3Q-10 C言語拡張システム：OPTEC －構文ツリーの書換え－

小島泰三　杉本明　平井健治　阿部茂
三菱電機株式会社　中央研究所

1．はじめに

筆者らはC言語を問題向きに拡張するための言語変換システムOPTEC(Object-oriented Pattern Translatorfor Extending C) を試作中である[1,2]。

C言語を問題向きに拡張するための枠組みとして、オブジェクト指向方式によるオーバローディングが注目されている。オブジェクト指向言語C++では、引数の型により実際に呼ばれる関数を選択するオーバロード関数や、オペランドの型により演算子の意味を決定するオペレータオーバローディングなどを導入している。これにより複素数やストリーム入出力などに対する問題向き表現をC言語に追加することに成功している。しかしながらC++では、関数呼び出しや既存の演算子に対してのみオーバロディングが可能であり、制御構文などについては固定されたものしか利用できず、C言語を問題向きに拡張するには限界があった。

これに対して、OPTECでは、型による書換えルールの決定に基づく言語パタン変換を用いてオーバローディングを一般化し、C言語をより柔軟に問題向きに拡張することを可能にしている。以下では、まずOPTECの概要を説明し、パタン変換における問題とOPTECで採用した方法について述べる。

2．システム概要

OPTECシステムの特長を以下に示す。

1. 新しい文や演算子を導入するシンタクス定義命令。
2. 演算子や式だけではなく、 if や while レベルの文構文に対するオーバローディングを指定できる。
3. 宣言や実行文の追加といった言語変換命令を用意。
4. 言語書換えルールの条件を柔軟に指定できる。

OPTECシステムは、パーザ部、ツリー書換え部、C言語ソース生成部の３つの部分から成り立っている。図１にOPTECへの入力例とその変換結果を示す。パーザ部では、シンタクス定義、書換えルール、及び問題向き記述を解析し、ルールと構文ツリーに変換された問題向き記述をツリー書換え部に渡す。ツリー書換え部では、このルールを用いて構文ツリーの書換えを行い、書換え後のツリーをC言語ソース生成部で出力する。

3．ツリーの書換え

ツリー書換え部では入力された構文ツリーからパタンマッチングにより書換えルールの探索を行い、書換え処理をする。

ルールはマッチング対象のパタン（ヘッドパタン）、書換え後のパタン（ボディパターン）、及び書換えを制御するコマンド群から成り立っている。入力された構文ツリーのすべてのサブツリーと大規模な数のルールヘッドパタンとを高速にパタンマッチングする必要がある。このため、Hoffmannの手法[3]に基づき、ヘッドパタンからマッチングオートマトンを構成し、１パスでサブツリーとマッチングするすべてのルールの組を検出する方法を採用している。

OPTECのパタンマッチングでは、ツリー上のノードの種類、及びノードに対して与えられるデータ型によってマッチングを行う。図２に図１-(4)の構文ツリー、及びマッチングに用いるパタン（マッチングパタン）を示す。

以下では、パタンマッチングによるC言語の書換え上の問題について述べ、OPTECで採用した方法を説明する。

3.1 型推定の問題

複数のヘッドパタンが同じツリーにマッチングする場合には、最長パタンを持つものを選択するのが適切である。また、上位レベルの文のパタンにより下位レベルの文の意味が決定される場合もある。従って、マッチングはトップダウンで行う必要がある。

ところが、マッチングにデータ型を使用しているため、ボトムアップな処理も必要となる。例えば、図１の例のように、サブツリーの書換えにより、ある中間ノードに新たな型が付加され、このためにマッチングするルールが新たに検出される場合がある。またC言語には暗黙の型推定が備わっており、int + float --> float などの型推定も

```
typedef struct _list {                                  -- (1)
 struct _list *next;
 int amount;
} *List;
$statement { forall $id in ( $expr ) $stmt }    -- (2)
$operator (prefix,10) new ;
$rule [ forall x in (&char'string) $stmt'body]  -- (3)
  ==> $workVariable(int'i,&char'p)
      [ for (i=0,p=string;i<strlen(p);i++)
          { char x = *(char *)(p+i); body } ]
$rule [ forall x in (List'l) $stmt'body]
  ==> [ for (x=list;x!=NULL;x=x->next) body ]
$rule [ new &char'str ]
  ==> $workVariable(&char'src,&char'dst)
  ==> [ (src=str,
         dst=malloc(strlen(src)+1),
         strcpy(dst,src),dst)]
forall ch in (new "string") putchar(ch);        -- (4)
       --- 以下変換結果 ---
static char *_1_wv_;                                    -- (5)
static char *_2_wv_;
static int   _3_wv_;
static char *_4_wv_;
for (_3_wv_=0,
     _4_wv_= (_1_wv_="string",
              _2_wv_=malloc(strlen(_1_wv_)+1),
              strcpy(_2_wv_,_1_wv_),
              _2_wv_) ;
     _3_wv_ < strlen(p); i++)
  { char ch = *(char *)(_4_wv_+_3_wv_); putchar(ch); }
```

図1 OPTECへの入力と変換結果

Object-Oriented Pattern Translator for Extending C
Taizo KOJIMA, Akira SUGIMOTO, Kenji HIRAI, Shigeru ABE
Mitsubishi Electric Corp.

ボトムアップに行う必要がある。このような型推定はサブツリーの書換え時にも必要である。

また、foo(式)のように、関数呼び出しのパタンを持った入力がある場合に、foo()が式に示された引数を持つ関数として宣言されていれば、C言語の型推定に従う必要がある。もし関数foo()が宣言されていない場合、C言語の規定に従えばfoo()はintを返す外部関数となり、foo(式)にはint型が与えられる。しかしながら、foo(式)にマッチングするルールがあれば、この場合には書換えを優先する方が良いと考えられる。従って、型推定を行うノードの順番により異なるルールが選択される場合が生じるので、ノード選択の戦略が必要である。

以上のように、型をベースとしたパタンマッチングにより書換え処理を行うには、ボトムアップの型推定との組合せ方が重要である。

3.2 書換え処理

上記問題に対処するため、OPTECで採用したツリーの書換え処理を以下に示す。

1. 複数のレベルの文にまたがるパタンはそれぞれの文ごとに分けて最上位の文のみを書換え対象とする。
2. 書換え対象の選択はトップダウンに行う。すなわち、上位の文に対する書換えルールは記号表を変更することにより下位の文の処理に影響を与えることができる。
3. タイプの確定したノードはそれ以上書換え対象とはならない。従ってあるノードの型を設定する場合、そのノードより下位のノードの型はすべて確定していなければならない。なお下位ノードを書換え対象としない場合　はこの限りではない。
4. C言語の基本型間の算術演算、論理演算などのC言語における一般的表現はルールに優先してボトムアップに型推定（強い型推定と呼ぶ）を行う。
5. ',' による関数引数並び、定義されていない関数名による呼び出しなどの型推定（弱い型推定と呼ぶ）は、適用可能ルールが存在しない場合に実行する。
   このとき型推定を行うノードは、型の付加により新たなルールが検出されるノードの中から選び出す。
   もし弱い型推定により強い型推定が可能であれば、強い型推定を行い、その後パタンマッチングを試みる。
6. 複数のルールがマッチングする場合には、マッチングしたツリーの包含関係、演算子の属性（左優先、右優先など）、マッチング条件の厳しさ（マッチングパタンの長さなど）を元にルールを選択する。

図３に書換え手順を示す。なお、図３は概念的なものを示しており、実際の処理では図中(5)の、弱い型推定の必要性の判別、推定後のマッチングの再開などについて、過去のオートマトンへの入力時の各状態の記録を用いて処理を行っており、この処理が書換え速度に大きく左右していることが実験的に得られている。

４．おわりに

本稿では、データの型を用いたパタンマッチングによるC言語の問題向き拡張用言語変換システムOPTECについて、その書換え処理の概要を述べた。現在OPTECシステムはCommon Lispによるプロトタイプを用いて、書換え処理のアルゴリズムの改良や高速化を中心に評価を行なっている。試作システムでは、例えばルール数15、構文および式の定義数7で、書換え対象が30ステップの入力を処理するのに、全体で36秒、書換え処理では９個のルールが起動され8秒であった。このうち、通常のプログラム開発では変更されることの少ない、型、ルール及び文定義などの処理時間は全体の60%であった。このため、あらかじめこれらのものを処理し保存する機構を導入することにより、速度の向上を計った。しかし、実用速度を得るためには、さらに効率的な書換え手法の開発や、C言語による実装などが課題として残されている。

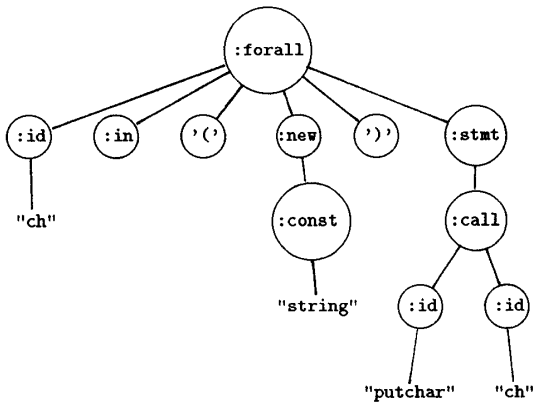
また、今後はルールの規模をさらに大きくし、より複雑なルールを用いた場合についての検討も行なう予定である。

参考文献

[1] 杉本 他,``オブジェクト指向方式による
C言語拡張システム：OPTEC(1)'',情処38全,(1984).

[2] 小島 他,``オブジェクト指向方式による
C言語拡張システム：OPTEC(2)'',情処38全,(1984).

[3] Hoffmann 他,``Pattern Matching in Trees'',
JACM, Vol. 29, No. 1.




```
:forall :id :in ( :new char* ) :call :id :id
```

図2 構文ツリーとそのマッチングパタン

```
process {
   if（ノードはブロック文）
       ブロックに含まれるすべての文に対して process を再帰的に適用;
   else if（ノードは文）
       ノードに rewrite を適用;
}
rewrite {
 loop:
   ツリーに対して強い型推定を再帰的に適用;                    -- (1)
   ツリーのマッチングパタンを取り出す;                        -- (2)
   オートマトンにマッチングパタンを入力しルール検出する;      -- (3)
   if（ルールは存在）{
     ルールの中から最も優先度の高いものを選択して、           -- (4)
     ツリーを書き換える;
     if（書き換えが成功）
       goto loop;
   }
   型の付加により新たな状態遷移が生じるノードを             -- (5)
   マッチングパタンから逆順に選択し、弱い型推定を行なう。
   このとき、{
     if（弱い型推定で型を付加されたノードが存在する）
         goto loop;
   }
   現在のノードのサブノードに対して process を再帰的に適用 -- (6)
}
```

図3 書き換えアルゴリズム