On general recursive functions

December 2004

Masako Takahashi
Division of Natural Sciences
International Christian University
Mitaka, Tokyo 181-8585, Japan

Abstract

We note that the notion of Herbrand-Gödel's general recursive functions[3] and that of McCarthy's definition of functions by recursion[9] share a fundamental viewpoint of computation that by a certain generalization of reccurrence formulas all and only computational processes can be described. In this respect, we propose to call the latter formalism also as general recursion. Their features in comparison with other mathematical formalisms of computation are discussed.

1. The origin of general recursiveness

In 1934, Gödel gave a series of lectures[3] on his results in 1931[2] and some related topics, and there he introduced a new notion called *general recursive functions*, which was obtained by modifying Herbrand's suggestion; if φ denotes an unknown function, and $\psi_1, ..., \psi_k$ are known functions, and if the ψ's and the φ are substituted in one another in the most general fashions and certain pairs of the resulting expressions are equated, then if the resulting set of functional equations has one and only one solution for φ, φ is a (general) recursive function. Gödel imposed the following restrictions on the Herbrand's idea; the left-hand sides of equations shall be of the form $\varphi(\psi_{i_1}(\vec{x}), \psi_{i_2}(\vec{x}), ..., \psi_{i_l}(\vec{x}))$, and for each l-tuple $\vec{k}$ of natural numbers there shall be one and only one m such that $\varphi(\vec{k}) = m$ is a derived equation. Note that in those days only total functions were considered in studies of computability. Also note that Gödel's derivation means 'call-by-value' evaluation (cf. [3], [4], or [6] for more details).

2. General recursiveness, λ-definability, and partial recursiveness

According to Kleene[7], the Gödel's notion is historically the second among mutually equivalent mathematical descriptions of the class of computable functions of natural numbers. The first is the notion of λ-definability, introduced by Church. In the memoir[7] Kleene recalls its dawn, as follows: In the fall of 1931-2, I was taking Church's course in which, besides reading Gödel 1931, I was made acquainted with Church's postulates for the foundation of logic. ... They included one ingredient that has proved to be extraordinarily fruitful. ... Before research was done, no one guessed the richness of this subsystem. (See also Rosser's memoir[10].)

During their study of the expressive power of λ-definable functions, Kleene found the μ-operator very useful for their study. He then attempted to investigate Gödel's notion by using the μ-operator, and he succeeded in proving the normal form theorem

for general recursive functions[4]; namely, every general recursive function can be expressed by means of the μ-operator (used just once) and primitive recursive functions via composition. By applying the result, Kleene succeeded also in establishing the equivalence of the general recursiveness and the λ-definability[5].

Kleene thus reached the notion of *partial recursive functions*, by removing the side condition which has always been associated with the μ-operator to guarantee the result be a total function. Hereafter, in particular under the strong influence of his comprehensive textbook[6] on mathematical logic, the notion of partial recursive functions has widely been accepted as the central notion among mutually equivalent mathematical formalizations of computable functions of natural numbers.

3. McCarthy's definition of functions by recursion

In early 1960's, McCarthy[9] introduced a new notion of *definition of functions by recursion*, which consists of a set of mutually recursive definitions of functions as in contemporary programming languages in which no side effect (hence no assignment statement) is involved. This notion may be considered as a further refinement of Herbrand-Gödel's general recursive functions in the following sense.

- In McCarthy's systems, each definition is of the form $\varphi(\vec{x}) = e$ where the defining expression e may contain *conditional expressions* as in the following example

$$\gcd(x, y) \quad = \quad \mathsf{if}\ (y = 0)\ \mathsf{then}\ x\ \mathsf{else}\ \gcd(y, \mathrm{mod}(x, y)),$$

  and simultaneous recursion is allowed .
- In the system, partial functions are defined.

In spite of these differences, both formalisms share a fundamental viewpoint of computation that by a certain generalization of recurrence formulas all and only computation processes can be described. In this respect, we propose to call the McCarthy's way of defining functions also *general recursion*.

4. Features of general recursion

As McCarthy[9] pointed out, the general recursion based on the conditional expressions has the following advantages.

(a) Functions in which one may be interested are more easily written down in the system and the resulting expressions are more brief and understandable than other formalisms such as partial recursive functions, λ-definitions, Turing machines, etc. This is because in order to control the flow of the calculation this formalism provides a direct means while in others one has to apply general methods such as integer calculation.

As other features of general recursion, one may think of the following.

(b) In order to analyze programs in procedural programming languages, the notion of general recursion provides an adequate way of describing the relation between intermediate values of variables during computation. This point is beneficial theoretically and practically. For example, from theoretical point of view, with this description at hand one can derive Kleene's type of normal form theorem smoothly. On the other hand, from practical point of view, such descriptions are useful for verification of programs.

(c) As mentioned earlier, both Herbrand-Gödel's formalism and McCarthy's reflect (or extend) the fundamental idea of recurrence formulas, and this aspect of computation might be educationally useful, in order for students (who studied recurrence formulas in high school) to comprehend what computers are. In other words, one can say that computers are machines to solve recurrence formulas.

(d) Needless to say, this formalism may not always be superior to others. For example, the description of the universal function by Kleene's formalism as in his normal form theorem would certainly be superior theoretically and also for better understanding of the subject matter.

References

[1] M. Davis (ed.): *The Undecidable — Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*, Raven Press, 1965.

[2] K. Gödel: Uber formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatsh. Math. Phys.*, Vol. 38, 1931, pp. 173-198. (English translation in [1], pp. 4-38.)

[3] K. Gödel: On undecidable propositions of formal mathematical systems, Lecture notes by S. C. Kleene and J. B. Rosser, Inst. for Advanced Study, Princeton, 1934. (Reprinted with corrections and postscriptum in [1], pp.39-74.)

[4] S. C. Kleene: General recursive functions of natural numbers, *Mathmatische Annalen*, vol. 112, 1936, pp. 727-742. (Reprinted with addendum in [1], pp. 236.)

[5] S. C. Kleene: Lambda definability and recursiveness, *Duke Mathematical Journal*, vol. 2, 1936, pp. 340-353.

[6] S. C. Kleene: *Introduction to Metamathematics*, Van Nostrand, 1952.

[7] S. C. Kleene: Origins of recursive function theory, *20th Ann. Symp. on Foundations of Computer Science*, IEEE, 1979, pp. 371-382.

[8] S. C. Kleene: Introductory note to 1934, *Kurt Gödel Collected Works*, Vol. I, edited by Feferman et al., Oxford University Press, 1986, pp. 338-345.

[9] J. McCarthy: A basis for a mathematical theory of computation, *Computer Programming and Formal Systems*, edited by P. Braffort, et al., North-Holland, 1963, pp. 33–70.

[10] J. B. Rosser: Highlights of the history of the lambda-calculus, *Proc. ACM Symposium on LISP and Functional Programming*, 1982, pp. 216-225.