I\leqslant31;LL+I-2\geqslant31\}\times\{Y(\varphi_1,\varphi_2)\mid\varphi_1=LL;\varphi_2=I\}\\&=\{Y(\varphi_1,\varphi_2)\mid\varphi_1=LL;\varphi_2=I;1\leqslant LL,I\leqslant31;LL+I-2\geqslant31\},\end{aligned}$$

出循环后,w 在整个程序段 P 中的区域为

$$\begin{aligned}\Phi^w_P&=\Omega^w_P\times Set_P(w)\\&=Dom(P)\times\{Y(\varphi_1,\varphi_2)\mid\varphi_1+\varphi_2-2\geqslant31;1\leqslant\varphi_1,\varphi_2\leqslant31\}\\&=\{Y(\varphi_1,\varphi_2)\mid1\leqslant\varphi_1,\varphi_2\leqslant31;\varphi_1+\varphi_2-2\geqslant31\},\end{aligned}$$

在语句Step 1~Step 9中的条件区域是 P 的定义域

$$\Omega_w=Dom(P).$$

如果嵌套在DO循环中的IF语句,其逻辑条件含有数组变量,并且数组引用的下标表达式中含有递归标量(否则视为标量处理),此时的逻辑条件是关于该数组特定区域的向量式的逻辑条件,需要对逻辑条件进行向量化扩展.理论上,向量化的逻辑条件可以蕴含相当丰富和复杂的语义,但在并行化常见的科学计算类程序时,特别需要处理好以下两种常见的模式.

(1) 向量式逻辑条件的条件归约.此时数组的下标表达式一般是严格单调的,设循环的迭代空间大小为 n,则根据各个被判定数组元素相对该逻辑条件是否成立可以有 2^n 种情形,但在编译器里通常只需考虑全部成立或全部不成立的情形.条件全部成立时,归约变量进行一般的归约运算,相应的 Ω 区域为各个分量满足逻辑条件时的联立;条件全部不成立时,归约变量不变.相应的 Ω 区域为各个分量不满足逻辑条件时的联立.参见例3.

(2) 向量式逻辑条件对应下的向量引用.此时,逻辑条件和向量引用中的数组下标表达式一般都是严格单调的,根据逻辑条件中数组元素的取值决定是否对相应的引用数组元素进行读写引用.这里也只需考虑向量式逻辑条件全部成立或全部不成立的情形,并且相应的 Ω 区域计算方法同上,而被引用的数组区域同一般的数组

区域计算方法相同.参见例4.

例3:

```
Step 1   KC=0
Step 2   DO K=1,9,1
Step 3       IF (RS(K).GT.CUT2) THEN
Step 4         KC=KC+1
Step 5       ENDIF
Step 6   ENDDO
```

Step 3中的条件是$RS(1:9)$的向量化逻辑条件.KC定义在Step 1～Step 6的范围是值不确定性定义.如果KC是符号变量,则需要对其进行符号分析.KC对向量条件:$RS(K).GT.CUT2, K=1,\ldots,9$进行了条件归约.因此$KC$的取值范围为$0 \leqslant KC \leqslant 9$.

$KC=0$对应条件Step 3全部不成立的$Dom(P)$中的一个区域

$$\Omega=\bigcap_{K=1}^{9}\{RS(K)\leqslant CUT2\};$$

$KC=9$对应$\forall K, 1\leqslant K\leqslant 9$条件Step 3都成立的区域

$$\Omega=\bigcap_{K=1}^{9}\{RS(K)>CUT2\}.$$

列4:

```
Step 1   DO K=2,5,1
Step 2     IF (RS(4+K).LE.CUT2) THEN
Step 3       RL(4+K)=SQRT(RS(4+K))
Step 4     ENDIF
Step 5   ENDDO
```

从Step 1～Step 5中的IF语句导致了程序段P中的一个条件写,$RL(6:9)$的定义和$RS(6:9)$满足$(RS(4+K).LE.CUT2)$的情况是一致的.当条件全部成立时,$\Omega_w=\bigcap_{K=6}^{9}\{RS(K)\leqslant CUT2\}$,数组区域$RL(6:9)$也全部被定义,条件全部不成立时,数组区域$RL(6:9)$也全部没有被定义.

需要指出的是,在上述两例中,由于Ω区域的表达式中出现了数组元素,并且是对一定范围中的每个元素都成立,例如,$\Omega_w=\bigcap_{K=5}^{9}\{RS(K)\leqslant CUT2\}$等,因此,有必要对现有的区域运算系统在表示和运算方法上进行相应的扩充.

1.4 过程间的传递

当被分析的程序段P含有过程调用时,需要在调用过程和被调用过程之间传递数据流信息.条件谓词的Ω区域主要是关于标量的约束关系,找出虚参数和实参数之间的对应关系并进行标量的替换即可.数组引用区域的跨过程传递则复杂得多,此时,虚参数组与实参数组可以有不同的形状定义,实参数组也无须从一个数组的首元素开始,因此,这里要解决数组区域的RESHAPE问题[4].

1.5 区域运算及区域覆盖关系的判定

区域之间的基本运算包括交、并、减.

n维空间中任意两个点集之间的集合运算和覆盖关系判定是很困难的.但在实际应用程序中,逻辑条件的Ω区域经常是线性凸区域,甚至是各维之间相互独立的正则区域.文献[5]中有关于Diophantine方程、不等式组的Omega测试法,可以用来测试整系数区域的相交和覆盖关系,特别是线性凸区域内整数点集之间的相交和覆盖关系判定.在文献[5]列出的参考文献中,有一般情形下线性凸区域之间的运算和覆盖关系判定的方法,这里不再赘述.当需要判定覆盖关系的两个空间区域非常复杂时,问题的复杂性源自逻辑谓词之间逻辑关系的复杂性.

2 区域覆盖技术在数据流分析中的应用

2.1 数组数据流分析中的区域覆盖技术

在数组数据流分析中可以利用 Ω 区域和 Φ 区域的覆盖关系来消解由条件分支语句导致的数据流不确定性.由文献[3]中的覆盖定理可以导出如下原理.

流不确定性消解原理. 对于程序段 P 中特定的读 r 和所有可能为 r 提供数据源的写集 $W:W=NaW_P$,判定 W 的一个特定的子集 W^* 是否满足 $W^*\xrightarrow{f}r$ 可以通过判定以下条件来确定:$\Phi_P\subseteq\Phi_P(W^*)$.判定单独一个写能否覆盖读时,只需判定 $\Omega_r\subseteq\Omega_w$ and $Set_P(r)\subseteq Set_P(w)$.

区域覆盖技术可以增强数组私有化的判定.

如果把数组引用区域从单纯的数组区域推广到 Φ 区域,文献[1]中的相关-覆盖方法可以推广到能够处理条件分支语句产生的、静态可消解的流不确定性.下面约定:BL 表示循环的次数,L_i 表示第 i 次循环迭代,L_v 表示任意的 L_i,$2\leqslant i\leqslant BL$.其他符号的含义与文献[1]相同,并且约定读写都是针对当前被考察数组的.限于篇幅,证明从略.

定理 2.1. 对于程序段 P,$UE_P\subseteq\bigcup_{r\in R_P}(\Phi_P(r)-\Phi_P(NaW_P^r))$.

定义 2.1. 数组在循环 L 中是写自覆盖的,如果$\forall\ 1\leqslant i\leqslant BL-1:\Phi_{L_i}(W)\subseteq\Phi_{L_{i+1}}(W)$.

基本判定准则. 循环 L 中,如果$\forall\ k,2\leqslant k\leqslant BL:UE_{L_k}\cap\bigcup_{i=1}^{k-1}\Phi_{L_i}(W_{L_i})=\varnothing$,该数组可以私有化.

判定准则 1. 循环 L 中,$r\in R_{L_v}$,如果 $\Phi_{L_v}(r)\subseteq\Phi_{L_v}(NaW^r_{L_v})$,$r$ 不妨碍私有化.如果任意 $r\in R_{L_v}$ 均不妨碍私有化,数组在循环 L 中是可私有化的.

判定准则 2. 循环 L 中,$r\in R_{L_v}$,如果数组在 L 中写自覆盖;且 $\Phi_{L_v}(W_{L_v})\subseteq\Phi_{L_v}(NaW_{L_v})$,那么 r 不妨碍私有化.如果任意 $r\in R_{L_v}$ 均不妨碍私有化,数组在循环 L 中是可私有化的.

2.2 符号分析中的区域覆盖技术

在计算函数模型下,条件分支语句定义的符号变量,其符号值可以表示为

$$Symbol\leftarrow\begin{cases}\Omega_{P_1}\xrightarrow{w}ExpVal_1\\ \cdots\\ \Omega_{P_n}\xrightarrow{w}ExpVal_n\\ \Omega_{other}\xrightarrow{w}ExpVal_o\end{cases}$$

$\Omega_{P_1},\ldots,\Omega_{P_n},\Omega_{other}$ 是 $Dom(P)$ 的一个划分.我们称这种表示法为区域函数法.

程序段 P 中,对符号变量 $Symbol$ 的读引用 $r:\Omega_{P_r}\xrightarrow{r}Symbol$,$r$ 能够读到 $ExpVal_i$ 的充要条件是:$\Omega_{P_r}\cap\Omega_{P_i}\neq\varnothing$;特别地,如果 $\Omega_{P_r}\subseteq\Omega_{P_i}$,则 r 读到的一定是 $ExpVal_i$;如果 $\Omega_{P_r}\subseteq\bigcup_{i\in D}\Omega_{P_i}$,则 R 读到的一定是 $\{ExpVal_i|i\in D\}$ 中的某个数值.我们用 $Symbol(\Omega_{P_r})$ 表示在状态条件 Ω_{P_r} 下读符号变量 $Symbol$ 所读取的变量值表达式.

例 5:

```
Step 1  IF (I.LE.6) THEN NB=1
Step 2  ELSEIF (I.GT.6 .AND. I.LE.26) THEN NB=2
Step 3  ELSEIF (I.GT.26 .AND. I.LE.64) THEN NB=3
Step 4  ELSE NB=4
Step 5  ENDIF
  ⋮
Step 10 IF (I.GT.32) ...=...NB...
```

$NB(I>32)$读到的数值可以确定为 3 或 4,如果 $I>64$,则可以进一步明确为 4.

文献[6]中推广的 Φ-function 表示法对应于区域函数表示法在 $n=1$ 时的情形.因此,文献[6]中介绍了表示

法的运算可以推广到区域函数表示法上,例如,求符号变量的极值:

$$\max(Symbol)\leqslant\max(ExpVal_1,\ldots,ExpVal_n,ExpVal_o),$$

$$\min(Symbol)\geqslant\min(ExpVal_1,\ldots,ExpVal_n,ExpVal_o).$$

定义两个符号表达式是相容的,如果经过变量的反向替换后,一个表达式的非常数项是另一个表达式非常数项的子集.比较两个相容符号变量的分配律:

$$Sym1>Sym2\Leftrightarrow\prod_{i=1}^{other}(Sym1(\Omega_i^{(2)})>ExpVal_i^{(2)}),$$

$\Omega_i^{(2)}$ 和 $ExpVal_i^{(2)}$ 表示第 2 个符号变量 $Sym2$ 中第 i 个 $\Omega_r\xrightarrow{w}ExpVal_i$ 相应的 Omega 区域和表达式.

当区域之间的覆盖关系过于复杂时,用近似的判定规则:

$$(\min(Sym1)>\max(Sym2))\Rightarrow(Sym1>Sym2).$$

区域函数表示法还可以方便地表示 Index Array.

例 6:

```
Step 1  DO J=1,JMAX
Step 2    JPLUS(J)=J+1
Step 3  ENDDO
Step 4  JPLUS(JMAX)=1
```

可以表示为:$A(K)=\{1\leqslant K\leqslant JMAX-1\xrightarrow{w}k+1;K=JMAX\xrightarrow{w}1\}$.

2.3 区域覆盖技术的应用范例

本例出自 Perfect Benchmarks 的 MDG.其子程序 INTERF 中的循环 DO 1000 占 MDG 总运行时间的 90%以上,由于传统的数据流方法无法确定数组 RL 是否可以私有化,使得该循环没有被并行执行.本例对该循环有保持问题本质的简化.

```
Step  1  DO I=1,NMOL-1,1
Step  2    DO J=I+1,NMOL,1
Step  3      KC=0
Step  4      DO K=1,9,1
Step  5        RS(K)=XL(K)*XL(K)+YL(K)*YL(K)+ZL(K)*ZL(K)
Step  6        IF (RS(K).GT.CUT2) THEN
Step  7          KC=KC+1
Step  8        ENDIF
Step  9      ENDDO
Step 10      IF (KC.NE.9) THEN
Step 11        DO K=2,5,1
Step 12          IF (RS(4+K).LE.CUT2) THEN
Step 13            RL(4+K)=SQRT(RS(4+K))
Step 14          ENDIF
Step 15        ENDDO
Step 16        IF (KC.EQ.0) THEN
Step 17          DO K=11,14,1
Step 18            FTEMP=AB2*EXP(-B2*RL(K-5))/RL((-5)+K)
Step 19            FF((-5)+K)=FF((-5)+K)+FTEMP
Step 20          ENDDO
Step 21        ENDIF
Step 22      ENDIF
Step 23    ENDDO
Step 24  ENDDO
```

在这个例子里,我们希望确定数组 RL 是否对于外层循环是可私有化的.因此,我们考虑的当前程序段 P 是

从 Step 3～Step 22.

Step 3～Step 9 定义的标量 KC 在 Step 10 和 Step 16 中被读引用,因此 KC 是典型的 IF 语句下的符号变量定义.需要推测其值的情况,参见第 1.3 节中的例 3.

Step 11～Step 15 中的条件写参见第 1.3 节中的例 4.

Step 16～Step 21 中的 IF 语句导致了程序段 P 中的条件读引用区域:$RL(6:9)$.

由条件读的条件谓词 $KC=0$ 可知,$RL(6:9)$的读 Omega 区域是 $KC=0$ 所对应的区域,$\Omega r=\bigcap_{K=1}^{9}\{RS(K)\leqslant CUT2\}$.该条件读的数组引用区域 $Set_P(r)-RL(6:9)$被 Step 11～Step 15 的条件写的全写区域 $Set_P(w)-RL(6:9)$所覆盖:$Set_P(r)=Set_P(w)$,且 $\Omega_r\subset\Omega_w$.

$$(\Omega_r=\bigcap_{K=1}^{9}\{RS(K)\leqslant CUT2\})\subset(\Omega_w=\bigcap_{K=6}^{9}\{RS(K)\leqslant CUT2\}).$$

由前面提出的流不确定性消解原理可知,此处条件读的确切数据源其根据是 Step 11～Step 15 的条件写,因此不会产生跨循环的流相关,由第 2.1 节中数组私有化的判定准则 1 可知,数组 RL 可以私有化.

在 SGI Challenge 4L(4 处理器)SMP 计算机上的实测表明,本例的分析变换使得 MDG 的加速比从 1.0 提高到了 3.7.

3 相关工作比较

由于逻辑条件在语义上的复杂性,即有的并行化编译器,例如,SUIF[7],AFT[8]在进行数据流分析时一般忽略条件分支语句的逻辑条件.在传统的科学计算程序中,控制流比较简单.在影响程序性能的关键代码段中,只有为数不多的条件分支语句,而且语义一般是值不确定性的,上述忽略对数据流分析的精度影响不大.随着科学计算程序的日益复杂以及并行化编译器潜在应用范围的扩大,人们开始注意到如何处理条件分支语句.

Parafrase-2 中的 Join-function 以及 Wolfe 提出的 φ-function,事实上并没有利用条件分支语句的逻辑条件.文献[2,6]对 φ-function 进行了扩充,使之能够携带逻辑条件,并在 Polaris[9]并行化编译器上应用于符号分析等问题.文献[10]提出的 GAR(guarded array region),是对数组区域在概念上进行扩充的一个尝试.但是在 GAR 中,作谓 Guard 的逻辑谓词,其运算采用逻辑推理系统;而数组区域表示为数组数据空间中的凸区域,运算是集合运算系统.两个不同的运算系统运用在同一个对象上,使得 GAR 在理论和实现上都相当复杂.此外,完备的逻辑推理系统集成到实用的编译器中,目前还是令人怀疑的.

本文在计算函数模型的框架下,把逻辑条件的语义运算转化为空间区域的集合运算,其形式与数组区域的运算是一致的.因此 Φ 区域与 GAR 相比是一个更加自然的对数组区域的概念拓广.Φ 区域的运算无需再实现一个新的运算系统,而只需在数组区域运算系统的基础上予以增强和扩充.通过本文的示例可知,Φ 区域覆盖法简洁、有效.此外,把符号变量的值表示为区域函数的形式,并利用区域覆盖的关系进行符号分析,其分析能力不亚于文献[6]中所提出的方法.

4 结 论

引入计算函数模型后,可以对数组区域进行概念上的拓广,提出数组引用的 Φ 区域这一新概念.Φ 区域同时包含了数组引用的引用元素区域的信息和引用的逻辑条件信息,因此,通过 Φ 区域的覆盖关系可以解决数组数据流分析中由条件分支语句导致的流不确定性.此外,由逻辑条件的 Omega 区域还可导出符号变量的取值信息.

区域覆盖技术在数组数据流分析中是比较成熟的技术.本文阐述了如何在实际程序中常见的程序构造下计算逻辑条件的 Ω 区域和数组引用的 Φ 区域,并利用 Ω 区域和 Φ 区域的覆盖关系在相对小的代价之下获得更精确的数据流信息.

参考文献

1 Chen Tong, Zang Bin-yu, Zhu Chuan-qi. A new method for array privatization. In: Tan Chee Klow ed. Proceedings of

High Performance Computing Conference'94. Singapore: National Supercomputing Research Center; National University of Singapore, 1994. 43～50

2 Tu Peng, Padua D. Automatic array privatization. In: Proceedings of the 6th International Workshop on Languages and Compilers for Parallel Computing. Berlin: Springer-Verlag, 1993. 500～521

3 Hu Shi-liang, Zang Bin-yu, Zhu Chuan-qi. Enhancing dataflow analysis with computation function model. Journal of Software, 2000,11(2):187～194
(胡世亮,臧斌宇,朱传琪.用计算函数模型增强数据流分析.软件学报,2000,11(2):187～194)

4 Creusillet B, Irigoin F. Interprocedural array region analysis. In: Huang C-H *et al* eds. Proceedings of the 8th International Workshop on Languages and Compilers for Parallel Computing. Columbus, Ohio: Springer-Verlag, 1995. 46～60

5 Pugh W. A practical algorithm for exact dependence analysis. Communication of the ACM, 1992,35(8):102～114

6 Tu Peng, Padua D. Gated SSA-based demand-driven symbolic analysis for paralleling compilers. In: Wolfe M, Nicole D *et al* eds. Proceedings of the International Conference'95 on Supercomputing. Bacelone: ACM Press, July 1995. 414～423

7 Wilson R, French R, Wilson C *et al*. SUIF: an infrastructure for research on paralleling and optimizing compilers. ACM SIGPLAN Notices, 1994,29(12):31～37

8 Zhu Chuan-qi, Zang Bin-yu, Chen Tong. An automatic parallelizer. Journal of Software, 1996,7(3):180～186
(朱传琪,臧斌宇,陈彤.程序自动并行化系统.软件学报,1996,7(3):180～186)

9 Blume B, Eigenmann R, Faigin K *et al*. Polaris: the next generation in paralleling compilers. In: Proceedings of the 7th International Workshop on Languages and Compilers for Parallel Computing. New York: Springer-Verlag, 1994. 141～154

10 Trung Nguyen, Gu Jun jie, Li Zhi-yuan. An interprocedural paralleling compiler and its support for memory hierarchy research. In: Huang C-H *et al* eds. Proceedings of the 8th International Workshop on Languages and Compilers for Parallel Computing. Columbus, Ohio: Springer-Verlag, 1995. 90～104

Region Coverage Method in Dataflow Analysis

HU Shi-liang ZANG Bin-yu LING Bing ZHU Chuan-qi

(*Parallel Processing Institute Fudan University Shanghai* 200433)

Abstract For a precise dataflow analysis within the framework of the computation function model, the logical relationship of branch conditions is represented as the coverage relationship of regions. In this paper, the authors discuss how to represent, compute and propagate the Ω region, Φ region of the conditional reference. Meanwhile, the methods of resolving the nondeterminism caused by conditional branch statements are also presented in order to get more precise dataflow information.

Key words Ω region, Φ region, array dataflow analysis, array privatization, symbolic analysis.