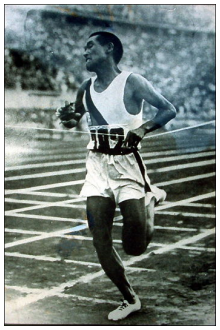
현대 컴퓨터의 모델,
튜링기계(Turing Machine)의 고안*

이광근†

06/2010

1936년. 손기정 선수가 베를린 올림픽 마라톤에서 우승하던 바로 그 해. 마라톤 경기(8월 9일)를 2달 정도 남긴 5월 28일.



손기정

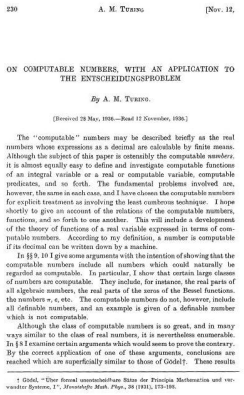
알란 튜링

손기정 선수와 동갑내기인 영국의 알란 튜링(Alan Turing)이라는 청년. 그는 다음 제목의 논문을 런던 수리학회(London Mathematical Society)에 제출한다.

> *"계산가능한 수에 대해서, 수리명제 자동생성 문제에 응용하면서"(On Computable Numbers, with an Application to the Entscheidungsproblem)*

*과학창의재단, "20세기 10대 과학 사건" 원고

†서울대 교수, ropas.snu.ac.kr/~kwang



230 A. M. TURING [Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. Turing.

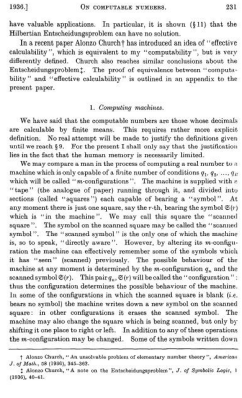
[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.



1936.] ON COMPUTABLE NUMBERS. 231

have valuable applications. In particular, it is shown (§11) that the Hilbertian Entscheidungsproblem can have no solution.

In a recent paper Alonzo Church† has introduced an idea of "effective calculability", which is equivalent to my "computability", but is very differently defined. Church also reaches similar conclusions about the Entscheidungsproblem‡. The proof of equivalence between "computability" and "effective calculability" is outlined in an appendix to the present paper.

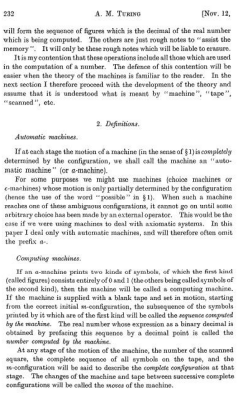
1. *Computing machines.*

We have said that the computable numbers are those whose decimals are calculable by finite means. This requires rather more explicit definition. No real attempt will be made to justify the definitions given until we reach § 9. For the present I shall only say that the justification lies in the fact that the human memory is necessarily limited.

We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions $q_1, q_2, \ldots, q_R$ which will be called "m-configurations". The machine is supplied with a "tape" (the analogue of paper) running through it, and divided into sections (called "squares") each capable of bearing a "symbol". At any moment there is just one square, say the r-th, bearing the symbol $\mathfrak{S}(r)$ which is "in the machine". We may call this square the "scanned square". The symbol on the scanned square may be called the "scanned symbol". The "scanned symbol" is the only one of which the machine is, so to speak, "directly aware". However, by altering its m-configuration the machine can effectively remember some of the symbols which it has "seen" (scanned) previously. The possible behaviour of the machine at any moment is determined by the m-configuration q_n and the scanned symbol $\mathfrak{S}(r)$. This pair $q_n, \mathfrak{S}(r)$ will be called the "configuration": thus the configuration determines the possible behaviour of the machine. In some of the configurations in which the scanned square is blank (i.e. bears no symbol) the machine writes down a new symbol on the scanned square: in other configurations it erases the scanned symbol. The machine may also change the square which is being scanned, but only by shifting it one place to right or left. In addition to any of these operations the m-configuration may be changed. Some of the symbols written down

† Alonzo Church, "An unsolvable problem of elementary number theory", *American J. of Math.*, 58 (1936), 345–363.

‡ Alonzo Church, "A note on the Entscheidungsproblem", *J. of Symbolic Logic*, 1 (1936), 40–41.



232 A. M. TURING [Nov. 12,

will form the sequence of figures which is the decimal of the real number which is being computed. The others are just rough notes to "assist the memory". It will only be these rough notes which will be liable to erasure.

It is my contention that these operations include all those which are used in the computation of a number. The defence of this contention will be easier when the theory of the machines is familiar to the reader. In the next section I therefore proceed with the development of the theory and assume that it is understood what is meant by "machine", "tape", "scanned", etc.

2. *Definitions.*

Automatic machines.

If at each stage the motion of a machine (in the sense of § 1) is *completely* determined by the configuration, we shall call the machine an "automatic machine" (or a-machine).

For some purposes we might use machines (choice machines or c-machines) whose motion is only partially determined by the configuration (hence the use of the word "possible" in § 1). When such a machine reaches one of these ambiguous configurations, it cannot go on until some arbitrary choice has been made by an external operator. This would be the case if we were using machines to deal with axiomatic systems. In this paper I deal only with automatic machines, and will therefore often omit the prefix a-.

Computing machines.

If an a-machine prints two kinds of symbols, of which the first kind (called figures) consists entirely of 0 and 1 (the others being called symbols of the second kind), then the machine will be called a computing machine. If the machine is supplied with a blank tape and set in motion, starting from the correct initial m-configuration, the subsequence of the symbols printed by it which are of the first kind will be called the *sequence computed by the machine*. The real number whose expression as a binary decimal is obtained by prefacing this sequence by a decimal point is called the *number computed by the machine*.

At any stage of the motion of the machine, the number of the scanned square, the complete sequence of all symbols on the tape, and the m-configuration will be said to describe the *complete configuration* at that stage. The changes of the machine and tape between successive complete configurations will be called the moves of the machine.

...

이 논문에서 튜링은 컴퓨터의 근본적인 디자인을 최초로 선 보인다. 이 때 튜링은 2년 전 캠브릿지(Cambridge)대학 수학 학부과정을 최우수 성적으로 마친 24세의 청년이었다. 그 해 9월 프린스턴(Princeton)대학 고등연구소(Institute for Advanced Study)로 유학 떠나기 직전이었다.

그런데 제목만으로는 도저히 컴퓨터와 관계없을 것 같은데, 어떻게 그 논문에 컴퓨터의 원천 설계도가 나타난 걸까? 그리고, 컴퓨터의 무엇이 그렇게 특별해서 컴퓨터의 모델을 최초로 고안한 튜링을 기념해서 그것을 "튜링기계(Turing Machine)"이라고 부르고, 컴퓨터분야의 노벨상을 "튜링상(Turing Award)"라고 부르는 걸까?

우선 컴퓨터라는 도구의 비범함에 대해서. 컴퓨터는 지금까지 인류가 발명한 도구 중에서 아주 독특한 능력을 가진 도구다. 일찍이 다른 도구에는 없던 능력. 그것은 컴퓨터 하나가 무한히 다양한 일을 할 수 있다는 것이다. 컴퓨터 한 대가 문서편집도 돕고, 동영상도 재생하고, 게임도 해주고, 인터넷 서핑도 해준다. 필요한 일은 무엇이든 소프트웨어로 만들기만 하

면 컴퓨터가 그 일을 해 준다. 이것이 컴퓨터의 비범함이다. 바퀴나 칼이나 자동차나 냉장고등과는 다른.

그런데 이런 비범한 도구를 최초로 설계한 논문의 제목이 영 이상하지 않은가? 재미있게도, 위의 논문은 "이러한 특이한 도구를 디자인 했으니 보라"고 주장한 논문은 아니다. 위의 논문은 그 당시 수학계에 내리친 청천벽력의 좌절을 전해들은 튜링이 색다른 방식으로 그 사실을 다시 증명해 본 것이다. 그런데 이 논문에서 컴퓨터의 원천 설계도가 슬그머니 드러난다. 아이러니하게도, 좌절을 증명하는 작품속에 인류의 정보혁명을 이끌 도구의 설계도가 주요 소품으로 등장했던 것이다.

그럼 튜링이 리바이브한 당시 수학계에 내리친 청천벽력은 무엇인가? 그 것은 쿠르트 괴델(Kurt Gödel)이 1931년에 증명한 다음의 사실이었다.

> *"기계적인 방식만으론 수학의 모든 사실들을 만들어 낼 수 없다."*

쿠르트 괴델

이게 무슨 말인가? 괴델의 증명이 있기 전까지는 하나의 꿈이 유럽 수학계를 떠돌고 있었다. 기계적인 방식으로 – 자동으로 – 모든 수학의 사실들을 술술 만들어 낼 수 있을 것 같다는 꿈. 그래서 수학자의 고된 짐을 벗을 수 있을 것 같다는 꿈. 이에 괴델이 내려친 벼락은 "꿈 깨시오"였던 것. 이 꿈은 절대 이루어질 수 없다고 증명해버린 것이다. 기계적인 방법만으로는 모든 수학적인 추론을 만들어낼 수 없다는 것이다. 수학자들이 깊이

생각해서 알아내는 사실들을 자동으로 척척 빠뜨리지않고 모두 만들어내는 기계는 불가능 하다는 것이다.

튜링은 이 좌절을 새롭게 증명하는 과정에서 간단한 기계부품들을 정의한다. 그리곤 이 부품들로 하나의 특별한 기계를 만들어 보이는데, 그 기계가 바로 컴퓨터인 것이다.

우선, 튜링은 그 증명에서 왜 기계부품들을 정의할 필요가 있었을까? 튜링의 증명은 단도직입적이다. 증명할 게 뭔가? 기계적인 방식만으론 모든 사실들을 만들 수 없다는 것이다. 이에 튜링은 "기계적인 방식"이 뭔지를 곧바로 정의해간다. 튜링은 아주 단순한 다섯 종류의 기계 부품들을 정의한다. 마치 레고 블락 셋트같은. 그리고 그 부품들로 만든 기계들로 돌릴 수 있는 것만을 "기계적인 방식"이라고 정의한다. 그러고는, 이 방식으로는 절대 돌릴 수 없는 계산 문제를 하나 보인다. 더군다나, 그 계산 문제의 답이 참인지 거짓인지 우리도 알 수 없는. 따라서, 기계적인 방식으로는 참인 사실을 하나도 빠뜨리지 않고 만들어 낼 수 있는 건 아니라는 결론을 이끌어 낸다.

이 증명의 중앙부 가파른 협곡을 통과하면서 튜링은 컴퓨터의 설계도를 펼쳐보이게된다. 튜링은 우선 자신이 정의한 "기계적인 방식"이 충분히 광범위하다는 것을 설득해야 했다. 그런 연후에야 "기계적인 방식"으로 만들어 낼 수 없는 사실이 있다는 게 의미있는 결론이 되므로. 그런데 그 정의가 충분히 광범위하다는 것을 어떻게 증명해 보일 수 있겠는가. 직관에 호소할 수 밖에. 그래서 튜링은 자기의 방식으로 돌아가는 기계들의 예를 보여주기 시작한다. 그리고는 그 정점의 예로, 하나의 특이한 "궁극의 기계"를 만들어 보인다. 이 정도의 기계까지 만들어 낼 수 있는 내 기계부품들은 그러므로 충분히 광범위한게 아니냐는 걸 암시하면서. 그리곤 그 기계를 타고 증명의 중심부 협곡을 아슬아슬 통과해 간다. 이 궁극의 기계를 가지고도 작동시킬 수도 없고 참인지 거짓인지도 알 수 없는 계산 문제가 있다는 것을 보이는 것으로. 그런데 바로 이 궁극의 기계에 컴퓨터의 원천 설계도가 담기게 된다.

이 궁극의 기계를 소개하기 전에 우선 튜링이 정의한 기계부품들이 무

엇인지 살펴보자. 그 기계부품들은 너무나 단순하다. 아래에 그 부품으로 만든 튜링기계의 한 예가 있다:

현재 상태	테잎 심볼	하는 일	움직이는 방향	다음상태
Q_1	A	쓰기 A	오른쪽	Q_3
Q_1	C	지우기	왼쪽	Q_4
Q_2	B	쓰기 C	제자리	Q_5
⋮	⋮	⋮	⋮	⋮
Q_n	A	쓰기 B	오른쪽	Q_1

부품들은 다섯 가지다. 무한히 많은 칸을 가진 테잎, 테잎에 기록되는 기호들("A", "B", "C"등. 유한 개), 테잎에 기록된 기호를 읽거나 쓰는 장치, 그 장치의 상태들("Q_1", "Q_2" 등. 유한 개), 그리고 기계의 작동규칙표(그림의 오른쪽 표). 기계마다 이 다섯가지 부품이 구체적으로 정해진다. 몇 개의 기호들을 테잎에 읽고 쓰게 될 것인지, 장치의 상태들이 몇개나 되는지, 그리고 작동규칙표는 무엇인지. 그리고 테잎의 시작 모습과 읽고쓰는 장치의 시작 상태, 그리고 테잎에서의 시작 위치가 정해진다. 이렇게 정의된 기계가 작동 규칙대로 작동을 하면서 정해진 계산을 진행하게 되는 것이다. 작동규칙표에는 기계의 작동 규칙이 정의되있다. 그림에서 표의 한 횡이 하나의 작동 규칙이고 그 규칙들이 유한개 모여서 작동규칙표를 이룬다. 작동 규칙에 표현되는 기계의 작동은 매우 간단한 일로만 제한된다. 테잎 칸의 기호를 읽고 쓰면서 테잎을 한 칸씩 좌우로 움직여 가는 일만 할 수 있다. 이러면서 기계의 상태가 매번 변경된다. (그림에서 박스 안에 시계 바늘이 현재의 상태를 가리킨다.) 기계가 어떤 상태에서 어떤 기호를 읽으면 어떤 기호를 다시 쓰고 테잎의 어느방향으로 움직이며 새로운 기계상태는 무엇이 되는지가 작동 규칙이된다.

튜링이 만들어 보인 "궁극의 기계"도 이렇게 구성되는 튜링기계의 하나이고 이 기계에 컴퓨터의 핵심능력이 처음으로 등장하게 되는 것이다. 컴퓨터의 핵심 능력은 하나의 컴퓨터가 모든 일을 할 수 있다는 점이다. "궁극의 기계"가 바로 그 능력을 처음으로 보여준 설계인 것이다.

이 궁극의 기계는 두개의 대담한 설계를 담고있다. 첫째는, 임의의 튜링기계의 정의(작동규칙표와 테잎의 초기상태)를 일렬의 테잎에 표현할 수 있다는 점, 그래서 임의의 다른 기계의 정의를 테잎에 입력으로 받도록 한 점. 둘째는, 테잎에 표현된 기계의 정의를 이해하고 그 동작을 그대로 흉내낼 수 있도록 작동규칙표를 정의한 점. 그래서 자기의 고정된 작동규칙표를 따르면 테잎에 기록된 임의의 기계를 그대로 흉내낼 수 있도록 한 점. 튜링은 이런 능력을 갖춘 기계를 자신이 정의한 기계 부품들로 만들어 보인다.

튜링은 이 궁극의 기계를 "보편만능의 기계(Universal Computing Machine)"라고 부른다. 튜링기계의 하나지만 모든 튜링기계를 흉내낼 수 있기 때문에. 어떤 계산이건 그에 해당하는 기계를 테잎에 입력으로 받아 그 작동을 그대로 흉내낼 수 있기 때문에. 그가 정한 범주에 드는 모든 기계적인 계산이 그 하나의 기계로 가능하기 때문에.

자 이제 튜링의 기계부품들을 되돌아보자. 그토록 단순한 부품들로 보편만능의 기계를 만들 수 있다는 것은 참 놀랍고 반가운 일이다. 튜링의 기계부품은 매우 단순하기 때문에 그 기계부품을 실재로 만들어 볼 수 있을게고, 그 부품으로 보편만능의 기계를 만들 수 있지 않겠는가.

이쯤에서 여러분의 컴퓨터를 보자. 그곳에는 튜링이 정의한 기계 부품들이 고스란히 구현되어 있다. 테잎은 메모리칩으로, 테잎에 읽고 쓰는 장치는 메모리칩과 입출력 장치로, 작동규칙표는 중앙처리장치(CPU)로. 특히 CPU는 보편만능의 기계에 해당하는 작동규칙표를 구현한 것이다. 따라서 실행시키고 싶은 일에 해당하는 기계를 메모리에 표현해 – 소프트웨어로 만들어 – 넣어주기만 하면 컴퓨터는 그 정의대로 일을 수행한다. 튜링의 보편만능의 기계, 그 실재가 여러분 무릎위에 놓인 컴퓨터인 것이다.

▷▷ 2036년. 튜링의 논문이후 100년. 그 동안 풀리지 않았던 의문. 튜링의 기계를 능가하는 컴퓨터가 가능할까? 즉, 튜링의 방식을 넘어서는 기계적인 계산이란게 있을까? 누가 알겠는가. 또 다른 24세의 학도. 어디선가 전혀 관계없을 듯 한 연구에 몰두하는 중에 튜링의 기계를 능가하는 컴퓨터를 암시하는 디자인을 슬쩍 펴보이는 일이 생길지. 희소식은 종종 의외의 곳에서 오지않던가. □