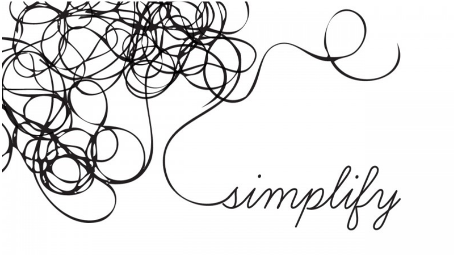
Mathematical Abstraction: Turing's Analysis of Computation

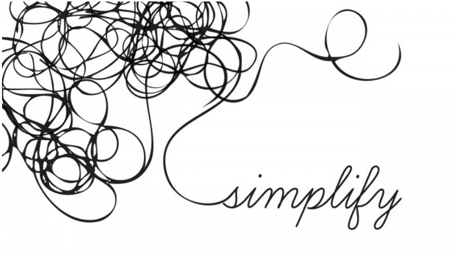
Rossella Marrano

Scuola Normale Superiore

The problem: mathematical modelling

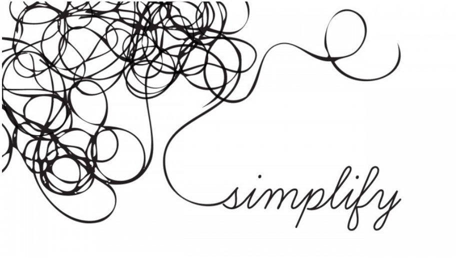


The problem: mathematical modelling



- *all models are wrong*
- *but some models are wronger than others*

The problem: mathematical modelling



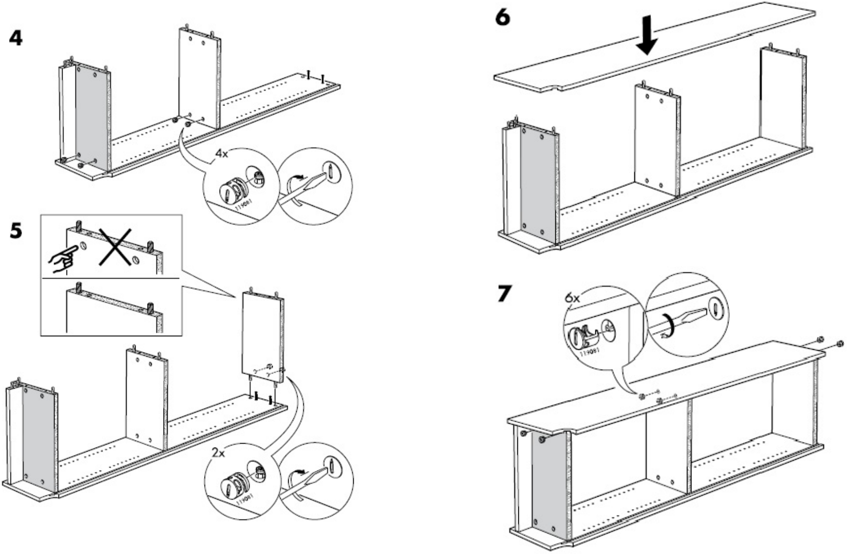
- *all models are wrong*
- *but some models are wronger than others*

Case-study

Turing's analysis of computation

- mathematical abstraction
- role of the agent

Algorithm (we all do it)



Algorithm (mathematicians have always been doing)

input: two positive integers a and b with $a \geq b$
output: greatest common divisor $GCD(a, b)$

Euclid (c. 300 b.C.)

Algorithm (mathematicians have always been doing)

Euclid (c. 300 b.C.)

input: two positive integers a and b with $a \geq b$
output: greatest common divisor $GCD(a, b)$

method:

Divide a by b:

$$a = q_1 b + r_1$$

if $r_1 > 0$, divide it by b:

$$b = q_2 r_1 + r_2$$

if $r_2 > 0$, keep dividing:

$$r_1 = q_3 r_2 + r_3$$

$$(\dots)$$

$$r_n = q_{n+2} r_{n+1} + r_{n+2}$$

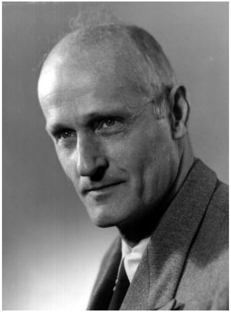
Continue until $r_k = 0$

The last non-zero remainder r_{k-1} is $GCD(a, b)$.

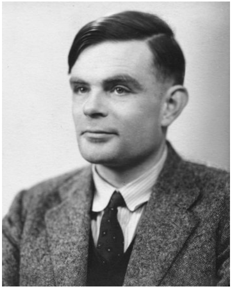
2000 years later: the confluence of ideas

2000 years later: the confluence of ideas

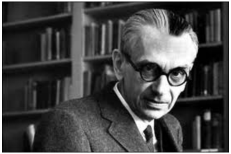
Alonzo Church
(1903-1995)



Stephen Kleene
(1909-1994)



Alan M. Turing
(1912-1954)



Kurt Gödel (1906-1978)

Turing's analysis

> If one is given a puzzle to solve one will usually, if it proves to be difficult, ask the owner whether it can be done. Such a question should have a quite definite answer, yes or no, at any rate provided the rules describing what you are allowed to do are perfectly clear. Of course the owner of the puzzle may not know the answer. One might equally ask, 'How can one tell whether a puzzle is solvable?', but this cannot be answered so straightforwardly. (Turing, 1954)

Turing's problem

Investigate the class of problems that can be effectively solved.

Turing's analysis

> ```
> If one is given a puzzle to solve one will usually, if it
> proves to be difficult, ask the owner whether it can be done.
> Such a question should have a quite definite answer, yes or no,
> at any rate provided the rules describing what you are allowed
> to do are perfectly clear. Of course the owner of the puzzle
> may not know the answer. One might equally ask, ‘How can one
> tell whether a puzzle is solvable?’, but this cannot be
> answered so straightforwardly. (Turing, 1954)
> ```

Turing's problem

Investigate the class of problems that can be effectively solved.

On Computable Numbers, with an Application to the Entscheidungsproblem, Proc. Lond. Math. Soc. (2) 42, 230–265, 1936.

Turing's analysis

> If one is given a puzzle to solve one will usually, if it proves to be difficult, ask the owner whether it can be done. Such a question should have a quite definite answer, yes or no, at any rate provided the rules describing what you are allowed to do are perfectly clear. Of course the owner of the puzzle may not know the answer. One might equally ask, 'How can one tell whether a puzzle is solvable?', but this cannot be answered so straightforwardly. (Turing, 1954)

Turing's problem

Investigate the class of problems that can be effectively solved.

On Computable Numbers, with an Application to the Entscheidungsproblem, Proc. Lond. Math. Soc. (2) 42, 230–265, 1936.

Central idea

> We may compare a man in the process of computing a real number to a machine [...].

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.

[...] 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 that the number of states of mind which need to be taken into account is finite.

[...] Let us imagine the operations to be performed by the computer to be split up into 'simple operations' which are so elementary it is not easy to imagine them further divided. Every such operation consists of some change of the physical system consisting of the computer and his tape. We know the state of the system if we know the sequence of symbols on the tape, which of these are observed by the computer (possibly with a special order), and the state of mind of the computer. We may suppose that in a simple operation not more than one symbol is altered. Any other changes can be split up into changes of this kind.

Abstraction hypotheses

I one-dimensional character of the support

II finite number of printed symbols

III finite number of observed symbols

IV finite number of states of mind

V all the performed operations are elementary ones:
- not more than one symbol is altered
- the squares whose symbols are changed are always the 'observed' squares
- changing observed square is one-step process

VI any calculation step is determined only by the computer's state of mind and the observed symbols

VII (implicit) no practical limitations

We may now construct a machine to do the work of this computer. To each state of mind of the computer corresponds an 'm-configuration' of the machine. The machine scans B squares corresponding to the B squares observed by the computer.

[...]

The move which is done, and the succeeding configuration, are determined by the scanned symbol and the m-configuration.

We may now construct a machine to do the work of this computer. To each state of mind of the computer corresponds an 'm-configuration' of the machine. The machine scans B squares corresponding to the B squares observed by the computer.

[...]

The move which is done, and the succeeding configuration, are determined by the scanned symbol and the m-configuration.

Human computer	Machine

We may now construct a machine to do the work of this computer. To each state of mind of the computer corresponds an 'm-configuration' of the machine. The machine scans B squares corresponding to the B squares observed by the computer.

[...]

The move which is done, and the succeeding configuration, are determined by the scanned symbol and the m-configuration.

Human computer

- paper

Machine

- tape

We may now construct a machine to do the work of this computer. To each state of mind of the computer corresponds an 'm-configuration' of the machine. The machine scans B squares corresponding to the B squares observed by the computer.

[...]

The move which is done, and the succeeding configuration, are determined by the scanned symbol and the m-configuration.

Human computer

- paper
- mental states

Machine

- tape
- m-configurations

We may now construct a machine to do the work of this computer. To each state of mind of the computer corresponds an 'm-configuration' of the machine. The machine scans B squares corresponding to the B squares observed by the computer.

[...]

The move which is done, and the succeeding configuration, are determined by the scanned symbol and the m-configuration.

Human computer

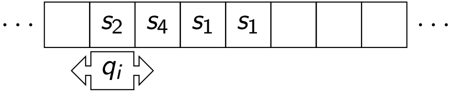
- paper
- mental states
- observation

Machine

- tape
- m-configurations
- scanning

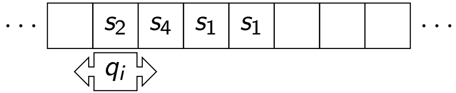
Turing Machine

- $q_0, \ldots, q_n$
- $s_1, \ldots, s_m$
- indefinitely extensible tape
- read-write head



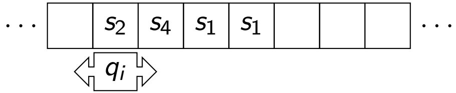
Turing Machine

- $q_0, \ldots, q_n$
- $s_1, \ldots, s_m$
- indefinitely extensible tape
- read-write head
- actions:
  - change symbol (print/erase)
  - move to the right (R)
  - move to the left (L)



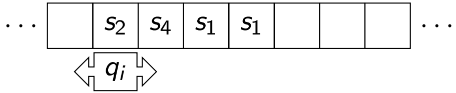
Turing Machine

- $q_0, \ldots, q_n$
- $s_1, \ldots, s_m$
- indefinitely extensible tape
- read-write head
- actions:
  - change symbol (print/erase)
  - move to the right (R)
  - move to the left (L)
- operations:
  - change state
  - halt



Turing Machine

- $q_0, \ldots, q_n$
- $s_1, \ldots, s_m$
- indefinitely extensible tape
- read-write head
- actions:
  - change symbol (print/erase)
  - move to the right (R)
  - move to the left (L)
- operations:
  - change state
  - halt



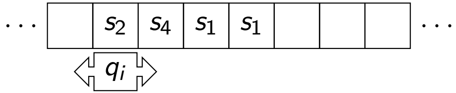
List of instructions

⟨ state, symbol, symbol to write or head action, next state ⟩

$\langle q_i, s_k, s_m, q_j \rangle$

Turing Machine

- $q_0, \ldots, q_n$
- $s_1, \ldots, s_m$
- indefinitely extensible tape
- read-write head
- actions:
  - change symbol (print/erase)
  - move to the right (R)
  - move to the left (L)
- operations:
  - change state
  - halt



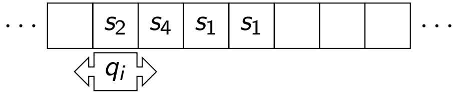
List of instructions

⟨ state, symbol, symbol to write or head action, next state ⟩

$\langle q_i,\ s_k,\ s_m,\ q_j \rangle$

Turing Machine

- $q_0, \ldots, q_n$
- $s_1, \ldots, s_m$
- indefinitely extensible tape
- read-write head
- actions:
  - change symbol (print/erase)
  - move to the right (R)
  - move to the left (L)
- operations:
  - change state
  - halt



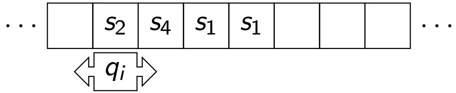
List of instructions

⟨ state, symbol, symbol to write or head action, next state ⟩

$\langle q_i,\ s_k,\ R,\ q_j \rangle$

Turing Machine

- $q_0, \ldots, q_n$
- $s_1, \ldots, s_m$
- indefinitely extensible tape
- read-write head
- actions:
  - change symbol (print/erase)
  - move to the right (R)
  - move to the left (L)
- operations:
  - change state
  - halt



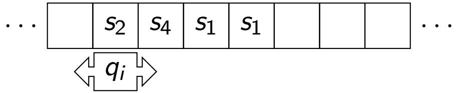
List of instructions

⟨ state, symbol, symbol to write or head action, next state ⟩

$\langle q_i,\ s_k,\ L,\ q_j \rangle$

Turing Machine

- $q_0, \ldots, q_n$
- $s_1, \ldots, s_m$
- indefinitely extensible tape
- read-write head
- actions:
  - change symbol (print/erase)
  - move to the right (R)
  - move to the left (L)
- operations:
  - change state
  - halt



List of instructions

⟨ state, symbol, symbol to write or head action, next state ⟩

$\langle q_i,\ s_k,\ L,\ q_j \rangle$

Turing-solvability

`start` input on the tape (in some notation)

Turing-solvability

`start` input on the tape (in some notation)

`calculation steps` input transformations

Turing-solvability

`start`	input on the tape (in some notation)
`calculation steps`	input transformations
`end`	output, if the machine halts; no output, otherwise.

Turing-solvability

`start`	input on the tape (in some notation)
`calculation steps`	input transformations
`end`	output, if the machine halts; no output, otherwise.

Turing-solvability

Turing-solvable problems are those for which a Turing machine gives an output.

Turing-solvability

`start`	input on the tape (in some notation)
`calculation steps`	input transformations
`end`	output, if the machine halts; no output, otherwise.

Turing-solvability

Turing-solvable problems are those for which a Turing machine gives an output.

There are T-unsolvable problems

There are T-solvable but unfeasible in practice problems

Church-Turing Thesis

Fact

All T-solvable problems are effectively solvable.

Church-Turing Thesis

Fact

All T-solvable problems are effectively solvable.

> `Logical computing machines can do anything that could be described as "rule of thumb" or "purely mechanical".`
>
> `(Turing, 1948)`

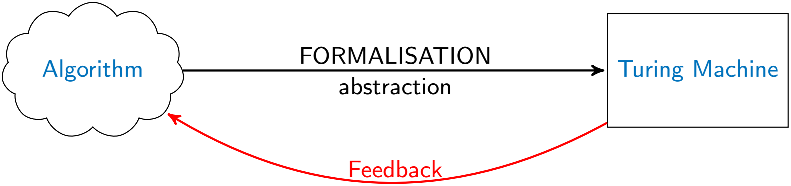
Thesis

All effectively solvable problems are T-solvable.

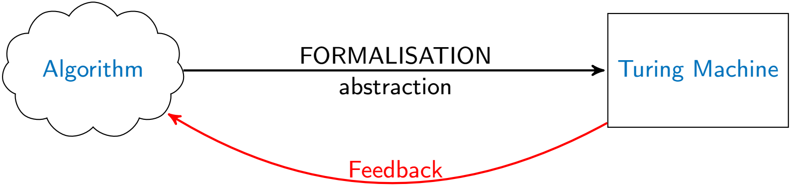
- strength and solid consensus
- the thesis cannot be proved


Algorithm
Turing Machine


Algorithm
FORMALISATION
abstraction
Turing Machine

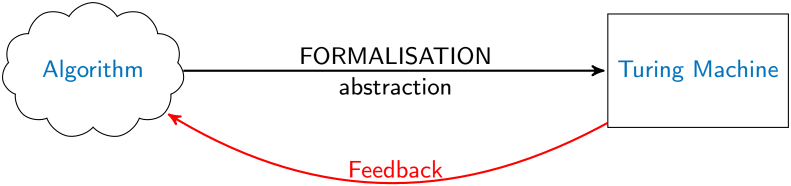

Algorithm
FORMALISATION
abstraction
Turing Machine
Feedback

Feedback I



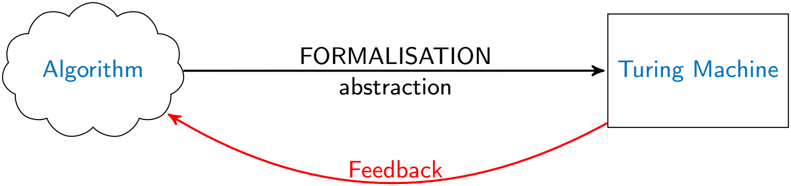
Feedback I

- Turing isolates the essential aspects of computation:



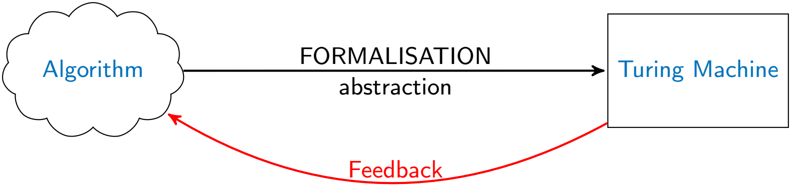
Feedback I

- Turing isolates the essential aspects of computation:
  - ✗ where the instructions come from,
  - ✗ psychological activities,



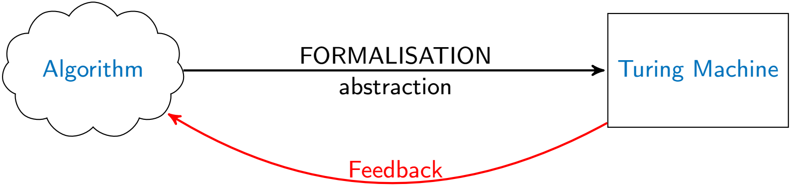
Feedback I

- Turing isolates the essential aspects of computation:
  - ✗ where the instructions come from,
  - ✗ psychological activities,
  - ✓ observable behaviour (input–output).

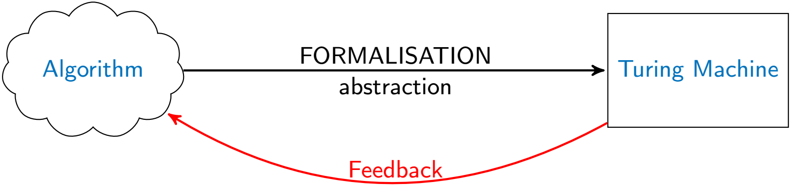


Feedback I

- Turing isolates the essential aspects of computation:
  - ✗ where the instructions come from,
  - ✗ psychological activities,
  - ✓ observable behaviour (input–output).
- The 'human nature' of the computer is irrelevant.

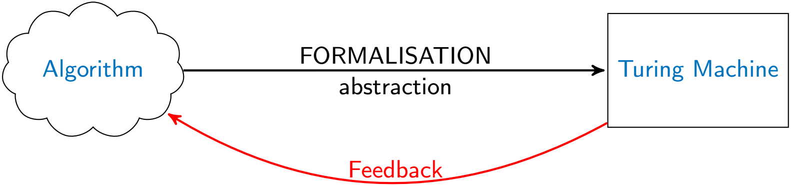


Feedback II (CTT)



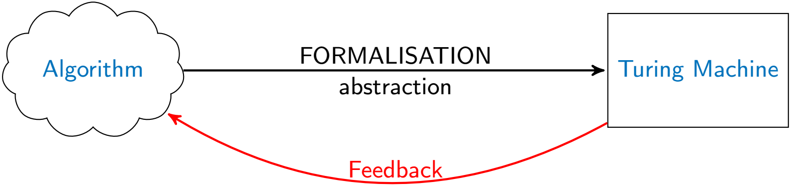
Feedback II (CTT)

- abstraction hypotheses do not change the nature of the intuitive concept we start with



Feedback II (CTT)

- abstraction hypotheses do not change the nature of the intuitive concept we start with
- mathematical abstraction in this case is costless

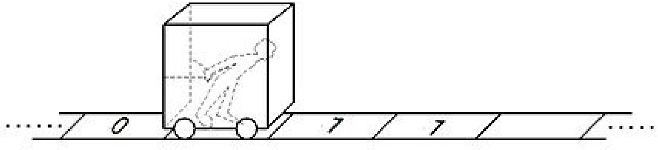


Feedback II (CTT)

- abstraction hypotheses do not change the nature of the intuitive concept we start with
- mathematical abstraction in this case is costless
- the intuitive notion (sharpened by the act of modelling) is itself a mathematical concept

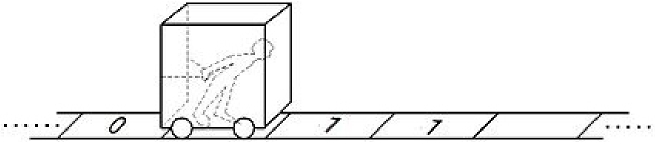
Analogy VS. Modelling

- Turing doesn't model the human being, he models the intuitive notion of algorithmic procedure



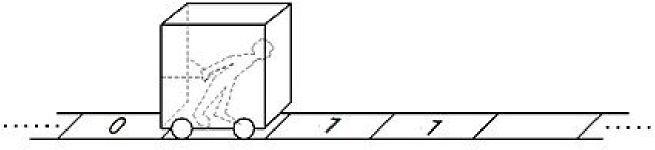
Analogy VS. Modelling

- Turing doesn't model the human being, he models the intuitive notion of algorithmic procedure
- Turing uses a person computing a real number as analogy



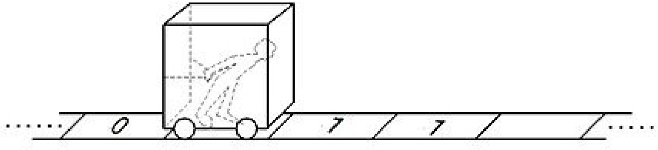
Analogy VS. Modelling

- Turing doesn't model the human being, he models the intuitive notion of algorithmic procedure
- Turing uses a person computing a real number as analogy
- through this analysis he defines the notion of *logical computing machine*



Analogy VS. Modelling

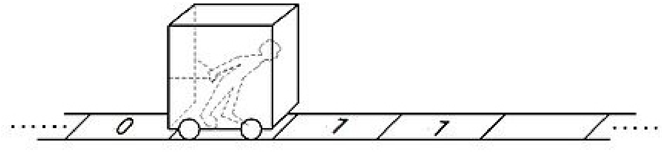
- Turing doesn't model the human being, he models the intuitive notion of algorithmic procedure
- Turing uses a person computing a real number as analogy
- through this analysis he defines the notion of *logical computing machine*
- Turing Machines are abstract algorithmic procedures



Analogy VS. Modelling

- Turing doesn't model the human being, he models the intuitive notion of algorithmic procedure
- Turing uses a person computing a real number as analogy
- through this analysis he defines the notion of *logical computing machine*
- Turing Machines are abstract algorithmic procedures

But we can take Turing Machines to be agents again!



TMs as idealised agents

- no resource limitations:
  - time
  - memory (potentially infinite tape)
- no cognitive limitations
- no mistakes
- no interactions with the environment

TMs as idealised agents

- no resource limitations:
  - time
  - memory (potentially infinite tape)
- no cognitive limitations
- no mistakes
- no interactions with the environment

Modulo CTT, this agent can solve any effectively solvable problem!

TMs as maximally idealised agents

- no resource limitations:
  - time
  - memory (potentially infinite tape)
- no cognitive limitations
- no mistakes
- no interactions with the environment

Modulo CTT, this agent can solve any effectively solvable problem!

Most demanding normative standard for rationality

In principle VS. feasible