3P-3

オブジェクト指向方式によるC言語拡張システム：OPTEC（2） 実現手法とXツールキット対応拡張例

小島泰三　平井健治　杉本明　阿部茂

三菱電機株式会社中央研究所

1．はじめに

C言語の問題向き拡張を容易化するため、言語パタンの書換えに基づく言語変換システム OPTEC （ Object-oriented Pattern Translator for Extending C) を試作中である。OPTEC の書換えはオブジェクト指向におけるオーバローディングの枠組みを一般化したものであって、パタン内に含まれるデータの型（あるいはクラス）に従って書換えルールが選択される。本システムの概要は関連発表で報告する。本稿では実現手法を述べ、Xウィンドツールキット対応へのC言語拡張例を検討する。

2．　OPTECによるC言語への変換処理

図1に OPTEC 内部の処理の流れを示す。以下、順に各部について説明する。

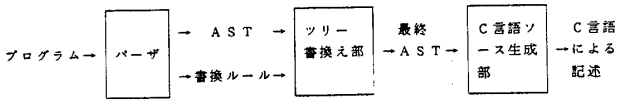

図1　OPTECの処理の流れ

2．1　パーザ部

パーザ部では再帰下降型の構文解析を行い、AST(abstract syntax tree)を構成する。変数や関数の宣言などはC言語のシンタックスを含んだものとしてあらかじめ定義している。これをもとにパーザ部では変数の型を決定する。実行文については OPTEC ではユーザが構文や演算子などを自由に定義できるので、これに従い構文解析を行う。

シンタックス定義命令や書換えルールの設定などの OPTEC のコマンドは、通常の型宣言や関数宣言などと混在する形でパーザ部に入力される。従ってシンタックス定義命令を入力した時点で、その後の構文解析を動的に変更しなければならない。このため構文解析はテーブルを参照する形で行っている。

2．2　ツリー書換え部

OPTEC の書換えルールは、

```
ヘッドパタン　is
    [適用条件判定コマンド]
    [アクションコマンド]
    書換えパタン
```

の形式をしている。パーザ部においてヘッドパタンも構文解析されこれ自体が一つのツリーとなる。そしてこれと書換え対象となるASTのサブツリーとのパタンマッチを行う。マッチングが成功すればそのルールを適用候補とする。

ヘッドパタンを構成するツリーのノードは、マッチすべき識別子（キーワードかデータ型名）か、任意のサブツリーとマッチングするパラメータである。これらがマッチする以外に他の適用条件が必要な場合には、OPTEC に用意した条件判定コマンドを用いる。アクションコマンドは、マッチしたASTのサブツリーを越える作用をASTに及ぼす場合に使用する。そして最後に書換え後のパタンを記述する。

対象となるASTを大規模な数のルールと高速にパタンマッチするため、ルールヘッドパタンからマッチングオートマトンを構成している。本手法は Hoffmann の手法[2]に基づいている。このオートマトンにASTの各要素を preorder 順に入力し、状態遷移を行う過程で適用候補ルールを検出する。

複数の適用候補ルールがある場合には、よりマッチング条件の厳しいものを選択し、アクションコマンド及び書換えを実行する。書換えの過程は、ASTの中間ノードに対する型推定とみなすこともできる。このため、C言語本来の型推定（int + int -> int など）も同時に行う。

2．3　C言語ソース生成部

最後に変換後のASTからC言語の記述を生成する。再帰的にC言語を生成していく時に、C言語では式（expression）とみなされる中間ノードの型が確定していない場合、警告メッセージを出力する。

3．Xウィンドツールキット対応のC言語拡張例

XウィンドV11R2上のXツールキット[3]の利用を容易化するため OPTEC によりC言語の拡張を試みた。Xツールキットはオブジェクト指向のクラス間継承や属性アクセスに対するフック関数など高度な機能を備えている。これにより作成されるダイアローグ用部品（widget）により、直接Xウィンドライブラリを使用するよりはアプリケーションプログラムの作成が容易になっている。しかしながら高度な機能を低レベルのC言語インタフェースによって提供しているため、プログラムにおいて煩雑な記述を行わなければならない。また少し内部的な機能を利用しようとすると、ツールキット内部の関数やその処理方法を理解することが必要である。

図2に OPTEC による拡張言語の記述例を示す。Xツールキット使用のヘッダファイルや OPTEC のコマンドは先頭の行の#include 文により入力する。図3は対応するC言語による記述である。このように拡張した言語では簡潔な記述

Object-Oriented Pattern Translator for Extending C (2)
Taizo Kojima, Kenji HIRAI, Akira SUGIMOTO, Shigeru Abe
Mitsubishi Electric Corp.

が可能である。しかもライブラリ関数は代表的なものを覚えるだけで良い。図２で用いたライブラリ関数を隠した初心者用 main ルーチンを提供することも可能である。

図４に使用した書換えルールの例を示す。 widget 部品のドキュメントに使用される属性名は、インクルードファイルにより String 型（char の配列）のコンスタントとして定義されている。図４ a）のルールはこれを利用したもので、Ｃ言語では誤り（代入の左辺が定数）となる式を解釈する。図４ b）のルールは必要なスタティック関数を生成する例として示した。図２の on Btn3Up ... の場合は Btn3Up をキーワードとしているので、図４ b）とは別のルールを適用する。２つのルールは、Widget 型の変数 me が宣言されていることを適用条件としている。

４．おわりに

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、処理方式の概要と応用例を述べた。 OPTEC 自体は現在 Common LISP を用いてプロトタイプの試作を行っている。ルール適用候補からの詳細な選択基準や、型の継承関係をマッチングにおいて高速に処理する手法の開発などの問題が残されている。

Ｘツールキット利用によるアプリケーション構築環境としては、マウス操作による widget レイアウトエディタやプロパティシートによる属性の設定など対話型手段も合わせ持つ必要があり、このための試作も行っている。しかしユーザ操作に対する動作の定義は専用言語による方が適切と考えている。 widget の動作の定義を実行時に対話的に追加、修正できるよう、OPTEC の周辺システムとしてインタプリタの開発を検討している。

参考文献

(1) 杉本、平井、小島、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 （１）、本全国大会予稿

(2) C.Hoffmann 他：Pattern Matching in Trees, JACM, Vol.29, No. 1, pp.68-95

(3) J.McComack 他：X Toolkit Intrinsics, X WINDOW V11R2 DESTRIBUTION TAPE


```
#include <XtRule.alt>
#include <X11/Box.h>
#include <X11/Command.h>

main (argc,argv)
unsigned int argc; char **argv;
{
    Widget topshell;
    topshell = XtInitialize(NULL,"Exam",NULL,0,&argc,argv);
    with (topshell) {
        with (new boxWidgetClass) {
            with (new commandWidgetClass named "quit") {
                XtNbackground = "dark green";
                XtNforeground = "yellow";
                on XtNcallback {
                    exit(0);
                }
                on Btn3Up {
                    with(the "test" of Parent) {
                        XtNbackground = "red";
                    }
                }
            }
            new commandWidgetClass named "test";
        }
    }
    XtRealizeWidget(topshell);
    XtMainLoop();
}
```

図２　Ｘツールキット対応拡張による記述例

```
String'to = String'from is                          --  rule (a)
    $syntax(<constant>,to)
    $requireVariable( Widget me )
    $declareVariable( int _arg_count = 0)
    $declareVariable( Arg _arglist[50])
    $postStatement(XtSetValues(me,_arglist,_arg_count))
    {
        _arglist[_arg_count].name = to;
        _arglist[_arg_count++].value =
                XtmConvertArg(me,to,XtRString,from);
    }

on String'name $$stmt is                            --  rule (b)
    $requireVariable( Widget me )
    $gensym(XtCallbackProc'proc,"SP")
    $declareExternal(
        static void proc(me,_client_data,_call_data)
        Widget me;
        caddr_t _client_data;
        caddr_t _call_data;
        {    $$stmt;   }
    )
    void'XtAddCallback(me,name,proc,NULL);
```

図４　使用した書換えルールの例

```
static void _SP1(me, _client_data,_call_data)
    .....  omitted
static void _SP2(me,_event,_params,_num_params)
    .....  omitted
main (argc,argv)
    .....  omitted
    {
        Widget _it = topshell;
        Widget me = _it;
        {
            Widget _it = XtCreateManagedWidget(0,boxWidgetClass,me,0,0);
            Widget me = _it;
            {
                Widget _it = XtCreateManagedWidget("quit",commandWidgetClass,me,0,0);
                Widget me = _it;
                int _arg_count = 0;
                Arg  _arglist[50];
                _arglist[_arg_count].name = "background";
                _arglist[_arg_count++].value = XtmConvertArg(me,"background","String","dark green");
                _arglist[_arg_count].name = "foreground";
                _arglist[_arg_count++].value = XtmConvertArg(me,"background","String","yellow");
                XtAddCallback(me,"callback",_SP1,0);
                XtmAddTranslation(me,"Btn3Up",_SP2);
                XtSetValues(me,_arglist,_arg_count);
            }
            XtCreateManagedWidget("test",commandWidgetClass,me,0,0);
        }
    }
    .....  omitted
```

図３　変換後のＣ言語による記述（部分）