Why is Computer Science a Science?

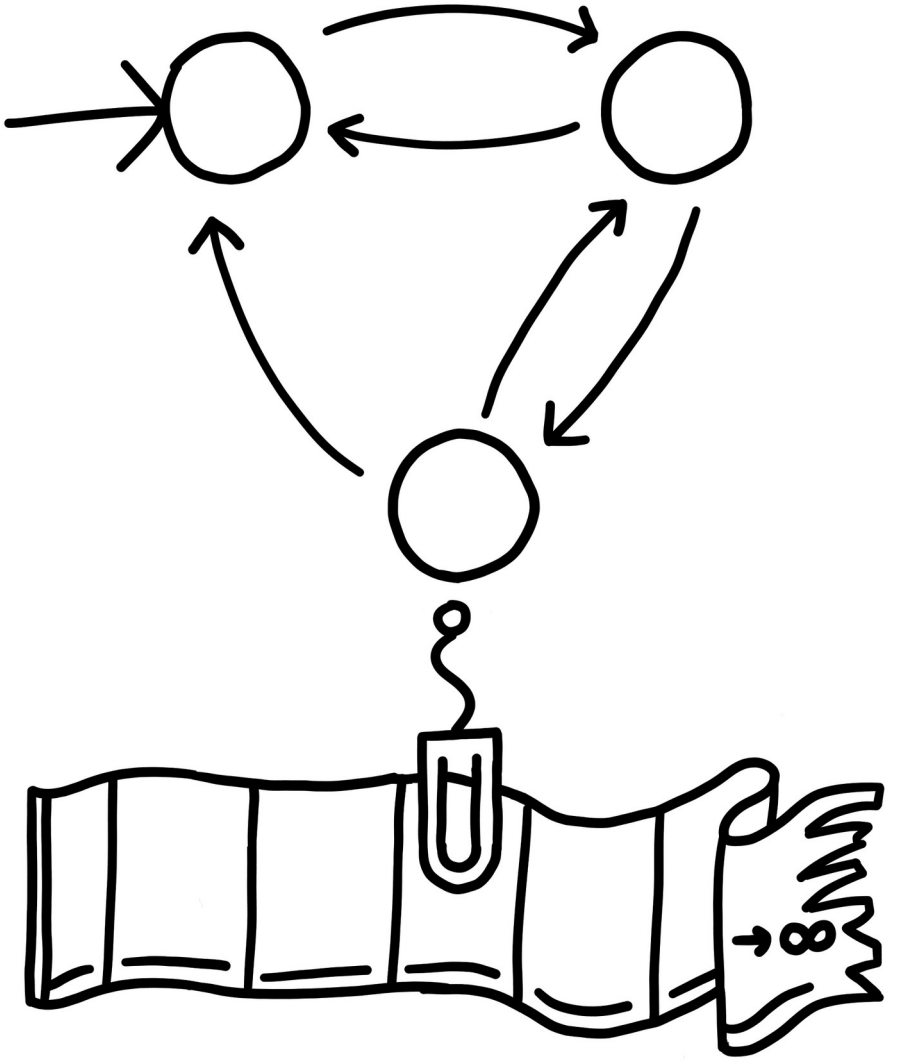
Abrahim Ladha

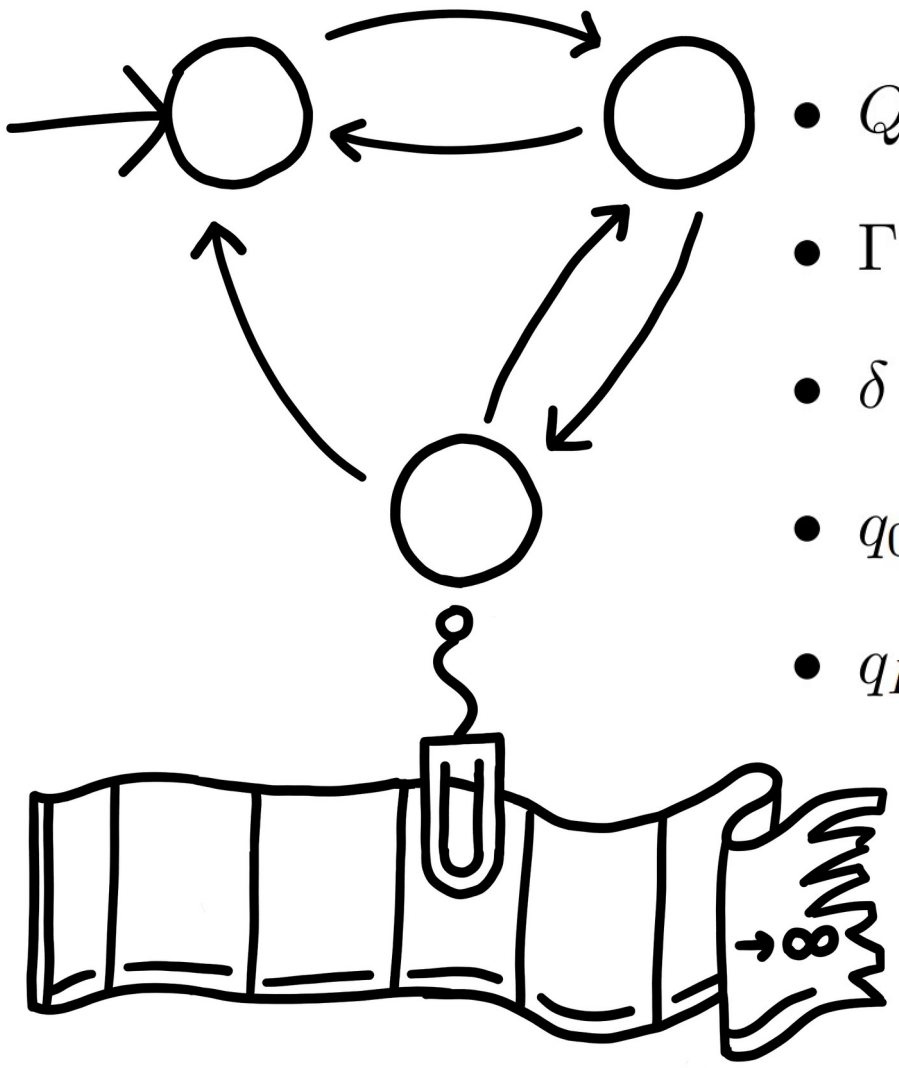
Goals For Today

- Why do we consider Alan Turing the founder of Computer Science?
- Why is Computer Science a Science, and not an Engineering?

Turing Machine

Single steps of conditional Read, Write, Move





A Turing Machine is a tuple $(Q, \Gamma, \delta, q_0, q_H)$

- $Q = \{q_0, ..., q_k\}$ is a finite set of states
- $\Gamma = \{0, 1, \sqcup\}$ finite tape alphabet
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
- $q_0 \in Q$ designated start state
- $q_H \in Q$ is the instruction to stop

LEMMA 1. *If $\theta_A(i_0)$ is defined, then $\theta_A(i_0) \in \{i_0\}_A$, $\theta_A(i_0) > 2i_0$.*

PROOF. If $\theta_A(i_0)$ is defined, then there is some value of t such that $T^A(i_0, K(t), L(t))$, $\theta(i_0) = K(t)$, and $\theta_A(i_0) > 2i_0$. Hence,

$$\bigvee_y T^A(i_0, \theta_A(i_0), y); \text{ that is, } \theta_A(i_0) \in \{i_0\}_A.$$

LEMMA 2. *If $\{i\}_A$ is infinite, then $\{i\}_A \cap S^A \neq \phi$.*

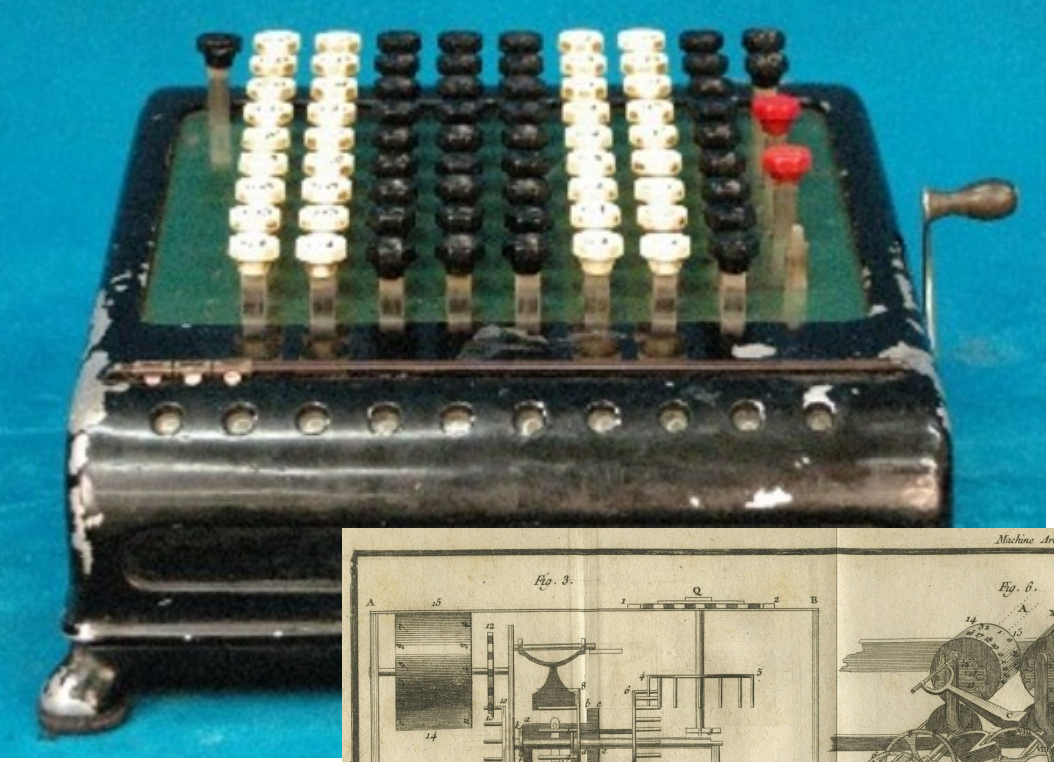
PROOF. If $\{i_0\}_A$ is infinite, there is a number m_0 such that $m_0 \in \{i_0\}_A$ and $m_0 > 2i_0$. Hence, $\bigvee_y T^A(i_0, m_0, y)$. Let y_0 be the least such y. (Actually, by the definition of the T-predicates there is at most one such y.) Then $T^A(i_0, m_0, y_0)$. Let $t_0 = J(m_0, y_0)$. Then

$$m_0 = K(J(m_0, y_0)) = K(t_0); \; y_0 = L(J(m_0, y_0)) = L(t_0).$$

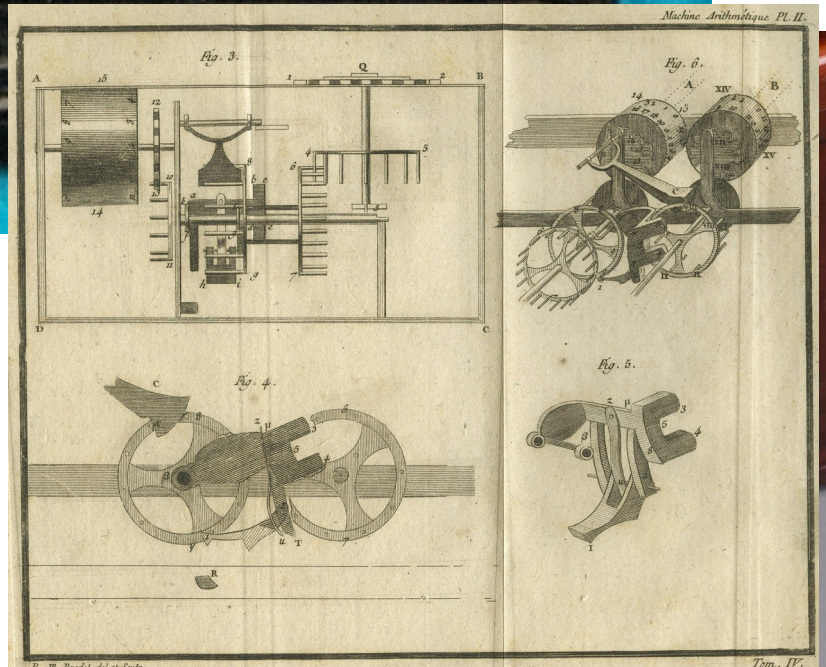
Hence, $T^A(i_0, K(t_0), L(t_0))$. Since $K(t_0) = m_0 > 2i_0$, $\theta_A(i_0)$ is defined. Hence, by Lemma 1, $\theta_A(i_0) \in \{i_0\}_A$. But, $\theta_A(i_0) \in S^A$. Hence, $\{i_0\}_A \cap S^A \neq \phi$.

For a set of recursion equations for F^* consists of the recursion equations for F together with the equations,

$$i_2(1, 2) = 2,$$
$$i_2(S(x), 2) = 1,$$
$$i_2(x, 1) = 3,$$
$$i_2(x, S(S(y))) = 3,$$
$$j_2(1, y) = y,$$
$$j_2(S(x), y) = x,$$
$$g_2(x, 1) = i_2(f_2(x, 1), 2),$$
$$g_2(x, S(y)) = i_2(f_2(x, S(y)), g_2(x, y)),$$
$$h_2(S(x), y) = x,$$
$$h_2(g_2(x, y), x) = j_2(g_2(x, y), y),$$
$$f_1(x) = h_2(1, x),$$



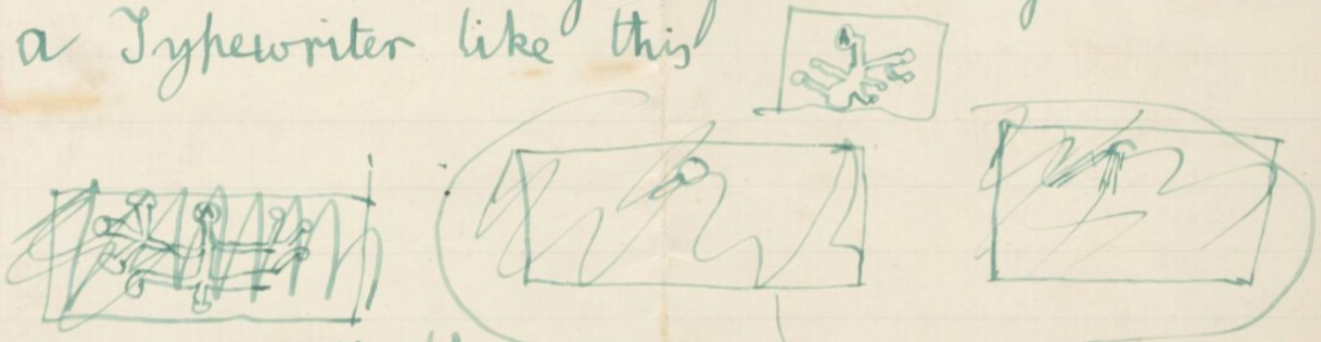

ROYAL


Machine Arithmétique Pl. II.
Fig. 3.
Fig. 6.
Fig. 4.
Fig. 5.
Tom. IV.

[3.a]

[No 3a]

It is luck I don't have to show up my letters* or Mr Darlington would conversecate the whole bottle# probably. This week I thought of how I might make a Typewriter like this



you see the the funny little rounds are letters cut out on one side slide along to the round (A) and along an ink pad and stamp down and make the letter, thats not nearly all though

Love from

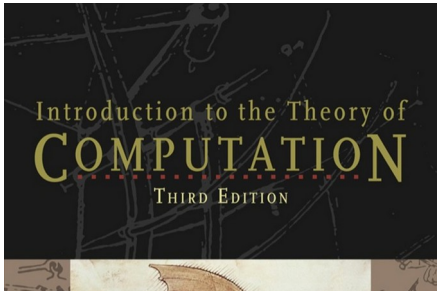
Alan

What is the Church-Turing Thesis?

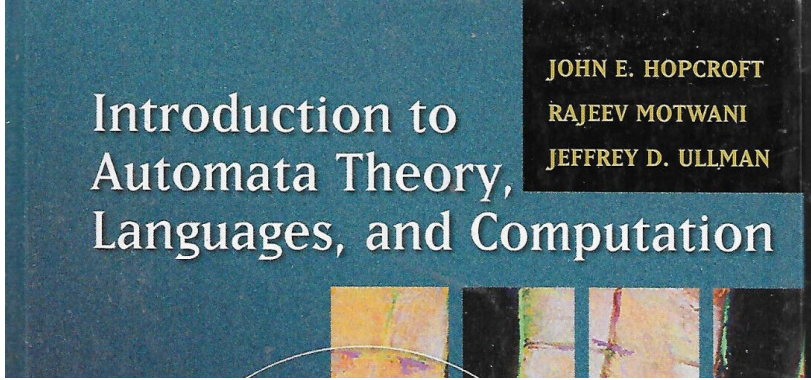
- “Algorithm” is an informal, intuitive notion of a process. A set of instructions to complete some task.
- A Turing Machine is a formal definition of a model of computation
- The Church-Turing Thesis asserts that the Turing machine captures the intuitive definition. That everything computable in the intuitive sense is computable by a Turing machine. But why?

Informally speaking, an *algorithm* is a collection of simple instructions for carrying out some task. Commonplace in everyday life, algorithms sometimes are called *procedures* or *recipes*.

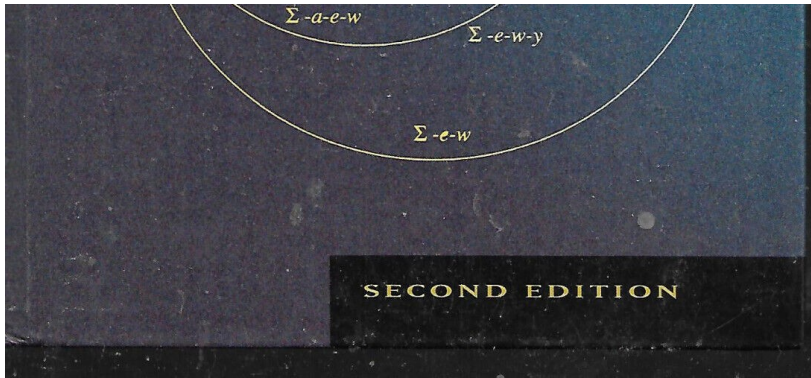
Even though algorithms have had a long history in mathematics, the notion of algorithm itself was not defined precisely until the twentieth century. Before that, mathematicians had an intuitive notion of what algorithms were, and relied upon that notion when using and describing them. But that intuitive notion was insufficient for gaining a deeper understanding of algorithms.



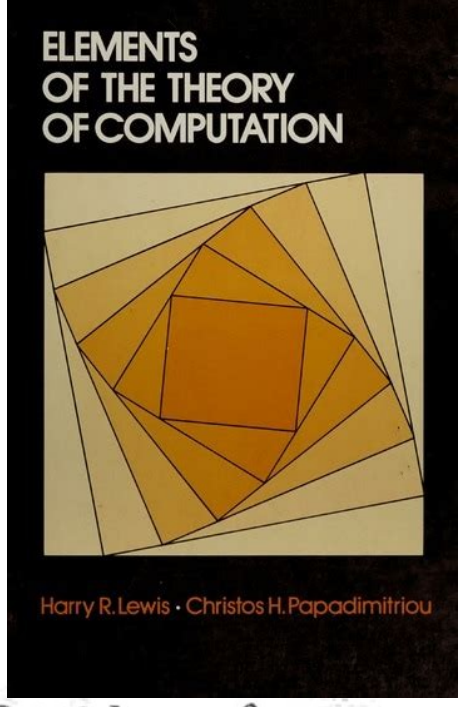
The definition came in the 1936 papers of Alonzo Church and Alan Turing. Church used a notational system called the λ-calculus to define algorithms. Turing did it with his "machines." These two definitions were shown to be equivalent. This connection between the informal notion of algorithm and the precise definition has come to be called the ***Church–Turing thesis***.



Interestingly, all the serious proposals for a model of computation have the same power; that is, they compute the same functions or recognize the same languages. The unprovable assumption that any general way to compute will allow us to compute only the partial-recursive functions (or equivalently, what Turing machines or modern-day computers can compute) is known as *Church's hypothesis* (after the logician A. Church) or the *Church-Turing thesis*.



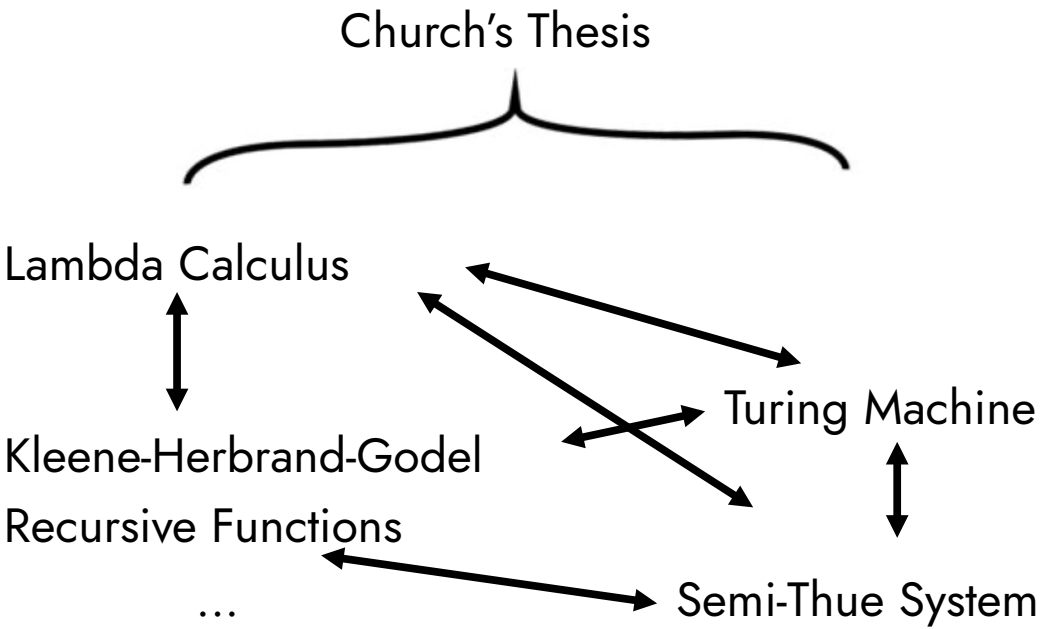
The situation is quite analogous to that met whenever one attempts to replace a vague concept, having a powerful intuitive appeal, with an exact mathematical substitute. (An obvious example is the area under a curve.) In such a case, it is, of course, pointless to demand a mathematical proof of the equivalence of the two concepts; the very vagueness of the intuitive concept precludes this.



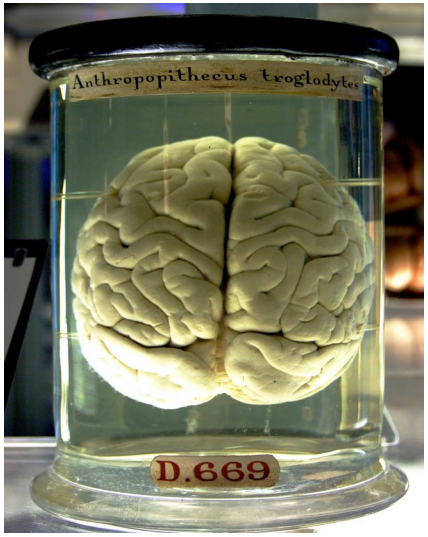
Turing machines can be imitated by grammars,
which can be imitated by μ-recursive functions,
which can be imitated by Turing machines.

The only possible conclusion is that *all these approaches to the idea of computation are equivalent*. This is Church's Thesis, extended now to methods quite different from those of the theory of automata.

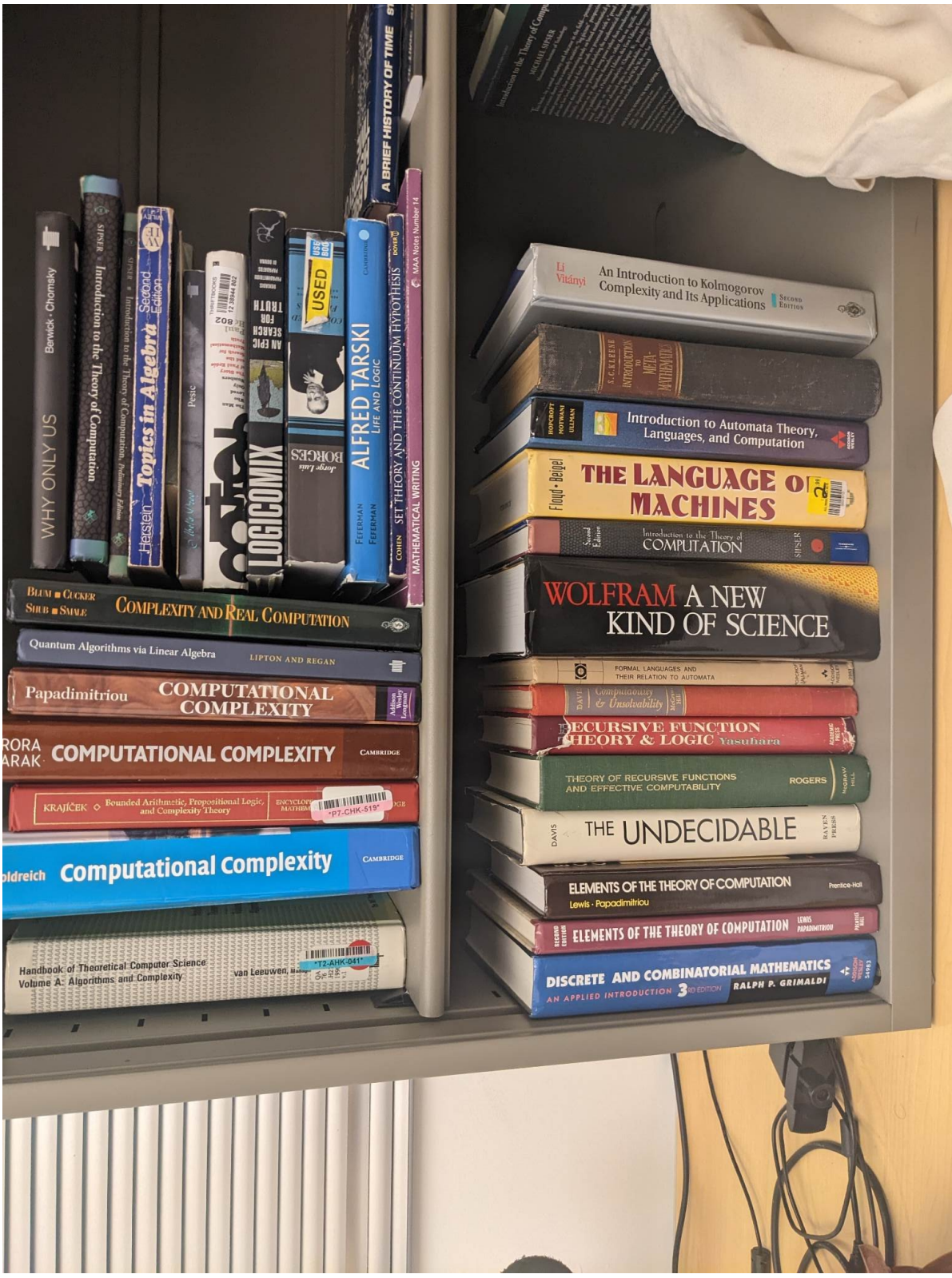
considering one aspect of computation by


Church's Thesis
Lambda Calculus
Turing Machine
Kleene-Herbrand-Godel
Recursive Functions
...
Semi-Thue System

=????


Anthropopithecus troglodytes
D.669

Human Mind


A BRIEF HISTORY OF TIME
WHY ONLY US
Berwick · Chomsky
Introduction to the Theory of Computation
SIPSER
Topics in Algebra
Second Edition
Herstein
Pesic
AN EPIC SEARCH FOR TRUTH
LOGICOMIX
USED
Jorge Luis BORGES
ALFRED TARSKI
LIFE AND LOGIC
FEFERMAN
SET THEORY AND THE CONTINUUM HYPOTHESIS
COHEN
MAA Notes Number 14
MATHEMATICAL WRITING
BLUM · CUCKER
SHUB · SMALE
COMPLEXITY AND REAL COMPUTATION
Quantum Algorithms via Linear Algebra
LIPTON AND REGAN
Papadimitriou
COMPUTATIONAL COMPLEXITY
RORA
ARAK
COMPUTATIONAL COMPLEXITY
CAMBRIDGE
KRAJÍČEK
Bounded Arithmetic, Propositional Logic, and Complexity Theory
Computational Complexity
CAMBRIDGE
Handbook of Theoretical Computer Science
Volume A: Algorithms and Complexity
van Leeuwen
Li Vitányi
An Introduction to Kolmogorov Complexity and Its Applications
SECOND EDITION
Introduction to Automata Theory, Languages, and Computation
HOPCROFT
MOTWANI
ULLMAN
Floyd · Beigel
THE LANGUAGE OF MACHINES
Introduction to the Theory of COMPUTATION
SIPSER
WOLFRAM A NEW KIND OF SCIENCE
FORMAL LANGUAGES AND THEIR RELATION TO AUTOMATA
DAVIS
Computability & Unsolvability
RECURSIVE FUNCTION THEORY & LOGIC Yasuhara
THEORY OF RECURSIVE FUNCTIONS AND EFFECTIVE COMPUTABILITY
ROGERS
DAVIS
THE UNDECIDABLE
ELEMENTS OF THE THEORY OF COMPUTATION
Lewis · Papadimitriou
Prentice-Hall
SECOND EDITION
ELEMENTS OF THE THEORY OF COMPUTATION
LEWIS
PAPADIMITRIOU
DISCRETE AND COMBINATORIAL MATHEMATICS
AN APPLIED INTRODUCTION 3RD EDITION
RALPH P. GRIMALDI

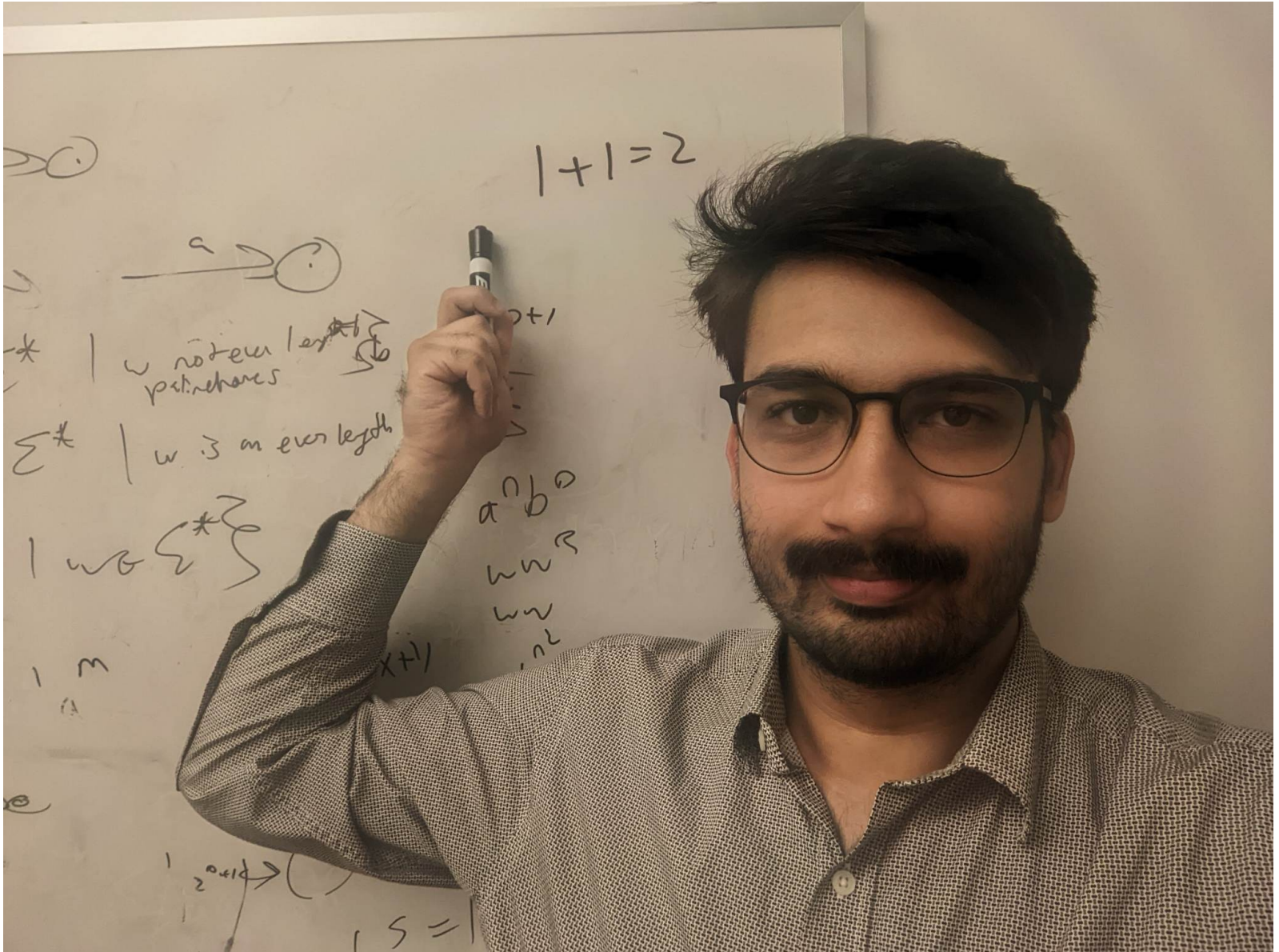
ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

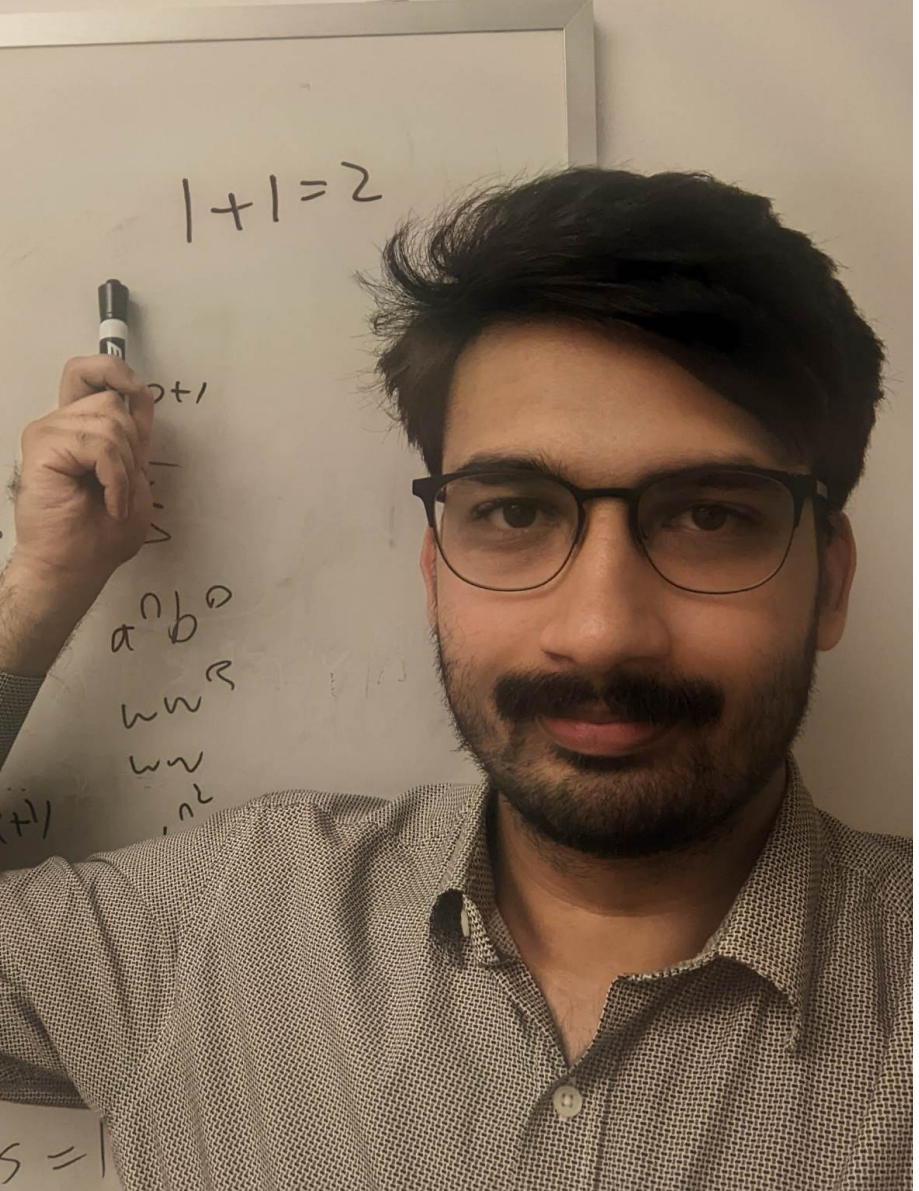
By A. M. TURING.

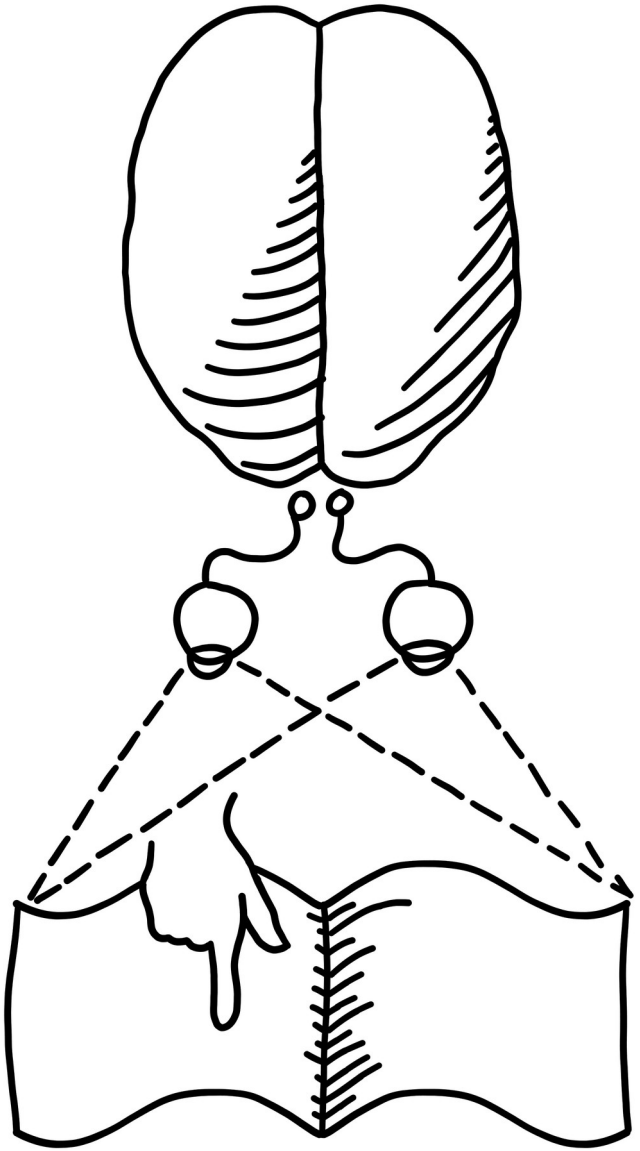
[Received 28 May, 1936.—Read 12 November, 1936.]

The Direct Appeal to Intuition

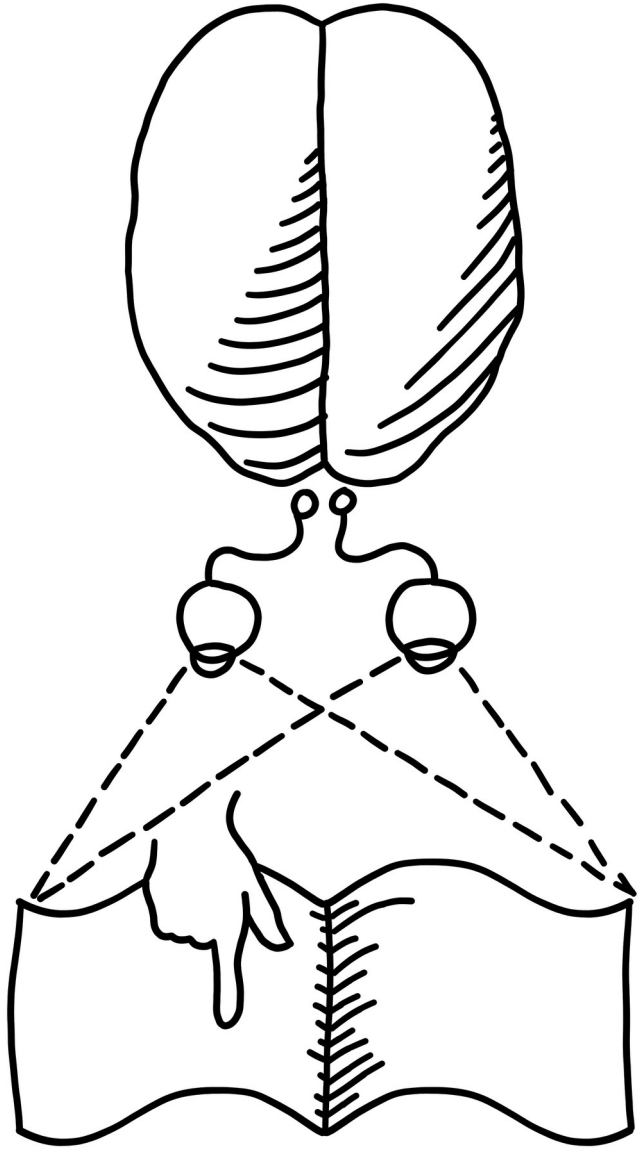
- First we will agree on an example of a computer, an object performing the action of computation
- We will make slight modifications so small to be undeniable
- We will argue that the composition of these modifications is still correct and conclude

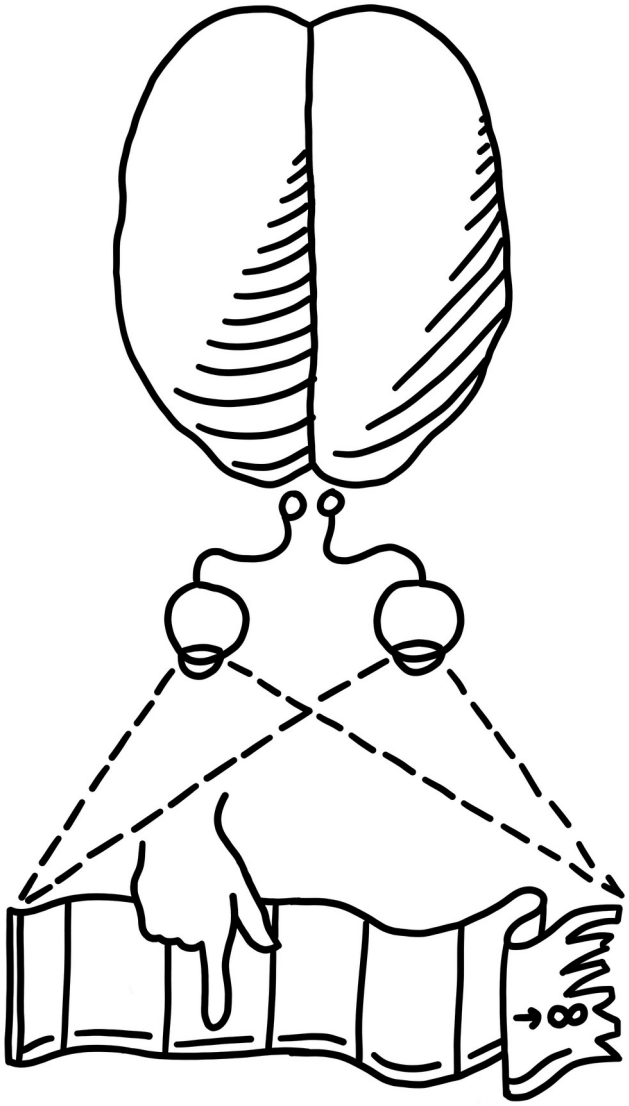

1+1=2


1+1=2



Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a child's arithmetic book. In elementary arithmetic the two-dimensional character of the paper is sometimes used. But such a use is always avoidable, and I think that it will be agreed that the two-dimensional character of paper is no essential of computation. I assume then that the computation is carried out on one-dimensional paper, *i.e.* on a tape divided into squares.

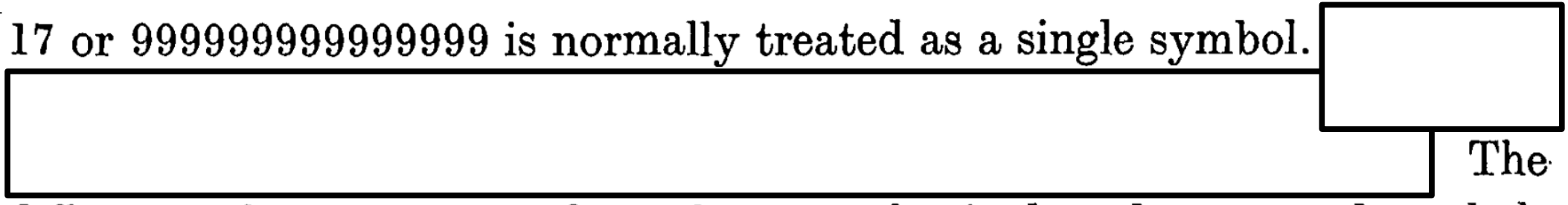




I shall also suppose that the number of symbols which may be printed is finite. If we were to allow an infinity of symbols, then there would be symbols differing to an arbitrarily small extent†. The effect of this restriction of the number of symbols is not very serious. It is always possible to use sequences of symbols in the place of single symbols. Thus an Arabic numeral such as

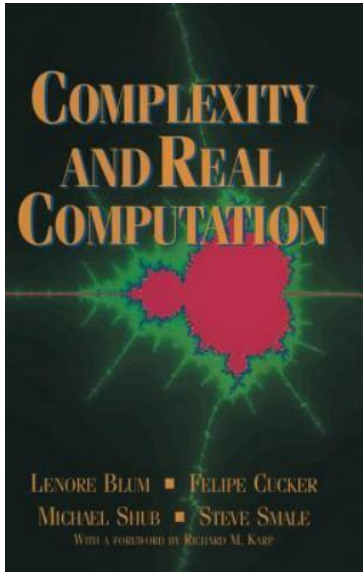
---

17 or 99999999999999 is normally treated as a single symbol.

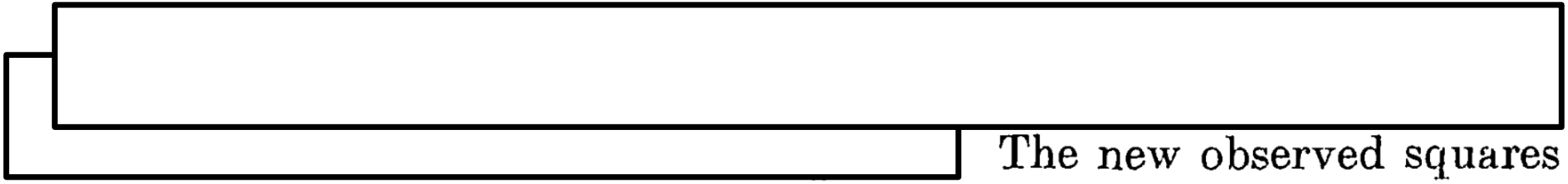


The differences from our point of view between the single and compound symbols is that the compound symbols, if they are too lengthy, cannot be observed at one glance. This is in accordance with experience. We cannot tell at a glance whether 9999999999999 and 99999999999999 are the same.

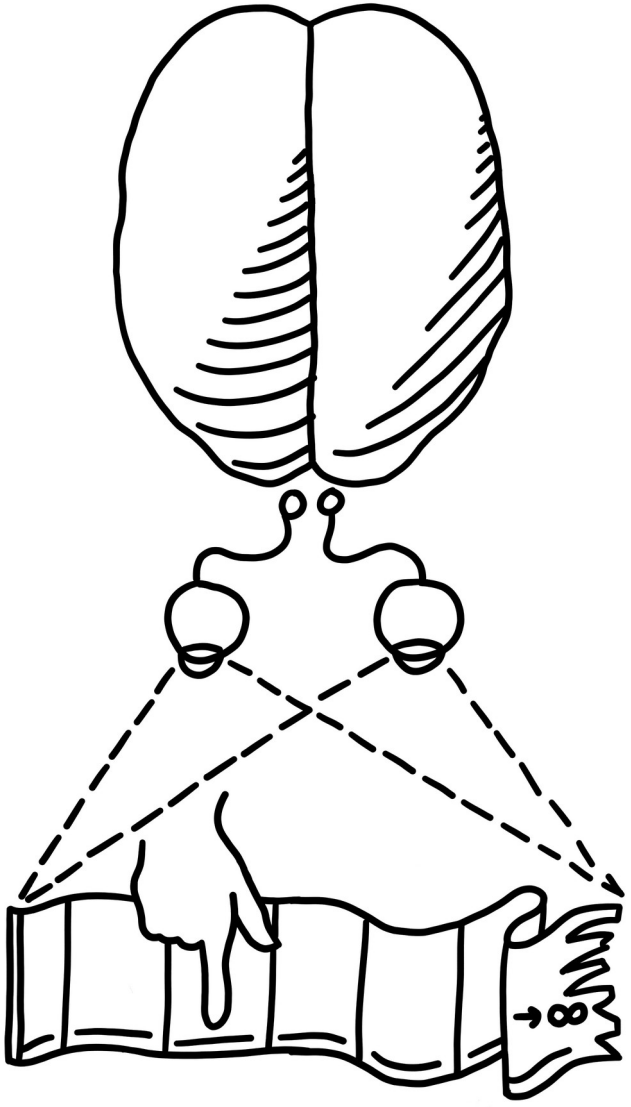
- Finite work is done in finite time, important for the development of computational complexity
- The amount of symbols necessary for computation must be finite

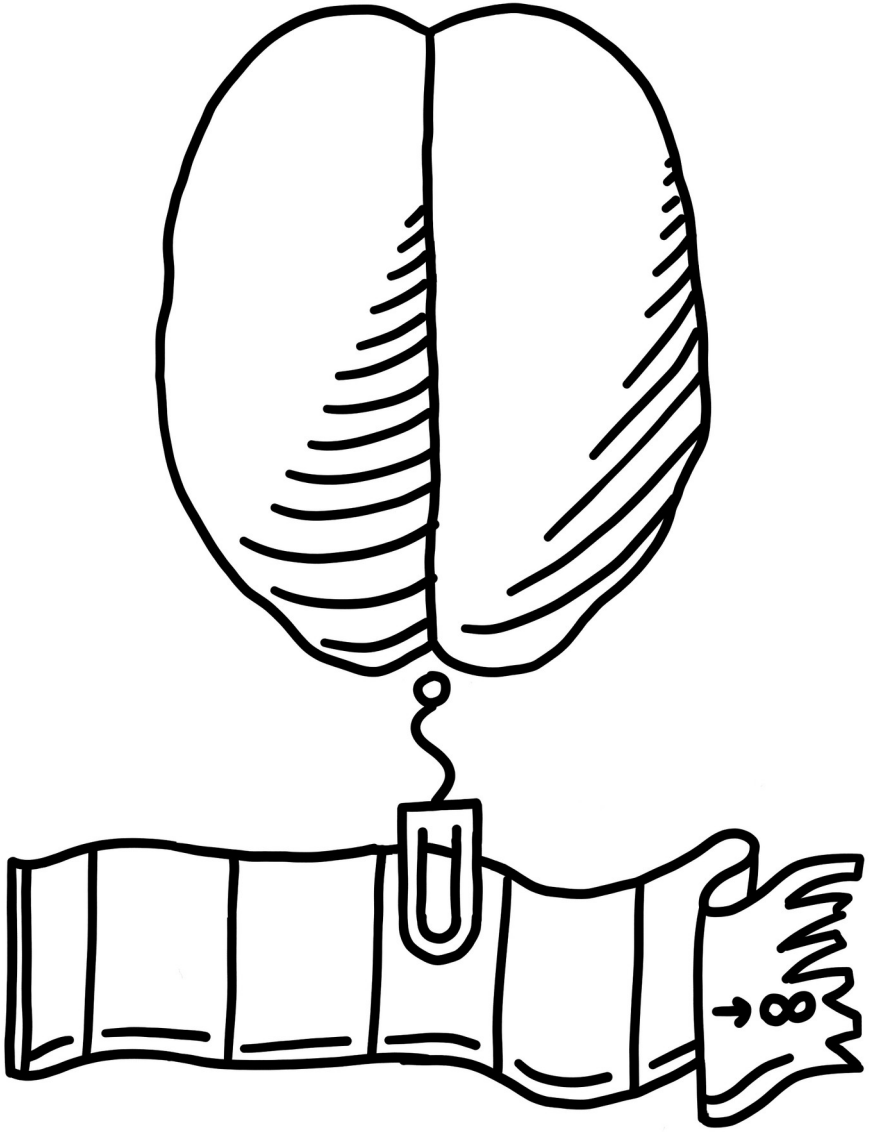


The behaviour of the computer at any moment is determined by the symbols which he is observing, and his "state of mind" at that moment. We may suppose that there is a bound B to the number of symbols or squares which the computer can observe at one moment. If he wishes to observe more, he must use successive observations. We will also suppose

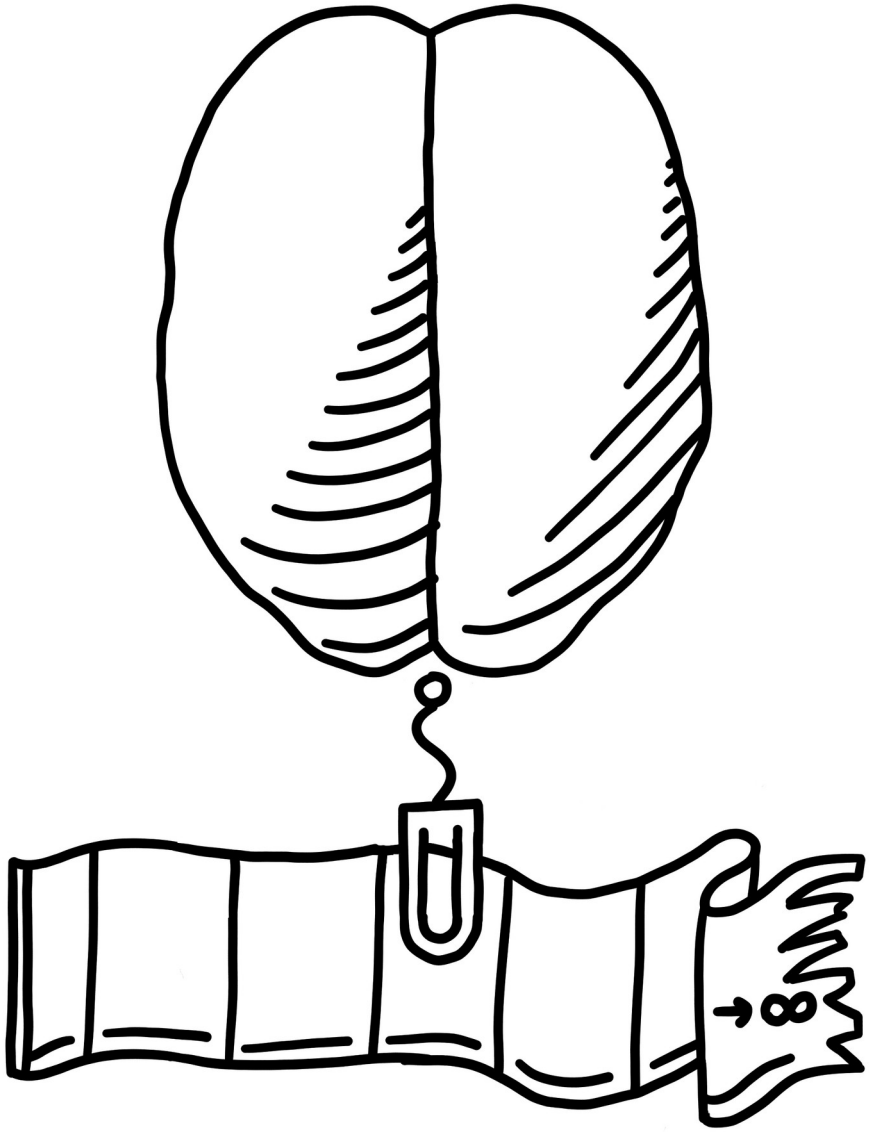


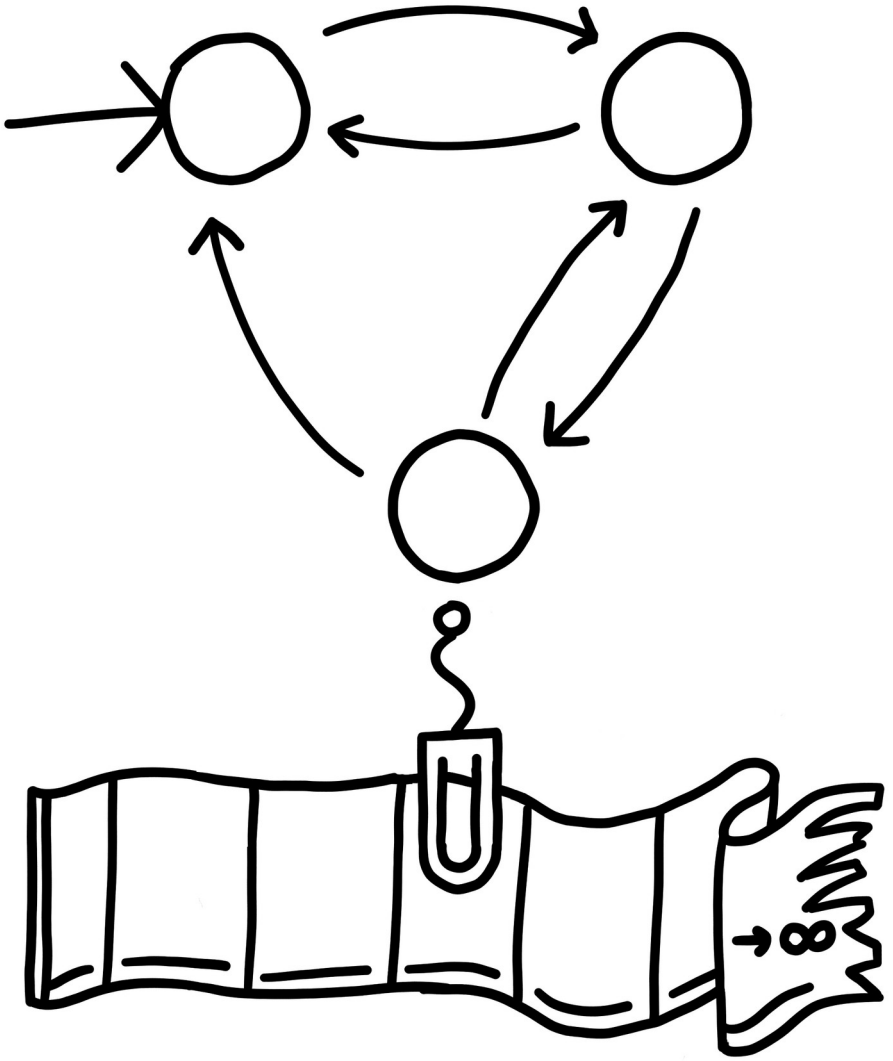
The new observed squares must be immediately recognisable by the computer. I think it is reasonable to suppose that they can only be squares whose distance from the closest of the immediately previously observed squares does not exceed a certain fixed amount. Let us say that each of the new observed squares is within L squares of an immediately previously observed square.

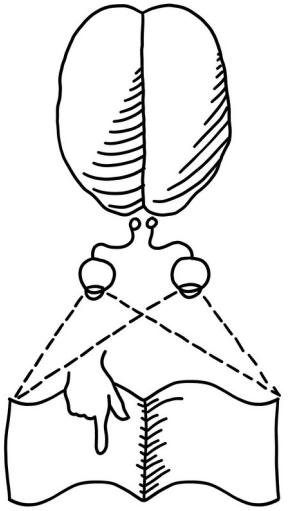




We will also suppose that the number of states of mind which need be taken into account is finite. The reasons for this are of the same character as those which restrict the number of symbols. If we admitted an infinity of states of mind, some of them will be "arbitrarily close" and will be confused.







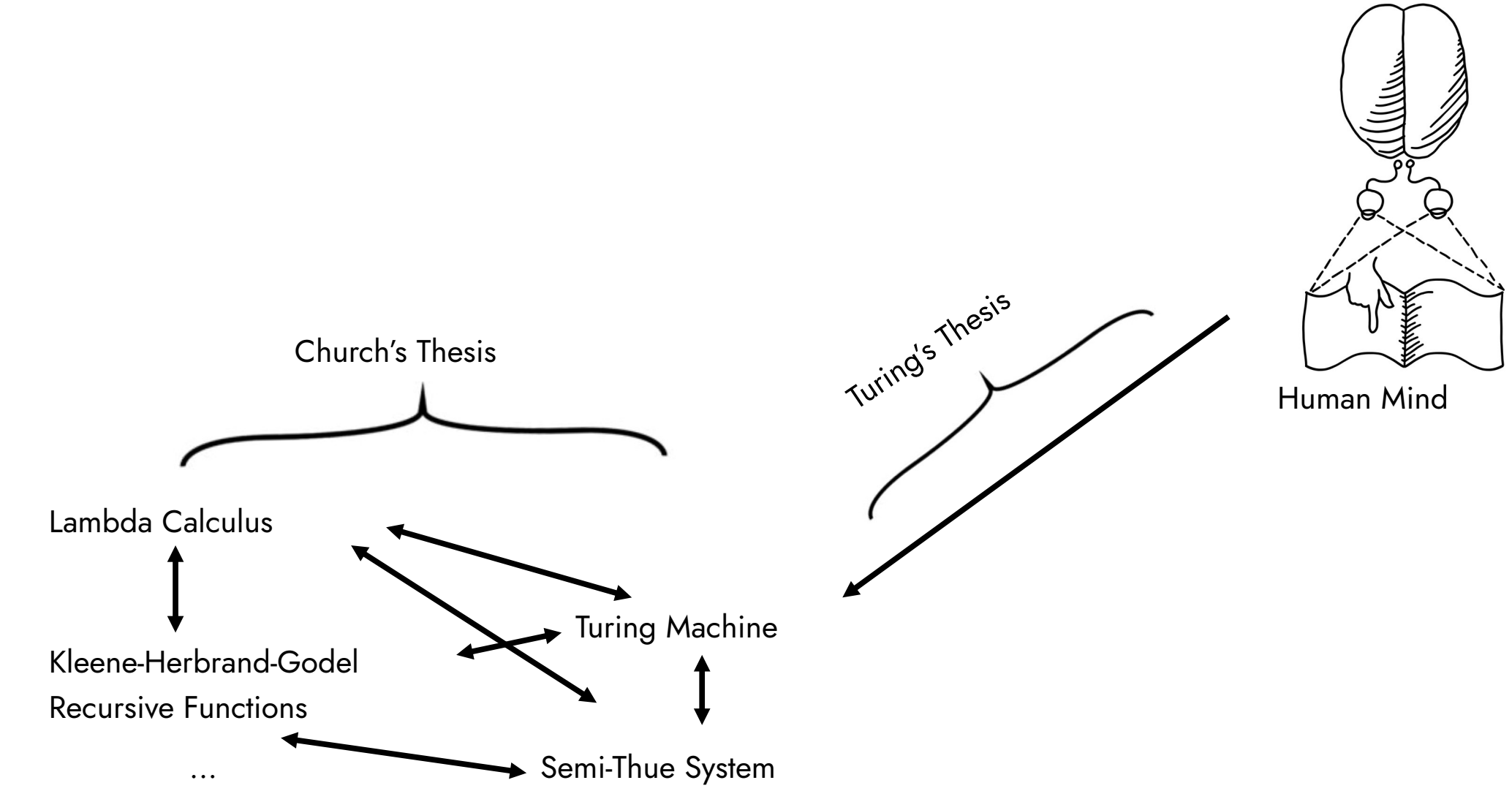
Human Mind

Lambda Calculus

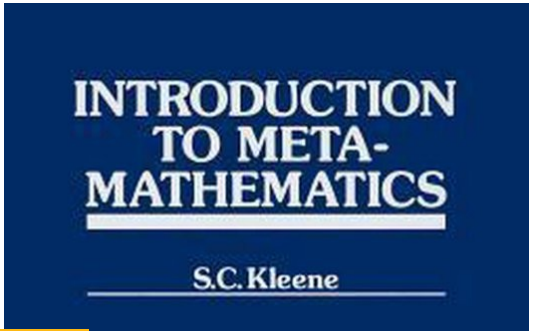
Kleene-Herbrand-Godel
Recursive Functions

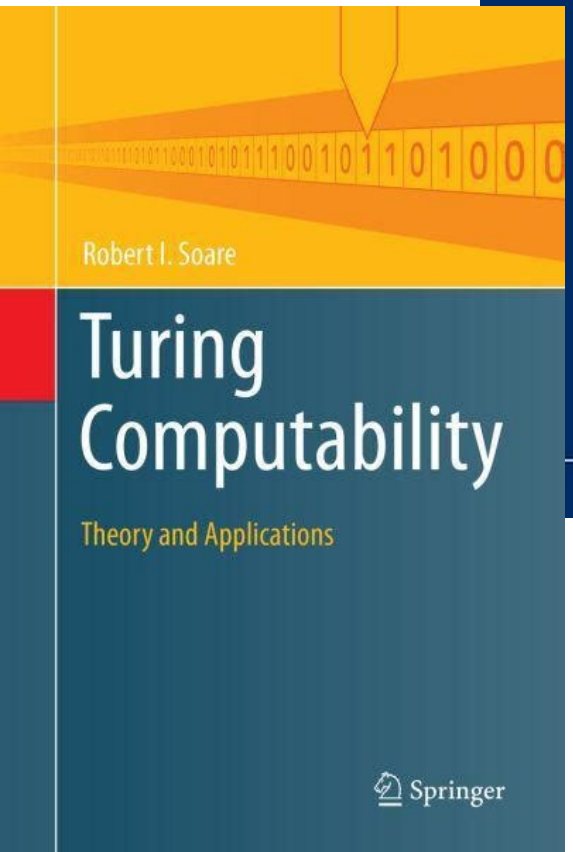
Turing Machine

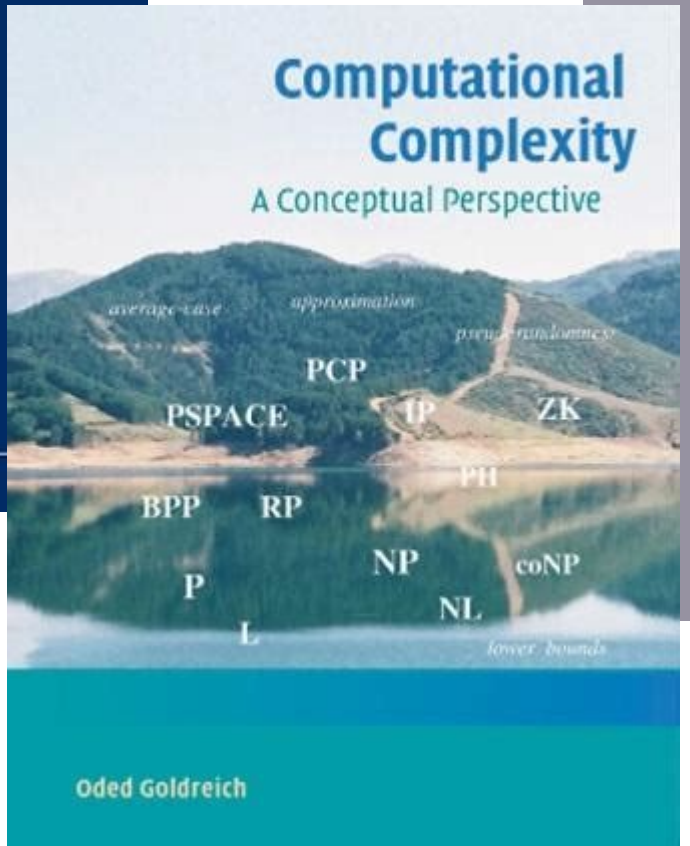
Semi-Thue System

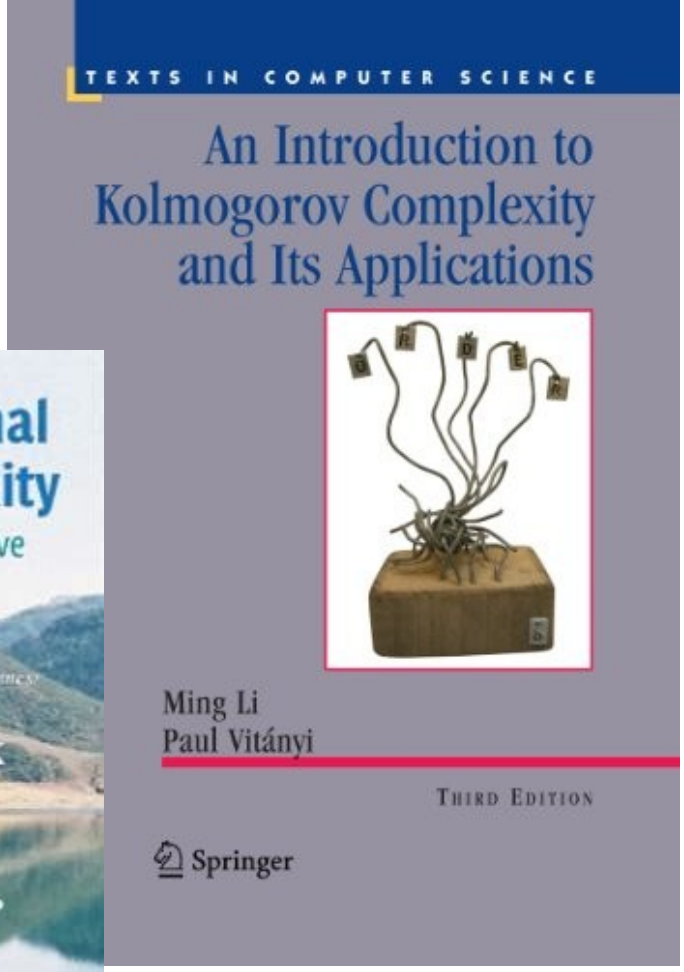

Church's Thesis
Turing's Thesis
Human Mind
Lambda Calculus
Turing Machine
Kleene-Herbrand-Godel
Recursive Functions
...
Semi-Thue System

- Church's Thesis: Any reasonable model of computation is equivalent to a Turing machine
- Turing's Thesis: The Turing machine is equivalent in power to the human mind
- Church-Turing Thesis: The definition of algorithm is independent of any specific formalism


INTRODUCTION
TO META-
MATHEMATICS
S.C. Kleene
North-Holland


Robert I. Soare
Turing
Computability
Theory and Applications
Springer


Computational
Complexity
A Conceptual Perspective
average-case
approximation
PCP
PSPACE
IP
ZK
PH
BPP
RP
NP
coNP
P
NL
L
lower bounds
Oded Goldreich


TEXTS IN COMPUTER SCIENCE
An Introduction to
Kolmogorov Complexity
and Its Applications
Ming Li
Paul Vitányi
Third Edition
Springer

Applications

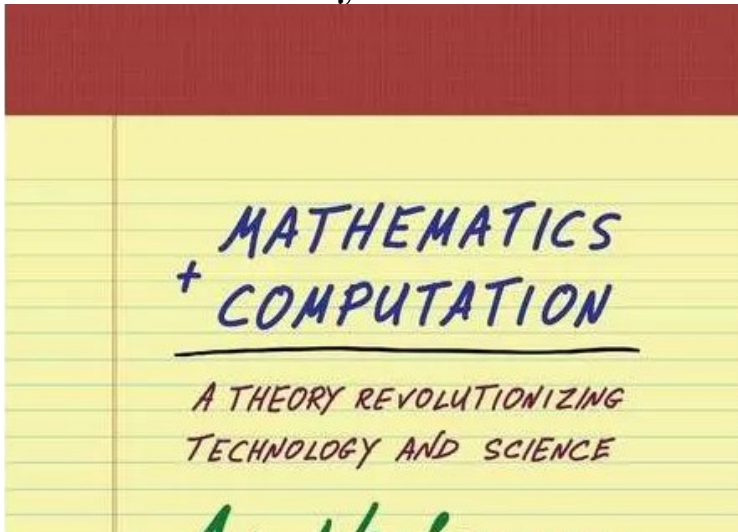
- Well specified pseudocode can always be implemented
- A problem unsolvable to a formal model (like a Turing machine) will be unsolvable to us as well.

A Recipe for All of Computer Science

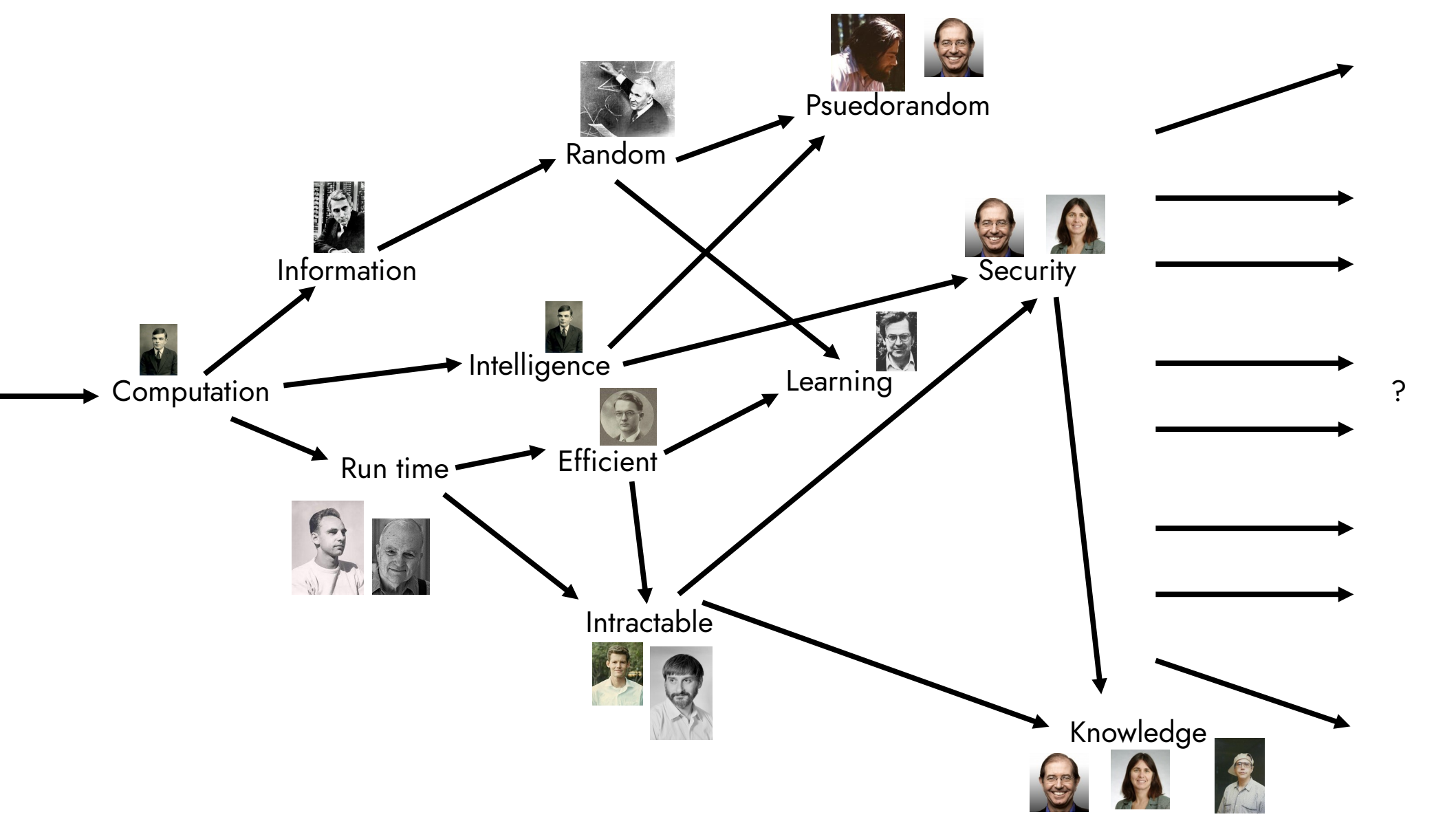
- Choose an abstract intuitive concept
- Construct a formal definition
- Argue that the intuitive concept corresponds to the formal definition
- Proofs or results involving the formal definition implicate the intuitive concept

I feel proud to belong to a field that has seriously taken on defining (sometimes redefining, sometimes in several ways) and understanding such fundamental notions that include: *collusion, coordination, conflict, entropy, equilibrium, evolution, fairness, game, induction, intelligence, interaction, knowledge, language, learning, ontology, prediction, privacy, process, proof, secret, simultaneity, strategy, synchrony, randomness, and verification.*

It is worthwhile reading this list again, slowly. I find it quite remarkable to contrast the long history, volumes of text written, and intellectual breadth that the concepts in this list represent, with the small size and the relative youthfulness of ToC, which has added so much to their understanding.



- Proof – Frege, Hillbert, Russell, Whitehead, (1900s)
- Truth – Tarski (1933)
- Computation, Algorithm – Turing (1936)
- Information, Communication, Noise – Shannon (1948)
- Intelligence – Turing (1950)
- Grammatical and Ungrammatical – Chomsky (1950s)
- Randomness – Kolmogorov (1963)
- Run-time, Complexity – Hartmanis and Stearns (1965)
- Efficient – Edmonds and Cobham(1965)
- Intractable, Difficult – Cook and Levin (1971, 1973)
- Pseudorandomness – Blum, Micali (1982)
- Secure – Goldwasser, Micali (1982)
- Learning – Valiant (1984)
- Knowledge – Babai, Goldwasser, Micali, Rackoff, Moran, (1989+)


Psuedorandom
Random
Information
Security
Computation
Intelligence
Learning
?
Run time
Efficient
Intractable
Knowledge

- Computer Science has nothing to do with computers. These are just tools.
- Chemistry is not about beakers
- We (Computer Scientists) have as big of a claim to understanding the beauty of the natural world as much as physicists or biologists do

Questions?


Bank of England
AA40 925300
Final
m-config. Symbol Operations m-config.
$q_i \quad S_j \quad PS_k, L \quad q_m \quad (N_1)$
$q_i \quad S_j \quad PS_k, R \quad q_m \quad (N_2)$
$q_i \quad S_j \quad PS_k \quad q_m \quad (N_3)$
$q_1 S_0 S_1 Rq_3;\ q_2 S_0 S_0 Rq_2;\ q_3 S_0 S_1 Rq_4;\ q_4 S_0 S_0 Rq_1;.$
Fifty Pounds
This is only a foretaste of what is to come
and only the shadow of what is going to be
Alan Turing 1912-1954
© THE GOVERNOR AND COMPANY OF THE BANK OF ENGLAND 2020
£50
AA40 925300