情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2010-AL-131 No.13
2010/9/22

計算理論から見たゲーデルとチューリング

田　中　榮　一

本論文は述語とその長さの関係を調べ，帰納的述語は有限長で，帰納的ではない述語は本質的に無限長であることについて述べている．ゲーデルの不完全性定理の証明に用いられた証明可能性述語とチューリング機械の停止問題を表す述語は帰納的ではないから，共に本質的に無限長論理式である．前者から不完全性定理の証明が誤っていることが分る．後者からチューリング機械の停止問題が決定不能であることが直感的に理解できる．

Gödel and Turing from the Viewpoint of the Theory of Computation

Eiichi Tanaka †1

This paper discusses the relation between a predicate and its length, and reports that the length of a recursive predicate is finitely long and that of a non recursive predicate is essentially infinitely long. It is well known that the provability predicate and the predicate for the halting problem of a Turing machine are not recursive. Therefore both predicates are essentially infinitely long. The former means that the proof of the first incompleteness theorem is incorrect. From the latter, it is intuitively understood that the halting problem of a Turing machine is undecidable.

1. Introduction

This paper is a sequel of the another paper [1] and discusses computability, provability and decidability from the viewpoint of the theory of computation. Kleene [2),3)] explains the first incompleteness theorem [4] using the predicate to express the computation of a Turing machine [5]. Kashima [6] states that any formula obtained by a recursively enumerable but not recursive relation is unprovable. We show that the length of a recursive predicate is finitely long, and that of a non recursive predicate is essentially infinitely long. It is well known that the provability predicate [4),7)] and the predicate to express the halting problem of a Turing machine [8] are not recursive. So the lengths of both predicates are essentially infinitely long. This means that the proof of the first incompleteness theorem is incorrect. Therefore it is natural that the provability predicate is neither proved nor refuted. Furthermore the undecidability of the halting problem of a Turing machine is intuitively understood.

†1 Kobe University

2. Preliminaries

We shall review a primitive recursive function and a recursive function. Let $\overline{x}$ denote $x_1, x_2, \cdots, x_n$.

(b1) Initial functions

The initial functions are defined for natural numbers.

(a) The zero function: $Z(x) = 0$ for all x.

(b) The successor function: $S(x) = x'$ for all x, where x' is the successor of x.

(c) The projection functions: $U_n^i(\overline{x}) = x_i$ for all $x_1, x_2, \cdots, x_n$.

(b2) Composition

Let $h_1, \cdots, h_r$ be r functions of n variables ($r \geq 1,\ n \geq 0$). Let g and f be a function of r variables and that of n variables, respectively. Define f as follows.

$$f(\overline{x}) = g(h_1(\overline{x}), \cdots, h_r(\overline{x})) \quad (1)$$

(b3) Primitive recursion

Let g and h be a function of n variables ($n \geq 0$) and that of $(n+2)$ variables, respectively. Define function f of $(n+1)$ variables as follows.

$$f(\overline{x}, 0) = g(\overline{x}) \quad (2)$$

$$f(\overline{x}, y') = h(\overline{x}, y, f(\overline{x}, y)) \quad (3)$$

Definition 1.
A function is primitive recursive, if it can be obtained by a finite applications of the operations of composition and primitive recursion beginning with initial functions (b1).

Vol.2010-AL-131 No.13
2010/9/22

Definition 2.
A subset of the set of n-tuples in a set A is called an n-ary predicate in A. If P is an n-ary predicate, we define an n-ary function R_P such that $R_{P(a)} = 0$, if $P(a)$, and $R_{P(a)} = 1$, if $\neg P(a)$. We call R_P the representing function of P. P is primitive recursive, if R_P is primitive recursive [9].

(b4) μ operator

Function $g(\overline{x}, y)$ is called regular, if there is a natural number y such that $g(\overline{x}, y) = 0$ for any $\overline{x}$. μ operator is to find the least y for a regular function. Asume that g is a regular function of $(n+1)$ variables. Define function f of n variables as follows.

$$f(\overline{x}) = \mu y(g(\overline{x}, y) = 0) \tag{4}$$

Definition 3.
A function is recursive, if it can be obtained by finite applications of the operations of composition, primitive recursion and a μ operator beginning with initial functions (b1). Similarly a recursive predicate can be defined.

The predicate logic for computing theory is defined using the following symbols. (c1) individual constants $(a, b, c, \cdots)$, (c2) variables$(x, y, z, \cdots)$, (c3) function symbols $(Z, S, U, +, *, \mu)$, (c4) predicate symbols $(=)$, (c5) logical symbols 1 ($\neg, \vee$), (c6) logical symbols 2 $(\exists)$, (c7) subsidiary symbols ((,), comma).

The predicate logic is basically after Shoenfield [9], but is slightly modified. In this paper the symbols of (c1) $\sim$ (c5) and (c7) are called the basic symbols for the theory of computation. Assume that defined function symbols and defined predicate symbols are rewritten using the basic symbols. A term and a formula are defined in the similar way as the predicate logic for the arithmetic [1].

Consider a predicate with an infinite existential quantifier $\exists x C(x)$. This predicate includes infinite number of formulas such as $C(0), C(1), \cdots$. This means that the computation of $\exists x C(x)$ does not end in finite operations. Therefore we exclude an existential quantifier from the basic symbols.

The lengths of a function, a formula and a predicate are similarly defined as defined in 1). An essentially infinite length predicate is also the same in 1).

3. From the Viewpoint of the Thory of Computation

In this section we shall study the lengths of a function and a predicate.
(1) A recursive function (predicate) is finite length
We shall confirm that the computation of a primitive recursive function and that of a recursive function terminate in finite operations.

The initial functions $Z(x) = 0$ and $S(x) = x'$ are determined, if x is given. $U_n^i(\overline{x}) = x_i$ is obtained, if $\overline{x}$ and i are given. If $\overline{x}, h_1, \cdots, h_r, g$ are given, a function $f(\overline{x})$ defined by composition can be computed.

Consider a primitive recursion. Assume that x and y are given. $f(x, y)$ is computed in the following way.

$$\begin{aligned} &f(x, 0) = g(x) \\ &f(x, 1) = h(x, 0, f(x, 0)) = h(x, 0, g(x)) \\ &f(x, 2) = h(x, 1, f(x, 1)) \\ &\quad \cdots \\ &f(x, y) = h(x, y-1, f(x, y-1)) \end{aligned} \tag{5}$$

The computation of a primitive recursion terminates in finite operations, if x and y are finite.

A function obtained by finite applications of a composition and a primitive recursion beginning with initial functions is a function with finite symbols. Therefore a primitive recursive function is a finite length function. If R_P is primitive recursive, P is a finite length predicate.

Vol.2010-AL-131 No.13
2010/9/22

Remark 1.
A primitive recursive function is a finite length function. If R_P is primitive recursive, P is a finite length predicate.

A regular function $g(\overline{x}, y)$ has y_0 such that $g(\overline{x}, y_0) = 0$, where y_0 is a finite value. Even if y_0 is not known, y_0 can be obtained by finite times computation of $g(\overline{x}, y)$ for $y = 0, 1, \cdots$. Define $p(z)$ such that if $z = 0, p(z) = 1$, and otherwise, $p(z) = 0$. The computation of $\mu g(\overline{x}, y_0)$ is expressed in the following way.

$$\mu g(\overline{x}, y_0) = 0 * p(g(\overline{x}, 0)) + 1 * p(g(\overline{x}, 1)) + \ldots + y_0 * p(g(\overline{x}, y_0)) \quad (6)$$

The function $\mu g(\overline{x}, y_0)$ is a finite length function.

From Remark 1 and (6), a recursive function is finite length. If R_P is recursive, P must be finite length.

Remark 2.
Both a recursive function and a recursive predicate are finite lengths.

(2) A non recursive function (predicate) is infinite length
Recall the following important and widely accepted understanding of a recursive function.
(c1) A recursive function is a computable function.
(c2) According to Church's thesis, a computable function is a recursive function.

The following two statements are self-evident.
(c3) From Remark 2, a recursive function is a finite length function.
(c4) From (c2) and (c3), a computable function is a finite length function.

Replacing a function with a predicate, we have similar statements.
(d1) A recursive predicate is a decidable predicate.
(d2) From (c2), a decidable predicate is a recursive predicate.
(d3) A recursive predicate is a finite length predicate.
(d4) A decidable predicate is a finite length predicate.

Consider an essentially infinite length function. It goes without saying that an essentially infinite length function is not computable. Therefore an essentially infinite length function is not a recursive function. A non recursive function must be an essentially infinite length function.

An essentially infinite length predicate is undecidable. So an essentially infinite length predicate is not a recursive predicate. Therefore a non recursive predicate must be an essentially infinite length predicate.

Lemma 1.
A non recursive predicate is an essentially infinite length predicate.

Gödel [6] states that "we can not assert that $Bew(x)$ is recursive" (Heijenoort, p.606 in 10)), where $Bew(x)$ is the provability predicate and x is a provable formula. Maehara (p.182 in 7)) shows the proof that $Bew(x)$ is not a recursive predicate. This fact means that the provability predicate is an essentially infinite length predicate. Then it can not be proved nor refuted.

Lemma 2.
The provability predicate is an essentially infinite length predicate. Therefore it can be neither proved nor refuted.

Predicate $\exists z T(x, x, z)$ is not recursive [8]. Therefore $\exists z T(x, x, z)$ is an essentially infinite length predicate. Then the halting problem of a Turing machine is undecidable. It is well known that the halting problem of a Turing machine is undecidable.

Lemma 3.
Predicate $\exists z T(x, x, z)$ is an essentially infinite length predicate. Therefore the halting problem of a Turing machine is undecidable.

(3) A finite length predicate is either provable or refutable
In this subsection, assume that T is the arithmetic and formula A is finite length. If A is valid in T, we write $T \models A$, otherwise, $T \models \neg A$. The completeness theorem of predicate logic is expressed in the following way.

Completeness theorem

$$T \vdash A \leftrightarrow T \models A \tag{7}$$

A finite length predicate is a finite length recursive predicate. From (d1), a recursive predicate is decidable. If A is true ($T \models A$), it is proved in T ($T \vdash A$). If A is false ($T \models \neg A$), it is refuted in T ($T \vdash \neg A$). The set F_T of finite length formulas derived from T is consistent. Summing up, a finite length predicate is provable or refutable. In other words, there is no finite length predicate that is neither proved nor refuted.

Remark 3.
There is no finite length predicate that is neither proved nor refuted.

The above observation is very fundamental and simple. Gödel published his paper [6] 80 years ago. We wonder why nobody has noticed this up to now.

4. Concluding Remarks

We can summarize the conclusions just obtained as follows:
(1) A recursive predicate is a finite length predicate and a non recursive predicate is an essentially infinite length predicate.
(2) The provability predicate is not recursive. Therefore it is an essentially infinite length predicate. This indicates that Gödel's proof of the incompleteness theorems is incorrect. Furthermore there is no finite length predicate that is neither proved nor refuted.
(3) The predicate to express the halting problem of a Turing machine is not recursive. Then it is an essentially infinite length predicate. Therefore the undecidability of the halting problem of a Turing machine is naturally understood.

Acknowledgment
We would like to express our thanks to sincere metamathematicians for their comments. Both constructive and critical comments were very stimulative.

References

1) Tanaka, E.; Reflections on Gödel and Turing, *IPSJ SIG Technical Report*, Vol. 2010-AL-131, No. 12, pp.1-6, (2010)
2) Kleene, S. C.; Recursive predicates and quantifiers, *Transaction of the American Mathematical Society*, 53, pp.41-74, (1936)
3) Kleene, S. C.; *Mathematical Logic*, John Wiley and Sons, NY (1967)
4) Gödel, K.; Über Formal Unentscheidbare Sätze der Principia Mathematica und Verwandter Systeme, *Monatshefte für Mathematik und Physik*, 38, pp.173-198, (1931)
5) Turing, T.; On Computable Numbers, with an Application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society*, ser.2, 42, pp.230-265, (1936)
6) Tanaka, K., Kashima, R.,Kadota,N., Kikuchi, M.; *Lectures on Foundations of Mathematics - The Incompleteness Theorems and the Development* (in Japanese), Nihonhyouronsha, Tokyo, (1997)
7) Maehara, S; *Introduction to Metamathematics* (in Japanese), Asakura-shoten, Tokyo, (1977)
8) Davis, M.; *Computability and Unsolvability*, McGraw-Hill, NY, (1958)
9) Shoenfield, J. R.; *Mathematical Logic*, Addison-Wesley, Massachusets, (1967)
10) Heijenoort, J.; *From Frege to Gödel*, Harvard University Press, Massachusetts, (1967)