The Korean Journal for History of Mathematics Vol. 17 No. 4 (Nov. 2004), 27-36

Gödel's Critique of Turing's Mechanism

Center for Cognitive Science, Yonsei University **Woosik Hyun**
godel@yonsei.ac.kr

Dedicated to Professor Taidong Han on the 80th Birthday.

This paper addresses Gödel's critique of Turing's mechanism that a configuration of the Turing machine corresponds to each state of human mind. The first part gives a quick overview of Turing's analysis of cognition as computation and its variants. In the following part, we describe the concept of Turing machines, and the third part explains the computational limitations of Turing machines as a cognitive system. The fourth part demonstrates that Gödel did not agree with Turing's argument, sometimes referred to as mechanism. Finally, we discuss an oracle Turing machine and its implications.

Key words: (oracle) Turing Machine, mechanism, Gödel's incompleteness theorem, non-computability

0. Mind to Computing Machine

A physical symbol system is an instance of a Universal Turing machine. Thus, we may deduce that cognition can be realized by a Universal Turing machine. The development of the first digital computer and of the automata theory originates with Alan Turing's work in "On Computable Numbers, with an Application to the Entscheidungsproblem(1936)"[1]. The chess programs of C. Shanon and A. Turing, LISP of J. McCarthy, Logic Theorist and General Problem Solver of A. Newell, H. A. Simon and J. C. Shaw, PROLOG of F. Green, R. Kowalski and A. Colmerauer, SOAR of A. Newell, J. Laird, and P. Rosenbloom, Automated Mathematician program of R. Davis and D. B. Lenat, and Boyer-Moore theorem prover of R. S. Boyer and J. S. Moore are just some of the significant theoretical and historical works worth noting in cognitive science.

Since Turing's abstract device is regarded as the embodiment of mathematical

thinking at the most fundamental level, then the computer and cognitive scientists' claim on the Turing machine as a conceptual tool is now at least as strong as the logician's one. Its significance for the computability theory is fundamental: within a finite time, the Turing machine is capable of any computation that can be done by any modern digital computer, no matter how powerful. Thus, for the theoretical study of the ultimate problem-solving capacity of the real computer, the Turing machine and its degree is a necessary condition.

Turing's analysis transformed the term *finite procedure* into *mechanical procedure.* Consequently, a function is computable, or effectively calculable, if it can be calculated by a finite mechanical procedure, that is, by a Turing machine. Kurt Gödel claimed: "a formal system can simply be defined as any mechanical procedure for producing formulas, called provable formulas"[2]. In this sense, a function is Turing computable if it is definable by a Turing machine[1]. According to this framework, a formal mind as a Turing machine yields m on input n if, when the machine is started on input n, it eventually halts, and at the moment when it halts, the tape represents m.

1. The Concept of Turing Machines

The Turing machine is a finite automaton with unlimited tape as a memory device. It is mathematically equivalent to the class of the Herbrand-Gödel-Kleene equation system, that is, the class of general recursive functions[3, 4]. Rather, Gödel endorsed the concept of Turing machines as a generally accepted property of effective calculability, not as general recursion defined by himself[5]. Turing devised an idealized human computing agent with the concepts of function produced by mechanical procedure. Turing's theorem states that any function calculable by an idealized human computer is Turing computable[1]. Furthermore, Turing's thesis asserts that if a function is informally computable, then it is computed by a Turing idealized human computer, meaning that every algorithm can be programmed on a one-tape Turing machine.

A Turing machine is characterized by the following:

(1) a list of states called by Turing machine configurations: a specification of how many states there are.

(2) a finite alphabet of symbols, including blank and stroke.

(3) a finite set of lists of instructions. Each instruction has the form of the quintuple (i, s, t, Φ, j), where i and j are numbers no greater than the number of states, s and t are elements of the alphabet, and Φ is either R(move one right) or L(move one left). The instructions may be read: if in state i and scanning a cell containing s, then replace s with t ,move as Φ directs, and go into new state j.

Let S be a finite set of symbols including Blank 0 and Stroke 1, and let $q_1, q_2, \ldots$ be symbols of states not in S. Then a Turing machine on S is a finite set of quintuples (q_i, s, t, Φ, q_j), where s and t are in S and Φ is one of the symbols "R(move one right) or L(move one left)," such that no two distinct quintuples have the same first two members. The symbol q_i represents the state i. Formally, a Turing machine is a mapping TM such that for some natural number n,

$$TM: \{0, 1, 2, \cdots, n\} \times \{0, 1\} \rightarrow \{0, 1\} \times \{L, R\} \times \{0, 1, 2, \cdots, n\}$$

where L stands for "move one left" and R "move one right."

Turing compared a man in the process of computing a real number to a machine that is only capable of a finite number of conditions $q_1, q_2, \ldots, q_r$ that is called "*m-configurations.*" According to Turing, the behavior of the machine is determined by the *m-configuration* q_n and the scanned symbol s_r. This pair (q_n, s_r) is called the *configuration.* Thus, the configuration determines the possible behavior of the machine. Since there are only finitely many pairs, the behavior of the machine is specified by a finite list.

UNIVERSAL TURING MACHINE (Turing 1936). For a recursive function F, there is a universal Turing machine UTM such that $F_{TM_n}(x) = F_{UTM}(n, x)$, for any Turing machine TM_n and for any natural numbers n and x. This means that there is a universal Turing machine that can simulate any Turing machine.

2. The Non-computability of Turing Machines

In 1936 paper, A. Turing proved that Hilbert's 10th problem, *Endscheidungs-problem*(decision problem), of discovering a method for establishing the truth or falsity of any statement in the first-order calculus, was impossible to solve by the Turing machine. Gödel stated,

> In consequence of later advances, in particular of the fact that, due to A. M. Turing's work, a precise and unquestionably adequate definition of the general concept of formal system can now be given, the existence of undecidable arithmetical propositions and the non-demonstrability of the consistency of a system in the same system can now be proved rigorously for *every* consistent formal system containing a certain amount of finitary number theory. (Gödel [2])

A function f is *Turing computable* if there is a Turing machine TM that computes f. For an n-place function f, TM computes f if and only if, any $x_1, \ldots, x_n$ of the natural numbers, TM produces $f(x_1, \ldots, x_n)$ on input x. It is well known that Turing computable function f is a decidable $(n+1)$-ary relation and a recursively enumerable relation.

Turing, furthermore, demonstrated the limitations of Turing computability, proving that there are unsolvable problems, e.g., the *Halting Problem*, in the Turing machine system[1]. There is a fundamental result in unsolvable problems: There exists a recursively enumerable set which is not recursive. The halting function for the Turing machine is a mechanical implementation of Gödel's undecidable sentences. Gödel credited that Turing's 1936 paper provides an adequate analysis of mechanical procedures and that, as a consequence of his work, a general formulation of the incompleteness theorems can be given[2].

UNSOLVABILITY OF TURING MACHINE (Turing 1936). There is no Turing machine M such that, for all e and n, if the Turing machine Gödel-numbered e produces something on input n then M produces 0 on input (e, n) if the Turing machine Gödel-numbered e produces nothing on input n then M produces 1 on input (e, n). Mathematically, this means: Let $K = \{x: \phi_x(x) \textit{ halts}\}$ where ϕ_x is partial recursive function computed by a Turing machine program with Gödel number e. Then K is recursively enumerable, but

not recursive.

This result is known as the effective unsolvability of the Halting problem for the Turing machines. This is equivalent to Church's theorem that the decision problem for first-order calculus is not solvable. Thus, the results show that Hilbert's Entscheidungsproblem can not be solved. Both Gödel's and Turing's theorems show the limitations of the first-order calculus system or of recursive universal machines. Thus, if the mind is a Turing machine and cognition is a Turing computable function, then the mind would not be able to compute such Halting functions, because they would not be in the class of cognition.

NON-COMPUTABILITY OF COGNITION AS TURING COMPUTABLE FUNCTION. Let human cognition be a Turing computable function. Then there is non-computable cognition on the natural numbers. This is obvious by Cantor's diagonal idea. Suppose that all cognitions are computable in the sense of Turing computability. Then there are countable numbers of cognition C. Thus, one can enumerate the cognitions C_1, C_2, ... Define a cognition C on natural numbers by $C(x) = C_x(x) + 1$. Since C is Turing computable, it must appear somewhere in our list of the cognition. Put $C = C_k$. Then $C(k) = C_k(k) + 1$ if and only if $C_k(k)$, which leads to a contradiction. Hence, C is not computable.

3. Gödel's Critique of Turing's Argument

Turing claimed that it is possible to construct a machine to do the work of the human computer[1]. According to Turing, to each state of mind of the human computer corresponds an m-configuration of the Turing machine. As Turing asserted:

> The behaviour of the computer [the human computer] at any moment determined by the symbols which he is observing, and his "state of mind" at that moment. We may suppose that there is a bound B to the number of symbols or squares which the computer can observe at one moment. If he wishes to that the umber of states of mind which need be taken into account is finite. The reason for this are of the same character as those which restrict the number of symbols. If we

> admitted an infinity of states of mind, some of them will be "arbitrarily close" and will be confused. Again, the restriction is not one which seriously affects computation, since the use of more complicated states of mind can be avoided by writing more symbols on the tape. (Turing [1], p.136)

His idea is based on that the simple operations of the Turing machine must include:

(a) Changes of the symbol on one of the observed squares.

(b) Changes of one of the squares observed to another square within squares of one of the previously observed squares.

According to Turing, it may be that some of these changes necessarily involve a change of state of mind. The most general single operation must therefore be taken to be one of the following:

(A) a possible change (a) of symbol together with a possible change of state of mind.

(B) a possible change (b) of observed squares, together with a possible change of state of mind. (Turing [1], p.137)

In "Some Remarks on the Undecidability Results(1972)," Gödel stressed that Turing gives an argument that mental procedures cannot go beyond mechanical procedures[6]. Although Gödel accepted Turing's analysis of the computability, he did not agree with Turing on this point. Gödel told that Turing's argument is inconclusive:

> What Turing disregards completely is the fact that *mind, in its use, is not static, but constantly developing*, i.e., that we understand abstract terms more and more precisely as we go on using them, and that more and more abstract terms enter the sphere of our understanding. (Italics in original, Gödel [6], p.306)

In Gödel's view, Turing disregarded temporal elements of the mental capability. In contrast, he avoided the term *static*. Rather, Gödel himself focused on the developing process as a significant capability of cognition. Consequently, Gödel

suggested two possibilities such as existence and convergence: (1) There may exist systematic methods of actualizing this development; (2) Turing's number of *distinguishable states of mind* may *converge toward infinity*. This process refers to forming of stronger and stronger axioms of infinity in set theory. According to Gödel's account, such developing processes would produce a non-recursive number -theoretic function.

The existence of *finite non-mechanical procedures* is not excluded by Turing's analysis. Gödel claimed:

> [T]he question of whether there exist finite *non-mechanical* procedures, not equivalent with any algorithm, has nothing whatsoever to do with the adequacy of the definition of "formal system" and of "mechanical procedure." (Italics in original, Gödel [2], p.370)

Gödel asserted that his incompleteness results do not limit the powers of human reason. According to Gödel, the incompleteness results do not establish any bounds for the powers of human reason, but rather than for the potentialities of pure formalism in mathematics([2], p. 370). Unlike Turing, Gödel chose a positive way to the higher powers of human cognition with respect to Turing's mechanism.

4. Beyond Gödel's Limitation?

To overcome the computational limitation of the Turing machine, in "Systems of Logic Based on Ordinals(1939)," Turing proposed an extension of his machine model[7]. This idea gave rise to important issues such as *arithmetical hierarchy* and *relative recursiveness* [8].

ORACLE TURING MACHINE. An *oracle Turing machine* is simply a Turing machine with an extra "read only" tape, called the *oracle tape*, upon which is written the characteristic function of some set O called the *oracle*, and whose symbols cannot be printed over. The old tape is called the *work tape*. The reading head moves along both tapes simultaneously. An Oracle Turing machine is a function OTM such that for some natural number n,

$$OTM: \{0, 1, 2, \ldots, n\} \times \{0, 1, 2\} \times \{0, 1\} \rightarrow \{0, 1\} \times \{L, R\} \times \{0, 1, 2, \ldots, n\},$$

where $0, 1, 2$ is the oracle tape alphabet, L stands for "move one left" and R for "move one right."

An oracle O for a function $f: N \rightarrow N$ is a device that, for a natural number $n \in N$, responds the value $f(n)$. Suppose A and B are arbitrary sets, and for all $n \in N$, $n \in A$ if and only if $f(n) \in B$. Then, we have a decision procedure for membership in A if we have a decision procedure for membership in B. If there exists a decision procedure which computes $f(n)$ from n using an oracle O for B, for all $n \in N$, then A is reducible to B via f, written $A \leq_f B$. The oracle tape, therefore, is a query tape. An oracle O for B is a external agent that will supply the correct answer to questions of the form "$x \in B$?" or not, for every $x \in N$. We can replace f by a Turing machine if f is recursive. By this, we can consider the problem of relative computability or relative reducibility. Although the oracle has a new and powerful feature, it is the least constructive approach. It is remarkable to note that

(1) O is not necessarily identified with an algorithm,

(2) B may not be recursive,

(3) f may accept members of N^N as arguments.

The oracle model is clearly more powerful than its predecessor, but it is also clear that the power comes from the addition of a function that was previously not computable. Subsequently, this leads to a recursive function that accepts members of the uncountable set as inputs, which raises the problem of relative computations on recursive infinite functions. The extension model, however, still cannot give us any real idea of how to compute the halting function. Moreover, such an infinite machine is beyond the scope of our debate, for it does not satisfy the assumptions underlying the finite machine, the type specified, or the consistency condition.

In Turing's view, for a given formal system S_1, one can add the statement $Con(S_1)$, consistency of S_1, as a new axiom to S_1 in order to obtain S_2.

Similarly we can obtain $Con(S_2)$, $Con(S_3)$, ... His finding implies that any true sentence is provable at some state in the transfinite iteration process. This process is clearly not complete within some finitary methods. Consequently, it is totally dependent on the assumption and philosophy of logicians.

References

1. Turing, A., "On computable numbers, with an application to the Entscheidungs-problem," *Proceedings of the London Mathematical Society*, 42(1936), 230-265.
2. Gödel, K., "Postscriptum to Gödel 1934 (1964)," *Collected Works I: Publications 1929-1936*, S Ferferman et al.(eds), Oxford University Press, 1986.
3. Gödel, K., "On undecidable propositions of formal mathematical systems(1934)," in *Collected Works I: Publications 1929-1936*, S. Ferferman et al.(eds), Oxford University Press, 1986.
4. Kleene, S.C., *Introduction to Metamathematics*, D. Van Nostrand, 1952.
5. Gödel, K., "Some basic theorems on the foundations of mathematics and their implications(1951)," in *Collected Works III: Unpublished Essays and Lectures*, S. Ferferman et al.(eds), Oxford University Press, 1995.
6. Gödel, K., "Some remarks on the undecidability results(1972)," in *Collected Works II: Publications 1938-1974*, S. Ferferman et al.(eds), Oxford University Press, 1990.
7. Turing, A., "Systems of logic based on ordinals," *Proceedings of the London Mathematical Society*, ser. 2, 45(1939), 161-228.
8. Soare, R., *Recursively Enumerable Sets and Degrees*, Springer-Verlag, 1987.

튜링의 기계주의에 대한 괴델의 비평

연세대학교 인지과학연구소 **현우식**

이 논문에서는 튜링의 기계주의에 대한 괴델의 비평을 다룬다. 여기에서 튜링의 기계주의란 튜링기계의 기호배열이 인간의 마음의 각 상태에 대응된다는 것을 의미한다. 첫째 부분에서는 계산으로서의 인지과정에 대한 튜링의 분석을 검토한다. 두 번째 부분에서는 튜링기계의 개념을 살펴보고, 세 번째 부분에서는 인지적 체계로서의 튜링기계가 갖는 계산적 한계를 설명한다. 네 번째 부분에서는 괴델이 튜링의 기계주의에 동의하지 않았음을 보이고, 마지막으로 오라클 튜링기계과 그 함의에 대하여 논의한다.

주제어 : (오라클) 튜링기계, 기계주의, 괴델의 불완전성정리, 계산불가능성
2000 Mathematics Subject Classification : 03D10 01A60 03A05 68Q05
ZDM Classification : E20 M50 R40