Post's (and Zuse's) "forgotten ideas" in the Philosophy and History of Computer Science.

Liesbeth De Mol
Boole centre for Research in Informatics, Ireland
Centre for Logic and Philosophy of Science, Belgium
elizabeth.demol@ugent.be

1. The Church-Turing thesis *historically*.

The Church-Turing thesis

"On computable numbers". From computor to computer.

⇒ **"What are the possible processes which can be carried out in computing a number?"**

- "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 $q_1, q_2, ..., q_R$ which will be called m-configurations. "
- 1-dimensional tape instead of 2-dimensional paper
- Finiteness conditions on (Gandy, 1988; Sieg, 1994):
  1. the number of symbols printed (boundedness condition)
  2. the number of states (boundedness condition)
  3. the number of cells that can be scanned simultaneously (boundedness condition)
  4. the number of symbols that can be changed simultaneously (locality condition)
  5. the number of moves to left or right that can be made in one time step (locality condition)

⇒ *Analysis that results in abstract computor, resulting in Turing machine*

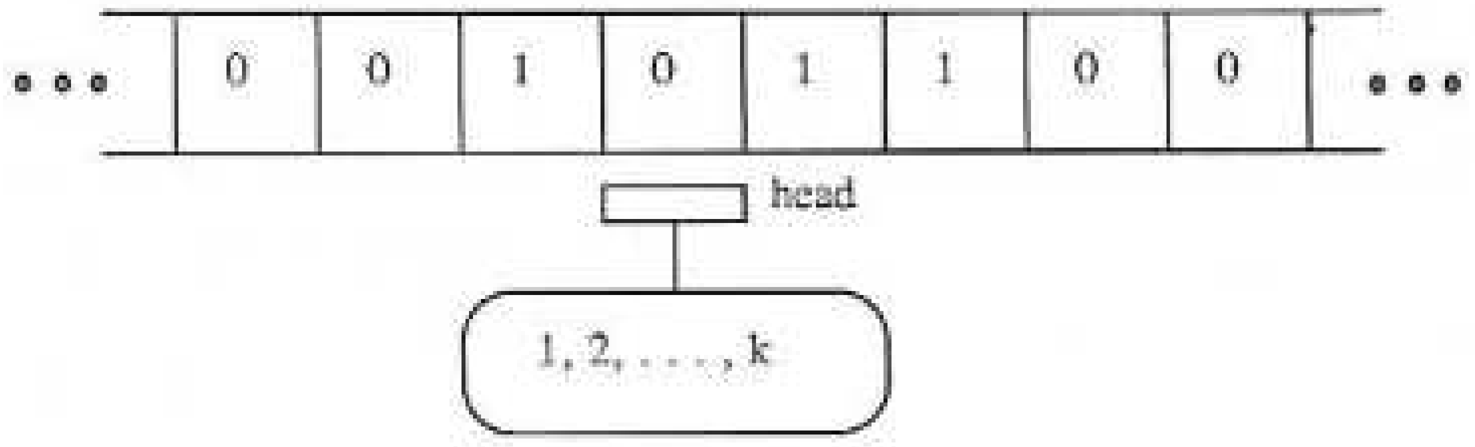


Figure 1: A Turing machine

The Church-Turing thesis
"On computable numbers". Turing's thesis.

"According to my definition, a number is computable if its decimal expansion can be written down by a machine."

Turing's thesis. *Any number (anything) that can be computed by a human being, can be computed by a Turing machine (and conversely)*

- This thesis (a definition for Turing!) imposes a fundamental limitation of what is computable. As a consequence there are uncomputable functions (e.g. halting problem)
- But "All arguments which can be given are bound to be, fundamentally, appeals to intuition, and for this reason rather unsatisfactory mathematically. The real question at issue is 'What are the possible processes which can be carried out in computing a number?' "

The Church-Turing thesis

The surprise of λ...

- Initial motivation: "Underlying the [formal calculus] which we shall develop is the concept of a function, as it appears in various branches of mathematics [...]" (Church, 1941)
- "We [Church and Kleene] kept thinking of specific such functions, and of specific operations for proceeding from such functions to others. I kept establishing the functions to be λ-definable and the operations to preserve λ-definability." (Kleene, 1981)
- "Before research was done, no one guessed the richness of this subsystem. Who would have guessed that this formulation, generated as I have described to clarify the notation for functions, has implicit in it the notion (not known in mathematics in 1931 in a precise version) of all functions on the positive integers (or on the natural numbers) for which there are algorithms?" (Kleene, 1981)

The Church-Turing thesis
Church's thesis

- "We now define the notion [...] of an *effectively calculable* function of positive integers by identifying it with the notion of a recursive function of positive integers (or of a λ-definable function of positive integers.)"
- "This definition is thought to be justified by the considerations which follow, *so far as positive justification can ever be obtained for the selection of a formal definition to correspond to an intuitive notion.* [m.i.]"

  Church's Thesis. *Every effectively calculable function is general recursive (λ-definable) and conversely.*

2. “Computing” the non-Turing computable. A philosophical debate.

“Computing” the non-Turing computable (Eberbach, Goldin, Wegner, 2004).

Strong Turing Thesis A Turing machine can do anything a computer can do ($\sim$ physical Church-Turing thesis)

Church-Turing thesis The formal notions of recursiveness, λ-definability, and Turing computability equivalently capture the intuitive notion of effective computability of functions over integers.

"Computing" the non-Turing computable (Eberbach, Goldin, Wegner, 2004).

Assumption Turing machines can *only* model algorithmic computations, but, *not every "computation" is algorithmic* (example: distributed client-server computation).

[I] The TM models *closed* computation, which requires that all inputs are given in advance

[II] The TM is allowed to use an unbounded but only *finite amount of time and memory* for its computation.

[III] Every TM computation starts in an identical *initial configuration*; for a given input, TM behaviour is fixed and does not depend on time.

$\Rightarrow$ Ergo, not every computation is Turing computable, and thus the strong Turing thesis must be false. "Though the Church-Turing thesis is valid in the narrow sense that Turing machines express the behavior of algorithms, the broader assertion that algorithms precisely capture what can be computed is invalid."

$\Rightarrow$ Alternatives? *Turing's forgotten ideas in computer science* (??) (Copeland, 1999), i.e., oracle machines.

3. Post's account of computability

A quick account of Post's account of an anticipation

Absolutely unsolvable problems and relatively undecidable propositions. Account of an anticipation (1920-21)

- Posthumuously published by Martin Davis in 1965
- Post's program in 1920: "Since *Principia* was intended to formalize all of existing mathematics, Post was proposing no less than to find a single algorithm for all of mathematics." (Martin Davis, 1994)
- Approach: Focus on form rather than meaning. Simplification through generalization, starting from *Principia* in order to prove the decidability of the Entscheidungsproblem (finiteness problem) for restricted functional calculus!
- The frustrating problem of "tag" and the reversal of Post's program (De Mol, 2006)

Definition of tag systems

Definition 1 *A v-tag system T, consists of a finite alphabet $\Sigma = \{a_0, a_1, ..., a_{\mu-1}\}$ of μ symbols, a deletion number $v \in \mathbb{N}$ and a finite set of μ words $w_{a_0}, ..., w_{a_{\mu-1}} \in \Sigma^*$, called the appendants, where any w_{a_i} corresponds with a_i.*

Computation step on a word $A \in \Sigma^*$: append the word associated with the leftmost letter of A and delete its first v symbols.

Definition of tag systems. A (relatively) famous Example

Let T_{Post} be defined by $\Sigma = \{0, 1\}, v = 3, 1 \rightarrow 1101, 0 \rightarrow 00$

$A_0 = 10111011101000000$

Definition of tag systems. A (relatively) famous Example

Let T_{Post} be defined by $\Sigma = \{0, 1\}, v = 3, 1 \rightarrow 1101, 0 \rightarrow 00$

$A_0 = 10111011101000000$
~~101~~110111010000001101

Definition of tag systems. A (relatively) famous Example

Let T_{Post} be defined by $\Sigma = \{0,1\}, v = 3, 1 \rightarrow 1101, 0 \rightarrow 00$

$A_0 = 10111011101000000$
~~101~~110111010000001101
~~110~~1110100000011011101

Definition of tag systems. A (relatively) famous Example

Let T_{Post} be defined by $\Sigma = \{0, 1\}, v = 3, 1 \rightarrow 1101, 0 \rightarrow 00$

$A_0 = 10111011101000000$
~~101~~110111010000001101
~~110~~1110100000011011101
~~111~~01000000110111011101

Definition of tag systems. A (relatively) famous Example

Let T_{Post} be defined by $\Sigma = \{0,1\}, v = 3, 1 \rightarrow 1101, 0 \rightarrow 00$

$A_0 = 10111011101000000$
~~101~~110111010000001101
~~110~~1110100000011011101
~~111~~01000000110111011101
~~010~~0000011011101110100

Definition of tag systems. A (relatively) famous Example

Let T_{Post} be defined by $\Sigma = \{0, 1\}, v = 3, 1 \rightarrow 1101, 0 \rightarrow 00$

$A_0 = 10111011101000000$

~~101~~110111010000001101

~~110~~1110100000011011101

~~111~~01000000110111011101

~~010~~0000011011101110100

~~000~~00110111011101000

Definition of tag systems. A (relatively) famous Example

Let T_{Post} be defined by $\Sigma = \{0, 1\}, v = 3, 1 \rightarrow 1101, 0 \rightarrow 00$

$A_0 = 10111011101000000$
~~101~~110111010000001101
~~110~~1110100000011011101
~~111~~01000000110111011101
~~010~~0000011011101110100
~~000~~001101110111010000
~~001~~$\underbrace{10111011101000000}_{A_0} \Rightarrow$ Periodicity!

Definition of tag systems. A (relatively) famous Example

Let T_{Post} be defined by $\Sigma = \{0,1\}, v = 3, 1 \rightarrow 1101, 0 \rightarrow 00$

$A_0 = 10111011101000000$

$\cancel{101}110111010000001101$

$\cancel{110}1110100000011011101$

$\cancel{111}01000000110111011101$

$\cancel{010}0000011011101110100$

$\cancel{000}001101110111010000$

$\cancel{001}\underbrace{10111011101000000}_{A_0} \Rightarrow$ Periodicity!

$\Rightarrow$ Turing machine vs. Abstract form that is hard to "encode" $\rightarrow$ analysis of the behavior. Simple forms complex behavior.

Experimentation on tag systems

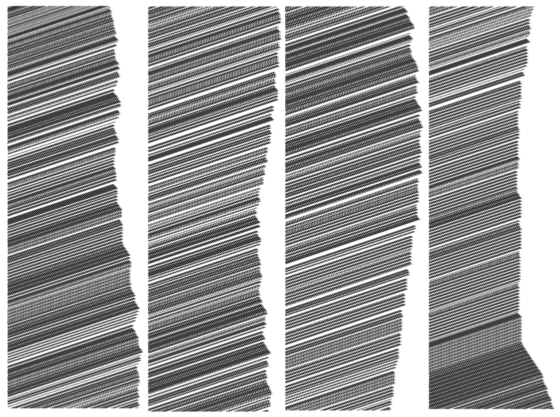


Figure 2: $v = 3, 1 \rightarrow 1101, 0 \rightarrow 1101$

A quick account of Post's account of an anticipation (continued)

- Development of normal systems and proof of the normal form theorem
- $\Rightarrow$ "In view of the generality of the system of *Principia Mathematica*, and its seeming inability to lead to any other generated set of sequences on a given set of letters than those given by our normal systems, we are led to the following generalization", i.e., Post's thesis I (Davis, 1982):

  > Every generated set of sequences on a given set of letters $a_1, a_2, ..., a_\mu$ is a subset of the set of assertions of a system in normal form with primitive letters $a_1, a_2, ..., a_\mu, a'_1, a'_2, ..., a'_\mu$, i.e., the subset consisting of those assertions of the normal system involving the letters $a_1, a_2, ..., a_\mu$.

- "Establishing this universality [the universality of the thesis] is not a matter of mathematical proof, but of psychological analysis of the mental processes involved in combinatory mathematical processes."

A quick account of Post's account of an anticipation (continued)

Given thesis I:

> "There is no finite method which would uniformly enable us to tell of an arbitrary normal system and arbitrary sequence on the letters thereof whether that sequence is or is not generated by the operations of the system from the primitive sequence of the system"
>
> "(**Theorem**) No normal-deductive-system is complete, there always existing a normal system S and enunciation P thereof such that P is not in S while $b(S, P)$ is not in the normal-deductive system."
>
> "(**Theorem**) Given any normal-deductive-system there exists another which second proves more theorems (to put it roughly) than the first."

$\Rightarrow$ Informal proofs of the fact that there are absolutely unsolvable decision problems and relatively undecidable propositions.

- 1936: thesis II, identifying solvability and formulation I (similar to Turing machines)

Significance of Post's work I: focus on absolutely unsolvable decision problems (the limitation implied by the thesis)

- Possible reason? Experience with own limitations to solve problem of "tag"
- "We offer this conclusion at the present moment as a *working hypothesis*. [...] The success of the above program would, for us, change this hypothesis not so much to a definition or to an axiom but to a *natural law*. (Post, 1936)
- "But to mask this identification under a definition hides the fact that a fundamental discovery in the limitations of the mathematicizing power of *Homo Sapiens* has been made and blinds us to the need of its continual verification." (Post, 1936)
- "the creativeness of human mathematics has a counterpart inescapable limitation thereof witness the absolutely unsolvable (combinatory) problems. Indeed, with the bubble of symbolic logic as universal logical machine finally burst, a new future dawns for it as the indispensable means for revealing and developing those limitations. For [...] Symbolic Logic may be said to be Mathematics become self-conscious."
- $\Rightarrow$ Question: how do the so-called models that claim to go beyond the Turing limit deal with this problem?

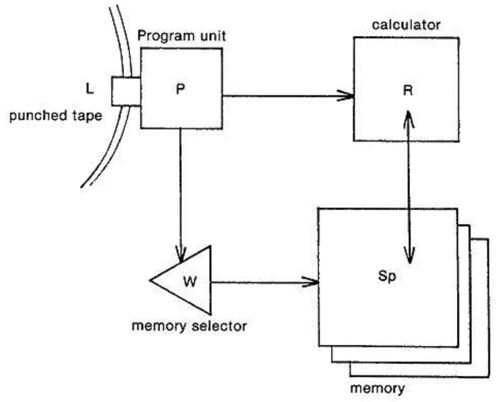
Significance of Post's work II: Focus on form and generated (possibly infinite) sets of sequences of letters

- Generality of the notion of computability. Computations are not only about algorithms and functions on integers.
- Infinite time and space. "TM computations are allowed to use an unbounded but only finite amount of time and memory" (Eberbach, Goldin, Wegner, 2004)
- Unpredictability and evolution over time. "TM behavior is fixed and does not depend on time" (Eberbach, Goldin, Wegner, 2004) Computability and computational processes!
- Weak universality and tag systems (or normal systems): finite vs. infinite input, computational power of the process vs. computational power of the input?

4. Zuse's account of computability

A very quick account of the work of one of the first computer pioneers, Konrad Zuse

- "I became aware of the tremendous number of monotonous calculations necessary for the design of static and aerodynamic structures. Therefore, I decided to design and construct calculating machines suited to solving these problems automatically." (Zuse, 1980)
- Overall plan for computing machine ∼ plan modern general-purpose computer



A very quick account of one of the first computer pioneers, Konrad Zuse

- Use of the binary system (from the pragmatic point of view of the engineer to build a simple machine) $\rightarrow$ switch off/on mechanism of relays $\neq$ ENIAC's "simulation" of decimal system with on/off vacuüm tubes
- Development of Bedingungskombinatorik (Conditional Combinatoric) and Dyadik $\Rightarrow$ The theoretical basis for building switching circuits and ultimately a computer ($\sim$ Shannon's *A Symbolic Analysis of Relay and Switching Circuits*)
- Construction of Z1, Z2, Z3 (the first universal general-purpose computer) and Z4 from 1936 – 1945.
- The logistic machine and the Plankalkül (forerunner of today's programming languages: "The result of these developments will be the general calculation machine, that solves general combinatorial problems and mechanical thinking problems on the basis of applied logistic. I call this development the logistic calculation machine"

A very quick account of one of the first computer pioneers, Konrad Zuse

The bit as the only primitive object

- $S0 \Rightarrow L$ or 0

- Composite objects, e.g., positive integers: $n \times S1.4 = n \times 4 \times S0$

- E.g. 83 = (L000, 00LL)

- Selection of components

$$\begin{array}{l|l} & \mathrm{V} \\ \mathrm{V} & 3 \\ \mathrm{K} & \\ \mathrm{S} & m \times 2 \times 1 \cdot n \end{array} \qquad \begin{array}{l|l} & \mathrm{V} \\ \mathrm{V} & 3 \\ \mathrm{K} & \mathrm{i} \\ \mathrm{S} & 2 \times 1 \cdot n \end{array}$$

K(omponenten index), S(truktur Index), V(ariablen Index)

Zuse's "thesis" (1945)

- "Calculation is: to form out of given "data" (Angaben) new "data" according to some rule (Vorschrift) $\sim$ information-processing"
- "Data can be many things like, for example, numbers, statements, names, addresses, military degrees, messages, coordinates,...They are differentiated through their structure. The most simple such structure has a yes-no-value. It appears that any complicated information can be constructed out of yes/no- values. Thus one is in need of a carefully constructed Structural Calculus (Strukturenkalkül)"
- "A calculation rule can be any schematical operation, deduction, formula, algorithm, command, ... The Plankalkül is developed as an exact formulation of such rules."

Significance of Zuse's "thesis": Generality of Computability

- Explicit differentiation between input/output and the process of the computation.
- Generality of the thesis: "In the following we want to develop a theory to mechanically solve schematic thinking tasks [..] Calculating with numbers belongs to the lowest level; the process of calculating is so schematic and clear that mechanical solutions are already applied to a considerable extent. "There is a steady way from simple numerical calculation to high-level thinking processes."
- "General considerations concerning the relations between calculating and thinking followed. I realized that there is no border line between these two aspects and by 1938 it was already perfectly clear to me that the development would progress in the direction of the artificial brain."
- Basic problem: "The extension of machine calculating beyond the fields hitherto known depends on the competence to *convert* [m.i.] general schematic thinking procedures into a form, suited to deliver the operational instructions for calculating machines"
- Notes going from computers used for bureaucratic tasks to assistance in more "artistic" work. "[...] *we must find the courage to consider [the computer] not as a subordinate tool, but as a key to surmounting difficulties.*

- (“evolutionary” programming (the germ cell that solves engineering problems); the calculating space; cellular automata.)

5. Discussion

Discussion

- Computations have led to a conceptual, mathematical and technological "revolution" $\Rightarrow$ slow historical process that changed the concept of computability as computing functions over integers to something far more general.
- "we cannot fully understand our own conceptual scheme without plumbing its historical roots, but in order to appreciate those roots, we may well have to filter them back through our own ideas." (J. Webb, 1981)
- *If* the world computes $\neq$ the world is computable
- re: On going beyond the Turing limit. Theory vs. practice.
- The thesis *is* and *as* a thesis.
- Significance of different (historical) models of and views on computability (even though they are logically equivalent) We cannot start and end with Turing machines!

References