数理解析研究所講究録
912巻 1995年 22-32

On the Difficulty of Writing Out Formal Proofs in PRA

北陸先端科学技術大学院大学 情報科学研究科 鹿島 亮 (Ryo Kashima)

In this paper we investigate the difficulty of the problem of writing out formal proofs of given formulas in **PRA**. We prove a fact saying that even if we have $\mathrm{Th}_{\mathrm{PRA}}$ as an oracle and even if the target is restricted to proofs of such simple formulas as $\forall x(f(x)=0)$, we will have no "effective" strategy to write out formal proofs in **PRA**. ($\mathrm{Th}_{\mathrm{PRA}}$ is the characteristic function of the set of the Gödel numbers of theorems in **PRA**.)

1 Introduction

Gödel proved the second incompleteness theorem by formalizing the proof of the first incompleteness theorem. This technique made an epoch in proof theory, and it has been widely used. But it is remarkable that giving a *perfect* proof of the second incompleteness theorem is *practically impossible* because it needs writing out *vast* formal proofs in arithmetic. Why must we write out formal proofs? Let T be a suitable system of arithmetic such that T has function symbols of all primitive recursive functions and that the incompleteness theorem holds for T. (For example, take the Primitive Recursive Arithmetic as T.) Now suppose we are given a problem "show $T\vdash f(m)=n$" where f is a primitive recursive function. Then instead of giving a detailed proof in T, we can use the fact:

$$f(m)=n \text{ is true} \quad\Leftrightarrow\quad T\vdash f(m)=n.$$

That is, showing that "$f(m) = n$ is true" is sufficient to show it is provable in T. However if the problem becomes a little complicated as "show $T \vdash \forall x(f(x) = n)$", then we cannot use the above strategy because in general

$$\forall x(f(x) = n) \text{ is true} \quad \overset{\not\Rightarrow}{\Leftarrow} \quad T \vdash \forall x(f(x) = n).$$

Indeed the incompleteness theorem shows the existence of a primitive recursive function f such that $\forall x(f(x) = n)$ is true but $T \nvdash \forall x(f(x) = n)$. Therefore to give a perfect positive answer to this problem, we must write out a proof figure of $\forall x(f(x) = n)$ in T. A proof of the second incompleteness theorem essentially contains such problems, and we feel that giving a perfect proof of the second incompleteness theorem is practically impossible. In this paper we try to express such "practical impossibility" objectively.

We will investigate the difficulty of the problem of writing out proofs of given formulas of the form $\forall x(f(x) = n)$ in T. First we define a function Ω by

$$\Omega(f) = \begin{cases} \text{a proof of } \forall x(f(x) = 0) \text{ in } T & \text{if } T \vdash \forall x(f(x) = 0), \\ 0 & \text{otherwise.} \end{cases}$$

(To be more exact, Ω's input is a Gödel number of a primitive recursive function f, and the output is the minimum Gödel number of a proof of $\forall x(f(x) = 0)$, or 0.) Our first result is that Ω *is not recursive.* (Corollary 3.2)

By the way, when one is asked to show either a proof of $T \vdash A$ or the fact that A is not provable, what he will do is perhaps based on his intuition,

- to try to generate some candidates of subproofs of a proof of A, and
- to decide whether the conclusions of the candidates of the subproofs are provable or not.

The intuition may be regarded as an oracle, in the terminology of recursion theory. This observation suggests us the possibility of studying the computability of the function Ω with some oracles. It is then easy to show that Ω *is recursive in* Th_T (i.e., recursive with Th_T as an oracle) where Th_T is the characteristic function of the set of the Gödel numbers of formulas provable in T. On the other hand, Ω *is not primitive recursive in* Th_T, and moreover Ω *is not primitive recursive in any class of functions* $\{f_1, f_2, ...\}$ *so long as each* f_i *has a recursive upper bound* (Theorem 3.3). This is our main result in this paper.

These results indicate that even if we have Th_T as an oracle and even if our target is restricted to proofs of such simple formulas as $\forall x(f(x) = 0)$, we will have no "effective" strategy to write out formal proofs in T.

2 Preliminaries

$\mathcal{N}$ will denote natural numbers $\{0, 1, 2, ...\}$, and "functions" will mean total functions over $\mathcal{N}$.

The class $\mathbf{P}$ of *primitive recursive functions* is defined as usual (see, e.g., [1]) to be the smallest class satisfying the following conditions:

- (Initial functions) The unary function $\mathcal{Z}$, the unary function $\mathcal{S}$, and the k-ary function $\mathcal{P}_i^k$ is in $\mathbf{P}$ $(1 \leq i \leq k)$ where

  $\mathcal{Z}(n) = 0$ (constant Zero),

  $\mathcal{S}(n) = n + 1$ (Successor),

  $\mathcal{P}_i^k(n_1, ..., n_k) = n_i$ (Projection).

- (Composition) If f is an m-ary function in $\mathbf{P}$ and $g_1, ..., g_m$ are k-ary functions in $\mathbf{P}$, then the k-ary function $\mathcal{C}[f, g_1, ..., g_m]$ is also in $\mathbf{P}$ where

$$\mathcal{C}[f, g_1, ..., g_m](n_1, ..., n_k) = f(g_1(n_1, ..., n_k), ..., g_m(n_1, ..., n_k)).$$

- (Primitive recursion) If f is a k-ary function in $\mathbf{P}$ and g is a $(k+2)$-ary function in $\mathbf{P}$, then the $(k+1)$-ary function $\mathcal{R}[f, g]$ is also in $\mathbf{P}$ where

$$\mathcal{R}[f, g](n_1, ..., n_k, 0) = f(n_1, ..., n_k)$$
$$\mathcal{R}[f, g](n_1, ..., n_k, \mathcal{S}(m)) = g(n_1, ..., n_k, m, \mathcal{R}[f, g](n_1, ..., n_k, m)).$$

Let F be a class of functions. If the condition

$$F \subset \mathbf{P}$$

is added to the above definition of $\mathbf{P}$, then functions in $\mathbf{P}$ are said to be *primitive recursive in* F. If f is a function being primitive recursive in F, then there is an expression consisting of

$$[,\ ],\ \mathcal{Z},\ \mathcal{S},\ \mathcal{P}_i^k,\ \mathcal{C},\ \mathcal{R},\ \text{and the } \textit{names} \text{ of the functions in } F$$

which defines f. We say the expression is the *description* of the function f being primitive recursive in F. Of course many descriptions may define one function.

The axiom system **PRA**(Primitive Recursive Arithmetic) (see, e.g., [2]) is defined as follows.

The symbols in the language of **PRA** are the following:

- Constant symbol: $\overline{0}$.

- k-ary function symbol $\overline{f}$ for each description f of k-ary primitive recursive function. For example, $\overline{\mathcal{S}}$ is an unary function symbol and $\overline{\mathcal{R}[\mathcal{P}_1^1, \mathcal{C}[\mathcal{S}, \mathcal{P}_3^3]]}$ is a binary function symbol. (The latter represents "+". From now on, the term $\overline{\mathcal{R}[\mathcal{P}_1^1, \mathcal{C}[\mathcal{S}, \mathcal{P}_3^3]]}(t_1, t_2)$ will be abbreviated to $t_1 + t_2$.)

- Variable symbols: $\mathbf{v}_0$, $\mathbf{v}_1$, $\mathbf{v}_2$...
- Predicate symbol: $=$.
- Logical connectives and quantifiers: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\forall$, $\exists$.

Terms and *formulas* are defined as usual from these symbols. Terms $\overline{0}, \overline{\mathcal{S}}(\overline{0}), \overline{\mathcal{S}}(\overline{\mathcal{S}}(\overline{0})), \ldots$ are called *numerals*, and they will be denoted by $\overline{0}, \overline{1}, \overline{2}, \ldots$, respectively. If $A(x)$ is a formula with free variable x, then $A(t)$ means the formula obtained from $A(x)$ by replacing x by the term t.

The axioms and inference rules in **PRA** consist of the following:

- Usual axioms and inference rules for classical first-order logic with $=$.
- Axioms for each function symbol:

$$\overline{\mathcal{Z}}(\mathbf{v}_0) = \overline{0}$$
$$\neg(\overline{\mathcal{S}}(\mathbf{v}_0) = \overline{0})$$
$$\overline{\mathcal{S}}(\mathbf{v}_0) = \overline{\mathcal{S}}(\mathbf{v}_1) \rightarrow \mathbf{v}_0 = \mathbf{v}_1$$
$$\overline{\mathcal{P}^k_i}(\mathbf{v}_1, \ldots, \mathbf{v}_k) = \mathbf{v}_i$$
$$\overline{\mathcal{C}[f, g_1, \ldots, g_m]}(\mathbf{v}_1, \ldots, \mathbf{v}_k) = \overline{f}(\overline{g_1}(\mathbf{v}_1, \ldots, \mathbf{v}_k), \ldots, \overline{g_m}(\mathbf{v}_1, \ldots, \mathbf{v}_k))$$
$$\overline{\mathcal{R}[f, g]}(\mathbf{v}_1, \ldots, \mathbf{v}_k, \overline{0}) = \overline{f}(\mathbf{v}_1, \ldots, \mathbf{v}_k)$$
$$\overline{\mathcal{R}[f, g]}(\mathbf{v}_1, \ldots, \mathbf{v}_k, \overline{\mathcal{S}}(\mathbf{v}_0)) = \overline{g}(\mathbf{v}_1, \ldots, \mathbf{v}_k, \mathbf{v}_0, \overline{\mathcal{R}[f, g]}(\mathbf{v}_1, \ldots, \mathbf{v}_k, \mathbf{v}_0))$$

- The induction axiom for each quantifier-free formula $A(x)$:

$$(A(\overline{0}) \wedge \forall x(A(x) \rightarrow A(\overline{\mathcal{S}}(x)))) \rightarrow \forall x A(x)$$

We assume a standard Gödel numbering function Gn; that is, Gn(α) codes each expression α in **PRA** (see, e.g. [2]). If Gn(α) $= n$, then $\ulcorner \alpha \urcorner$ will denote the numeral $\overline{n}$.

The following Propositions 2.1–2.3 are well-known. See, e.g., [1] and [2] for the proofs.

Proposition 2.1 *Let f be a description of a k-ary primitive recursive function. Then*

$$\mathbf{PRA} \vdash \overline{f}(\overline{n_1}, ..., \overline{n_k}) = \overline{m} \quad \textit{if} \quad f(n_1, ..., n_k) = m$$

$$\mathbf{PRA} \vdash \neg(\overline{f}(\overline{n_1}, ..., \overline{n_k}) = \overline{m}) \quad \textit{if} \quad f(n_1, ..., n_k) \neq m$$

for all $n_1, ..., n_k, m \in \mathcal{N}$.

Proposition 2.2 *For any k-ary recursive predicate R, there is a formula $\overline{R}(x_1, ..., x_k)$ such that*

$$\mathbf{PRA} \vdash \overline{R}(\overline{n_1}, ..., \overline{n_k}) \quad \textit{if} \quad R(n_1, ..., n_k) \textit{ holds}$$

$$\mathbf{PRA} \vdash \neg\overline{R}(\overline{n_1}, ..., \overline{n_k}) \quad \textit{if} \quad R(n_1, ..., n_k) \textit{ does not hold}$$

for all $n_1, ..., n_k \in \mathcal{N}$.

Proposition 2.3 *For any formula $A(x)$ with at most one free variable x, there is a sentence B such that* $\mathbf{PRA} \vdash B \leftrightarrow A(\ulcorner B \urcorner)$.

3 Main result

Let Prov be the primitive recursive binary predicate defined by

$$\text{Prov}(m, n) \Leftrightarrow n \text{ codes a formula } A \text{ and } m \text{ codes a proof of } A \text{ in } \mathbf{PRA}.$$

Then we give a precise definition of the function Ω:

$$\Omega(n) = \begin{cases} \mu y[\text{Prov}(y, \text{Gn}(\forall \mathbf{v}_0(\overline{f}(\mathbf{v}_0) = \overline{0})))] & \text{if } n = \text{Gn}(\overline{f}) \text{ for some unary} \\ & \text{function symbol } \overline{f} \text{ such that} \\ & \mathbf{PRA} \vdash \forall \mathbf{v}_0(\overline{f}(\mathbf{v}_0) = \overline{0}), \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 3.1 *There is no formula $Q(x)$ with a free variable x such that*

$$\mathbf{PRA} \vdash Q(\ulcorner\overline{f}\urcorner) \quad \textit{if} \quad \mathbf{PRA} \vdash \forall \mathrm{v}_0(\overline{f}(\mathrm{v}_0) = \overline{0}) \tag{1}$$

$$\mathbf{PRA} \vdash \neg Q(\ulcorner\overline{f}\urcorner) \quad \textit{if} \quad \mathbf{PRA} \nvdash \forall \mathrm{v}_0(\overline{f}(\mathrm{v}_0) = \overline{0}) \tag{2}$$

for any unary function symbol $\overline{f}$.

Proof By "Rosser's technique" we show that the existence of such $Q(x)$ yields contradiction.

Let A be a formula. Then we have a primitive recursive function Pr_A such that

$$\mathrm{Pr}_A(m) = \begin{cases} 1 & \text{if } \mathrm{Prov}(m, \mathrm{Gn}(A)) \text{ and } \forall y \leq m(\neg\mathrm{Prov}(y, \mathrm{Gn}(\neg A))), \\ 0 & \text{otherwise.} \end{cases}$$

A precise description of Pr_A is as follows. Let $\mathrm{prove}(m,n)$ and $\mathrm{nonproved}(m,n)$ be the primitive recursive functions:

$$\mathrm{prove}(m,n) = \begin{cases} 1 & \text{if } \mathrm{Prov}(m,n), \\ 0 & \text{otherwise,} \end{cases}$$

$$\begin{cases} \mathrm{nonproved}(0,n) & = \ 1, \\ \mathrm{nonproved}(\mathcal{S}(m),n) & = \ \mathrm{nonproved}(m,n) * (1 - \mathrm{prove}(\mathcal{S}(m),n)). \end{cases}$$

(We are assuming that 0 does not code any proofs.) Then

$$\mathrm{Pr}_A(m) = \mathrm{prove}(m, \mathrm{Gn}(A)) * \mathrm{nonproved}(m, \mathrm{Gn}(\neg A)).$$

From now on, "Pr_A" will denote such a description.

There is a primitive recursive function g such that

$$\mathrm{g}(\mathrm{Gn}(A)) = \mathrm{Gn}(\overline{\mathrm{Pr}_A}) \tag{3}$$

holds for any formula A. Then let $\overline{\mathrm{g}}$ be a function symbol of such g. By Proposition 2.3 we obtain a sentence P such that

$$\mathbf{PRA} \vdash P \leftrightarrow Q(\overline{\mathrm{g}}(\ulcorner P \urcorner)). \tag{4}$$

Now we consider two possible cases:

$$\text{(Case 1)}\quad \mathbf{PRA} \vdash \forall \mathbf{v}_0(\overline{\mathrm{Pr}_P}(\mathbf{v}_0) = \overline{0}). \tag{5}$$

$$\text{(Case 2)}\quad \mathbf{PRA} \nvdash \forall \mathbf{v}_0(\overline{\mathrm{Pr}_P}(\mathbf{v}_0) = \overline{0}). \tag{6}$$

In case 1 we have

$$\mathbf{PRA} \vdash Q(\ulcorner \overline{\mathrm{Pr}_P} \urcorner) \tag{7}$$

by (1). By the way, (3) and Proposition 2.1 imply

$$\mathbf{PRA} \vdash \overline{\mathrm{g}}(\ulcorner P \urcorner) = \ulcorner \overline{\mathrm{Pr}_P} \urcorner. \tag{8}$$

Then (4), (7), and (8) imply

$$\mathbf{PRA} \vdash P$$

i.e., there is a proof of P in **PRA**. Let m be the Gödel number of a proof of P, then we have

$$\mathbf{PRA} \vdash \neg(\overline{\mathrm{Pr}_P}(\overline{m}) = \overline{0})$$

by the definition of Pr_P, consistency of **PRA**, and Proposition 2.1. So we have

$$\mathbf{PRA} \vdash \exists \mathbf{v}_0 \neg(\overline{\mathrm{Pr}_P}(\mathbf{v}_0) = \overline{0})$$

but this is impossible because of (5) and the consistency of **PRA**.

In case 2 we have

$$\mathbf{PRA} \vdash \neg Q(\ulcorner \overline{\mathrm{Pr}_P} \urcorner) \tag{9}$$

by (2). Then (4), (8), and (9) imply

$$\mathbf{PRA} \vdash \neg P$$

i.e., there is a proof of $\neg P$ in **PRA**. Let m be the Gödel number of a proof of $\neg P$, then we have

$$\mathbf{PRA} \vdash \overline{\mathrm{Pr}_P}(\overline{i}) = \overline{0}, \quad \text{for } i = 0, 1, \cdots, m-1 \tag{10}$$

by the definition of Pr_P, consistency of **PRA**, and Proposition 2.1. Moreover we have

$$\mathbf{PRA} \vdash \forall \mathbf{v}_1(\overline{\mathrm{Pr}_P}(\mathbf{v}_1 + \overline{m}) = \overline{0}) \tag{11}$$

because the formula

$$\forall \mathbf{v}_1(\overline{\mathrm{nonproved}}(\mathbf{v}_1 + \overline{m}, \ulcorner \neg P \urcorner) = \overline{0})$$

is proved in **PRA** by using the induction axiom. On the other hand, we have

$$\mathbf{PRA} \vdash \forall \mathbf{v}_0(\mathbf{v}_0 = \overline{0} \vee \mathbf{v}_0 = \overline{1} \vee \cdots \vee \mathbf{v}_0 = \overline{m-1} \vee \exists \mathbf{v}_1(\mathbf{v}_0 = \mathbf{v}_1 + \overline{m}))^1. \tag{12}$$

Then, (10), (11) and (12) imply

$$\mathbf{PRA} \vdash \forall \mathbf{v}_0(\overline{\mathrm{Pr}_P}(\mathbf{v}_0) = \overline{0}),$$

and this contradicts (6). ∎

Corollary 3.2 *The function Ω is not recursive.*

Proof By Proposition 2.2 and Lemma 3.1. ∎

Theorem 3.3 *Let F be a class of functions such that each f in F has a recursive upper bound, i.e., there is a recursive function g for f such that $\forall n_1 \cdots \forall n_k(f(n_1, ..., n_k) \leq g(n_1, ..., n_k))$. Then the function Ω is not primitive recursive in F.* [2]

Proof We show that if Ω were primitive recursive in F, then we could construct an algorithm to compute Ω. This together with Corollary 3.2 proves the theorem.

[1]This formula is provable in Robinson's arithmetic **Q** (see, e.g., [1]). Note that **Q** has an axiom $\forall x(\neg(x=0) \rightarrow \exists y(x = \overline{S}(y)))$, which is provable in **PRA** by applying the induction axiom to the formula $\neg(x=0) \rightarrow x = \overline{S}(\overline{\mathrm{prev}}(x))$ where prev(0)=0 and prev($S(n)$)=n.

[2]In an earlier version of this paper, the statement of this theorem is weaker and inelegant: " ... each f in F is either a recursive function or a function whose range is $\{0, 1\}$." The present form is suggested by Professor Kojiro Kobayashi, to whom the author would like to give his thanks.

Suppose α is a description of a k-ary function which is primitive recursive in F, and $n_1, ..., n_k \in \mathcal{N}$. We will call the expression

$$\alpha(n_1, ..., n_k)$$

a *redex*. We now define the rules of *reduction*, i.e, rewriting a redex:

1. (Deterministic reduction)

   (a) $\mathcal{Z}(n) \;\Rightarrow\; 0$

   (b) $\mathcal{S}(n) \;\Rightarrow\; n+1$

   (c) $\mathcal{P}_i^k(n_1, ..., n_k) \;\Rightarrow\; n_i$

   (d) $\mathcal{C}[\alpha, \beta_1, ..., \beta_m](n_1, ..., n_k) \;\Rightarrow\; \alpha(\beta_1(n_1, ..., n_k), ..., \beta_m(n_1, ..., n_k))$

   (e) $\mathcal{R}[\alpha, \beta](n_1, ..., n_k, m) \;\Rightarrow$
   $\beta(n_1, ..., n_k, m{-}1, \beta(n_1, ..., n_k, m{-}2, \beta(\cdots \beta(n_1, ..., n_k, 1, \alpha(n_1, ..., n_k)) \cdots)))$

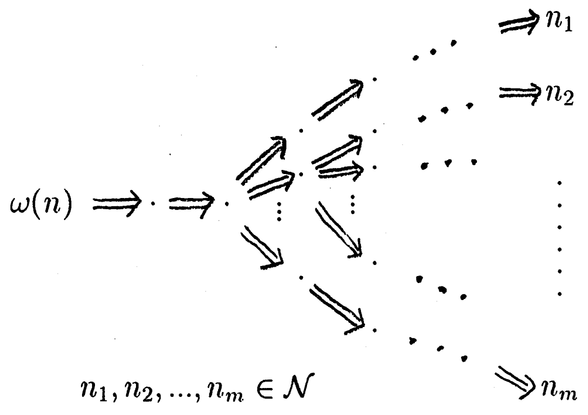
2. (Nondeterministic reduction)
   If f is the name of a function in F then there is a recursive upper bound g of f, and then there are $(m{+}1)$ ways to reduce the redex $f(n_1, ..., n_k)$:

   $$f(n_1, ..., n_k) \;\Rightarrow\; 0, 1, ..., m$$

   where $m = g(n_1, ..., n_k)$.

If the function Ω were primitive recursive in F, there is a description ω of Ω. Then we can effectively compute the value of $\Omega(n)$ for a given n, as follows:

If $n \neq \mathrm{Gn}(\overline{f})$ for any unary function symbol $\overline{f}$, then $\Omega(n) = 0$. If $n = \mathrm{Gn}(\overline{f})$, then we make a "reduction tree" which starts from $\omega(n)$:



$\cdot \Rightarrow \cdot$ means a deterministic reduction of a redex.

means a nondeterministic reduction of a redex.

Note that the length of each path in the reduction tree is finite because the depth of the nests of "[ ]" in $\omega(n)$ must decrease by reductions. Therefore, by König's Lemma, the reduction tree is finite, and we can effectively compute the values of $n_1, \ldots, n_m$. Then, for $i = 1, \ldots, m$, we examine whether n_i satisfies $\mathrm{Prov}(n_i, \mathrm{Gn}(\forall \mathbf{v}_0(\overline{f}(\mathbf{v}_0) = \overline{0})))$. If such n_i exists, then $\Omega(n) = \min\{n_i \mid \mathrm{Prov}(n_i, \mathrm{Gn}(\forall \mathbf{v}_0(\overline{f}(\mathbf{v}_0) = \overline{0})))\}$, otherwise $\Omega(n) = 0$.

This algorithm is complete because there must be at most one n_i satisfying the above condition if $\Omega(n) \neq 0$. ∎

参考文献

[1] P. Odifreddi, Classical Recursion Theory, North-Holland (1992).

[2] C. Smoryński, Self-Reference and Modal Logic, Springer (1985).