7P-9

ベクトルコンパイラにおける変数のデータフロー解析の方式

田中義一、金田　泰、安村通晃
（日立製作所中央研究所）

1．はじめに

近年、技術計算の分野にはベクトルプロセッサと呼ばれるスーパーコンピュータの進出が目覚ましい。ベクトルプロセッサのためのコンパイラ、すなわち自動ベクトル化コンパイラは，FORTRANプログラム中のDOループ部分を解析して，ベクトル演算用オブジェクトの生成を行なうものであり、従来プログラミング言語からの高速ハードウェア機構へのアクセスを可能としている。

ところで、従来の自動ベクトル化コンパイラでは代入文とCONTINUE文のみからなるDOループをベクトル化の対象としていた。しかし、最近のベクトルプロセッサ（たとえば、当社のHITAC M-280HIAP, S-810）には、条件文をベクトル化できるハードウェア機構を有しており、この機構を生かすための自動ベクトル化コンパイラ技術を確立する必要があった。

本発表は、条件文のベクトル化のために必要な技術の中から、単純変数のデータ参照関係の検出方式について述べたものである。

2．変数のデータ参照関係解析の必要性

ベクトル演算を適用できるためには、変数の定義と使用，または定義と定義の順序関係が変わらないことが必要である。このため、変数間の順序関係を検出することはベクトルコンパイラにとって必須の技術であったが、条件文をサポートするに際し、制御フローまで考慮した解析が必要になった。つまり、問題は条件文を含むループ中に変数の定義が存在する場合、ループ入口から出口までのパス上で　Φ－「あるパス上で使用が定義に先行」（図1－1）、Ψ－「すべてのパス上で定義が使用に先行」（図1－2）、Ω－「ΦまたはΨ以外の場合」（図1－3）　のいずれの関係にあるかを解析することである。ΨまたはΩの関係にあればベクトル演算化可能であり、Φの関係にあれば、このままではベクトル演算化できない。

```
DO 10 I = 1, 10
 IF(A(I).EQ.B(I)) THEN
    A(I) = S + B(I)
  ELSE
    S = A(I)
 END IF
10 CONTINUE
```

(1)

```
DO 10 I = 1, 10
 IF(A(I).EQ.B(I)) THEN
    S = A(I)
  ELSE
    S = B(I)
 END IF
10 CONTINUE
```

(2)

```
DO 10 I = 1, 10
 IF(A(I).EQ.B(I)) THEN
    S = A(I)
 END IF
10 CONTINUE
```

(3)

図1　変数の出現の順序関係

3．変数のデータ参照解析のアルゴリズム

ブロック（制御の移動のない一連の文のかたまり）の出口での変数Sに関する順序関係を示す値R，D，Uをもつ量REL（S）を次の様に定義する。

REL（S）＝ ループ入口からブロック末尾までのパス上で
- R：変数SがΦの関係にある。
- D：変数SがΨの関係にある。
- U：変数SがΩの関係にある。

このＲＥＬ$_i$（Ｓ）を求めるために、各ブロックについて次の量を求めておく。

$$GEN_i(S) = \begin{cases} R：ブロック_i で変数Sの使用が定義に先行（使用のみも含む） \\ D：ブロック_i で変数Sの定義が使用に先行（定義のみも含む） \\ U：ブロック_i には変数Sはない。 \end{cases}$$

これらの量の間に右図に示すような演算$\cap^*, \cup^*$を導入すると、次の等式が成り立つ。

$$REL_i(S) = (\mathop{\cap^*}_{j \in Pred(i)} REL_j(S)) \cup^* GEN_i(S)$$

注）

（演算項）

$\cap^*$	R	U	D
R	R	R	R
U	R	U	U
D	R	U	D

（被演算項）

$\cup^*$	R	U	D
R	R	R	R
U	R	U	D
D	D	D	D

図２　演算$\cap^*, \cup^*$の定義

この方程式を解くためのアルゴリズムを図３に示す。もしプログラムに逆方向分岐がない場合、ループ入口ブロックよりdepth first orderingに従ってブロックをスキャンすれば繰り返し数は一回でよい。また逆方向分岐を含む時の収束性は次の様に説明できる。値Ｄ，Ｕ，Ｒの大小関係をＤ≧Ｕ≧Ｒ，Ｄ≧Ｄ，Ｕ≧Ｕ，Ｒ≧Ｒの様に定めると、$REL_i(S) \overset{\leq}{=} min(REL_j)\ \cancel{GEN_i}$（ただし、$GEN_i = D$かつ$\cap^* REL_j = U$の時を除く）である。また$GEN_i = D$かつ$\cap^* REL_j = U$となった時それ以後$REL_i = D$またはＲであるが、Ｒと一度なった時はブロック$_i$の$REL_i$の値が影響する範囲のブロック$_k$の$REL_k(S)$は以後Ｒとなる。このことから収束性が証明される。

また、解は一般に一意的でない。たとえば内部ループをもつ場合、逆方向分岐をもつブロック$_i$の$REL_i(S)$の値をＲとした場合も等式は成り立つ。今の場合、意味のあるのは最大解であるから、$REL_i(S)$の初期値としてＤとした図３のアルゴリズムでもとめることができる。

```
begin
 REL0(S) = U   (0: ループ入口ブロック)
 for each block i do REL_i(S)=D
 repeat
  UNCHANGED=true
  for each block i do
   begin
    new = (∩* REL_j(S)) ∪* GEN_i(S)
          j∈Pred(i)
    if new ≠ REL_i(S) then
     begin
      UNCHANGED = false
      REL(S) = new
     end
   end
 until UNCHANGED
end
```

図３　解析アルゴリズム

４．適用例

上述のアルゴリズムを適用した例を図４に示す。$REL_5(S) = U$であり変数Ｓは存在するパスでは定義が先行することを示している。

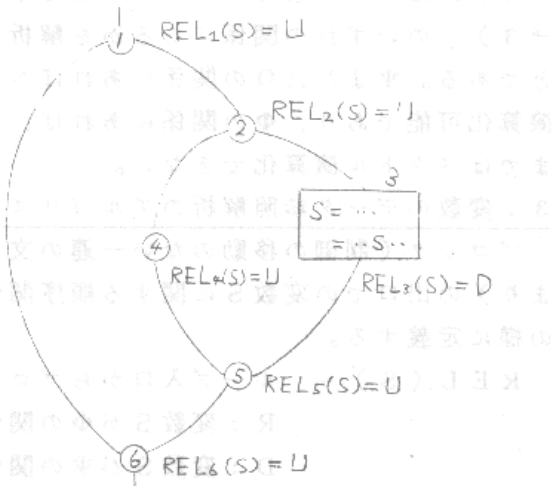

図４　アルゴリズムの適用例

５．おわりに

本報告で述べた解析方式により、条件文等の複雑な構造を含むDOループ内の変数のデータ参照関係が明らかとなった。

６．参考文献

A. V. AHO & J. D. ULLMAN: PRINCIPLES OF COMPILER DESIGN, ADDISON WESLEY

注）Pred(i)：ブロック i の Predecessor ブロック
例）図４で　Pred(6) = 1, 5