并行化编译中递归标量的优化处理*

王 诚 臧斌宇 朱家菁 朱传琪

(复旦大学并行处理研究所 上海 200433)

摘要 提出了一种并行化编译中统一处理递归标量的通用方法.该方法将递归标量的处理转化为差分方程(组)的求解,然后利用Z变换与反Z变换来求解方程(组).提高了并行化编译器对递归标量的处理能力,有利于对串行程序的自动并行化.

关键词 递归标量,相关性分析,差分方程,强连通分枝,Z变换.

中图法分类号 TP314

相关性分析是并行化编译中的重要环节,相关性分析能力直接影响到并行化编译器的并行化能力,而对数组变量进行相关性分析是相关性分析的关键.数组的相关性分析依赖于对数组下标表达式的分析.已有的相关性分析方法只能够分析含有循环常量或循环参数的下标表达式,对于含有循环变量的下标表达式则通过表达式替代来消除.由于递归标量的值不仅依赖于循环参数,而且依赖于该变量上一次循环的值,因而数组下标中的递归标量不能通过简单的替代来消除,其值的不确定性影响了数组变量的相关性分析,而且递归标量本身也会产生跨循环的相关,影响并行化.递归标量在数组下标中广泛存在,如果不能对它进行有效的处理,许多可并行化的循环只能因信息不足做保守估计,而不能自动并行化.将递归标量变换为等价的非递归标量,消除其不确定性,可以有效地减少循环之间的相互依赖,提高数组变量相关性分析的精度,也有利于unimodular变换、变量归约及向前替代等并行技术的实现,成为提高并行编译器并行化识别能力的有效手段.

递归标量不能通过常数、表达式的传播或其他常规方法来消除.到目前为止,尚没有统一、有效的方法来处理递归标量,已有的方法都只能处理某些个别和特殊的情况,缺乏普遍性,而且都存在一定的缺点.有些方法通过模式识别来识别固定模式,然后对其进行表达式的替换[1],例如:对于X的循环初值为$X0$,I为循环参数,初值与步长均为1,可识别X=X-1模式,将其替换为X=X0+I;识别X=X+I模式,将其替换为X=X0+I*(I+1)/2等.该方法只能处理简单的多项式,对于多维递归标量则无法处理.也有些方法先识别递归标量表达式的形式,求出一般通式,再利用特殊值求待定系数[2],例如:对于X=2*X+1,变换为差分方程X(I+1)=2*X(I)+1,由差分方程的特征值可求出其一般通式为X(I)=C0*2**I+C1,此处C0,C1为待定系数,代入X(0)=X0,X(1)=2*X0+1,可求出C0=X0+1,C1=-1,所以可替换为X=(X0+1)*2**I-1.该方法处理范围仅限于一部分多项式的情况,而且由于多维递归标量的前几次循环受标量初值的影响而出现异常,无法满足一般通式,因而需要展开前几次循环实例(见例2).该方法难以确定异常出现的循环次数,从而难以选取适当的特殊值以求其待定系数.本文提出了对递归标量的统一的通用处理方法,通过将递归标量标准化为差分方程(组),然后利用控制理论中常用的Z变换与反Z变换的方法来求解差分方程(组),对递归标量进行优化,增强了对递归标量的处理能力,克服了其他递归标量处理方法的缺点.由于我们通过求解差分方程(组)来处理递归标量,因而有坚实的数学基础做保证,而且对多维与一维递归标量可同样处理.而对于受初值影响而需要展开的循环次数也可从方程的解中直接得出.我们对各方程(组)按数据依赖关系进行拓扑分类依次处理,一方程的解可通过

* 本文研究得到国家自然科学基金、国家863高科技项目基金、国家攀登计划基金和上海市重点学科与学术带头人基金资助.作者王诚,1971年生,硕士生,主要研究领域为并行处理.臧斌宇,1965年生,副教授,主要研究领域为并行处理.朱家菁,女,1972年生,硕士生,主要研究领域为并行处理.朱传琪,1943年生,教授,博士生导师,主要研究领域为并行处理.

本文通讯联系人:王诚,上海 200433,复旦大学并行处理研究所

本文1997-09-29收到原稿,1997-12-26收到修改稿

© 中国科学院软件研究所 http://www.jos.org.cn

变量的替代传至引用其变量的另一方程，因而可以按数据依赖对递归标量进行依次处理.而对于嵌套循环，也可由内向外对各循环进行依次处理.该方法的处理能力仅依赖于 Z 变换与反 Z 变换的能力，对于所有可以变换为有理多项式或底数为有理数的指数表达式的递归标量，本文中介绍的算法都能够处理.由于当前对数组的相关性分析只限于处理数组下标为循环参数的线性、幂及简单的指数表达式的情况，因而该算法对递归标量的处理能够充分满足相关性分析的要求.

下面，我们通过几个例子来加以说明.

```
例 1:    DO 10 I=1,100
         N=N+1          (1)
         A(N)=A(N)+I
         ...
      10 CONTINUE
```

```
例 2:    DO 10 I=1,100
         M=P            (2)
         P=N            (3)
         N=P+1          (4)
         A(M)=A(M)+I
         B(N)=B(N)+1
      10 CONTINUE
```

递归标量是指循环中的整型标量，它的赋值直接或间接地依赖于上一次循环中该标量的值.例 1 中，标量 N 就是递归标量，因为它的赋值依赖于其上一次循环的值.我们可以通过对递归标量进行处理，将其变为非递归标量.例如，语句(1)可等价为

$$N=N0+I, \tag{5}$$

其中 $N0$ 为 N 的进入循环时的初始值.

例 2 中，M，N，P 都是递归标量，因为它们的值都间接地相互依赖于上一次循环的值.对其进行的处理，在展开一次循环实例后，也可等价地变为非递归标量的形式：

```
      A(P0)=A(P0)+1
      B(N0+1)=B(N0+1)+1
      DO 10 I=2,100
      M=N0+I-2
      P=N0+I-1
      N=N0+I
      A(N0+I-2)=A(N0+I-2)+I
      B(N0+I)=B(N0+I)+1
   10 CONTINUE
```

其中 $M0$，$P0$，$N0$ 分别为 M，P，N 进入循环时的初始值.该例中，由于 M 的值受 P 初值的影响而出现异常，所以展开了第 1 次循环.

由此可见，展开前几次循环后，有些递归标量的赋值可变为初始值与循环参数的表达式，从而变为非递归标量.在程序的并行化编译中，递归标量的存在常常会阻碍程序的并行化.递归标量本身的值依赖于上一次循环的值，从而产生跨循环的相关，使循环不能并行化.递归标量还常常出现在数组的下标中，其值的不确定性也妨碍了对数组的相关性的分析.如在例 1 中，如果不能知道 N 与 I 的关系，循环就不能并行化.如将式(1)变为式(5)，再代入数组变量 A 的下标中就不难看出，变量 N 和 A 都没有跨循环的数据相关，因而可以并行执行.对于例 2 也是如此.

本文第 1 节讨论如何将递归标量标准化为差分方程(组)，第 2 节给出利用 Z 变换与反 Z 变换对方程进行求解的算法，第 3 节介绍一个实际的求解例子，第 4 节则介绍相关的工作及将来的发展.

1　递归标量标准化为差分方程

对于递归标量的优化处理，首先要将其标准化为差分方程(组)，然后解出其通解形式，代入原程序.通常，递归标量的值可等价于某一递归方程(组)的解.

例如，式(1)可等价于如下差分方程：

$$N(k+1)=N(k)+1(k\geqslant 0),$$

其中 $N(k)$ 为第 k 次循环时对 N 的赋值，$N(0)$ 为进入循环时 N 的初始值，下同.

© 中国科学院软件研究所 http://www.jos.org.cn

例 2 中，式(2)～(4)也可写为如下差分方程组：

$$\begin{bmatrix} M(k+1) \\ P(k+1) \\ N(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} M(k) \\ P(k) \\ N(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (K \geqslant 0).$$

下面，我们通过一个例子具体介绍将递归标量标准化为差分方程的过程.

例 3：
```
      DO 10 I=1,N
      B=2*B
      A=-A-B+C
      C=D+1
      B=-3*A-B+C
      D=I
10    CONTINUE
```

首先，通过改名将循环中的变量赋值变为单赋值语句. 如下所示，循环中存在对同一标量的两赋值语句 $S1$，$S3$：

```
X=...      S1
...        S2
X=...      S3
```

将 $S1$ 变为

```
Y=...
```

同时将 $S1$ 与 $S3$ 之间对 X 的引用变为 Y. 例 3 中对 B 有两处赋值，对第 1 处赋值的 B 改名为 E，并将两赋值之间的对 B 的引用改为 E，则变为：

```
      DO  10 I=1,N
T1    E=2*B
T2    A=-A-E+C
T3    C=D+1
T4    B=-3*A-E+C
T5    D=I
10    CONTINUE
```

对所有循环中的标量单赋值语句，将所有对变量 X 的赋值变为对 $X(k)$的赋值，赋值以前的引用变为 $X(k)$，赋值以后的引用变为 $X(k+1)$，则例 3 中的赋值变为：

$$\begin{aligned} E(k+1) &= 2B(k) \\ A(k+1) &= -A(k) - E(k+1) + C(k) \\ C(k+1) &= D(k) + 1 \\ B(k+1) &= -3A(k+1) - E(k+1) + C(k+1) \\ D(k+1) &= k+1 \end{aligned}$$

对于所得赋值语句 $S1,S2,\ldots,Sn$，将 $S1$ 代入 $S2,S3,\ldots,Sn$，将 $S2$ 代入 $S3,S4,\ldots,Sn$，等等，依次类推，即可得到如下差分方程：

$$\begin{bmatrix} x_1(k+1) \\ \cdots \\ x_m(k+1) \end{bmatrix} = (\lambda_{ij}^*) \begin{bmatrix} x_1(k) \\ \cdots \\ x_m(k) \end{bmatrix} + \begin{bmatrix} u_1(k) \\ \cdots \\ u_m(k) \end{bmatrix} \quad (k \geqslant 0).$$

对于例 3，即可得

$$E(k+1) = 2B(k) \tag{6}$$
$$A(k+1) = -A(k) - 2B(k) + C(k) \tag{7}$$
$$C(k+1) = D(k) + 1 \tag{8}$$
$$B(k+1) = 3A(k) + 4B(k) - 3C(k) + D(k) + 1 \tag{9}$$
$$D(k+1) = k+1 \tag{10}$$

再根据数据依赖关系构造有向图，对有向图求强连通分枝，强连通分枝的凝聚图和凝聚图的拓扑分类后，按拓扑分类顺序依次处理如下：

由式(10)可得(D_0 为 D 的循环初始值，下同)

© 中国科学院软件研究所 http://www.jos.org.cn

$$D(k)=\begin{cases}D_0, & k=0\\ k, & k>0\end{cases}.$$

我们定义 k 的函数 $L(n,k,C_0,C_1,\ldots,C_n)$ 为

$$L(n,k,C_0,C_1,\ldots,C_n)=\begin{cases}C_k, & k\leqslant n\\ 0, & k>0\end{cases},$$

则

$$D(k)=k+L(0,k,D_0)\quad(k\geqslant 0).\tag{11}$$

此处的 L 函数即为差分方程受初值影响而产生的异常,它只影响递归标量的前 n 次循环,因而 n 即为需展开的循环次数.

把式(11)代入式(8)得

$$C(k)=\begin{cases}D(k-1)+1=k+L(1,k,0,D_0), & k>0\\ C_0, & k=0\end{cases},$$

即

$$C(k)=k+L(1,k,C_0,D_0)\quad(k\geqslant 0)\tag{12}$$

把式(6)、(12)代入式(7)得

$$\begin{aligned}A(k+1)&=-A(k)-2B(k)+C(k)\\ A(k+1)&=-A(k)-2B(k)+k+L(1,k,C_0,D_0)\quad(k\geqslant 0)\end{aligned}\tag{13}$$

把式(6)、(12)代入式(9)得

$$\begin{aligned}B(k+1)&=3A(k)+4B(k)-3C(k)+D(k)+1\\ &=3A(k)+4B(k)-3[k+L(1,k,C_0,D_0)]+k+L(0,k,D_0)-1,\\ B(k+1)&=3A(k)+4B(k)-2k+1+L(1,k,-3C_0+D_0,-3D_0)\quad(k\geqslant 0).\end{aligned}\tag{14}$$

令

$$x(k)=\begin{pmatrix}A(k)\\ B(k)\end{pmatrix},$$

由式(13)、(14),求差分方程组可得

$$x(k+1)=\begin{pmatrix}-1 & -2\\ 3 & 4\end{pmatrix}x(k)+\begin{pmatrix}k\\ -2k+1\end{pmatrix}+L\left(1,k,\begin{pmatrix}C_0\\ D_0-3C_0\end{pmatrix},\begin{pmatrix}D_0\\ -3D_0\end{pmatrix}\right)\quad(k\geqslant 0),$$

求解此差分方程即可将例 3 中的所有递归标量变为非递归标量.

2　差分方程的求解

下面讨论对差分方程的求解.假设已求得方程:

$$x(k+1)=Ax(k)+u(k),\tag{15}$$

A 为 m 阶常数方阵,$u(k)$ 只含有循环参数和循环不变量且标准化为有限个 Rk^nc^k 或 $L(n,k,D_0,D_1,\ldots,D_n)$ 的和,即

$$u(k)=\sum_{i=0}^{s}R_ik^{n_i}c_i^k+L(n,k,D_0,D_1,\ldots,D_n),$$

其中 R_i,D_i 为 m 维向量,c_i 为常数,n_i 为非负整数.

对函数 $f(k)$,定义其 Z 变换 $F(z)$ 为:

定义 1. $F(z)=Z[f(k)]=\sum_{i=0}^{\infty}f(i)z^{-i}$. 逆变换 Z^{-1} 定义为:$Z^{-1}[F(z)]=f(k)$. 其中 k 为原函数自变量,z 为变换后函数自变量,小写字母 f 为原函数,相应的大写字母 F 为其 Z 变换后的函数,下同.

为了本文的完整性与便于阅读,我们列出求解上述差分方程所要用到的引理.这些引理的证明可参考文献[3,4].

引理 1. $f(k),g(k)$ 为任意函数,则 Z 变换有如下性质:

© 中国科学院软件研究所 http://www.jos.org.cn

①$Z[af(k)+bg(k)]=aZ[f(k)]+bZ[g(k)]$;

②$Z[f(k+1)]=zZ[f(k)]-zf(0)$;

③$Z[C_k^k c^k]=c^i z/(z-c)^{i+1}\quad (c\neq 0)$;

④$Z[L(n,k,d_0,d_1,\ldots,d_n)]=\sum_{i=0}^{n} d_i/z^i$.

C 为组合符号,即 $C_k^i=k!/[(k-i)!\,i!]$,$!$ 为阶乘,a,b,c,d_i 为常数,i,n 为非负整数.

引理 2. 由任意 m 阶方阵 A 可得

$$(zI-A)^{-1}=\Big(\sum_{i=1}^{m} B_i z^{m-i}\Big)\Big/\Big(\sum_{i=0}^{m} a_i z^{m-i}\Big) \tag{16}$$

B_i,a_i 满足递推式:$a_0=1,B_1=I_m,a_i=-tr(AB_i)/i,B_{i+1}=AB_i+a_iI_m$,其中 $tr(R)$ 为方阵 R 对角线元素之和,即 $R=(r_{ij})_{m\times m}$,则 $tr(R)=r_{11}+r_{22}+\ldots+r_{mm}$.

引理 3. 对任意 n 次多项式 $r(z)$ 以及 m 次多项式 $p(z)$,存在 u 次多项式 $q(z)$ 和 v 次多项式 $t(z)$,使得

$$r(z)=q(z)p(z)+t(z)\text{且 } v<m.$$

定义 2. 对引理 3 中的 $r(z)=q(z)p(z)+t(z)$,定义

$$q(z)=r(z)\%p(z),\qquad t(z)=r(z)\bmod p(z).$$

引理 4. 对任意 n 次多项式 $r(z)$,i 次多项式 $q_i(z),i=0,1,\ldots,n$,存在 $a_i(i=0,1,\ldots,n),a_0\neq 0$,使得

$$r(z)=\sum_{i=0}^{n} a_{n-i}q_i(z).$$

引理 5. n 次多项式 $p(z)$ 和 m 次多项式 $q(z)$ 互素,对任意 s 次多项式 $r(z)$,如果 $s<m+n$,则存在唯一的 v 次多项式 $v(z)$ 和 w 次多项式 $w(z)(v<m,w<n)$,使得

$$v(z)p(z)+w(z)q(z)=r(z).$$

引理 6. 对于 n 次多项式

$$q(z)=\sum_{i=0}^{n} a_i z^{n-i}=a_0\prod_{l=0}^{m}(z-d_l)^{i_l},$$

如果所有的 a_i 都是整数,d_l 为有理数 $d_l=p_l/q_l$(p_l,q_l 为整数且互素),则有 p_l 整除 a_n,q_l 整除 a_0.

下面,我们应用以上引理来求式(15)的 $x(k)$.

对式(15)两边进行 Z 变换,应用引理 1 中 Z 变换的性质(1)、(2)得

$$Z(x(k+1))=Z(Ax(k)+u(k)),\qquad zX(z)-zx(0)=AX(z)+U(z),$$

$$X(z)=(zI-A)^{-1}zx(0)+(zI-A)^{-1}U(z), \tag{17}$$

可求得 $$x(k)=Z^{-1}[X(z)]=Z^{-1}[(zI-A)^{-1}zx(0)+(zI-A)^{-1}U(z)].$$

首先,我们做 Z 变换,求式(17)中的 $X(z)$.

定义 3. 称 $t(z)=p(z)/q(z)$ 为 n 次真分式,如果 $p(z)$ 为 s 次多项式,$q(z)$ 为 n 次多项式,并且 $n>s\geqslant 0$.

定理 1. 式(17)中的 $X(z)$ 可表示为

$$X(z)=\sum_{l=0}^{m} T_l zx(0)t_l(z)+\sum_{l=0}^{m} S_l z s_l(z). \tag{18}$$

T_l 为 $m\times m$ 矩阵,S_l 为 m 维向量,$t_l(z),s_l(z)$ 为真分式.

证明:因为 $q_i(k)=C_k^i$ 为 i 次多项式,由引理 4 及引理 1 可得

$$Z[k^n c^k]=Z\Big[\sum_{i=0}^{n} a_{n-i}C_k^i c^k\Big]=\sum_{i=0}^{n} a_{n-i}Z[C_k^i c^k]=\sum_{i=0}^{n} a_{n-i}c^i z/(z-c)^{i+1},$$

$$U(z)=Z[u(k)]=Z\Big[\sum_{i=0}^{s} R_i k^{n_i}c_l^k+L(n,k,D_0,D_1,\ldots,D_n)\Big]=\sum_{l=0}^{s}\sum_{i=0}^{n_l} R_l a_{n_l-i}c_l^i z/(z-c_l)^{i+1}+\sum_{i=0}^{k} D_i/z.$$

由引理 2 中式(16)可知,

$$(zI-A)^{-1}=\Big(\sum_{i=1}^{m} B_i z^{m-i}\Big)\Big/\Big(\sum_{i=0}^{m} a_i z^{m-i}\Big)=\sum_{i=1}^{m} B_i\Big[z^{m-i}\Big/\Big(\sum_{j=0}^{m} a_j z^{m-j}\Big)\Big]=\sum_{i=1}^{m} B_i t_i(z),$$

所以

© 中国科学院软件研究所 http://www.jos.org.cn

$$X(z)=(zI-A)^{-1}zx(0)+(zI-A)^{-1}U(z)$$
$$-\sum_{i=1}^{m}B_izx(0)[t_i(z)]+\sum_{i=0}^{m}\sum_{l=0}^{s}\sum_{j=0}^{n_l}[B_iR_la_{n_l-j}c^l]z[t_i(z)/(z-c_l)^{j+1}]+\sum_{i=1}^{m}\sum_{l=0}^{k}[B_iD_l]z[t_i(z)/Z^{l+1}].$$

由定义 3 可知，$t_i(z)$，$t_i(z)/(z-c_l)^{j+1}$，$t_i(z)/z^{l+1}$ 为真分式. □

定理 1 的证明是构造性的，不难从中得出求 $X(z)$ 的算法.

下面我们通过式(18)的反 Z 变换求 $x(k)$.

定理 2. 任意真分式 $t(z)$ 必可表示为有限个 $g/(z-d)^i$ 的和，即

$$t(z)=\sum_{l=0}^{s}g_l/(z-d_l)^{i_l+1}\quad(g_l,i_l\text{ 为常数，且 }i_l\geqslant 0).$$

证明：设 $t(z)=p(z)/q(z)$，$p(z)$ 为 s 次多项式，$q(z)$ 为 n 次多项式，$s<n$，对 n 进行归纳：

$n=1$ 时，$q(z)=a_0z+a_1$，$a_0\neq 0$，因为 $s<n$，所以 $s=0$，$p(z)=p$，$t(z)=(p/a_0)/(z-a_1/a_0)$，定理成立.

假设当 $n<k$ 时，定理成立. 当 $n=k$ 时，由引理 6，$q(z)$ 可表示为：$q(z)=(z-z_0)^ir(z)$. z_0 为 $q(z)=0$ 的根，$i>0$，$r(z)$ 为 $n-i$ 次多项式，并且 $r(z)$ 与 $(z-z_0)^i$ 互素.

因为 $s<n=(n-i)+i$，由引理 5 可知，存在唯一的 v 次多项式 $v(z)$ 和 w 次多项式 $w(z)$，使得

$$v(z)(z-z_0)^i+w(z)r(z)=p(z)\quad(v<n-i,w<i),$$

则
$$t(z)=p(z)/q(z)=[v(z)(z-z_0)^i+w(z)r(z)]/[(z-z_0)^ir(z)]$$
$$=w(z)/(z-z_0)^i+v(z)/r(z).$$

由引理 4，$w(z)$ 可表示为

$$w(z)=\sum_{l=1}^{w}a_l(z-z_0)^{w-l},\quad w(z)/(z-z_0)^i=\sum_{l=0}^{w}a_l/(z-z_0)^{l+i-w},\qquad l+i-w>0,$$

因为 $v<n-i<k$，由归纳假设得

$$v(z)/r(z)=\sum_{l=0}^{s}g_l/(z-d)^{i_l+1}(i_l\geqslant 0),$$

所以
$$t(z)=\sum_{l=0}^{w}a_l(z-z_0)^{l+i-w}+\sum_{l=0}^{s}g_l(z-d_l)^{i_l+1}.$$ □

由定理 1、定理 2 可知

$$X(z)=\sum_{l=1}^{m}T_lt_l(z)zx(0)+\sum_{l=0}^{s}S_ls_l(z)z=\sum_{l=1}^{m}\sum_{t=0}^{n_l}T_lg_{l_t}z/(z-d_{l_t})^{i_{l_t}+1}x(0)+\sum_{l=0}^{s}\sum_{t=0}^{r_l}S_lh_{l_t}z/(z-e_{l_t})^{j_{l_t}+1},$$

$$x(k)=Z^{-1}[X(z)]=\sum_{l=1}^{m}\sum_{t=0}^{n_l}T_lg_{l_t}x(0)Z^{-1}[z/(z-d_{l_t})^{i_{l_t}+1}]+\sum_{l=0}^{s}\sum_{t=0}^{r_l}S_lh_{l_t}Z^{-1}[z/(z-e_{l_t})^{j_{l_t}+1}].$$

由引理 1 中 Z 变换性质③、④，可得

$$Z^{-1}[z/(z-c)^l]=\begin{cases}C_k^{l-1}c^{k-l+1}, & c\neq 0\\ L(l-1,k,0,\ldots,1), & c=0\end{cases},$$

即可求得 $x(k)$.

3　实际求解的例子

对例 3 求得的差分方程

$$x(k+1)=\begin{pmatrix}-1 & -2\\ 3 & 4\end{pmatrix}x(k)+\begin{pmatrix}k\\ -2k+1\end{pmatrix}+L\left(1,k\begin{pmatrix}C_0\\ D_0-3C_0\end{pmatrix},\begin{pmatrix}D_0\\ -3D_0\end{pmatrix}\right)$$

求 Z 变换得

$$X(z)=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}z^2/(z^2-3z+2)x_0+\begin{pmatrix}-4 & -2\\ 3 & 1\end{pmatrix}z/(z^2-3z+2)x_0+\begin{pmatrix}1\\ -2\end{pmatrix}z^2/(z^2-3z+2)/(z-1)^2+$$
$$\begin{pmatrix}0\\ 1\end{pmatrix}z/(z^2-3z+2)/(z-1)^2+\begin{pmatrix}0\\ 1\end{pmatrix}z^2/(z^2\quad 3z+2)/(z-1)+\begin{pmatrix}-2\\ 1\end{pmatrix}z/(z^2-3z+2)/(z-1)+$$
$$\begin{pmatrix}C\\ D_0-3C_0\end{pmatrix}z/(z^2-3z-2)-\begin{pmatrix}2C_0-D_0\\ -2D_0\end{pmatrix}1/(z^2-3z+2)+\begin{pmatrix}2D_0\\ 0\end{pmatrix}1/(z^2\cdot 3z+2)/z.$$

© 中国科学院软件研究所 http://www.jos.org.cn

求反 Z 变换得

$$X(k)=\begin{pmatrix}3-2^{k+1} & 2-2^{k+1}\\ -3-3\times 2^{k} & -2-3\times 2^{k}\end{pmatrix}x_0+\begin{pmatrix}-k(k-1)/2\\ k+k(k-1)/2\end{pmatrix}+\begin{pmatrix}C_0+D_0\\ -D_0\end{pmatrix}L(0,k,1)+$$
$$\begin{pmatrix}D_0\\ 0\end{pmatrix}L(1,k,0,1)+\begin{pmatrix}-3 & 2^{k+1}\\ 3-3\times 2^{k}\end{pmatrix}C_0+\begin{pmatrix}-1\\ 1\end{pmatrix}D_0.$$

由于结果中有 $L(1,k,0,1)$，所以展开第 1 次循环，则例 3 可消除递归标量而等价变形为：

```
A=-A0-2*B0-C0
C=D0+1
B=3*A0-4*B0+1+D0-3*C0
D=1
DO 10 I=2,N
A=(3-2**(I+1))*A0+(2-2**(I+1))*B0-I*(I-1)/2+(2**(I+1)-3)*C0-D0
C=I
B=(3*2**I-3)*A0+(3*2**I-2)*B0+I*(I+1)/2+(3-3*2**I)*C0+D0
D=I
10 CONTINUE
```

4 结 论

递归标量对相关性分析等其他并行化技术产生严重影响，详细情况可见参考文献[5,6]. 本文讨论了利用 Z 变换及反 Z 变换将递归标量变为非递归标量的方法，该方法的处理能力与计算复杂度依赖于 Z 变换与反 Z 变换的算法. 本文给出一种 Z 变换与反 Z 变换的具体算法，由于当前对数组的相关性分析只限于处理下标为循环参数的线性、幂及简单的指数表达式的情况，本文中介绍的算法能够处理所有可以变换为多项式或指数表达式的递归标量，因而该算法对递归标量的处理能够充分满足相关性分析的要求. 本文介绍了将递归标量转化为差分方程的过程，文中的例子中没有涉及控制相关，对于存在控制相关的问题则有待于进一步的讨论.

参考文献

1 Padua David A, Wolfe Michael. Advanced compiler optimizations for supercomputers. Communications ACM, 1986, 29 (12): 1184～1202

2 Pottenger B, Eigenmann R. Idiom recognition in the Polaris parallelizing compiler. In: Wolfe M ed. Proceedings of the International Conference'95 on Supercomputing. New York: ACM Press, 1995. 444～448

3 Lovaglia Anthony R, Preston Gerald C. Foundations of Algebra and Analysis. New York and London: Harper & Row Publishers, 1966

4 Reid J Gary. Linear System Fundamentals, Continuous and Discrete, Classic and Modern. New York: McGraw-Hill Book Company, 1983

5 Wolfe Michael. High Performance Compilers for Parallel Computing. New York and London: Addison-Wesley Publishing Company, Inc. 1996

6 Allen John R, Kennedy Ken. Automatic translation of Fortran programs to vector form. ACM Transactions on Programming Languages and Systems, 1987, 9(4): 491～542

Induction Scalar Optimization in Parallelizing Compiler

WANG Cheng ZANG Bin-yu ZHU Jia-jing ZHU Chuan-qi

(*Parallel Processing Institute Fudan University Shanghai* 200433)

Abstract In this paper, a general method is put forward to process the induction scalars in paralleling compiler. This method changes the processing of induction scalars to the solving of difference equations and uses Z transformation and inverse Z transformation to solve equations. It improves the paralleling compiler's ability to process the inductive scalars, which is helpful to the automatic parallelization of serial programs.

Key words Induction scalar, dependence analysis, difference equation, strongly connected components, Z transformation.

© 中国科学院软件研究所 http://www.jos.org.cn