項書換え系に基づく定理証明支援環境の構築*

5X－1

中川中§　澤田寿実§　石曽根誠§　二木厚吉°　*RazvanDiaconescu*°
§(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

等式論理を基盤とする仕様記述言語では、操作意味論を項書換え系 [5, 1] に求めて仕様を直接実行することが可能である。CafeOBJ[2, 7] は等式論理を拡張した論理系を基盤とする仕様記述言語であり、等式論理に限定すれば同様の実行が可能である。

拡張された論理系では、等式に加え、遷移関係を定義する公理 [6]、内部状態を持つ物体の振る舞いを定義する公理 [3] が使われる。それら公理を検査するには上記の実行機構はそのままでは使えない。我々はそうした公理を扱うためのブール演算を幾つか用意し、また実行機構を介して定理証明を支援する仕組みの構築に取り組んでいる。

2　項書換え系と実行

仕様を項書換え系の定義とみての実行は記号計算にほかならない。陳腐な例を使えば、自然数の加算の定義

```
mod! NAT { [ Nat ]
  op 0 : -> Nat
  op s : Nat -> Nat
  op _+_ : Nat Nat -> Nat
  eq N:Nat + 0 = N .
  eq N:Nat + s(M:Nat) = s(N + M) .
}
```

が与えられたとき、3＋2の計算は

```
s(s(s(0))) + s(s(0)) -->
s(s(s(s(0))) + s(0)) -->
s(s(s(s(s(0)))) + 0) -->
s(s(s(s(s(0)))))
```

という記号書換え操作によって実現される。CafeOBJ システムでは、この計算は

```
reduce in NAT : s(s(s(0))) + s(s(0)) .
```

というコマンドによって起動される。

3　等式の証明

上記の実行機構を等式論理の証明に使うため、_==_ というブール演算を与え、両辺の書換え結果が一致するとき、そしてそのときに限り真を返す、と定義する。この演算を利用することにより、実行機構を定理証明に援用することが可能である。陳腐な例を続ければ、3＋2＝5はコマンド

```
reduce in NAT : s(s(s(0))) + s(s(0))
                == s(s(s(s(s(0))))) .
```

の起動結果が真になることで確かめられる。

加算の結合則のように定理に変数が含まれる場合、また帰納法を使う必要がある場合には、上記のように単一の実行コマンドを使うだけでは一般に不十分であり、一時的に「任意の要素」を表す定数を宣言したり、帰納仮定を記述する方法が必要である。それらを使えば加算の結合則は次のように示される。

---

*Term Rewriting System Helps Proving Theorems: Ataru T. Nakagawa, Toshimi Sawada, Makoto Ishisone, Kokichi Futatsugi, Razvan Diaconescu

†SRA Software Engineering Laboratory

‡Japan Advanced Institute of Science and Teleology, Hokuriku

```
open NAT
ops n m p : -> Nat .
reduce n + (m + 0) == (n + m) + 0 .
eq n + (m + p) = (n + m) + p .
reduce n + (m + s(p)) == (n + m) + s(p) .
close
```

上で1番目のreduceの起動が帰納底、2番目のそれが帰納歩、その直前の等式が帰納仮定である。この宣言・コマンド系列は全体として帰納法を使った証明譜を構成する。

4 遷移関係の証明

非決定的な遷移関係は、たとえば以下のように定義出来る。

```
mod! CHOICE { extending (NAT)
  op _|_ : Nat Nat -> Nat
  trans N:Nat | M:Nat => N .
  trans N:Nat | M:Nat => M .
}
```

ここでtransは遷移関係を定義する公理である。遷移関係に関する証明を支援するため、CafeOBJ ではブール演算_==>_が用意され、反射律、合同律が公理として与えられる。上記演算_|_が加算に関して合同であることは次のように示される。

```
open CHOICE
ops n m p : -> Nat .
reduce n + (m | p) ==> n + m .
close
```

5 振る舞いの証明

観察同値に基づく演算の定義は以下の例のように与えられる。

```
mod* { protecting (NAT)
  *[ Counter ]*
  bop add : Nat Counter -> Counter
  bop read : Counter -> Nat
  eq read(add(N:Nat, C:Counter)) =
       N + read(C) .
}
```

ここでCounter、add、read は観察同値により定義されるソート、演算と宣言され、等式は観察同値の定義と解釈される。観察同値に関する証明を支援するため、CafeOBJ ではブール演算 _=*=_ が用意され、双帰納法 [8, 4] による証明が可能である。例として add を2回適用する効果と一度だけ和を適用する効果とが観察上同一の振る舞いをすることは以下の証明譜で示される。

```
open COUNTER
ops n m : -> Nat .
op c : -> Counter .
reduce add(n,add(m,c)) =*= add(n + m,c) .
close
```

6 今後の課題

遷移関係の推移律は項書換え系の枠内で扱うことが困難であり、その扱いは今後の課題である。また双帰納法は新しい研究分野であり、新たな成果をシステムに取り込む余地はまだ多分に残されている。

参考文献

[1] Dershowitz, N. and Jouannaud, J.-P., "Rewrite Systems", *Handbook of Theoretical Computer Science, Vol.B: Formal Models and Semantics*, The MIT Press/Elsevier Science Publishers, 1990, pp.245–320

[2] Futatsugi, K. and Diaconescu, R., *CafeOBJ Report*, a technical report in preparation, Japan Advanced Institute for Science and Teleology, 1997

[3] Goguen, J. and Malcom, G., *A Hidden Agenda*, technical paper in preparation

[4] Jacobs, B. and Rutten, J., "A Tutorial on (Co)Algebras and (Co)Induction", EATCS Bulletin No.62, 1997, pp.222–259

[5] Klop, J.W., "Term Rewriting Systems: A Tutorial", *Bulletin of the EATCS*, Vol.32, EATCS, 1987, pp.143–182

[6] Meseguer, J., "Conditional Rewriting Logic: Deduction, Models and Concurrency", *Proc. 2nd International CTRS Workshop*, Lecture Notes in Computer Science 516, 1991, pp.64–91

[7] Nakagawa, A.T., Sawada, T., and Futatsugi, K., *CafeOBJ Manual*, SRA, 1997

[8] Rutten, J.J.M.M., *Universal Coalgebra: a Theory of Systems*, Technical Report CS-R9652, CWI, Amsterdam, 1996