誌上討論

ブロック構造の処理プログラムについて*

小　島　　惇**　　石　内　祥　介**

ALGOL 60 コンパイラーのブロック構造の処理について，先に本誌上で野崎氏が一つの例を紹介されたが***，筆者が WALT で用いた方法は，identifier の登録あるいは消去に多少手間がかかってもそのテーブルの索引に要する時間を最小にすることが全体として能率的であるという考え方に基いたもので，野崎氏の三つの公準のうち，PII については多少優れ，PIII については同等であるが，処理プログラムのステップ数は多少多いと思われる．

テーブルの索引に要する時間を最小にするためには必要最小限の identifier をテーブルにのせておくこと，かつ索引すべきテーブルは分割しないことなどが必要である．ここで述べるアルゴリズムは，このような要求を満たすための方式を原理的に説明するもので，identifier の登録あるいは消去に伴なう array, procedure, switch などの処理は省略する．

1. 用語の説明

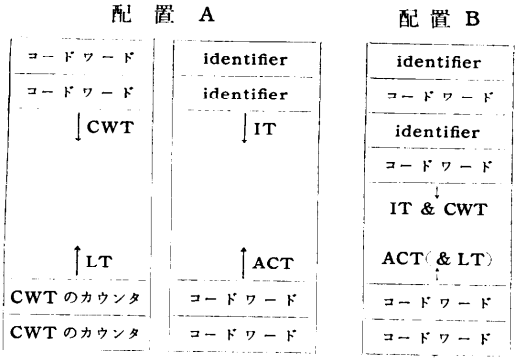
IT: 登録されている identifier のテーブル

CWT: 登録されている identifier の諸情報を含むコードワードのテーブル．アクセス・タイムの問題がなければ，IT と CWT は同じテーブルを使用し，identifier とそのコードワードを並べることができる．

ACT: CWT の補助テーブルで，同じ identifier が内側のブロックで宣言されたとき，先に宣言された方のコードワードを納める．

LT: ACT に入る情報が CWT のどこから取出されたかを記憶する．この情報を ACT のコードワードに含めることができれば LT は要らない．

以上の四つのテーブルは次のように配置する．この理由は同じ identifier を多数回使用すれば IT と CWT は短くなるが ACT と LT は長くなり，同じ identifier の使用が少なければ IT と CWT は長くなるが ACT と LT は短くなり，全体として効果的にテーブルを使用できるからである．



N: IT と ACT に許される最大の大きさ．

IT end: 登録されている identifier の数＝IT および CWT の大きさ．

ACT end: ACT および LT の大きさ．

LC: non-own タイプの simple variable に割付ける番地を value として持つ．

OWNC: own タイプの simple variable に割付ける番地を value として持つ．

Tb: コードワードの原形で identifier のタイプ，種類，BF などの情報を含む．

BF: その時点までの (**begin** の数−**end** の数)

bf: CWT から ACT にコードワードを移すとき bf: ＝BF−1 としてそのコードワードに入れる．

なお，以下の記述中，A{B} は B がいくつかの情報を含んでいるとき，その中の A という情報だけを取り出すことを意味するものとする．

2. アルゴリズム

procedure declaration (I);

comment I という identifier が宣言されたとき，IT を調べて同じ identifier があるかどうかみる．

* On a Method for Processing the Block Structure of ALGOL Compiler, by Jun Kojima and Yosuke Ishiuchi (Electronic Computation Centre, Waseda University)

** 早稲田大学電子計算センター

*** 番地割付の諸問題: 野崎昭弘: 情報処理, Vol. 3, No. 6, 1962, p. 312.

```
あれば，そのコードワードをCWTから取り出して ACT に移し，新しいコードワードをその場所に入れる．同じ identifier が無ければ，I とそのコードワードを IT と CWT の最後に入れる．;
begin integer counter;
      switch SW 1:=SV 1, A 1, S 1, P 1, L 1;
      swich  SW 2:=SV 2, A 2, S 2, P 2, L 2;
TABLE SEARCH 1:
   begin counter:=IT end-1;
 1: if counter<0 then go to 2 else if IT
   [counter]=I then go to 3 else counter:=
   counter-1; go to 1 end;
 2: begin counter:=IT end;
       IT [counter]:=I;
       IT end:=counter+1; go to 4 end;
 3: begin ACT [ACT end]:=CWT [counter]
                +(BF-1);
       LT [ACT end]:=counter;
       ACT end:=ACT end-1 end;
 4: if TYPE{Tb}=OWN then
          go to SWI [KIND{Tb}]
    else go to SWZ [KIND{Tb}];
SV 1: begin CWT [counter]:=Tb+OWNC;
      OWNC:=OWNC+1; go to END end;
SV 2: begin CWT [counter]:=Tb+LC;
      LC:=LC+1; go to END end;
   ……simple variable 以外は省略……
END: end
procedure block end;
comment まず ACT のコードワードのうち，bf=BF であるものを CWT に戻し，次いで CWT のコードワードの BF が現在の BF より大きい identifier を消去する．(simple variable の外は，消去に際して必要な処理の記述を省略する．);
begin integer counter, i;
TABLE SEARCH 2:
begin comment ACT のコードワードの bf が現在の BF と等しければこれを CWT に移し，CWT にあったコードワードを消す．ACT の bf は小さい方から順に並んでいるから，テーブルを逆にサーチして bf<BF となったときこのサーチを終る．;
   counter:=ACT end+1; i:=0;
 1: if counter>N∨bf{ACT [counter]}<BF
   them go to TABLE SEARCH 3;
   if TYPE{CWT (LT [counter])}≠OWN
   then i:=i+1;
   CWT〔LT [counter]〕:=ACT [counter]
          -bf{ACT [counter]};
   ACT end:=counter;
   counter:=counter+1;
   go to 1 end;
TABLE SEARCH 3:
   begin comment CWT のコードワードに含まれる BF が現在の BF より大きければ対応する identifier の登録を取り消す．この段階で，消去すべき identifier はかならずテーブルの終りの方に並んでいるから，テーブルを逆に調べて行き BF{CWT [counter]}≦BF となったとき処理を終る．;
      counter:=IT end-1;
 2: if counter<0∨BF{CWT [counter]}<BF
      then go to END;
      if TYPE{CWT [counter]}≠OWN
      then i:=i+1;
      counter:=counter-1; go to 2 end;
END: LC:=LC-i; IT end:=counter+1 end
```

参考文献

1) Compiling Technique に関するレポート(1)：小島惇，藤野喜一：情報処理学会アルゴル研究会．

(昭和38年6月11日受付)