2H－8

言語間共通ベンチマークプログラムの生成方式

青野　博、伊藤　琢巳、菅原　昌久
NTTソフトウェア研究所

１．はじめに

コンパイラの性能評価は開発、機能拡張時あるいは、購入コンパイラ選定時などに行われいるが、その作業は各言語毎にベンチマークテストプログラム（以下ＴＰという）群を作成し、走行させるのが一般的である[1]。しかしこの場合、過去に使用された他言語のＴＰを流用することができず変換に手間がかかる、あるいは新たにＴＰを開発する手間がかかる、等の問題点がある（図１）。

この手間を減らし実行したいＴＰを簡単に得られるようにする、つまり評価する処理系に依存しないでＴＰを生成するために、ある言語で記述されたＴＰ群をもとに評価対象言語に対応したＴＰ群を生成するジェネレータの開発を行った。

本稿では以下、ジェネレータの構成、実現法、Ａｄａ、ＣＨＩＬＬ、Ｃの３言語間共通ベンチマークＴＰ生成への応用例を報告する。

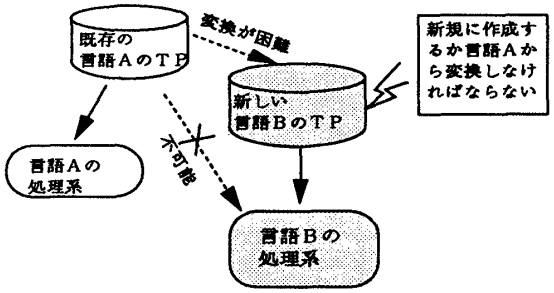


図１．　従来のＴＰ作成時の問題点

２.ジェネレータ概要

本章では、ＴＰジェネレータの構成、その入力を記述するためのＴＰ定義データ記述言語について述べる。

（１）構成概要

ジェネレータの構成を図２に示す。図２に示されるようにジェネレータは、ある言語（後述）で記述されたＴＰ定義データ群と変換器から構成される。変換器は、出力すべき言語の言語仕様定義データとＴＰ定義データ群を用いて言語対応ＴＰソース集合を生成、出力する。

言語仕様データを入れ換えるだけで、多くの言語の処理系に対応したＴＰをＴＰ定義データ群から簡単に生成することが可能となる。また、１つのTP定義データ群は、１つのTPソースに対応するため、それを追加すれば生成される

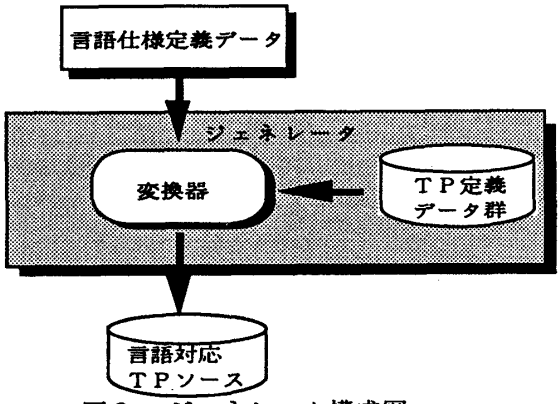


図２．　ジェネレータ構成図

TPソースの追加も可能である。

（２）ＴＰ定義データ記述言語

ＴＰ定義データは、TPソース生成の「種」であるため、評価対象となる多くの言語の共通仕様の形で記述されていた方が変換を効率良く容易に行うことができる[2]。そこで、今回は手続き型言語に的を絞り構造化言語の一つであり多くの言語のベースとなっているＰａｓｃａｌをベースにした言語を採用した。

ＴＰデータ群記述言語で書いたプログラムの例を以下に示す。（例１）

```
program bubble;
var buffer : array[1000 ] of integer;
temp : integer;
i,j,k : loop;
begin
    start_test ;
    for i:=0 to 10 do
    begin
        for i:=0  to 999 do
            buffer[j]:=999-j;
        for k:=0 to 998 do
            if buffer[j]>buffer[j+1] then
            begin
                temp := buffer[j];
                buffre[j] := buffer[j+1];
                buffer[j+1] := temp
            end
    end;
    stop_test;
    feature_times
end.
```

例１．　ＴＰ定義データ記述言語で記述したbubble sortの例

下線の部分がＰａｓｃａｌとは異なっている点である。これらが異なっている理由は、言語によって配列の宣言方法やループカウンタに使用する変数の扱いが異なるためである。また、ベンチマークとして必須である時間計測を行うために表１に示す５つの手続きをＴＰ定義データ記述言語に追加した。

Generation of Benchmark Programs Common to Multiple Languages.
Hiroshi.AONO,Tokumi.ITOH,Masahisa.SUGAHARA
NTT Software Laboratories.

表1　時間計測用に追加した手続き

手続き名	機能
start_control	ダミーステートメントのみを含むループの時間計測を開始する。
stop_control	ダミーステートメントのみを含むループの時間計測を終了する。
start_test	実際にテストしたい処理を含むループの時間計測を開始する。
stop_test	実際にテストしたい処理を含むループの時間計測を終了する。
feature_time	各ループにかかった時間の差を計算し、テスト対象の処理のCPU_TIMEを計算し、出力する。

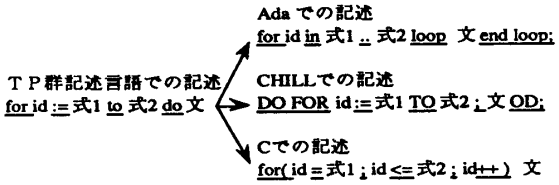
3.変換器の実現

変換器は言語仕様定義データから対象言語とＴＰ定義データ記述言語との対応リストを読み込み、それを基にＴＰデータ群に含まれるＴＰを対象言語に変換する。

本章では、その特徴的な部分について述べる。

（1）言語仕様定義データの表現

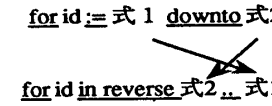
言語仕様定義データはＴＰ定義データ記述言語に対応して一つのステートメント単位で表現されている。ｆｏｒ文に関する言語仕様データの例を以下に示す（例2）。この例では、for..文　までが、一つのステートメントとなっている。同様に他のステートメントについても言語仕様定義データとして登録されている。



例2．　Ada,CHILL,Cにおけるｆｏｒ文の表現

（2）変換の方法

ＴＰ定義データ群に含まれるＴＰは、ステートメント単位で対象言語に変換されていく。具体的には、(a)順にＴＰ定義を読み込み、(b)idや式などを評価し、(c)その後対象言語の言語仕様データに従って、一括して変換を行う。このため変換は、効率の良い1パス置換アルゴリズム[1]で実行される。例えば、例3のようにdowntoという予約語がfor文中に出現しても、1パスで変換が可能となる。



例3．　downtoをはさんで式1と式2が逆転

（3）特別な変換

例えば、CHILLの場合ではloop文からexitする場合ラベルが必要となるため、変換器は自動的にラベルの生成を行っている。このような特別な変換は、今のところ言語仕様定義データ上で記述せず、特別扱いをして処理を行うようにしている。

4.3言語共通ベンチマークTP作成への応用

ジェネレータによるＴＰ生成の応用例としてＡｄａ、ＣＨＩＬＬ、Ｃの3言語共通ベンチマークＴＰを作成した。

表1の時間計測用の関数を使用するために、Adaの場合はwith節、use節、ＣＨＩＬＬの場合はFROM文、SEIZE文、Ｃの場合は#include "measure_time.h"をソースファイルの先頭に出力するようにした。Ada,Cの出力例を下に示す。（例4）

```
with measure_time;
use measure_time;
procedure bubble is
type type_1 is array(0..999)of integer;
buffer:type_1;
temp:integer;
begin
start_test;
for i in 0 .. 10 loop
  for j in 0 .. 999 loop
    buffer(j) := 999-j;
  end loop;
  for k in 0 .. 998 loop
    for j in reverse k .. 998 loop
      if buffer(j)>buffer(j+1) then
        temp := buffer(j);
        buffer(j) := buffer(j+1);
        buffer(j+1) := temp;
      end if;
    end loop;
  end loop ;
end loop;
stop_test;
feature_time;
end bubble;
```

```
#include "measure_time.h"
int buffer[1000];
int temp;
int i,j,k;
main()
{
start_test();
for(i=0;i<=10;i++)
  {
  for(j=0;j<=999;j++)
    buffer[j]=999-j;
  for(k=0;k<=998;k++)
    for(j=998;j>=k;j--)
      if(buffer[j]>buffer[j+1])
      {
      temp=buffer[j];
      buffer[j]=buffer[j+1];
      buffer[j+1]=temp;
      }
  }
stop_test();
feature_time();
}
```

例4．　ＡｄａとＣの出力例

5．まとめ

1つの言語で記述されたＴＰ群をもとに評価対象言語に対応したＴＰを生成するジェネレータにより、処理系に依存しないTPの作成が可能になり、異なる言語間の性能評価も可能である。

今回開発した変換器はプロトタイプであり、構造体、ポインタ、その他いくつかの必要と思われる機能に関しては実現していない。そこで今後の課題としては、変換器に構造体、ポインタなどの機能を追加し、評価可能範囲を広げることがあげられる。

参考文献

［1］"あなたの言語処理系選びは正しいか"、NIKKEI BYTE,1988

［2］John R Wolberg、" ソフトウェア変換ハンドブック" (岸和孝訳)、啓学出版、1988