Turing 기계

1931 년 가을에 영국의 A. Turing(1912–1954)은 Cambridge 대학의 일지망이던 Trinity college[1]에서는 장학금을 받지 못하여 입학하지 않고, 이지망이던 King's college 에 장학생으로 입학한다. 그는 1933 년 가을 학기에 유명한 천체물리학자인 A. Eddington 교수의 강의를 듣다가, 강의 시간에 잘 이해되지 않던 부분에 대하여 연구 논문을 쓴다. 이것이 확률론과 통계학에서 유명한 Central Limit Theorem 인데 [Weaver], Turing은 이 정리가 1922 년에 핀랜드 수학자 Lindeberg에 의하여 이미 증명되었다는 것을 나중에야 알게 된다 [http://www.turing.org.uk], [Zabell]. Turing은 1935년에, 위상수학자이고 논리학에 관심을 가지던 Max Newman 교수의 강의에서 Hilbert 프로그램과 Gödel의 정리에 대한 소식을 듣는다. 그리고 Hilbert 의 판정문제(Entscheindungsproblem, Decision Problem)이 아직 미해결이라는 것도 배운다.

Turing은 달리기를 좋아 하였는데, 어느날 달리기를 마치고 쉬면서 "셈(헤아림)이란 무엇인가?"라는 질문에 대한 답을 떠올린다.[2]

셈이란 이성적인 사고를 뜻하는 것으로 그것을 구현되는 과정은 다음과 같다: 우선 마음이나 기계의 "상태"가 있어야 하고, 외부에서 들어오는 입력에 따라 상태가 바뀌고 출력이 나타나고, 다른 입력을 받아들이게 된다.

$$\begin{array}{rcccc} & & \dot{2} & \dot{4} & 6 \\ + & & 7 & 8 & 9 \\ \hline & 1 & 0 & 3 & 5 \end{array}$$

Turing은 모든 것을 단순화하여 입출력을 담당하는 것을 기다란 종이 테이프로 생각하였다. 이 테이프는 일정한 크기의 사각형 칸들로 연결되어 있고, 각 칸에는 문자가 쓰여 있다. 문자 집합도 단순하게

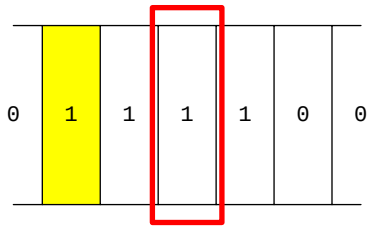
$$C := \{0, 1\}$$

로 두자. 그러면 문자를 읽고 쓰는 장치인 "헤더"가 칸을 읽고는 상태에 따라 (원래 있던 문자는 지우고) 0 또는 1 을 그 칸에 출력하고 다음 칸(오른쪽 칸) 또는 이전 칸(왼쪽 칸)으로 이동한다.

우리가 사용하는 문자 중 0 은 테이프의 빈 칸을 나타내는 것으로 사용한다.

[1] I. Newton이 다녔던 학교

[2] *On computable numbers, with an application to the Entscheidungsproblem*, Proc. London Math. Soc. **42** (1963), 230–65; a correction **43** (1937), 544–6. http://www.abelard.org/turpap2/tp2-ie.asp



이제 상태 집합을

$$S := \{s_1, s_1, \ldots, s_n\}$$

으로 두자. 상태집합은 반드시 **초기상태** s_1 을 포함하지만 **정지상태**(STOP, s_0)은 S 에 포함되지 않는 것으로 생각한다. 기계는 정지 상태에 이르면 작동을 멈춘다. 이러한 기계장치는 함수

$$M : S \times C \to C \times \{L, R\} \times (S \cup \{s_0\})$$

에 해당한다.

상태 집합의 기수(cardinal number)가 n 인 기계를 n-**상태 기계**라 부른다. n-상태 기계의 가지수는 모두

$$(4(n+1))^{2n}$$

이다. 물론 이 중에는 정상적으로 작동하는 것도 있고, 불량품도 있다.

기계들의 가지수는 가산수이다.

정상적인 기계 M 은 초기상태 s_1 에 있을 때 입력된 테이프의 "초기 위치"에서 작동하기 시작하여 정지상태 s_0 에 이르면 작동을 멈추고 임무를 완수하게 된다.

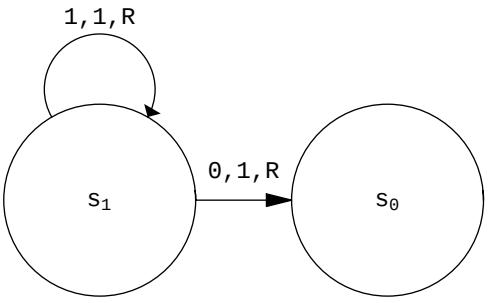
1 보기

1.1 더하기 일

보기를 들어 자연수 n 을 입력하였을 때 $n+1$ 을 출력하는 기계 $f(n) = n+1$ 은 다음과 같이 만들 수 있다.

	0	1
s_1	1 R s_0	1 R s_1

이러한 기계는 다음과 같은 그림으로 나타내기도 한다.



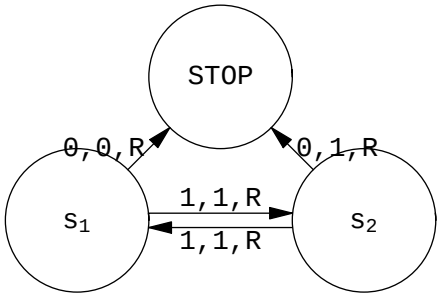
1.2 홀짝

다음은 입력한 1 의 개수가 홀수인지 또는 짝수인지를 판정하는 기계

$$f(n) = \mathrm{Mod}(n, 2)$$

이다. 이때 멈추고 나서 헤더의 왼쪽 칸에 나타난 수가 0 이면 짝수개라는 것을 뜻하고 1 이면 홀수개라는 것을 뜻한다.

	0	1
s_1	0 R s_0	1 R s_2
s_2	1 R s_0	1 R s_1



1.2.1 연습문제

1. 입력된 것을 지우고 홀짝을 판정하는 기계를 만들라.
2. 입력을 한 칸씩 오른쪽으로 이동시키는 기계를 만들라.

1.3 더하기 기계

두 자연수를 더하는 기계 $f(m, n) = m + n$ 을 만들어 보자.

	0	1
s_1	1 R s_2	1 R s_1
s_2	0 L s_3	1 R s_2
s_3	0 R s_0	0 R s_0

이 때, 오른쪽을 1 로, 왼쪽을 0 으로 표시하고, 상태 s_1, s_2, s_3, s_4, s_0 을 각각 $\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{0}$ 으로 표시하여, 위 기계를

$$\{\{1, 1, \mathbf{2}, 1, 1, \mathbf{1}\}, \{0, 0, \mathbf{3}, 1, 1, \mathbf{2}\}, \{0, 1, \mathbf{0}, 0, 1, \mathbf{0}\}\}$$

나타내면, 처음 원소는 상태 s_1 인 경우에 입력 0,1 에 대한 행동을 표시한 것이고, 마지막 원소는 상태 s_3 인 경우에 입력에 따른 행동을 표시한 것이다.

모든 기계는 이와 같이 영이상의 정수로 이루어진 유한 수열로 나타낼 수 있다.[3]

[3] 이러한 유한 수열 $\{n_1, n_2, n_3, \dots\}$ 은 소수열 $2, 3, 5, 7, 11, \dots$ 을 이용하여 자연수 $n = 2^{n_1} 3^{n_2} 5^{n_3} \cdots$ 으로 바꿀 수 있다.

1.3.1 연습문제

1. 위 더하기 기계는 n, m 이 모두 1 이상일 때에 작동한다. n, m 이 0 일 때에도 작동하도록 고쳐보라.

2. 다음 기계는 입력을 두배하는 것($f(n) = 2n$)임을 밝히라.

$$\{\{0,1,1,1,1,2\},\{1,1,0,0,1,3\},\{1,0,4,1,1,3\},\{1,1,1,1,0,4\}\}$$

3. Kronecker의 델타함수

$$\delta(n,m) = \begin{cases} 1 & (n = m) \\ 0 & (n \neq m) \end{cases}$$

을 나타내는 기계를 서술하라.

1.4 복사 기계

함수 $f(n) = (n, n)$ 을 나타내는 Turing 기계를 만들어 보자.

state	0	1
1	0 R 0	1 R 2
2	0 R 3	1 R 2
3	1 L 4	1 R 10
4	0 L 4	1 L 5
5	0 R 3	1 R 6
6	0 L 11	0 R 7
7	0 R 7	1 R 8
8	1 L 9	1 R 8
9	0 L 4	1 L 9
10	0 R 10	1 L 6
11	1 L 11	1 L 0

다음은 위 프로그램을 이용하여 3 을 복사하는 과정을 그린 것이다. 가장 왼쪽에 있는 수는 작업의 단계를 나타내는 것이고, 밑줄 있는 부분은 '헤더'가 읽기 직전의 상태를 말한다. 그리고 빈칸은 모두 0 인 것으로 간주한다.

steps		•							
1		1	1	1					
2		1	1	1					
3		1	1	1					
4		1	1	1	_				
5		1	1	1		_			
6		1	1	1	_	1			
7		1	1	1		1			
8		1	1	1		1			
9		1	1	1		1			
10		1	1		_	1			
11		1	1			1			
12		1	1			1	_		
13		1	1			1	1		
14		1	1		_	1	1		
15		1	1	_		1	1		
16		1	1			1	1		
17		1	1			1	1		
18		1	1			1	1		
19		1		_		1	1		
20		1			_	1	1		

steps								
21		1				1	1	
22		1				1	1	
23		1				1	1	_
24		1				1	1	1
25		1				1	1	1
26		1			_	1	1	1
27		1		_		1	1	1
28		1	_			1	1	1
29		1				1	1	1
30	_	1				1	1	1
31		1				1	1	1
32		1	_			1	1	1
33		1		_		1	1	1
34		1			_	1	1	1
35		1				1	1	1
36		1			_	1	1	1
37		1		_		1	1	1
38		1	_	1		1	1	1
39		1	1	1		1	1	1
40	_	1	1	1		1	1	1

1.4.1 연습문제

1. 복사기계를 이용하여 뒤집기 기계 $f(m,n)=(n,m)$ 을 만들라.
2. 시작하면 $1, 11, 111, 1111, \ldots$ 을 계속하여 찍어내는 기계를 만들라.[4] (여기에서 쉼표(,)는 빈칸을 나타낸다.)

1.5

Von Neumann은 복사기 위에 글쓰인 종이를 올려 두면 글쓰인 종이가 복사 되어 나오지만, 진정한 복사기란 복사기 자신을 복제하는 기계라야 한다는 생각을 하였다. Von Neumann은 자기 복제 기계를 1940년대에 프로그램하였다.[5] Von Neumann의 생각은 세포자동자(CA, Cellular Automata)로 발전하고 J. Conway가 생명게임(Game of Life)을 만들어 더욱 유명해진다. Mathematica라는 프로그램으로 유명한 S. Wolfram 은 1차원 CA를 연구·발전시켰다.

1.6 Busy Beaver Problem

이 문제는 1962년에 Tiber Rado가 처음으로 생각하였다. 함수 $B(n)$ 은 다음과 같이 정의한다. 문자 집합으로 $\{0,1\}$ 을 사용하는 n-상태 기계에 빈 테이프를 입력하여 기계가 작동을 멈추었을 때 출력된 1 의 개수를 생각하자. 이때 기계를 잘 만들어서 가장 1 의 개수가 많이 나오는 경우에 그 개수를 $B(n)$ 으로 정의한다.

$$B(1)=1,\ B(2)=4,\ B(3)=6,\ B(4)=13$$

[4] 앞의 것을 복사한 다음 1을 추가하는 기계를 만들면 된다.
[5] Watson 과 Crick 에 의한 DNA의 이중나선구조는 후일에 발견되었다.

임이 알려져 있지만, $n \geq 5$ 인 경우에 $B(n)$ 의 값은 알려져 있지 않다 [J. Casti], [Rosen], [Rucker].

1.6.1 연습문제

1. 다음 기계
$$M : (s_1, 0) \mapsto (1, R, s_0)$$
을 생각하고 $B(1) = 1$ 을 보이라.

2. 기계
$$M : \begin{cases} (s_1, 0) \mapsto (1, R, s_2) \\ (s_1, 1) \mapsto (1, L, s_2) \\ (s_2, 0) \mapsto (1, L, s_1) \\ (s_2, 1) \mapsto (1, R, s_0) \end{cases}$$
을 생각하고 $B(2) = 4$ 임을 보이라.

3. $B(3) = 6$, $B(4) = 13$ 임을 보이라.

2 계산 가능 수

Turing은 실수 r 이 "계산가능하다"는 것을 다음과 같이 정의하였다: 기계를 잘 만들면 r 을 이진법으로 표현하였을 때, 임의의 자리수 n 에 대하여 소수점 아래 처음부터 n번째 자리까지 정확하게 찍어낼 수 있다.

이러한 정의에 의하면 대수적 수는 모두 **계산 가능 수**임을 알 수 있다. 원주율 π 나 자연상수 e 도 모두 계산가능 수이다.

2.1 Babylonian Algorithm 과 개평법

cf. Euler, Algebra (세제곱근)

2.1.1

Turing 기계를 만들 수 있는 방법은 가산개이고, 각 기계가 유한 시간 동안 셈을 하고 결론을 알려 주는 방법도 가산개이므로 결국 계산 가능수는 가산개이다. 그러므로 계산 불가능한 실수가 대부분이라는 것을 알 수 있다.

3 Church-Turing Thesis

Turing의 기계는 "알고리즘(algorithm)" 또는 "프로그램"이라 불러도 된다.[6] 현재 우리 주위에 있는 컴퓨터들은 모두 Turing 기계라고 보아도 좋다.[7]

[6] 825년경에 페르시아의 al-Khowarizmi 가 인도의 십진법을 사용하여 산술의 기본연산을 하는 법에 대한 책을 썼다. 알고리즘이라는 말은 그의 이름에서 유래한다. "al-Khowarizmi"는 "Khowarizm 에서 온 사람"이라는 뜻이다 [Schwartzman], [Knuth].

[7] 기억 용량이 부족할 경우에는 더 보탤 수 있다고 가정한다.

Turing이 생각한 셈 이외에도 K. Gödel의 회기함수(recursive function), A. Church의 λ-calculus, E. Post의 기계 등 여러가지 다른 방법이 발표되었으나, 다행히(?) 이들의 생각은 모두 동치임이 밝혀졌다.

Church-Turing의 주장에 의하면 어떠한 셈도 Turing Machine이 할 수 있다. 이 주장은 "모든 생각은 결국 셈을 통하여 얻는다"는 Leibniz의 생각을 보완한 것이다.

4 보편 Turing 기계

Turing은 특수 목적만을 위한 기계뿐 아니라 일반 목적을 위하여 사용할 수 있는 보편기계(Universal Machine)를 생각하였다. 이러한 기계는 특수 목적을 위한 기계의 프로그램과 입력을 읽은 다음, 마치 자신이 특수 목적 기계처럼 행동한다.

5 정지문제

Turing 은 다음과 같은 문제를 생각을 하였다.

> **정지문제(Halting Problem)**: 기계를 잘 만들어 임의의 기계가 임의의 입력에 대하여 정지하는지 또는 정지하지 않는지를 (유한 시간안에) 판정할 수 있을까?

Turing은 정지문제의 답이 불가능이라는 것을 "칸토르의 대각화 방법"을 이용하여 다음과 같이 증명하였다.

어떤 기계 U 가 "기계 M 과 입력 n"을 입력하면 "M 이 입력 n 에 대하여 정지하는지 또는 정지하지 않는지"를 판정한다고 하자. 이제

$$U(M,n) := \begin{cases} 1, & M \text{ 이 } n \text{ 에 정지할 경우} \\ 0, & M \text{ 이 } n \text{ 에 정지하지 않을 경우} \end{cases}$$

로 정의하자. 기계의 가지수는 가산수이고, 따라서 U 가 판정할 수 있는 기계를 나열하여 $M_1, M_2, \ldots$ 등으로 두면 $U(M,n)$ 의 표는, 예를 들어, 다음과 같을 수 있다.

n	1	2	3	$\cdots$
M_1	0	1	1	$\cdots$
M_2	1	1	0	$\cdots$
M_3	0	0	1	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	

이제 다음과 같은 기계 M_* 를 생각하자: M_* 는 입력 n 에 대하여,

- $U(M_n, n) = 0$ 이면 정지하고
- $U(M_n, n) = 1$ 이면 정지하지 않는다.

그러면 기계 M_* 는 위 도표에 나오지 않고, 따라서 U 가 판정할 수 없는 기계이다.[8]

그러므로 "모든 기계가 입력에 따라 정지하는지 또는 아닌지"를 판정하는 기계는 존재하지 않는다.

6 Turing 기계와 형식 체계

Turing 기계와 형식체계는 일대일 대응을 이룬다.

Turing 기계	형식 체계
입력	공리
프로그램	추론 규칙
출력	정리

그러므로 Hilbert 의 Entscheidungsproblem 의 답은 "불가능"이다.

7 확장이진법

지금까지 우리가 살펴본 기계는 0 과 1 만을 입출력 기호로 사용하고, 더군다나 0 은 테이프의 빈공간까지 나타내고 있다. 따라서 입력은 기본적으로 1 만을 사용하였는데, 이러한 "일진법"식의 기수법은 방법은 지극히 비효율적인 것이다.

우리가 일상적으로 사용하는 인도-아라비아 기수법(記數法)은 십진법이다. 이러한 진법에서 "백(百, C)"을 나타내는데 사용하는 숫자의 개수는 백개가 아니고 $1+2=1+\log_{10}100$ 이며, "천(千, M)"을 나타내는 데 사용하는 숫자의 개수는 $1+3=1+\log_{10}1000$ 이다. 십진법을 사용하지 않고 일진법으로 이러한 수들을 입력한다면 아주 비효율적인 것이라는 것을 알 수 있다.

일반적으로 자연수 n 을 b-진법으로 표현하는데 사용되는 숫자의 개수는, $b>1$ 일 때, $O(\log n)$ 이다.[9] 진법의 밑수가 2 이상이기만 하면, 기수법에는 본질적인 차이가 없다.

그러므로 우리의 기계에서도 일진법식 표현을 사용하지 않고 이진법 또는 그 이상의 진법을 사용하는 것이 바람직함은 더 말할 것도 없다.

그러나 이진법에서는

$$11,\ 101,\ 1001,\ 10001,\ 100001,\ 1000001,\ \ldots \tag{1}$$

등의 표현을 가지는 수들이 한없이 많이 있고, 따라서 기계는 "과연 언제 입력이 끝나는가?"를 알 수 있는 방법을 가져야만 한다. 이 문제를 해결하기 위하여 이진법을 확장한 "확장 이진법"을 사용하기로 한다. "이진법수"에서 1 을 모두 '10'으로 바꾸면 "확장이진법수"가 된다. 그러므로 (1) 을 확장이진법으로 표현하면

$$1010,\ 10010,\ 100010,\ 1000010,\ 10000010,\ 100000010,\ \ldots$$

[8] U 를 책으로 보고, 이 책의 제1쪽에는 기계 M_1 이 정상 작동하여 정지하는 입력들의 수를 모두 적어두고, 일반적인 n 쪽에는 기계 M_n 이 정상 작동하여 정지하는 입력들의 수를 모두 적어둔다고 하자. 이제 자연수 n 에 n 쪽에 나타나지 않는 그러한 n 만으로 이루어진 쪽지는 U 라는 책에는 어느 쪽에도 나타나지 않는다. 즉, U 가 판정하지 못하는 쪽지를 발견한 셈이다.

[9] 정확하게는 $1+\lfloor\log_b n\rfloor$ 이다.

이 된다. 확장이진법수는 1 이 연속적으로 반복되지 않는다. 그러므로 일진법수 11, 111, 1111, ... 등을 다른 용도로 사용할 수 있다. 예를 들면 11 을 "쉼표"를 나타내는 기호로 하고, 111 을 "입력의 끝"을 나타내는 기호로 할 수 있다.

이러한 방법에서는 입력뿐 아니라 기계의 상태 집합과 작동 지침까지 서술할 수 있고, 따라서 모든 튜링기계를 0 과 1 의 유한 수열로 표현할 수 있다 [Penrose].

8 Chomski 와 언어구조

N. Chomski 는 형식체계에서 기호와 공리의 역할이 언어체계에서 기호와 문법의 역할과 마찬가지라는 것을 보고, 문법을

- regular grammar
- context-free grammar
- context-sensitive grammar
- phrase-structure grammar

등으로 분류하였다. 이들은 모두 Turing 기계의 분류라고 말 할 수 있다.

9 Turing 테스트

Turing 은 "언제 기계가 지능을 가졌다고 말할 수 있을까?"라는 질문에 다음과 같이 답하였다. 우선 기계와 사람을 커튼 뒤에 숨겨 두고, 양쪽에 키보드에 입력한 질문을 하여 그 답을 화면으로 받아 본다. 이러한 작업을 몇 번하여 질문자가 기계를 사람이라고 판정한다면, 그 기계는 지능을 가졌다고 말할 수 있다. 이러한 테스트를 "Turing 테스트" 또는 "모방 게임"이라고 한다.

10 계산복잡도

Turing 기계에 입력 n 을 하였을 때 기계가 작업을 마칠 때까지 움직이는 회수를 그 작업의 **시간 복잡도**라 한다. 만약 기계가 멈추지 않으면 시간 복잡도는 무한대이다.

앞에서 살펴본 복사기계

$$n \mapsto (n, n)$$

는 입력 n 에 대하여 작동하는 시간—즉, 헤더가 움직이는 회수가 $2(n^2 + 3n + 2)$ 임을 어렵지 않게 보일 수 있다. 이와 같이 작동 시간이 입력한 자료의 "비트(bit)" 수에 대한 다항식으로 주어지는 기계(또는 알고리즘)을 **다항시간 기계**(또는 다항시간 알고리즘)이라 부른다.

어떠한 작업을 완수하는 데 시간이 얼마나 많이 걸리는가 하는 문제는 기계에 따라 다를 수 있다. 문자 집합을 세개를 사용하는 기계도 있고, 입력 장치로 평면 격자를 사용할 수도 있다. 마치 우리가 복잡한 셈을 할 때, 사용하는 연습장처럼. 기계에 의존하지 않는 시간의 개념이 바로 다항 시간이라는 개념이다.

11 집단문제

11.1 다항시간 문제

"집단 문제(mass problem)"를 풀이하는 다항시간 알고리듬이 존재하면 그 문제를 **다항시간 문제**라고 부른다.

11.1.1 정돈 문제

예를 들어 주어진 자료 $(n_1, n_2, \ldots, n_k)$ 를 작은 것부터 차례로 나열하는 알고리즘을 생각하여 보자. 이것은 평범하게 프로그램을 짜면 많아야 $n(n+1)/2$ 시간 안에 해결할 수 있다. 그러므로 정돈(sorting) 문제는 다항시간 문제이다.

11.1.2 유클리드 알고리듬

자연수 쌍 (m, n) 의 최대공약수를 구하는 문제도 다항시간 문제이다. Lamé의 정리에 의하면 유클리드 알고리듬의 시간 복잡도는 입력한 비트수에 비례한다 [Rosen].

11.1.3 행렬 연산

두 행렬을 곱하거나 행렬식을 구하는 문제 등도 다항시간 문제이다.

11.2 미정다항시간 문제

"4,003,997 은 소수인가요?"라는 질문에 답을 하려면 무척 오랜 시간이 걸린다. 하지만 "$4,003,997 = 1999 \times 2003$ 입니까?"라는 질문에 답하는 것은 쉽다. 이러한 문제처럼 그 풀이는 쉽지 않더라도 일단 정답으로 제시하는 것이 진정한 답인지를 **검증**하는 데에 다항시간이 걸리는 문제를 미정다항시간(nondeterministic polynomial time) 문제라 부른다.

11.3 P 와 NP

다항시간 집단 문제 전체 집합을 P 라 하고, 미정다항시간 집단문제 전체 집합을 NP 라 두자. 그러면 $\mathrm{P} \subset \mathrm{NP}$ 임은 분명하다. 만약 "$\mathrm{P} = \mathrm{NP}$ 인가?"라는 질문에 답을 하면, Clay 수학연구소에서 100만불 현상금을 탈 수 있다.[김]

12 Chaitin 과 Algorithmic Information Theory

IBM 의 G. Chaitin(체이틴)은 1960년대초에 Columbia 대학에서 우수한 고등학생들을 위한 Computer Programming 강의에 참가하였다. 강의에서 교수는 "프로그램을 짜면서 문제를 푸는 것"을 연습시켰다. 프로그램을 짜는 일은 지루하고 재미 없는 것이 보통이지만 학생들에게 가장 짧은 프로그램을 이용하여 문제를 푸는 것으로 경쟁시켜 강의의 활기를 돋게 하였다. 하지만 Chaitin 은 "매주 우승하는 프로그램이 존재할 수 있는 프로그램 중 가장 짧은 것이라는 것을 어떻게 증명할까?" 하는 것을 아무도 생각한 적이 없다고 말하였다. 1965년 City University of New York (CUNY) 의 학생이던 Chaitin 은 수의

복잡성(complexity)을 정의하였다. (cf. J. Casti, Five MORE Golden Rules, p. 246.) 그는 어떤 수열을 더 이상 짧은 표현으로 서술할 수 없을 때 random 이라 정의하였다. Chaitin 은 기계나 형식 체계의 **정보량**(complexity, algorithmic information content)을 정의하였고, 이를 이용하여 정보량이 일정한 형식 체계가 판정할 수 있는 것의 한계를 설명하였다. 이러한 아이디어는 Richard-Berry-Russell의 역설을 풀이하여 얻은 것이다.

13 Continuous Turing Machine

디지털 기계 대신에 아날로그 기계를 개발하여야 한다?

참고문헌

김홍종, *길거리의 행인을 위한 백만불 현상문제 소개: P = NP ?*, 대한수학회 소식지, 2001.1. (http://www.math.snu.ac.kr/~hongjong/잡필/PversusNP/PversusNP.html)

J. Casti, *Five Golden Rules*, John Wiley & Sons, Inc., 1996. (한태식, 권기호, 김정헌 옮김, "20세기 수학의 다섯가지 황금율" 경문사, 1999.)

K. Devlin, *Mathematics: The New Golden Age*, Penguin Group Ltd., London, 1998. (허민 옮김, *수학: 새로운 황금시대*, 경문사, 1995.)

R. Feynman, *Lectures on Computation*, Addison Wesley, 1996.

V. Klee and S. Wagon, *Unsolved Problems*, Math. Assoc. Amer., 1991.

D. Knuth, *Fundamental Algorithm, The Art of Computer Programming*, Addison-Wesley, 1968, 1973.

J. Peterson, *The Mathematical Tourist*, Freeman, 1988.

____, *Islands of Truth, A Mathematical Mystery Cruise*, Freeman, 1990.

R. Penrose, *Shadows of Mind*, Oxford Univ. Press, 1994.

J. Peterson, *The Mathematical Tourist*, Freeman, 1988.

____, *Islands of Truth, A Mathematical Mystery Cruise*, Freeman, 1990.

K. Rosen, *Discrete Mathematics and its applications*, McGraw-Hill, Inc., 1988, 1995.

R. Rucker, *Mind Tools*, 1998. (김량국 옮김, 사고 혁명, 열린 책들, 2001)

S. Schwartzman, *The Words of Mathemtaics*, Math. Assoc. Amer., 1994.

S. Smale, *The Theory of Computation*, Mathematical Research — Today and Tomorrow, Springer-Verlag, Lect. Notes in Math. 1525 (1991), 59–69.

____, *Mathematical Problems for the Next Century*, in Mathematics: Frontiers and Perspectives, V. Arnold, M. Atiyah, P. Lax, and B. Mazur, editors, Amer. Math. Soc. (2000), 271–294.

A. Turing, *Computing Machinery and Intelligence*, Mind, 1950.

W. Weaver, *Lady Luck: The Theory of probability*, Dover, 1963.

S. Zabell, *Alan Turing and the Central Limit Theorem*, Amer. Math. Monthly (1995), 483–494.