1N－1

数学的帰納法を用いる定理証明器の実装

佐藤純一 山口文彦 中西 正和
慶應義塾大学大学院 理工学研究科 計算機科学専攻

1. はじめに

Bundy は、数学的帰納法を用いて証明を行う必要がある数式を証明する探索的な方法として、Rippling と Fertilize という手法を用いることを提案した。これは、数学的帰納法の公理の Step-case と呼ばれる部分を証明するための手法である。

本研究では、数学的帰納法を用いて証明を行う必要のある一階述語論理式で表された自然数の算術の数式を Rippling と Fertilize を用いて変換し、それを数学的帰納法を含む公理系から導出を用いて証明する自動証明器を実装する。

2. Bundy の手法

Bundy は、探索を用いて数学的帰納法による証明を行うための手法を提案している。

2.1 証明の戦略

数学的帰納法には多数の異なった規則があるが、最も一般的なものとして次のものがあげられる。

$$\frac{P(0),\quad P(n) \rightarrow P(s(n))}{P(n)}$$

数学的帰納法の証明は、二つの case に分けられる。それを図示すると図 1 のようになる。

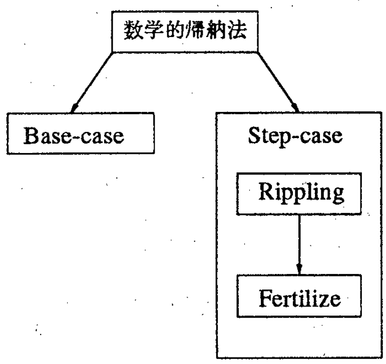

図 1: 数学的帰納法の戦略

Implementation of Theorem Prover using Mathematical Induction
Junichi SATO, Fumihiko YAMAGUCHI, Masakazu NAKANISHI
Department of Computer Science, Faculty of Science and Technology, Keio University 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa 223-8522, Japan

最初の副目標を Base-case、次の副目標を Step-case という。Step-case において、$P(n)$ は数学的帰納法の仮説、$P(s(n))$ を数学的帰納法の結論という。

Step-case では、Rippling と Fertilize が行われる。Rippling は、数学的帰納法の結論の中に数学的帰納法の仮説が現れるまで行われる。Fertilize では、数学的帰納法の結論の証明において数学的帰納法の仮説を適用する [1]。

2.2 Rippling

Rippling は、数学的帰納法の証明の Step-case を処理するための戦略である。これを用いて数学的帰納法の結論を書き換えて、数学的帰納法の仮説を適用できるようにする。特に、後続者関数 $s(\ldots)$ を含む数学的帰納法の結論を、中に数学的帰納法の仮説の複製を含むように書き換える。つまり、$P(s(n))$ を $Q[P(n)]$ に書き換える。Rippling は、数学的帰納法の仮説が現れるまで行われる [1]。

2.3 Fertilize

一般的には、数学的帰納法の仮説が当てはまる表現が数学的帰納法の結論の外側に動かされるとは限らないが、Rippling が完全ならば、数学的帰納法の結論を証明するために数学的帰納法の仮説を用いることができる。この戦略を Fertilize という [1, 2]。

3. 導出を用いた数学的帰納法での証明

導出を用いた数学的帰納法での証明を行うための公理系と証明に用いる手法について述べる。

3.1 ロビンソン算術

以下の式は、ロビンソン算術の公理である。

- $\forall x \forall y (x \neq y \rightarrow s(x) \neq s(y))$
- $\forall x (0 \neq s(x))$
- $\forall x (x \neq 0 \rightarrow \exists y x = s(y))$
- $\forall x (x + 0) = x$
- $\forall x \forall y (x + sy) = s(x + y)$
- $\forall x (x \times 0) = 0$
- $\forall x \forall y (x \times s(y)) = [(x \times y) + x]$

ここで、0 は自然数 0、s は後続者関数 $+1$ であり、$+$ は加法の関数、$\times$ は乗法の関数である。これをロビンソン算術が真となる解釈と呼ぶ。

ロビンソン算術は、個別の事例には強さを発揮するが、一般性には弱いところがある。

3.2 数学的帰納法を加えた公理系

また数学的帰納法の推論規則は、以下のような二階述語論理式とみなすことができる。

$$\forall P\{[P(0) \lor \forall x(P(x) \to P(s(x)))] \to \forall x P(x)\}$$

この二階述語論理式は、ロビンソン算術を真とする解釈の元で真となる。そこで、この公理をロビンソン算術の公理に加えて新しい公理系を作る。すると、ある二階の論理式がこの公理系から論理的に導かれるのは、その式がロビンソン算術が真となる解釈の元で真になるときであり、またそのときに限ることが証明できる [4]。

導出を用いて証明を行うために用いる手法として数学的帰納法の公理の式を用いるために、証明したい論理式をある 1 つの全称記号で束縛された個体変項を項に持つ一階述語論理の素論理式と見なす。その素論理式の述語記号を用いて、数学的帰納法の公理に普遍例化を行う [5]。これによって、数学的帰納法の公理が二階述語論理式から一階述語論理式となり、一階述語論理式を証明する方法を用いることができる。

3.3 証明したい論理式と導出に用いる恒真式

導出原理を用いて証明を行うときには、証明したい論理式の否定式の節集合が充足不能であることを示せば良い [3]。その証明したい式の否定式とともに、いくつかの恒真式を導出に用いる節集合に公理として加える。その恒真式は、数学的帰納法の公理に普遍例化を用いたときに与えた素論理式と証明したい式が同値であるという論理式から得られる。その同値であることを示す論理式を、与えた素論理式を $\forall x P(x)$ とすると、x が $0, x, s(x)$ の三通りの場合すべてについての同値であることを示す論理式を加える。また、証明したい論理式の否定を節集合に加えるのではなく、証明したい論理式と同値な与えた素論理式の否定を加えることにする。

3.4 数学的帰納法の適用

導出を用いた証明を行うときは論理式の節集合を用いるため、論理式を冠頭連言標準形に変形する必要がある。数学的帰納法で証明する必要がある数式は全称命題であるので、証明したい数式が内部に数学的帰納法を用いて証明を行う必要がある数式である場合には、その式を冠頭標準形に直すことで、証明したい数式の内部にある全称命題の全称記号を外部に移すことができる。よって、数式のもつ全称記号がかかっているすべての個体変数について数学的帰納法を用いて証明を行うことに帰着させることができる。従って、数学的帰納法の公理を適用する回数は一回だけで良い。

4. 実装

以上のことを考慮し、Lisp を用いて実装を行なった。実装された定理証明器は、和の結合法則や積の結合法則などの数学的帰納法を用いて証明を行う必要がある数式を証明することが可能である。

5. 考察

実装を行った証明器の考察を行う。

5.1 証明器の完全性

Rippling によって同値変形できる数式は限られているため、定理であっても証明できるとは限らない。よって、数学的帰納法を用いて証明する必要がある論理式は、定理であっても証明できるとは限らず、完全性がいえない。

5.2 複雑な数式の証明

より複雑な数式を証明するためには、より多くの演算記号を扱えるように公理を加える必要がある。Rippling では、等号、不等号、加算、乗算等が扱える [1, 2]。その他の演算子については、公理を拡張して扱えるように新たに公理を考える必要があると考えられる。また、Lisp で用いるようなリスト処理を行う関数を証明できるようにするためにもその関数を扱えるように公理を拡張する必要があると考えられる。

参考文献

[1] Bundy, A., Stevens, A., van Harmelen, F., Ireland, A., Smaill, A. : Ripping: A heuristic for guiding inductive proofs, Artificial Intelligence 62, pp. 185-253, 1993.

[2] Bundy, A.: A subsumption architecture for theorem proving?, Phil. Trans. R. Soc. Lond. A, Vol. 349, Iss. 1689, pp. 71-84, 1994.

[3] 有川節夫, 原口誠: 述語論理と論理プログラミング, 知識工学講座, オーム社, 1988.

[4] Jeffrey, R.(戸田山和久 訳): 形式論理学, 産業図書, 1995.

[5] Nolt, C., Rohatyn, D.(加地大介 訳): 例題で学ぶ現代論理学 1, マグロウヒル, 1994.