3P-2

オブジェクト指向方式によるC言語拡張システム：OPTEC（1） 概要

杉本明　平井健治　小島泰三　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 Translator for Extending C）を試作中である。本システムでは、問題向き構文からC言語への変換をルールとして記述する。すなわち言語翻訳過程に対するプログラミング（メタプログラミング）により、使用する言語自体を問題向きにカスタマイズする。本稿ではその特長を中心に概要を述べる。関連発表[1]でソフトウェア構成とC言語のXウィンドツールキット対応への拡張例を報告する。

2．背景と目的

現在、筆者らはソフトウェア開発の対象領域及び開発従事者に対してカスタマイズ可能なソフトウェア開発環境 ALTAIR システムの構築を行っている。カスタマイズ可能とすべきものとしては対話方式やツールなどがあるが、最も重要なものは使用するプログラミング言語自体であろう。

ALTAIR システムは、強く型付けされたコンパイラ言語によるソフトウェア開発の支援を目的としている。そして移植性の点から基本言語としてC言語を選択した。ＬＩＳＰなどのようにプログラムの表現構造が単純で、プログラムをデータと同様に処理できる場合には、強力なマクロ機能をインタプリタにより備えることができる。従って言語の問題向き拡張は比較的容易である。しかしC言語自体のマクロ機能は簡単なパタン変換しか扱えず、カスタマイズの手段としては不十分である。

最近、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ーローディング（overloading）が注目されている。オブジェクト指向言語 C++ [2]では、演算子の意味をオペランドのタイプにより決定するオペレータオーバーローディングにより、複素数演算やストリーム入出力に適した表現方法をC言語に導入している。しかしながら C++ をもとに問題向きに拡張する場合、次のような問題点がある；(1) シンタックスが固定されているので、新たな演算子や構文を追加することができない、(2) またオーバーローディングが式(expression)のレベルに限られているため、制御構文を汎用（generic）にすることができない。

これに対して OPTEC はオーバーローディングの枠組みを拡張した、型による書換えルールの決定に基づく言語パタン変換システムである。

3．ＯＰＴＥＣの特長

OPTEC の入力は外部定義や実行文などの通常の言語パートと、翻訳過程に対する命令や変換ルールの設定を行うメタコマンドからなる。以下に C++ と比較した場合の主な特長を述べる。

3．1　シンタックス定義命令

次の命令は、新たなシンタックスを定義するメタコマンドの例である。

```
$operator (postfix,300) cm,inch;
$expression get <expr> from <expr>;
$statement repeat <id> in (<expr>) <statement>;
```

最初の例では、結合優先度 300 を持つ後置単項演算子として、 cm と inch を定義している。例えば 10 cmと記述した場合の実際の単位変換方法は後述するルール記述を用いて指定する。次の例は通常の演算子定義では扱えない式（expression）を定義している。この中の get とfrom はキーワードとなり、 <expr> は受理するシンタックスクラスの指定である。最後の例はC言語の if や for のような構文の定義である。

3．2　構文レベルのオーバーローディング

演算子や式だけではなく、同一の構文であっても、この中に含まれる表現のデータ型により処理方法（C言語による実現）を選択する。図1は前述の repeat 文に対する書換えルールの例である。$apply から始まるメタコマンドには3つの書換えルールを記述している。ルールの中で予約語 is の前がマッチすべきパタン（ヘッド）であり、後が書換え後のパタン（ボディ）である。ヘッドの引数の中で int'x は引数となる式の型が int であることを示す。

```
typedef struct _list {
    struct _list *next;
    String name;
    int  amount;
} *List;

$apply repeat_rule {

  repeat int'x in (int'num) $$stmt is
       for ( x = 0; x < num; x++ ) $$stmt;

  repeat int'x in (List'list) $$stmt is
     { List temp = list;
       while (!temp) {
            x = temp->amount;
            $$stmt;
       }
     }

  repeat List'x in (List'list) $$stmt is
    for(x = list; x != NULL; x = x->next)
            $$stmt;
}
```

図1　構文レベルの変換ルールの例

Object-Oriented Pattern Translator for Extending C (1)
Akira SUGIMOTO, Kenji HIRAI, Taizo Kojima, Shigeru Abe
Mitsubishi Electric Corp.

例えば、図1の後で、

```
repeat x in (y) do_something(x);
```

と記述した場合、x と y の型により書換えルールが選択される。

3．3　宣言や実行文の追加命令

OPTEC は言語パタンを書換えるだけではなく、ルールのヘッドにマッチしない箇所にも宣言や実行文の追加といった形で作用する命令を用意している。例えば、大きなサイズの構造体、例えば行列間の演算を＋にオーバローディングしたとして、

```
Matrix a,b,c,d;
a = b + (c + d);
```

のような記述を行ったとする。実際に c + d を処理する関数を foo とすると、foo の出力を扱う形式は次の3つが考えられる。

1) Matrix foo(c,d) or foo(&c,&d)
2) Matrix *foo(&c,&d)
3) Matrix *foo(&c,&d,&out)

1) は結果の構造体をコピーすることになるので効率が悪い。2) はポインタを戻り値とするので結果を渡す点では効率は良いが、出力する構造体のメモリ領域をアロケートすることになり、ガルベージコレクションの必要が生じる。最後は引数として出力に使用する作業変数へのポインタを渡す方式である。しかしこのためには変数 out を、問題とする表現形式（c + d）の処理後も有効であるように宣言する必要がある。OPTEC では次のようにルールの中でこれを指定することができる。

```
Matrix'x + Matrix'y is
    $workVariable(Matrix out)
    &Matrix'foo(&x,&y,&out);
```

$workVariable 命令はマッチした式を含む文内で有効かつ他とは独立な作業変数を宣言する。また &Matrixは Matrix へのポインタを示す型名である。実体とその型へのポインタはマッチングでは同様に扱われ、OPTEC の内部処理により所定の型変換が行われる。このルールを適用すると、a = b + (c + d) は

```
{ Matrix _out1, _out2;
  a = *(foo(&b,foo(&c,&d,&_out1),&_out2))); }
```

と変換される。

3．4　ルールの特殊化（スペシャライゼーション）

ある記述に対してマッチする書換えルールのヘッドが複数ある場合、より適用条件を限定したパタンヘッドを持つものが選択される。これによりオブジェクト指向におけるクラス間のメソッド継承が実現できるが、より広い範囲にも応用できる。例えば3.3節の例で作業変数 _out2 は実は余計である。この場合、b + (c + d) の結果を直接 a に入れれば良い。このような最適化には次のような書換えルールを追加すればよい。

```
Matrix'z = Matrix'x + Matrix'y is
    &Matrix'foo(&x,&y,&z);
```

このルールがマッチする場合、3.3節のルールより優先される。従って a = b + (c + d) は

```
{ Matrix _out1;
  foo(&b,foo(&c,&d,&_out1),&a)); }
```

のように最適化することができる。

4．おわりに　――　問題向き言語間の継承

本稿ではオブジェクト指向言語で行われているオブジェクトの型による実現選択の枠組みを一般化した、C言語拡張システム OPTEC について目的と特長を述べた。OPTEC は別な意味でもオブジェクト指向言語の考え方を形を変えて導入している。すなわち、クラス間継承と似た、C言語を起点とした問題向き言語間の階層と継承関係である。ある言語からより特殊化した問題向き言語を構築する場合、親となる言語のシンタックスやセマンティクスを継承する。その上で新たなシンタックスの定義や書換えルールの追加、変更によって、より小さい問題領域に有効な特殊化を行う。現在検討を行っている問題向き言語の階層を図2に示す。この中で Alhard は YACC と LEX を使用した専用トランスレータの試作を一度おこなっており[3]、OPTEC の仕様はこの時の問題点の検討をもとにしている。

参考文献

(1) 小島、平井、杉本、阿部：オブジェクト指向方式によるC言語拡張システム：ＯＰＴＥＣ　（２）、本全国大会予稿
(2) B.Stroustrup: The C++ Programming Language, Addison-Wesley, 1986
(3) 杉本他：ハードウェア動作記述言語：ALHARD、情処３７回全国大会予稿、昭和６３年後期

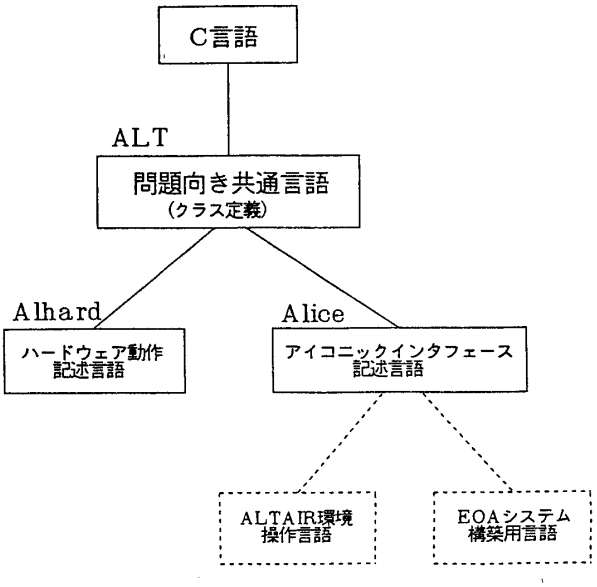

図2　ＡＬＴＡＩＲ問題向き言語間の継承階層