4D-6

SIMPOSのプログラミング環境
―コール・カウンタ―

川合　英夫　　近山　隆　　中島　克人
(三菱電機(株))　((財)ICOT)

1. はじめに

我々はSIMPOSのプログラミング環境を開発中である。SIMPOSは現在その基本機能開発の段階から、機能拡張及び改良の段階へ移行しつつある。このため、プログラムの実行中にプログラムのどの部分がどれくらいの頻度で呼び出されるかを知ることが出来れば、プログラムの改良等に役立てることが出来る。そこで、ESP の実行単位である述語の呼び出し回数を、任意の計測期間(任意の時点から任意の時点まで)について計測し、その結果を見やすい形で出力するツール、コール・カウンタを作成した。コール・カウンタはウィンドウを用いて、簡単な操作で述語呼び出し回数の計測及び結果の出力を行なうことが出来る。

本稿では、このコール・カウンタの計測方式及びその機能について述べる。

2. 計測方式

ESP の機械語KL0 のマイクロ・インタプリタにはコール・カウンタ用のフラグが用意されている。

マイクロ・インタプリタはこのフラグが“ON”の場合に、述語呼び出しのヘッド・ユニフィケーションを行なう時点でその述語の呼び出しカウントに“1”を加え、フラグが“OFF”であればカウントを行なわない。また、failした場合のアルタナティブ・クローズの再実行もカウントを行なわない。

なお、このフラグはコール・カウンタが操作するため、述語呼び出し回数の計測対象となるプログラムに、計測のための変更を加える必要はない。

コンパイラによってKL0 に変換された述語(プロシジャ)は図1に示されるような形式で主記憶上におかれており、各述語の属性等の情報を保持するベクタのうちの一語が述語呼び出し回数のカウンタとして使用される。

コール・カウンタでは、上に述べたように、述語呼び出し回数のカウントをマイクロ・インタプリタによって行うため、計測機能をソフトウェアで実現するのに較べて非常に高速であり、カウントを行なうことによる速度の低下は10パーセント程度である。

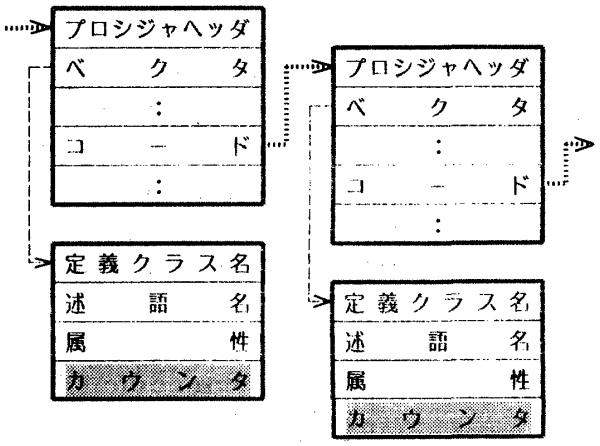

図1　KL0 の内部表現

3. ユーザ・インタフェース

コール・カウンタの操作はコール・カウンタ・ウィンドウを用いて行なう。コール・カウンタ・ウィンドウは図2に示され、コール・カウンタに対して以下の操作が可能である。

(1) START
　述語呼び出し回数の計測を開始する。

(2) STOP
　述語呼び出し回数の計測を停止する。

(3) RESET
　すべての述語のカウント値を“0”とする。

(4) 出力下限カウント値の指定
　何回以上呼び出されている述語を出力対象とするかを指定する。

(5) 出力下限順位の指定
　ソートした結果の上位何位までの述語を出力対象とするかを指定する。

(6) 出力対象クラス名の指定
　どのクラスで定義されている述語を出力対象とするかを指定する。複数クラス名の指定や、ワイルドカードによる指定も可能である。

(7) ソートモードの指定

Programming Environment of SIMPOS ···Call counter

Hideo Kawai, (Mitsubishi Electric corp.)　Takashi Chikayama, (ICOT)　Katsuto Nakajima (ICOT)

述語呼び出し回数の多い順にソートするか、述語名のアルファベット順にソートするかを指定する。

(8) 出力先の指定

ファイルあるいはプリンタが指定できる。

(9) OUTPUT

計測結果の編集及び出力を行なう。

(10)EXIT

コール・カウンタから抜ける。

上記の(4) ～ (8)の指定は、(9) で出力を行なう前であれば自由に変更できる。また、一度計測結果を出力してもカウンタをリセットあるいは計測を再開しない限り、各述語のカウント値は保存されているため、(4) ～ (8)の指定を変更して何度でも出力することができる。リセットせずに計測を再開した場合には、カウント値は前の値に加算される。

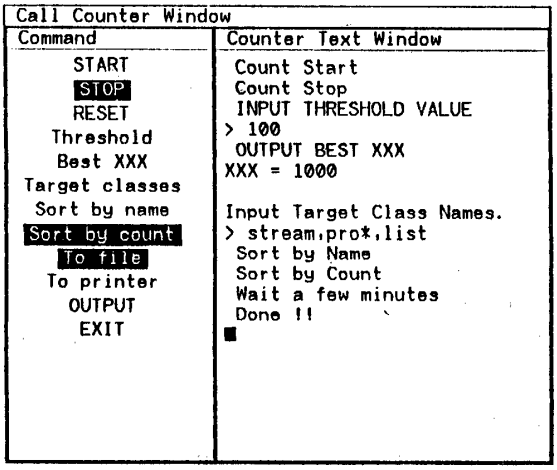

図2　コール・カウンタ・ウィンドウ

4．計測結果の編集及び出力

計測結果の編集を行なう際に、コール・カウンタはSIMPOSのライブラリ・サブシステムの機能を用いる。

ライブラリ・サブシステムでは、PSI 上に存在するすべての述語（プロシジャ）を管理しており、ライブラリ機能によってすべてのプロシジャ・ヘッダへのポインタを辿ることが出来る。

コール・カウンタはこのポインタを順に辿ることによって、図1に示されるような述語ごとの個有の情報（その述語が定義されているクラス名、述語名、属性及び呼び出しカウント値）を得ることができる。コール・カウンタは計測結果の出力に先立って、まず述語の個有情報の中の呼び出しカウント値及び定義クラス名が、ユーザの指定した出力下限カウント値（3の(4)）及び出力対象クラス名（3の(6)）を満たすか否かを判定し、これを満たした述語の個有情報を出力の候補としてリスト型式につないでゆく。

次に、コール・カウンタはこのリストをユーザが指定したソートモード（3の(7)）に従ってソートし、各述語ごとに順位、呼び出し回数、計測期間中の総呼び出し回数に対する比率、定義クラス名及び述語名等をユーザが指定した出力先（3の(8)）に出力する。なお、出力は、ユーザが指定した出力下限順位（3の(5)）に達するか、出力候補がなくなった時点で打ち切られる。出力例は図3に示される。

また、出力先をファイルとした場合、このファイルはエディタで編集可能であり、改めてプリンタやフロッピーディスクに出力することも出来る。

5．おわりに

FSP の述語呼び出し回数を計測し、計測結果を見やすい形で出力するツール、コール・カウンタを作成した。

コール・カウンタはSIMPOSの改良等に実際に使用されており、かな漢字変換やファイルシステム等の高速化にも利用された。今後、SIMPOSの他の部分の高速化にも利用される予定である。

```
PREDICATE CALL COUNTER OUTPUT LIST

①ORDER    ②COUNT      ③RATE       ④(ACCUM)      ⑤CLASS / NAME / TYPE / ARITY
   1:       32024       16.88%      (16.88%)       bitmap_operation / wait_transfer / local / 0
   2:        6807        3.58%      (20.47%)       character / range / local / 3
   3:        3646        1.92%      (22.39%)       bitmap_operation / start_io / local / 16
   3:        3646        1.92%      (24.31%)       bitmap_operation / operation_table / local / 4
   5:        3000        1.58%      (25.89%)       ready_queue / sweep / local / 3
   6:        2875        1.51%      (27.41%)       process / self / cmp / 2
   7:        2740        1.44%      (28.85%)       window / list / cmp / 2
   8:        2425        1.28%      (30.13%)       as_doubly_linked_circle / hook / imp / 2
             1000        0.59%      (51.05%)       process / sub_priority / imp / 2
  32:        1061        0.55%      (51.61%)       context / get_number / imp / 2
  33:        1012        0.53%      (52.14%)       as_doubly_linked_circle / empty / imp / 1
  34:        1005        0.52%      (52.67%)       stream / put / imp / 2

Total :       189691   ⑥
Best XXX :     10000   ⑦
Threshold :     1000   ⑧
Sort Mode :  by_count  ⑨
```

①：順位　②：呼び出し回数値　③：総呼び出し回数値に対する比率
④：③の積算値　⑤：定義クラス名／述語名／タイプ／引数個数
⑥：総呼び出し回数値　⑦：出力下限順位　⑧：出力下限カウント値
⑨：ソートモード

図3　コール・カウンタの出力例