G-22

Bernstein の定理の証明の検証
ソフトウェア Isabelle を用いて
Verifying the proof of Bernstein's theorem
By using the software, Isabelle

○馬場彰太郎[1]，保谷哲也[2]
*Shotaro Baba[1], Tetsuya Hoya[2]

Abstract: Bernstein's theorem is a theorem of Set Theory. Isabelle is software to help user to proof propositions of mathematics, and it is one of generic proof assistants. Isabelle contains a part of Mathematical theorems, and a method of proving propositions in Isabelle is a combination of the theorems made by using commands. The purpose of this article is to show how the proof of Bernstein's theorem is verified with Isabelle.

１．概要

Bernstein の定理は集合論における定理である．Isabelle は数学における命題の証明の支援や検証を行うソフトウェアであり，証明支援システムの一つである．本稿では，ソフトウェア Isabelle における証明の過程と Bernstein の定理の内容を説明したのち，その定理のいくつかある証明のうち一つを，Isabelle を用いて検証する過程を示す．（検証する証明については，Wikipedia に掲載されているものを採用した.）

２．Isabelle による証明過程と定理の説明

２．１．Bernstein の定理[1]

「単射の写像 $\boldsymbol{f: A \to B}$ が存在し，かつ単射の写像 $\boldsymbol{g: B \to A}$ が存在するならば，全単射の写像 $\boldsymbol{h: A \to B}$ が存在する」が Bernstein の定理の内容である．

２．２．Bernstein の定理の証明

$\boldsymbol{A, B}$ を集合とし，$\boldsymbol{f: A \to B}$，$\boldsymbol{g: B \to A}$ をそれぞれ単射の写像とする．集合 $\boldsymbol{C_n}$，集合 $\boldsymbol{C}$，$\boldsymbol{h\_func}$ をそれぞれ以下のように定義する[1].

$$\boldsymbol{C_n = \begin{cases} A - g(B) & (n = 0) \\ g(f(C_{n-1})) & (n > 0) \end{cases}} \qquad (1)$$

$$\boldsymbol{C = \bigcup_{n \in N} C_n} \qquad (2)$$

$\boldsymbol{x \in A}$ として，

$$\boldsymbol{h\_func(x) = \begin{cases} f(x) & (x \in C) \\ g^{-1}(x) & (x \notin C) \end{cases}} \qquad (3)$$

$\boldsymbol{f, g}$ が単射の写像という条件から，上記のように定義された $\boldsymbol{h\_func}$ が集合 $\boldsymbol{A}$ から集合 $\boldsymbol{B}$ への写像でありかつ全単射であることをいえば証明が完了する．

２．３．Isabelle による命題の証明過程[2]

Isabelle は証明支援システムの一つであり，数学の命題は形式化され表現される．それらの証明は Isabelle によるプログラミングで行われる．以下，その簡単な例を挙げる．

lemma $Simple\_example$: "$[\![A \subseteq B; x \in A]\!] \Longrightarrow x \in B$"
apply $(rule\ subsetD[of\ A\ B\ x], assumption+)$
done

１行目の **lemma** は命題を導入するコマンドであり，１行目における "" で囲まれた部分は導入された命題の内容を表している．"" の中の矢印の右側の〚 〛で囲まれた部分は仮定を表し，左側の部分は結論を表している．２行目の **apply** は命題に対する操作の実行を宣言するコマンドであり，（ ）の中にその方法を記述する．最後の行 **done** は証明の終了を宣言するコマンドである．

３．検証の方法

２．１．２．２．で紹介した定理や命題をそれぞれ **Figure1**, **Figure 2** のように形式化する[2]．２．２．についてのみその方法について概略を述べる．**(1)** は，$\boldsymbol{C_0, C_1, \cdots, C_n, \cdots}$ のように再帰的に定義されるので，形式化すると **(1)′** となる．次に **(2)** では，集合 $\boldsymbol{C}$ は $\boldsymbol{C_0, C_1, \cdots, C_n, \cdots}$ の和集合として定義されるので，**(2)′**

１：日大理工・学部・数学　２：日大理工・教員・数学

に形式化される．最後に **(3)** は，もし $\boldsymbol{x \in C}$ なら $\boldsymbol{h\_func(x)}$ は $\boldsymbol{f(x)}$ と定義され，$\boldsymbol{x \notin C}$ なら $\boldsymbol{h\_func(x)}$ は $\boldsymbol{g^{-1}(x)}$ と定義される．$\boldsymbol{x \notin A}$ のとき，$\boldsymbol{h\_func(x)}$ は定義されない．ゆえに，**(3)′** に形式化される．これら形式化された命題を用いて証明の検証を進めていく．なお，この証明の形式化の一部については小林英恒著の Set1.thy における，Isabelle を用いた集合論の命題の形式化の一部を参考にして行った[3]．

４．結論

Bernstein の定理の証明の検証に成功した．本検証に必要であるコマンドを減らし処理の効率化を図るとともに，数学におけるその他の命題についても証明の検証を行っていくことが今後の課題である．

５．参考文献

[1] “Cantor-Bernstein-Schroeder theorem”, http://www.en.wikipedia.org/wiki/Cantor-Bernstein-Schroder_theorem

[2] Tobias Nipkow, Lawrence C. Paulson, Markus Wenzel, “A Proof Assistant for Higher-Order Logic”, Springer Verlag, P9-P18, P66-P93, 2009.

[3] 小林英恒，“Set1.thy”，2007.

Figure 1. Formalization of Bernstein’s theorem

lemma $Bernstein$: "$[\![\exists f.\, f \in A \to B \ \wedge inj\_on\ f\ A; \exists g.\, g \in B \to A \ \wedge\ inj\_on\ g\ A]\!] \Longrightarrow \exists h.\, h: A \to B \wedge bij\_to\ f\ A\ B$"

Figure 2. Part of the proof check

```
(* Formalizing the proposition of (1) *)
consts Cset :: "['a ⇒ 'b, 'b ⇒ 'a, 'a set, 'b set, nat] ⇒ 'a set"     (* Defining the type *)
primrec   (* Defing the set C_n recursively *)
Cset_0 : "Cset f g A B 0 = A − g ` B"     (* Formalizing the set C_0 *)
(* Formalizing the set C_{n+1} by using the set C_n *)
Cset_Suc : "Cset f g A B (Suc n) = (cmp g f ) ` Cset f g A B n"
```

(**1**)′

```
(* Formalizing the proposition of (2) *)
constdefs   (* Defining the set C non-recursively *)
Cset_Union :: "['a ⇒ 'b, 'b ⇒ 'a, 'a set, 'b set] ⇒ 'a set"     (* Defining the type *)
"Cset_Union f g A B == {x. ∃n ∈ N. x ∈ Cset f g A B n}"     (* Formalizing the set C *)
```

(**2**)′

```
(* Formalizing the proposition of (3) *)
constdefs   (* Defining h_func non-recursively *)
h_func :: "['a ⇒ 'b, 'b ⇒ 'a, 'a set, 'b set] ⇒ ('a ⇒ 'b)"  (* Defining the type *)
(* Formalizing h_func *)
"h_func f g A B == λx. if x ∈ A then
                    (if x ∈ Cset_Union f g A B then f(x) else (invfun g B A) (x) ) else undefined"
```

(**3**)′

```
(* The following program proves the proposition, C_n ⊆ C *)
lemma Cset_in_C : "Cset f g A B n ⊆ Cset_Union f g A B"
apply (simp only: Cset_Union_def)
apply (rule subsetI, rule CollectI)
apply (rule_tac x = n in bexI, assumption)
apply (rule n_in_Nset)
done
```