代数仕様言語 CafeOBJ のパラメータ化機構*

2D－3

谷津弘一[†,1]　中川中[†,‡]　本間毅寛[‡]　澤田寿実[‡]　二木厚吉[§]

[†] 情報処理振興事業協会 (IPA) 技術センター

[‡] (株)SRA ソフトウエア工学研究所

[§] 北陸先端科学技術大学院大学

1. はじめに

ソフトウェアシステムの仕様を記述する上で、仕様の抽象化を支援するパラメータ化機構は、仕様の本質的な部分の理解を深める点、仕様の再利用を促進する点において、極めて重要な機構である。

情報処理振興事業協会 (IPA) において開発中の代数仕様言語 CafeOBJ[4, 5] は、代表的な代数仕様言語である OBJ[1] に José Meseguer が提案した書換え論理 [2] に基づく一方向の書き換えを表す書換え規則を導入し、拡張した言語である。CafeOBJ のパラメータ化機構は、OBJ のパラメータ化機構にある種の高階性を加えたものである。この高階性により、CafeOBJ では Standard ML[3] の functor モジュールにおける sharing constraint と同等の記述が可能となっている。

本稿では、CafeOBJ のパラメータ化機構、すなわち、パラメータ付きモジュールとその具象化について、高階性に重点を置いて説明する。本稿で説明されない CafeOBJ の構文については、[5, 4] をご参照いただきたい。

2. CafeOBJ のパラメータ付きモジュール

パラメータ付きモジュールは、モジュールから非本質的な部分を捨象することによって得られる、汎用的なモジュールである。

```
module TRIV is sort Elt . endm

module LIST[X :: TRIV] is
  sort List .
  subsort Elt < List .
  op nil : -> List .
  op _,_ : List List -> List .
  attr _,_ : [ assoc id: nil ] .
endm
```

TRIV はソート Elt をただ 1 つ持つだけのモジュールであり、LIST はリストを表すパラメータ付きモジュールである。LIST に現われる X :: TRIV は、パラメータとその制約を表す。X がパラメータであり、TRIV は X に代入される実モジュールに対する制約である。CafeOBJ では、パラメータに代入される実モジュールが表す代数のクラスは、制約が表す代数のクラスのあるサブクラスと同型でなければならない。パラメータ X に実モジュールを代入することにより、様々な要素のリストを表すモジュールを作ることができる。LIST では、リストの要素についての情報はモジュール本体から捨象され、パラメータの制約に移されている。

CafeOBJ のパラメータ化機構は高階性を有する。すなわち、CafeOBJ のパラメータ付きモジュールでは、パラメータの制約となるモジュールがパラメータを持つことを許される。例として、リストの各要素に与えられた関数を作用させる、map 関数を表すモジュールを紹介しよう。

```
module FUN[X Y :: TRIV] is
  op f_ : Elt.X -> Elt.Y .
endm

module MAP[X Y :: TRIV,
           L1  :: LIST[X],
           L2  :: LIST[Y],
           F   :: FUN[X,Y]] is
  op map_ : List.L1 -> List.L2 .
  var E : Elt.X .
  var L : List.L1 .
  eq map nil = nil .
  eq map (E,L) = (f E),(map L) .
endm
```

異なるパラメータが同じ名前のソートや演算を持つモジュールで制約されている場合、そのソートや演算はパラメータ名の修飾により区別される。例えば FUN では、Elt.X や Elt.Y のようにソート名の後ろにパラメータ名を修飾させて、X を通して導入されるソート Elt と Y を通して導入されるソート Elt が区別される。

MAP のパラメータ L1、L2、および F を制約するモジュールはそれぞれパラメータを持っている。そして、L1 と F の制約の間ではパラメータ X が共有され、L2 と F の制約の間ではパラメータ Y が共有されている。

3. パラメータ付きモジュールの具象化

CafeOBJ では、ビューと呼ばれるものを介して実モジュールがパラメータに代入される。ビューは、パラメータの制約と実モジュールとの間の、ソートと演算の対応を与える。

```
module NAT is
  sort Nat .
  op 0 : -> Nat .
  op s_ : Nat -> Nat .
endm
```

---

*The Parameterization Mechanism of An Algebraic Specification Language CafeOBJ: Hirokazu Yatsu, Ataru T. Nakagawa, Takehiro Honma, Toshimi Sawada, Kokichi Futatsugi

†Information-Technology Promotion Agency, Japan(IPA) Software Technology Center

‡Software Research Associates,Inc. Software Engineering Labolatory

§Japan Advanced Institute of Science and Technology

[1] 日本ユニシス (株) より出向中

```
view TRIV2NAT from TRIV to NAT is
  sort Elt to Nat .
endv
```

TRIV2NATは、LISTのパラメータの制約TRIVからNATへのビューである。このビューにより、TRIVのソートEltとNATのソートNatが対応付けられている。

パラメータ付きモジュールを具象化して実モジュールを作るには、次のようにビューを与えればよい。

```
make NATLIST is LIST[TRIV2NAT] endm
```

CafeOBJには、暗黙のビューと呼ばれるものがある。パラメータの制約がTRIVのようにソート1つしか持たない場合には、実モジュールをそのままビューとして与えることができる。

```
make NATLIST is LIST[NAT] endm
```

TRIVのソートEltからNATの最初に宣言されたソート(すなわち、Nat)への対応を表すビューが自動的に作られる。

代入する実モジュールが制約で定義されているソートと演算を含む場合にも、同じように実モジュールをそのままビューとして与えることができる。この場合には、同じ名前のソートと演算を対応付けるビューが自動的に作られる。

次のように、ビューの中身を直接代入することもできる。

```
make NATLIST is
  LIST[view to NAT is sort Elt to Nat . endv]
endm
```

この場合、ビューや制約の名前を記述する必要はない。

高階のパラメータ付きモジュールの具象化は、高階でないパラメータ付きモジュールの具象化とさほど変わらない。

```
module DOUBLE is
  using NAT .
  op double_ : Nat -> Nat .
  var N : Nat .
  eq double 0 = 0 .
  eq double (s N) = s s (double N) .
endm
```

DOUBLEは、与えられた自然数を2倍する関数を表すモジュールである。このモジュールを使ってMAPを具象化すると次のようになる。

```
make MAP-DOUBLE is
  MAP[NAT,NAT,LIST[NAT],LIST[NAT],
      view to DOUBLE is op f_ to double_ . endv]
endm
```

この具象化により、リストの各要素を2倍する、自然数のリスト上のmap関数を表すモジュールが得られる。

ここで注意して欲しいのは、最初の2つのビューにより、パラメータXおよびパラメータYを制約する2つのTRIVとNATの間で、ソートEltとソートNatが対応付けられていることである。パラメータXはLIST[X]とFUN[X,Y]の間で、パラメータYはLIST[Y]とFUN[X,Y]の間でそれぞれ共有されているので、残りの3つのビューでEltとNatを対応付ける必要はない。このため、LIST[NAT]へのビューでは暗黙のビューが使え、最後のDOUBLEへのビューでは演算fとdoubleを対応付けるだけで十分になっている。MAP-DOUBLEは、意味的には次のモジュールと等しい。

```
module MAP-DOUBLE is
  using NAT .
  using LIST[NAT] .
  using DOUBLE .
  op map_ : List -> List .
  var N : Nat .
  var L : List .
  eq map nil = nil .
  eq map (N,L) = (double N),(map L) .
endm
```

ここで、直接輸入されている、あるいはLIST[NAT]やDOUBLEを通じて輸入されているNATは同一視される。

CafeOBJではパラメータ付きモジュールのカリー化も許される。

```
make MAP' is MAP[NAT] endm
```

と具象化すると、次のようなパラメータ付きモジュールが得られる。

```
module MAP'[Y  :: TRIV,
            L1 :: LIST[NAT],
            L2 :: LIST[Y],
            F  :: FUN[NAT,Y]] is
  using NAT .
  op map_ : List.L1 -> List.L2 .
  var E : Nat .
  var L : List.L1 .
  eq map nil = nil .
  eq map (E,L) = (f E),(map L) .
endm
```

これは、MAPにおけるXの出現をNATに置き換えたものになっている。

4. おわりに

MAPの例でいうと、高階性がなければ、その記述は煩雑になり、汎用性は著しく減少する[5]。仕様記述言語のパラメータ化機構にとって、本稿で示したような高階性は必須である。

謝辞

本稿で報告した試みはIPA技術センターにおける実行可能な形式仕様言語システムの研究開発プロジェクトで行われた。

参考文献

[1] J.A.Goguen, T.Winkler, J.Meseguer, K.Futatsugi and J.-P.Jouannaud, Introducing OBJ, Technical Report SRI-CSL-92-03, Computer Science Laboratory, SRI International, 1992.

[2] J.Meseguer, Conditional Rewriting Logic as a Unified Model of Concurrency, Theoretical Computer Science, Vol.96, 1992.

[3] L.C.Paulson, ML for the Working Programmer, Cambridge University Press, 1991.

[4] 谷津, 本間, 中川, 二木, 実行可能な形式仕様言語CafeOBJの概要, 第12回IPA技術発表会論文集, 1993.

[5] 実行可能な形式仕様言語システムの研究開発に関する研究報告書, 7技-156, 情報処理振興事業協会, 1996.