中 国 科 学
1983 年 3 月　　A 辑　　第 3 期

串行运算向量化的下标追踪法

范 植 华
（长沙工学院计算机研究所）

摘　　要

本文为电子计算机向量运算识别器研制了一种识别与构造算法，这种算法自成体系，被命名为“下标追踪法”. 跟坐标方法[1]和时性关系矩阵方法[2]相比较，它不仅增强了对于赋值语句循环的识别能力，而且把识别范围扩张到 IF 与 GOTO 语句，正在诸如 151-3/4 大型序列机一类的系统[3]上加以实施.

文中介绍下标追踪法的基本思想、基本理论和基本方法，主要提出构造时序层次的公理系统——繁衍规则，证明了年长顺序定理与判别准则.

一、可向量化的精确描述

对于可向量化程序段的基本要求如文献[1]所述，本文主要讨论在运算类型不变，数组元素加工群不变的条件下的向量化. 可以调动语序，增添必要的数据暂存语句.

文中所谓的“不可向量化”则指“不能在满足上述条件的前提下向量化”，而非绝对地不可向量化.

跟向量化问题密切相关的另一要素，是源语言和目标语言的表达能力. 本文以 FORTRAN 77[4] 作为源语言，以扩充有类似于 CRAY 数组处理语句[5]的 FORTRAN 77 作为目标语言.

二、限 制 条 件

除下标是整型的要求外，首先讨论具有如下特征的 **A0** 型单层 *DO* 循环：

1. 循环控制变量的始值和步长在编译阶段是可计值的.

2. 循环体中出现的下标表达式线性依赖于循环控制变量，系数在编译阶段是可计值的.

3. 循环体中不含 GOTO，IF，CALL 和 *I/O* 等语句（因而也就无需含有 CONTINUF 语句）.

下面的例子给出这一类循环的典型结构.

例 1. *DO* 11 $I = 2, 100, S$

11 $A(I) = (A(I-1) + A(I+1))/2.0,$

其中，S 是编译时可计值的整常量.

例 2. *DO* 12 $I = 1, 10$

本文 1982 年 6 月 23 日收到.

$A(I) = B(I) * C(I) + D(I) **2$

$A(2*I+9) = E(I) **2 + 3*A(I) + 5$

$A(2*I+10) = 2*F(I) **2 + A(I) + 1$

$G(3*I-2) = B(2*I-1) * C(I) + A(2*I+9) - A(2*I+10)$

$G(3*I-1) = A(I) * A(2*I+9)$

12 $G(3*I) = A(I) * A(2*I+10)$

例 3. $DO \quad 13 \quad I = 1, N$

$A(I) = B(I) * C(I)$

$C(I) = B(I-1)$

13 $B(I) = A(I+1) * D$

例 4. $DO \quad 14 \quad I = 1, N$

$A(I) = 1$

$A(2*I) = 2$

14 $A(3*I) = 3$

例 5. $DO \quad 15 \quad I = 51, 100$

$A(I) = B(I-50) * 0\,5$

15 $B(I) = A(I) * D$

类似于文献[6]的分析，它们的一般形式是：

$$\left.\begin{array}{l} DO \quad LB \quad I = I^0, I^1, S \\ A_1(I_1) = f_1(A_{11}(I_{11}), A_{12}(I_{12}), \cdots, A_{1n_1}(I_{1n_1})) \\ A_2(I_2) = f_2(A_{21}(I_{21}), A_{22}(I_{22}), \cdots, A_{2n_2}(I_{2n_2})) \\ \quad \vdots \\ LB \quad A_m(I_m) = f_m(A_{m1}(I_{m1}), A_{m2}(I_{m2}), \cdots, A_{mn_m}(I_{mn_m})) \end{array}\right\} \tag{1}$$

其中，I^0 和 S 编译时可计值，I_i 和 I_{ij} 均为 I 的线性函数，系数编译时可计值，A_i 和 A_{ij} 均可重名，即作为同一个数组的不同出现，每个 A_{ij} 至少跟一个 A_k 重名，即不考虑没有定值性出现的数组的引用性出现[6]，LB 是作为标号的正整数，f_i 表示函数依赖规则.

设 $I_i = c_i * I + d_i$，$I_{ij} = c_{ij} * I + d_{ij}$. 由下标值前后顺序的变化可知，当条件：

$$\left(\left\{\frac{d_j - d_i}{c_i - c_j}\,\middle|\, \begin{array}{l} A_i, A_j \text{ 同名,} \\ \text{且 } c_i \neq c_j \end{array}\right\} \cup \left\{\frac{d_{ik} - d_i}{c_i - c_{ik}}\,\middle|\, \begin{array}{l} A_i, A_{ik} \text{ 同名,} \\ \text{且 } c_i \neq c_{ik} \end{array}\right\}\right)$$

$$\bigcap (I^0, I^1) = \varnothing \tag{2}$$

(不可计值的量替之以无穷)

不成立时，欲向量化多半要考虑循环区间(不是循环体)的分解. 由于这些交点将引起语序的变化，我们不妨称之为“变序点”. 就是说，倘若上述交集非空，则可用变序点分割循环区间.

三、时 序 层 次

我们采用拟人化的说法描述数组元素之间的数据依赖关系.

设有一个赋值语句：

$$A(2) = B(1) * C(3)$$

我们称 $A(2)$ 有一次定值性出现，$B(1)$ 和 $C(3)$ 各有一次引用性出现[2]. $A(2)$是 $B(1)$ 和 $C(3)$ 的子女，$B(1)$ 和 $C(3)$ 是 $A(2)$ 的父母，$B(1)$ 和 $C(3)$ 发生一次结合（跟人类的情形不同的是父母不一定恰为两个）.

假设接着又执行一个赋值语句:

$$A(2) = A(1) * B(2)$$

我们称这个 $A(2)$ 是第一个 $A(2)$ 的再生，第一个 $A(2)$ 是这个 $A(2)$ 的前身. 在递归的情况下，前身兼作父母.

在沿着串行运算的语义追踪 A0 型循环的过程中，元素的出现构成一个队列. 模仿人类的情形，我们在这个队列里建立繁衍结构. 第一个定值的元素算作第一辈的第一个人，接踵而来的不以任何定值元素作为父母的定值元素均跟他同辈. 一个人父母的最低辈若为 n，他的辈份最高为 $n+1$. 任何一个人不得超过前身及其子女的辈份. 同辈人之间依执行先后确定兄弟关系. 显然，未定值就引用的元素所引用的是他的始值. 我们把他算作原始人，记作第 0 辈，排在第一辈人的上方. 在原始人跟他的第一次定值性出现之间，以及在同一个人的两次相继的定值性出现之间，用再生链和前身链双向链接. 同一个人的几个父母用结合链相接. 一个人若同时作为几个人的父母，为了使其子女能够形成各自父母的结合链，则要把这个人复制，用同体链相接. 这样一来，在元素的出现队列里存在着由各种链形成的繁衍层次.

综上所述，出现队列里的繁衍规则如下:

1. 未定值就引用的出现属于第 0 辈.

2. 任何一个定值性出现尽量安排在非 0 的高辈份上，但是要满足两个前提:

(1) 低于所有父母的辈份.

(2) 不高于前身及其子女的辈份.

3. 同辈人依执行的先后挂兄弟链.

为了在机器实现中采用定长记录，还需要:

4. 首次引用一个定值出现只挂链，不添新人.

5. 再次引用一个定值出现要复制它.

关于出现队列及繁衍层次，成立两个定理.

定理 1. 任给一个 A0 型 FORTRAN DO 循环，其元素的出现队列是唯一确定的.

证. A0 型 FORTRAN DO 循环的执行语序是确定的，即反复地从循环体的第一个语句执行到最后一个语句.

跟 SETL 一类语言不一样，FORTRAN 不提供无序集合之类的数据结构，也就没有诸如 ∀（任取一个）一类操作结果不确定的运算，因而每个语句引用的和定值的元素都是确定的.

综合上述两点，定理 1 得证.

定理 2. 任给一个 A0 型 FORTRAN DO 循环，其元素的出现队列在子女关系下具有层次结构.

证. 子女关系是定义在元素出现有序对集合上的一个谓词 c:

$$c(\alpha, \beta) \Longleftrightarrow \beta \text{ 是 } \alpha \text{ 的子女},$$

把第 i 辈人的集合记作 G_i，谓词 c 具有下列性质:

1. $\{\alpha \mid c(\alpha, \beta)\ \mathrm{AND}\ \beta \in G_0\} = \varnothing$,

即第 0 辈人不是任何人的子女.

2. $(c(\alpha, \beta)\ \mathrm{AND}\ \beta \in G_i\ \mathrm{AND}\ i \geqslant 1)\ \mathrm{IMP}\ \alpha \in \bigcup_{0 \leqslant j < i} G_j$

即第 i 辈人至低是第 $i-1$ 辈人的子女.

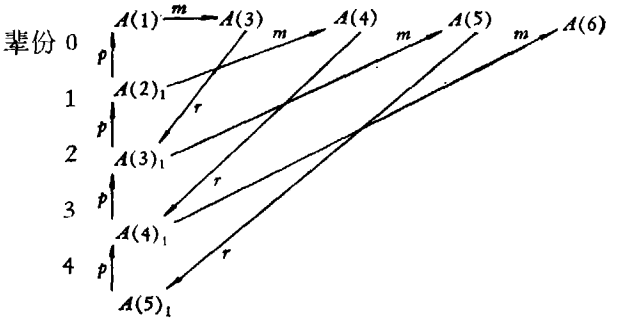
这正是层次结构的数学描述,定理 2 得证.

定义 1. 其元素的出现队列在子女关系下形成的层次结构,称作 A0 型 FORTRAN DO 循环的时序层次.

定理 1 和定理 2 保证了这个定义的合理性.

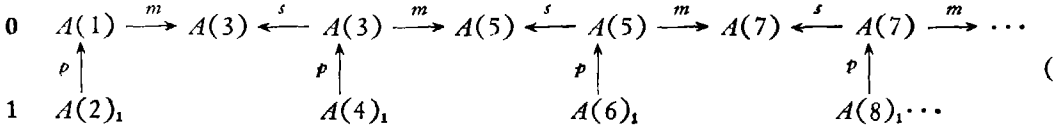
让我们考虑几个构造时序层次的例子. 其中,定值性出现的右下标为语句序号,箭头表示各种链,上面的字母含义是: p——parent 父母链, m——married 结合链, r——reborn 再生链, s——same 同体链. 子女关系隐示于布局之中. 实现时,在异辈人之间还有 c——child 子女链,在同辈人之间还有 y——young 弟链和 e——elder 兄链相接.

例 6. 在例 1 中令 $s = 1$.



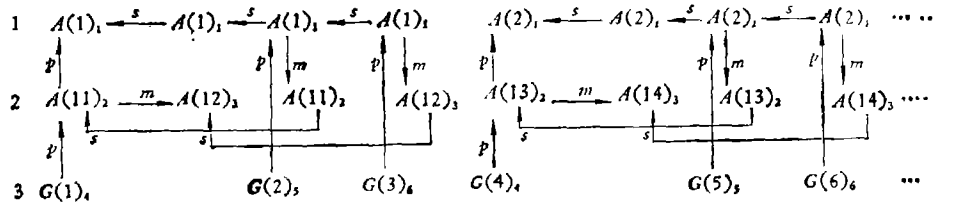
(3)

例 7. 在例 1 中令 $s = 2$.



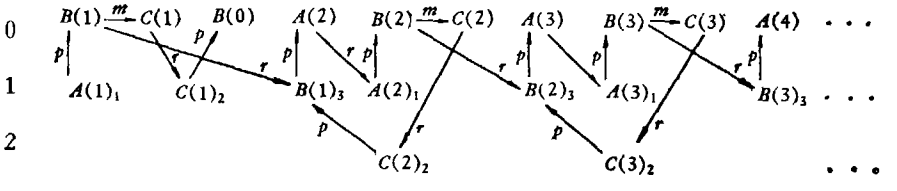
(4)

例 8. (例 2 的继续).



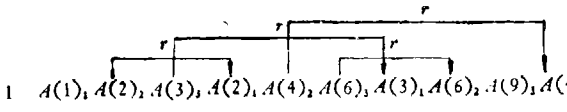
(5)

例 9. (例 3 的继续).



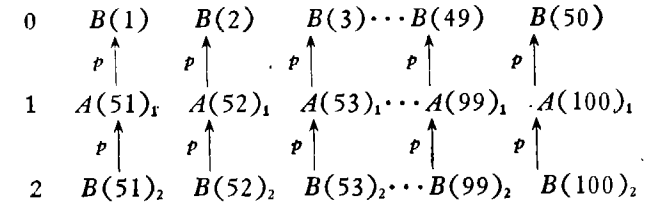
(6)

例 10　(例 4 的继续).



(7)

例 11　(例 5 的继续).



(8)

四、年长顺序定理

元素的出现队列是根据串行程序的语义编排的，即早执行的在前，晚执行的在后，我们称之为语义顺序．然而，由各种链构造起来的时序层次又给出一种“年长”顺序：先长辈后晚辈，先兄姐后弟妹．两种顺序的关系如何？下面的定理确认二者计算结果的一致性．

定理 3　(年长顺序定理).

任给一个 A0 型循环．按其时序层次的年长顺序定值不改变语义顺序的计算结果．

证．我们要证明下述两点：1.各定值性出现的计算结果不变．2.(如果有的话)同一元素多次定值的次序不变．

把数学归纳法施于辈份之上．

第 1 辈，根据繁衍规则 2.(1)，第 1 辈人的父母只能在第零辈，即 DO 循环的入口值．因此，第 1 辈的计算结果不会改变．倘若其中包含有同一元素的多次定值，那么，规则 3 维持本辈中原有的多次定值次序，规则 2.(2) 又保证不会有夹在中间的多次定值之一落到低辈上去．所以，定理对于第 1 辈人成立；

设定理对于 1—n 辈人均成立，现在证明对 $n+1$ 辈人也成立．

根据规则 2.(1)，$n+1$ 辈人的父母最低在 n 辈，它们或者是入口值，或者依据归纳假设计值正确．因此，$n+1$ 辈的计算结果不会改变．倘若 $n+1$ 辈的某个人有前身，那么，依据规则 2.(2)，前身或者在前辈，或者同辈．如果同辈，则由规则 3 保证前身作为哥哥排在前面．于是，不仅每次定值本身正确，而且多次定值的次序不变．所以，定理对于 $n+1$ 辈人也成立．

根据数学归纳法，定理对于所有非零辈份成立，即对于所有的定值均成立．定理 3 证毕．

语义顺序和年长顺序虽然都是串行的，二者却有本质区别：前者只是形成数据依赖关系，丝毫未考虑数据依赖关系；后者则在串行的范畴内对语义顺序加以根本改造，成为以数据依赖关系为主，以语义顺序为辅的年长顺序．不言而喻，这是一种过渡状态，向量化的工作将以辈份为基础进行．

例 12.　(例 3，9 的继续)　根据时序层次(6)，我们可以先用第 1 语句计算 $A(1)_1$，然后用第 2 语句计算 $C(1)_2$，再用第 3 语句计算 $B(1)_3$，把语句号记在右下角，接着计算 $A(2)_1$，$B(2)_3$，$A(3)_1$，$B(3)_3$，…，最后计算 $C(2)_2$，$C(3)_2$，…．定理 3 保证了这样的计算顺序

维持原计算结果不变.

例 13. (例 5, 11 的继续) 根据时序层次 (8), 我们可以先用第 1 语句计算 $A(51)_1$, $A(52)_1$, $\cdots$, $A(99)_1$, $A(100)_1$, 再用第 2 语句计算 $B(51)_2$, $B(52)_2$, $\cdots$, $B(99)_2$, $B(100)_2$. 同样地, 定理 3 保证了原计算结果不变.

五、可向量化判别准则

在本文的意义下, 向量化是以语句为单位进行的. 欲把循环体内的一个语句向量化, 也就是把它通过循环实现的操作用一个向量运算语句, 对整个加工群并发地同时完成. 这样做的必要前提是, 涉及该语句的操作能够调动到一起来顺序执行. 比方说, 象第四节例 13 那样, 把第 1 语句的计算都提到前面执行, 把第 2 语句的计算都推到后面执行.

正如第三节例题所示, 按照异辈之间自上而下, 同辈之间从左到右的规则, 我们能够把时序层次清晰地排列在纸面上. 这样一来, 任意一个定值性出现, 纵向固定于某一辈份中, 横向固定于同辈人行列的某一位置上.

但是, 在不改变计算结果的前提下, 每个定值性出现在时序层次中又有一定的"活动余地": 异辈之间, 向上不可挤入最年幼的父母所在的辈份, 向下不得迈进最年长的子女所在的行列; 同辈之间, 朝左不可逾越前身或其子女, 朝右不得跨过自己的或父母的再生.

然而, 程序设计的经验告诉我们, 使用临时存储, 我们能够把一个定值性出现跟前身的子女或父母的再生交换位置. 例如:

$$A = X,\ B = A * C,$$
$$A = Y.$$

这段程序中, B 是第二个(定值) A 的前身的子女, 第二个 A 是 B 的父母的再生. 它们可以交换次序, 得到对于 A 和 B 的计值等价的程序段:

$$A = X,\ T = A,$$
$$A = Y,\ B = T * C.$$

综上所述, 在时序层次中, 一个定值性出现能够在父母以下, 子女以上, 前身之右下, 再生之左上漂浮. 仅当跨越前身的子女或父母的再生时, 要注意相关数据的暂存, 即可保证计算结果不变. 我们把定值性出现的这一漂浮范围, 称作它在时序层次中的生存带.

定义 2. 任给一个 A0 型循环的时序层次. 定值性出现在各自生存带中的漂浮, 称作对时序层次施行等价变换.

一般而言, 时序层次的初始状态是杂乱无章的, 本文所谓的可向量化则要求把它"修整"得井井有条, 那就是说具有标准形状. 例如在第四节例 12 中, 第二语句的计算都在后面, 只有 $C(1)_2$ 例外, 如果能够把它也调后, 该语句即可向量化.

定义 3. 时序层次的标准形指的是: 由一个语句产生的所有 (定值性) 出现, 连片地位于一个辈份之上.

等价变换是修整时序层次的手段.

于是, 我们得到 A0 型循环在本文的意义下可向量化的判别准则.

定理 4. (判别准则) 一个满足条件 (2) 的 A0 型循环可向量化的充要条件, 是其时序层次能够在等价变换下达到标准形.

证．充分性．设有一个 A0 型循环，其时序层次能够经过等价变换达到标准形，即每个语句产生的出现连片地集中于一个辈份上．从第 1 辈开始，自上而下考察时序层次．倘若某个辈份上只有一个语句产生的出现，那么它们之间无子女关系；又由于对下标表达式有线性递增的限制，它们之间不存在再生关系．所以，它们的相对位置能够随意调换，无数据依赖关系．把这个循环执行的语句改写成类型相同的向量运算语句，即把该语句向量化；倘若某个辈份上有两个或两个以上语句所产生的出现，那么，由于它们是各自连片的，依照先左后右的次序，象前面一样把这些语句向量化．如果通过等价变换化到标准形的过程中需要临时存储，那么，在相应位置添加暂存语句即可实现向量化．由于满足条件(2)，即不包含变序点，这样的处理保持运算类型和数组元素加工群不发生变化．

必要性．设有一个 A0 型循环(在本文的意义下)可向量化，即循环体内每个语句通过循环完成的操作，都能够用同类型的向量运算对它的加工群同时操作．那就是说，由它产生的出现之间既无子女关系也无再生关系，而且能够调动到一起执行不改变计算结果．这意味着，时序层次能够经过等价变换把它们连片地排在一个辈份上，即达到标准形．

让我们考察使用判别准则的几个例子．

例 14 (例 1，例 6 的继续)．

在时序层次(3)中，同一个语句产生的出现 $A(2)_1$，$A(3)_1$，$A(4)_1$，…，依次具有子女关系，分列于不同的辈份上．因此，这个时序层次不能经过等价变换把它们集中于一个辈份而达到标准形，DO 11 $I=2$，100 循环在本文的意义下也就不可向量化．所反映的事实是：该循环在数学上是递推的，串行运算每一步所获得的结果立即用于该语句的下一执行遍中．

例 15 (例 1，例 7 的继续)．时序层次(4)本身就是标准形的．在本文的意义下，DO 11 $I=2$，100，2 循环可向量化．

例 16 (例 2，例 8 的继续)．在时序层次(5)中，第 4，5，6 语句产生的出现都在第 3 辈；第 2，3 语句的都在第 2 辈；第 1 语句的都在第 1 辈．为了达到标准形，我们要解决左右位移修整第 2，3 辈的问题．

从第 3 辈开始．$G(4)_4$ 根本没有前身，从而不存在左限界，能够经过等价变换逾越 $G(3)_6$ 和 $G(2)_5$ 向 $G(1)_4$ 靠拢，把第 4 语句产生的所有出现连片．类似地，第 5，6 语句的，以及第 2 辈上第 2，3 语句的均可连片．于是，DO 12 循环可向量化．

例 17 (例 3，例 9 的继续)．在时序层次(6)中，第 2 辈上全是第 2 语句产生的出现．第 2 语句的唯一例外是 $C(1)_2$，它没有再生，也没有子女，能够降到第 2 辈上来．它的父母 $B(0)$ 没有再生，不存在数据暂存问题．

第 1 辈上还剩下第 1，3 语句产生的出现．为了使 $A(2)_1$ 向 $A(1)_1$ 靠拢，应当把它跟 $B(1)_3$ 对调．但是，后者是其前身的子女，须暂存 A 的前身(向量运算，要存就存一个加工群，不考虑暂存单个元素)．类似地，使 $A(3)_1$ 向左靠拢，…，第 1，3 语句产生的出现分离开来，从而达到标准形．

由此得知，DO 13 循环可向量化．

例 18 (例 4，例 10 的继续)．在时序层次(7)中，$A(2)_2$ 是 $A(2)_1$ 的前身，两者不能对调，$A(2)_1$ 也就不能左移向 $A(1)_1$ 靠拢．于是，把 $A(1)_1$ 跟 $A(2)_2$ 对调，使 $A(1)_1$ 逐步向 $A(2)_1$ 靠拢，…，终于把第 1 语句产生的出现移动到最右边连成片．剩下第 2，3 语句产生的出

现交错排列. 由于$A(6)_3$ 是 $A(6)_2$ 的前身，同理，第 2 语句产生的出现不能在左边连片. 最后的结果是：第 3 语句产生的出现在左边，第 2 语句的在中间，第 1 语句的在右边连片而达到标准形. DO 14 循环可向量化.

例 19.(例 5，例 11 的继续). 时序层次(8)本身就是标准形的，DO 15 循环可向量化.

参 考 文 献

[1] Lamport, L., The Parallel Execution of Do loops. *Com. ACM*, 17 (1974). 2: 83 页.
[2] 吴明霞、陈火旺，科学探索，1 (1981)，4:33.
[3] 慈云桂，电子学报，1980，4:29 .
[4] White, F. C. et al., *American National Standard Programming Language FORTRAN*, American National Standards Ins., 1978.
[5] Charles Wetherell, *Array Processing for FORTRAN*, Cray Research Inc., 1980.
[6] 吴明霞，陈火旺，计算机学报. 4 (1981)，3:174.