5X－3

等式論理系における制約解消による証明記述の枠組み*†

瀬尾 明志‡
日本ユニシス株式会社§

中川 中¶
(株) SRA ソフトウェア工学研究所‖

1 はじめに

代数仕様言語 CafeOBJ[1] で記述された形式仕様の様々な性質を Knuth-Bendix 完備化、構造帰納法、項書き換え系などの等式論理のための検証系（ツール）を使って証明するための記述の枠組みを提案する。

この枠組みは制約解消 [2] の考え方を証明記述に適用した『証明=編集』パラダイムに基づいている。すなわち証明された証明記述が持つべき文書間の関係を先に "制約" という形で証明前の文書中に関係付け、その "制約" で記された関係になるように文書を変化させること ("解消") によって証明を進めていくというアプローチである。その証明までの手順の概略は次のようになる。

1. 文書中に仕様を記述する。
2. 証明したい定理を文書中の部分間の制約として関係付ける。制約には検証系の情報も記述する。
3. 制約を解消する。すなわち指定された検証系により証明を行う。
4. 制約解消の結果が他の制約の部分になることも可能であり、その場合は制約の解消が繰り返し行われる。

このように証明作業は入れ子になった制約を繰り返し解消することによって進められる。これにより利用者は次のような柔軟性を得ることができる。

- ある定理は複数の制約によって関係付けられることができ、また複数の定理が一つの制約によって関係付けられることもできる。
- 証明の順序はボトムアップでもトップダウンでもそれらの併用でも構わない。
- 全ての証明を一度に行うのではなく、利用者が全体の戦略や個々の証明に関して判断を下しながら対話的に証明を行うことができる。

*A scheme for describing proofs by constraint solving in equational logic

†本研究は情報処理振興事業協会 (IPA) が実施している「創造的ソフトウェア育成事業」の一環として行われたものである。

‡Akishi Seo
§Nihon Unisys Ltd.
¶Ataru T. Nakagawa
‖SRA Software Engineering Laboratory

2 Forsdonnet

本方式では、仕様、その仕様の性質の証明、説明文や図表、および他の文書へのリンクなどを Forsdonnet[1] と呼ぶ文書に記述する。Forsdonnet は分散したマシン上に置かれることを前提にしており、文書間のリンクはネットワークをまたいで行うことができる。また Web からのアクセスも可能にするために、その書式は HTML の拡張となっている。Forsdonnet の HTML に対する拡張は以下の 4 種類のタグを加えたことである。

`<MODULE>`, `</MODULE>`
　CafeOBJ のモジュールやビューを囲むタグ

`<MODREF>`, `</MODREF>`
　モジュールの輸入を示すタグ

`<CONSTRAINT>`, `</CONSTRAINT>`
　制約を記述するタグ

`<TARGET>`, `</TARGET>`
　制約に必要となる付加情報を記述するタグ

例えば、以下の例ではモジュール FACT を定義し、その FACT の中で RAT を輸入していることも定義している。TARGET タグでは、制約解消の際に使われる CafeOBJ 処理系へのコマンドが付加情報として定義されている。CONSTRAINT タグでは、文脈を FACT とし、証明系を CafeOBJ とし、付加情報を CMD で示される部分とした制約が定義されている。

```
<MODULE name=FACT>
module FACT {
  pr (<MODREF context=#RAT>RAT</MODREF>)
  op _! : Nat -> NzNat { memo }
  op fact : Nat -> NzNat
  eq 0 ! = 1 .
  eq N:NzNat ! = N * (p N !) .
  eq fact(0) = 1 .
  eq fact(N:NzNat) = N * fact(p N) .
}
</MODULE>
<TARGET name=CMD>
red 10 ! .
</TARGET>
<CONSTRAINT context=#FACT solver=CafeOBJ
            target=#CMD>
</CONSTRAINT>
```

[1]Formal Specification Document on Network

3 制約解消

制約の解消を行うと、その結果は CONSTRAINT タグの間に挿入される。以下は先に制約 C1 を解消して最終的なゴールから構造帰納法による問題を生成し、その問題を制約 C2 によって解消する例である。

```
<MODULE name=TINY-NAT>
  [ Nat ]
  op 0 : -> Nat
  op s_ : Nat -> Nat {prec: 1}
  op _+_ : Nat Nat -> Nat
  vars L M N : Nat
  eq M + 0 = M .
  eq M + s N = s(M + N) .
}
</MODULE>

<MODULE name=ASSOC>
module ASSOC {
  <TARGET name=PR1>
  protecting (<MODREF context=#TINY-NAT>
                      TINY-NAT</MODREF>)
  </TARGET>
  vars L M N : Nat
  eq L + (M + N) = (L + M) + N .
}
</MODULE>

<CONSTRAINT name=C1 context=#ASSOC
            target=#PR1 solver=obligation>
</CONSTRAINT>
<CONSTRAINT name=C2 context=#ASSOC
            target=#C1 solver=CafeOBJ>
</CONSTRAINT>
```

制約 C1 を解消すると CONSTRAINT タグ C1 内に結果として問題が以下のように挿入される。

```
<CONSTRAINT name=C1 context=#ASSOC
            target=#PR1 solver=obligation>
open TINY-NAT
ops l m n : -> Nat .

** base case, n=0: l+(m+0)=(l+m)+0
reduce l + (m + 0) == (l + m) + 0 .

** induction step
eq l + (m + n) = (l + m) + n .
reduce l + (m + s n) == (l + m) + s n .
close
</CONSTRAINT>
```

次に生成された問題を制約 C2 で解消すると CONSTRAINT タグ C2 内は次のようになる。

```
<CONSTRAINT name=C2 context=#ASSOC
            target=#C1 solver=CafeOBJ>
-- reduce : l + (m + 0) == (l + m) + 0
true : Bool

-- reduce : l + (m + s n) == (l + m) + s n
true : Bool
</CONSTRAINT>
```

これにより全体が証明されたことになる。このように証明はインタラクティブに進められる。

4 実装

本方式に基づいた実装はほぼ完了しており、その実装においては Forsdonnet は Emacs と Netscape 上で操作することができる。Emacs 版は w3.el を元に作られているので、Netscape 版だけでなく Emacs 版も Web ブラウザとして利用することができる。

また本稿で説明した範囲を含んだ CafeOBJ 処理系を分散環境で統合的に扱うための環境の構築 [3] が進められており、いくつかの検証系も開発されている。

5 まとめ

仕様とその仕様の性質を証明するための枠組みを提案した。この枠組みでは記述と証明に関して規制を与えているが、それは自由度の高いものであり、逆に柔軟な記述や証明を可能にしている。記述に関しては形式的な仕様だけでなく説明文や絵や表や他の文書へのリンクも記述することができ、証明に関しては証明順序を気にすることなく証明木を徐々に構築していく感覚で対話的に証明を行うことができる。

また汎用的に設計されているため、検証系などとの独立性が高く、新たな検証系が現れても容易に取り込むことができる。対象言語に関しても CafeOBJ 以外の言語を対象にすることが可能である。

謝辞

本研究に対して貴重な助言を下さった東京大学の萩谷昌己教授に感謝いたします。

参考文献

[1] Kokichi Futatsugi and Răzvan Diaconescu. CafeOBJ report. Technical report, Japan Advanced Institute for Science and Teleology, 1997.

[2] M. Hagiya and H. Kakuno. Proving as editing. In *Proceedings of User Interfaces of Theorem Provers*, July 1996.

[3] Ataru T. Nakagawa. Manipulating CafeOBJ on networks. In *12th Workshop on Algebraic Development Techniques*, Tarquinia, June 1997.

[4] Ataru T. Nakagawa and Akishi Seo. Directing proofs by documenting and pointing: CafeOBJ on network. In *Proceedings of User Interfaces of Theorem Provers*, 1997.