4G−6　ソフトウェア設計エキスパートシステムSOFTEXにおける パラメタライズ・プログラミング

佐藤 明良　山之内 徹　渡辺 正信

日本電気(株)

1 はじめに

筆者らは、オペレーティングシステムなどの基本ソフトウェアの設計を自動化する目的で、ソフトウェア設計エキスパートシステムSOFTEXの研究開発を行なっている[1]。本稿では、SOFTEXにおけるパラメタライズ・プログラミング(parameterized programming)について報告する。その特徴は、(1)部品のパラメタの一般化、(2)変換ルールによる部品のインスタンシエーション、(3)パラメタ値の型推論による自動決定、である。

2 パラメタライズ・プログラミング

パラメタライズ・プログラミング[2]とは、ソフトウェア部品の再利用を支援する技術である。この技術のアイディアは、ソフトウェア部品中の状況に応じて変化する部分をパラメタライズしておき、具体的なプログラムを作る時にそのパラメタに適当な値を代入することによって部品をカスタマイズして所望のプログラムを生成することである。これにより、ソフトウェア部品の再利用性を向上させ、あわせて品質や保守性を向上させることを目的とする。

パラメタライズ・プログラミングを実現する最も単純な例としては、C言語の #define マクロ がある。また他の例として、Adaにおける汎用単位(generic unit)、GNU C++ のcontainer class prototypes[3]や AT&T C++ にもStroustrupがparameterized typesを導入するアイディアを提案している[4]。

3 SOFTEXのパラメタライズ・プログラミング

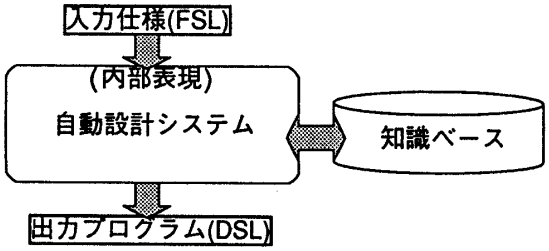

図1. SOFTEXシステム構成

SOFTEXは、知識ベースアプローチによるソフトウェア設計エキスパートシステムである。FSLと呼ばれる仕様記述言語で記述された仕様を入力とし、C言語とほぼ一対一に対応するDSL言語によるプログラムを自動合成する。合成過程は、まず、入力仕様が意味ネットワークによる内部表現に展開される。変換ルールがソフトウェア部品を利用して内部表現を段階的に詳細化する。詳細化された内部表現をDSLプログラムとして出力する(図1)。

この際、ソフトウェア部品知識ベース中に蓄えられたパラメタライズ構造が、変換ルールによって入力仕様に適合するようにカスタマイズされる。変換ルールによるカスタマイズはパラメタ間の型推論によって制御される。

SOFTEXパラメタライズ・プログラミングの特徴を以下にまとめ、それぞれについて説明する。

1. **一般化されたパラメタ** — パラメタライズ構造のパラメタにはFSL抽象構文の任意の構造を指定できることによって、抽象部品化を支援する。
2. **変換ルールによるインスタンシエーション** — パラメタライズ構造をインスタンシエートする際の展開知識をif〜thenルールで自然に知識化することができる。また、異なった展開規則の部品でも部品としてのインタフェースを共通化することを促進できる。
3. **パラメタ値の型推論による自動決定** — パラメタライズ部品間のパラメタの整合性を自動的にとることができる。

3.1 一般化されたパラメタ

SOFTEXのパラメタライズ構造は、ソフトウェア部品知識ベースに蓄えられたソフトウェア部品である。パラメタライズ構造の一般形は以下のようである。

type < *parameters* > *structure*

*type*は構造*structure*の型を指定する。*parameters*はパラメタである。パラメタはFSL抽象構文の任意の構造を指定できることによって一般化されている。*structure*はパラメタを用いて記述された構造、例えば関数定義やクラス定義である。

簡単な例として、2つの変数の値を交換するパラメタライズ構造(パラメタライズ関数)exchを以下に示す。

```
func <type t>
void exch(<t*> x, <t*> y) {
  <t> tmp;
  tmp = *x;  *x = *y;  *y = tmp;
}
```

ここで、funcはこの構造が関数型であることを表している。<type t>はパラメタ記述であり、パラメタtがタイプ型であることを表す(パラメタのタイプ指定はなくてもよい)。

3.2 変換ルールによるインスタンシエーション

上述した、パラメタライズ関数exchは、例えば、SOFTEXの入力仕様中に以下のような、#inst文指定があると

```
#inst exch<int> int_exch
main() {
  int x, y;
  ...
  int_exch(&x, &y);
  ... }
```

exchは具体的なプログラムにインスタンシエートされ、次のように展開される。

Parameterized programming in a sofware design expert system SOFTEX

Akiyoshi SATO, Toru YAMANOUCHI, Masanobu WATANABE

NEC Corporation

```
void exch<int>(int* x, int* y) {
  int tmp;
  tmp = *x;  *x = *y;  *y = tmp;
}
```

この展開は単純な文字列の置換であるが、SOFTEXでは、パラメタライズ構造にインスタンシエートのための変換ルールを記述することができ、この変換ルールによって、展開の知識を知識化することができる。以下にインスタンシエーション手順を示す。

[インスタンシエーション手順]
if パラメタライズ構造 p のインスタンシエート文がある
  if p 中に条件部が真となる変換ルールが存在する
    変換ルールを実行する
  else デフォルトの置換を実行する

例えば、exch のパラメタが char の時だけ特別に効率的な関数 c_exch が使用できるという知識があるとする。これを変換ルールとして以下のように記述することができる。

```
rule func::exch<type t> {
  if:   t == char
  then: unfolding{
          c_exch(char* x, char* y); }
}
```

パラメタのインスタンシエート指定が char である時に、この変換ルールが起動され、c_exch 関数が unfolding すなわち、展開される。例えば、入力仕様中に以下のような指定があると、

```
#inst char_exch exch<char>
```

関数 exch 中の変換ルール func::exch が起動され、

```
#inst char_exch exch<char>
#define exch<char> c_exch
```

unfolding 指示により、#define 文 として展開されるので、単純に展開するよりも最適化がはかられている。

3.3 パラメタ値の型推論による自動決定

パラメタライズ構造のインスタンシエートは、通常は、#inst 文によって仕様中で明示的にパラメタに値を設定する。しかし、SOFTEXでは明示的にパラメタを設定しなくても型変数を用いて暗示的に指定することができる。以下に、パラメタ値の推論手順を示す

[パラメタ値の推論手順]
パラメタライズ構造 p のパラメタの型変数 t を求める
if t に値が束縛されていない
  t の参照制約により型推論する

たとえば、

```
typev t;            // 型変数 t の宣言
#inst exch<t> t_exch
main() {
  ...
  t t1;             // t 型の変数 t1,t2 の宣言
  t1 = 1;           // -----(1)
  ... }
```

という仕様が入力された場合、(1) の代入文から t1 の型は int であると推論できるので パラメタライズ関数 exch のパラメタ t は int であると推論できる。

4 関連研究との比較

1. 一般化されたパラメタ
   GNU C++ の container class prototypes[3] や AT&T C++ の parameteried types[4] では、パラメタライズできるのは型だけであり、複雑な部品を記述することが難しい。
   Ada の generic unit では、型だけでなく変数やサブルーチンなどもパラメタライズできることは、SOFTEX と同様であり、より複雑で抽象化された部品を知識化することができる。
2. 変換ルールによるインスタンシエーション
   Ada の generic unit は、部品のインスタンシエートは単純な文字列置換であり、問題領域に依存した知識を知識化することが難しい。
   OBJ[2] は view という機構によって変換の対応を記述できる点は SOFTEX の変換ルールと対応し、問題領域に依存したインスタンシエーション規則を知識化できる。
3. パラメタ値の型推論による自動決定
   型推論を行なう言語システムは ML や Miranda などがあるが、ソフトウェア部品合成システムに応用した例は少ない。
   部品合成を自動化するためには、部品間のパラメタの関係を記述する必要がある。SOFTEX では、型変数によって部品間の関係を記述することができ、型推論によって部品間の整合性を保つことができる。

5 おわりに

ソフトウェア設計エキスパートシステム SOFTEX におけるパラメタライズ・プログラミングとその機構について述べた。その特徴は、(1) 一般化されたパラメタ、(2) 変換ルールによるインスタンシエーション、(3) パラメタ値の型推論による自動決定である。この 3 つの特徴を有することによって、リストやキューなどの container クラスだけではなくて、特定の問題領域に依存したソフトウェア部品を知識化でき、自動合成可能になると考えられる。

本機構を AI ツール CL[5] を用いてインプリメントした。現在、実際の基本ソフトウェアの設計に SOFTEX を適用することによって、本機構の有効性・効果について検討・評価を行なっている。

謝 辞

本研究の機会を与えていただき、有益な議論をしていただいた日本電気(株) 小池 誠彦部長、岡 浩部長、高村 淳課長に感謝いたします。

参考文献

[1] 山之内 徹, 他: ソフトウェア設計エキスパートシステム:SOFTEX-TSS コマンドへの適用 -, 1990 年度人工知能学会全国大会.
[2] Goguen,J.: Parameterized Programming, *CSLI-84-10,* Stanford Univ. (1984).
[3] Lea,D.: *User's Guide to GNU C++ Library,* Free Software Foundation,Inc., (1988).
[4] Stroustrup,B.: Parameterized Types for C++, *1988 USENIX C++ Conference,* pp.1-18 (1988).
[5] Watanabe,M., Yamanouchi,T., Iwamoto,M. and Ushioda,Y.: CL:A Flexible and Efficient Tool for Constructing Knowledge-Based Expert Systems, *IEEE Expert,* fall, pp.41-50 (1989).