

3B-4

構成的数学体系 RPT に基づく超数学の定理の形式化

亀山幸義

東北大学

1 はじめに

我々の目的は、洗練された論理体系に基づき、数学の定理やプログラムの性質の証明を計算機上で実行できる環境を構築し、ひとつのシステムの中で、定理証明やプログラムの合成、変換等を行うことである。ここで、基礎とする論理体系は構成的(直観主義的)なので、証明された定理に含まれるアルゴリズムを、プログラムの形で自動的に抽出することができる。従って、このシステムの中では、証明の作成作業はプログラミングと同一視できることになる。

構成的数学の体系は、近年、プログラムのための論理として活発に研究されており、計算機上の実現の例も、Martin-Löf の型理論に基づいた Constable らの Nuprl[1] や、

Feferman の T_0 に基づいた林の PX[2] などがある。また、我々も、上記の目的のために、体系 SST と関数型言語 Λ を用いたシステムを提案している [3]。

本研究の主な特徴は、

- 論理体系に含まれるプログラム言語を用いて、証明システムを実現したため、システムの性質をそれ自身の中で論じることができる。
- 扱う対象は、普通の数学の定理だけでなく、超数学(証明論)の定理を自然に形式化し、証明することを含んでいる。

という2点である。これらの特徴を示す例として、 Λ の項に対する Church-Rosser の定理の証明をあげる。また、その他の超数学の定理の形式化について簡単にのべる。

2 Frege 構造と RPT

我々が用いる体系は、佐藤の RPT(Reflective Proof Theory)である [4]。RPT は、P. Aczel の Frege 構造の基本思想を受けつぎ、それを拡張した体系である。[4] では RPT は形式的体系ではなく、Martin-Löf の型理論と同様に意味論だけが与えられている。

意味論の領域としてまず、項の全体が定義される。これは、本質的には型のない λ 項の世界と同じであるが、これを用いて証明システムを実現するため、真偽値、対、if や let(パターンマッチに用いる)の項が導入されている。また、 Λ などの論理記号に対応する項はこれらを用いて適当に表現されているとする。計算規則は省略する。RPT の項全体からなる関数型プログラム言語を Λ とよぶ。 Λ における計算は、lazy evaluation である。

RPT の基本的な言明(Judgement)は、以下の2種類である。(a, b は項、i は自然数である。)

$$\vdash_i b$$

は、項 b が第 i-level の論理式(Proposition)であることを表し、

$$a \vdash_i b$$

は、b が第 i-level の Proposition で、かつ、項 a がその証明であることを表す。

次に、それぞれの論理記号の意味が説明される。たとえば、 $p \vdash_i A \wedge B$ は、「A と B がともに正しい Proposition であり、それらの証明を対にしたものが p であること」と定まる。

添字の i は、Meta-ness を表す。我々はしばしば「A が論理式なら $A \supset A$ は tautology である」という言い方をするが、通常の論理体系では、この文自体は Meta 定理となってしまう。しかし、RPT では、「t が i-level の論理式であること」が $i+1$ -level の論理式として記述できる。従って、上の事実も、

$$\lambda x. \lambda y. 0 \vdash_{i+1} \forall x. (\vdash_i x) \supset (\lambda z. z \vdash_i x \supset x)$$

という1個の Judgement で表現できる。

この他、集合は、Propositional Function と同一視することにより導入される。自然数、リスト、S 式などのデータ構造を取り扱うため、帰納法が用意されている。従って、HA など算術の体系を RPT の中で解釈することができる。

3 RPT の形式的体系

先のべたように、[4] では、RPT の意味論のみが与えられているが、RPT の証明システムを計算機上で実現するため、形式的体系を与える必要がある。不完全性定理より、完全な形式化は不可能であるが、RPT の意味論より自然に定まる推論規則を自然演繹体系であらわし、形式化された RPT とした。そのうち特徴的な規則をあげる。

$$\frac{a \vdash_i b}{a' \vdash_i b'} (a = a', b = b')$$

$$\frac{\begin{array}{c} [x \vdash_i a] \\ \vdash_i a \vdash_i b \end{array}}{\vdash_i a \supset b}$$

Formalizing Metamathematical Theorems based on Constructive Logic RPT

Yukiyoshi Kameyama

Tohoku University

$$\frac{a \vdash_i \text{app}(b, c)}{[a.c] \vdash_i \exists b}$$

最初の規則は、簡約化により等しい項に置きかえても良いことを示す。2番目の規則より、 $a \supset b$ は、 a が偽の Proposition なら、 b が Proposition でなくても、Proposition になる。最後の規則は存在の導入規則であり、後にふれる。

4 計算機上への実現

最初に、 Λ の処理系を実現した。初期の版は、Kyoto Common Lisp 上に実現したが、速度の観点から現在の版は C 言語で実現している。

次に、 Λ によって RPT の証明システムを実現した。 Λ はこのために、パターンマッチの機能などを持つよう拡張されている。

このシステムを用いて、定理の証明を実際に試みた。

5 数学および超数学の形式化

我々の形式化の対象には、普通の数学だけでなく超数学も含まれる。RPT においても、Bishop 流の構成的数学などを展開することが原理的に可能である。しかし、ここでは、RPT や Λ 自身に関する超数学的定理の証明を試み、証明システム自体や RPT の形式化の妥当性のチェックに役立てることとした。

Church-Rosser の定理 a, b, c を Λ の項とすると、 $a \supset^* b$ かつ $a \supset^* c$ ならば $b \supset^* d$ かつ $c \supset^* d$ となる項 d が存在する。 $(\supset)$ は Λ の 1-step 簡約、 $\supset^*$ はその推移的閉包。

N. Shankar は、Boyer-Moore 証明系を用いて、 β 簡約のみの純粋な λ 計算に対する Church-Rosser 性を証明した [5]。Boyer-Moore システムは、ある程度自動証明をしてくれるので証明の手間が省けるが、定理を quantifier-free な論理式に変形する、等の部分は必ずしも簡単ではないし、わかりやすくもない。

我々は、手作業による佐藤の証明 [4] にない、 $\supset_p$ (並列簡約) と a^* (a 中の並列に簡約できる全ての基を簡約したもの) を定義して、

- $a \supset b$ ならば $a \supset_p b$
- $a \supset_p b$ ならば $a \supset^* b$
- $a \supset_p b$ かつ $c \supset_p d$ ならば $a_x[c] \supset_p b_x[d]$
- $a \supset_p b$ ならば $b \supset_p a^*$

- $a \supset^* b$ ならば $b \supset_p^*(\dots(a^*)\dots)$

という手順で証明した。もちろん、これらは informal な表現であり、実際の証明中では、たとえば、1行目の主張は、 $\forall a. \forall b. \text{Term}(a) \wedge \text{Term}(b) \supset \dots$ という Proposition p におきかえ、 $a \vdash_i p$ とな a を見つける必要がある。計算機による証明を試みる意義は、証明の誤りをチェックすることの他、このように隠れた仮定を明示的に示して推論を明確にする働きがある。

ここで用いた方法は、 a^* という簡約を用いることにより、Shankar の証明より簡単である。我々の結果の特徴は、 Λ 自身の Church-Rosser 性を証明したことである。また、これにより、(RPT の無矛盾性を仮定すれば) Λ が無意味な言語でないことが示された。

別の課題として、証明図の正規化定理があげられる。

正規化定理 RPT で $a \vdash_i b$ が証明できれば、正規な証明 (normal proof) が存在する。

型理論の場合と同様、RPT でも、 $a \vdash_i b$ の証明図と a とは、ほぼ同じ構造を持っている。しかし、RPT の場合は、上記の存在記号の導入規則により、証明項の中に、正規形を持たない項が含まれることがある。従って、両者を別の概念として形式化して証明する必要がある。正規化定理は無矛盾性を導くので、体系内で (自分自身の中で) 証明することはできない。従って、RPT から帰納法を除いた体系の正規化定理を RPT で形式化して証明する、ということが考えられる。

6 まとめ

RPT と Λ を中心とする計算機上の証明システムについて述べてきた。具体的な例題としてそれ自身興味のある Church-Rosser の定理の証明について述べた。

文献

- [1] Constable et al, Implementing Mathematics with the Nuprl Proof Development System, Prentice-Hall, 1986.
- [2] Susumu Hayashi and Hiroshi Nakano, "PX: A Computational Logic", MIT Press, 1988.
- [3] Masahiko Sato and Yuki Yoshi Kameyama, "Constructive Programming based on Λ ", TFKIP, 1989.
- [4] Masahiko Sato, "Reflective Proof Theory", 1990.
- [5] N. Shankar, "A Mechanical Proof of the Church-Rosser Theorem", JACM, vol. 35, No. 3, 1988.