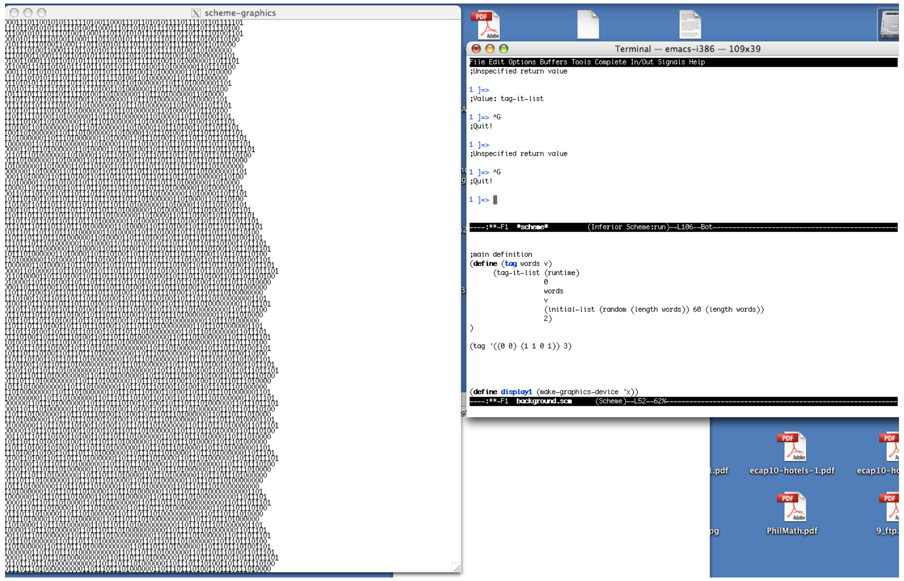
” Beyond ” Turing computability: a historical perspective

Liesbeth de Mol

“Beyond” Turing computability: a historical perspective


scheme-graphics
Terminal — emacs-i386 — 109x39
File Edit Options Buffers Tools Complete In/Out Signals Help
ecap10-hotels-1.pdf
PhilMath.pdf

```
;Unspecified return value

1 ]=>
;Value: tag-it-list

1 ]=> ^G
;Quit!

1 ]=>
;Unspecified return value

1 ]=> ^G
;Quit!

1 ]=>

----:**-F1  *scheme*         (Inferior Scheme:run)--L106--Bot-------------------------------------------


;main definition
(define (tag words v)
      (tag-it-list (runtime)
                   0
                   words
                   v
                   (initial-list (random (length words)) 60 (length words))
                   2)
)

(tag '((0 0) (1 1 0 1)) 3)



(define display1 (make-graphics-device 'x))
----:**-F1  background.scm      (Scheme)--L52--62%-----------------------------------------------------
```

Liesbeth De Mol

CNRS - UMR 8163 Savoirs, textes, Langage

Université de Lille 3, France

liesbeth.demol@univ-lille3.fr

Introduction

Topic Discussion of alternative models to computability by Church and Post

Method? Analysis from the bottom-up – focus on the practice, how that shapes the (content of the) results and, indirectly, their later use.

Motivation and why you might care

- (locally) “Challenge” the “classical” story – “beyond” Turing – understanding computability without assuming the classical story
- (locally) What does it mean to compute?
- (globally) “[...] time after time I found that *because* of my ignorance of these antecedents, *I had not, nor could have, really understood those ideas*. All the logical analysis in the world will not reveal the intentions behind ideas, and without these intentions one all too easily misunderstands and misjudges the ideas and theories of a writer no longer living. [...] one also finds that current ideas and results can illuminate older and crustier ideas. The lesson seems to be this: 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.” (Judson Webb, 1980)

1. Church-Turing thesis

What is the Church-Turing thesis?

⇒What was it about?

	Identification	**Vague notion**	**Formal device**
Church:	definition	eff. calculability	λ-def. & gen. rec. functions
			$\Updownarrow$
Turing:	definition	computability	Turing machines

⇒Why?

- Context of mathematical logic and more specifically Hilbert's formalism, *NOT* computer science (20s and 30s)
- **Motivation:** "It appears that there is no way of finding the general criterion for deciding whether or not a well-formed formula a is provable. [...] the undecidability is even the *conditio sine qua non* for the contemporary practice of mathematics, using as it does heuristic methods, to make any sense. **The very day on which the undecidability does not obtain any more, mathematics as we now understand it would cease to exist; it would be replaced by an absolutely mechanical prescription**, by means of which anyone could decide the probaility or unprovability of any given sentence. Thus we have to take the position: it is generally undecidable, whetehr a given well-formed formula is provable or not. (Von Neumann, 1927)

Why Turing rules!

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.)"

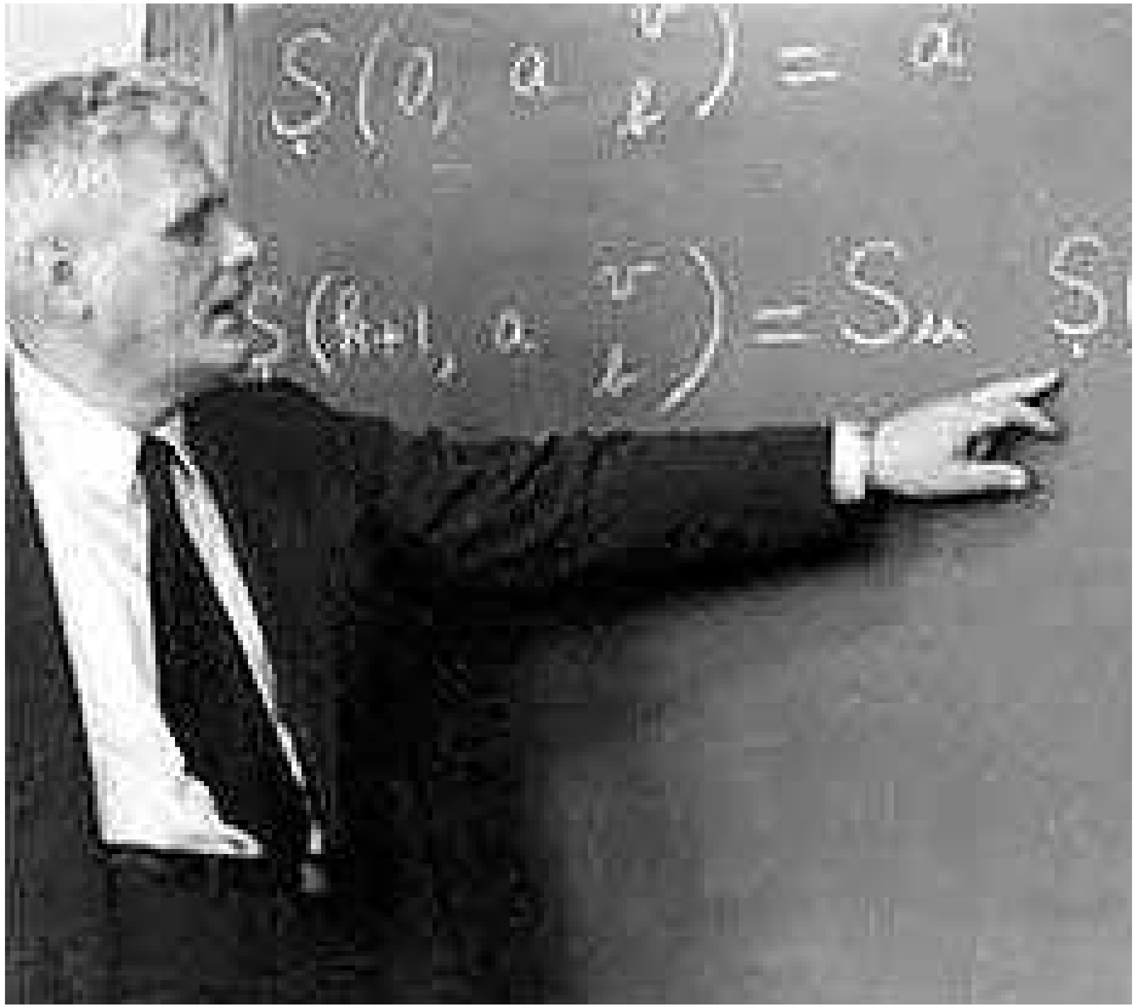
Turing's thesis According to my definition, a number is **computable** if its decimal expansion can be written down by **a machine**"

$\Rightarrow$ "*[I]t was Turing alone who [...] gave the first convincing formal definition of a computable function*" (Soare, 2007). Why?

- **Turing's main question:** "The real question at issue is: What are the possible processes which can be carried out in computing a number?" (Turing, 1936) -
- **Turing's approach:** Analysis of human computer as satisfying number of conditions (locality and boundedness conditions). These can be formalized by a (Turing) machine

$\Leftrightarrow$ **Church's 'approach'?**: Thesis only *after* a thorough analysis of λ-calculus and recursive functions (bottom-up)

Church’s thesi/es



Church’s background in a nutshell (1)....

Interst in foundations

It was Veblen who urged me to study Hilberts work on the plea, which may or may not have been fully correct, that he himself did not understand it and he wished me to explain it to him. At any rate, I tried reading Hilbert. Only his papers published in mathematical periodicals were available at the time. Anybody who has tried those knows they are very hard reading. **I did not read as much of them as I should have, but at least I got started that way.**

“Quasi-heuristic” approach to tackle foundational problems

(independence of AC)“*The object of this paper is to consider the possibility of setting up a logic in which the axiom of choice is false. [...] [I]f a considerable body of theory can be developed on the basis of one of these postulates without obtaining inconsistent results, then this body of theory, when developed, could be used as* **presumptive evidence** *that no contradiction exists*” (Church, 1927)

(consistency set of postulates for mathematics) “*Our present project is to develop the consequences of the foregoing set of postulates, until a contradiction is obtained from them, or until the development has been carried so far consistently as to make it* **empirically probable** *that no contradiction can be obtained from them.*” (Church, 1933)

Church's background in a nutshell (2)....
Motivations for a set of postulates (1932/33)

Yet another formalization of mathematics *after* (!) Gödel "In this paper we present a set of postulates for the foundation of formal logic" (Church, 1932)

Going beyond Gödel "I was seeking to do the very thing that Gödel proved impossible" (Church in a letter to Dawson, July 25, 1983)
"[...] This is conceivable on account of **the entirely formal character of the system** which makes it possible to abstract from the meaning of the symbols and to regard the proving of theorems (of formal logic) as a game played with marks on paper according to a certain arbitrary set of rules" (Church 1933) $\sim$ Post

Formal system such "[...] that every combination of symbols belonging to our system, if it represents a proposition at all, shall represent a particular proposition, unambiguously, and without the addition of verbal explanations."

$\Rightarrow$ **Introduction of the λ-operator – an abstraction operator – to denote functions**: **Ex.** Numbers or functions? "$x^4 + x$ is smaller than 1000" vs. "$x^4 + x$ is a primitive recursive function" $\Rightarrow \lambda x.x^4 + x$

$\Rightarrow$ **Notion of function application** If E and F are lambda terms, so is (EF), viz. applying a function to an argument.

... Church's thesi/es

"Our object is **to prove empirically (!) that the system is adequate for the theory of positive integers**" (Kleene, 1935)

Argument by example The results of Kleene are so general and the possibilities of extending them apparently so unlimited that one is **led to the conjecture that a [λ-] formula can be found to represent any particular constructively defined function of positive integers whatever**. (Church in a letter to Bernays, January 23, 1935)

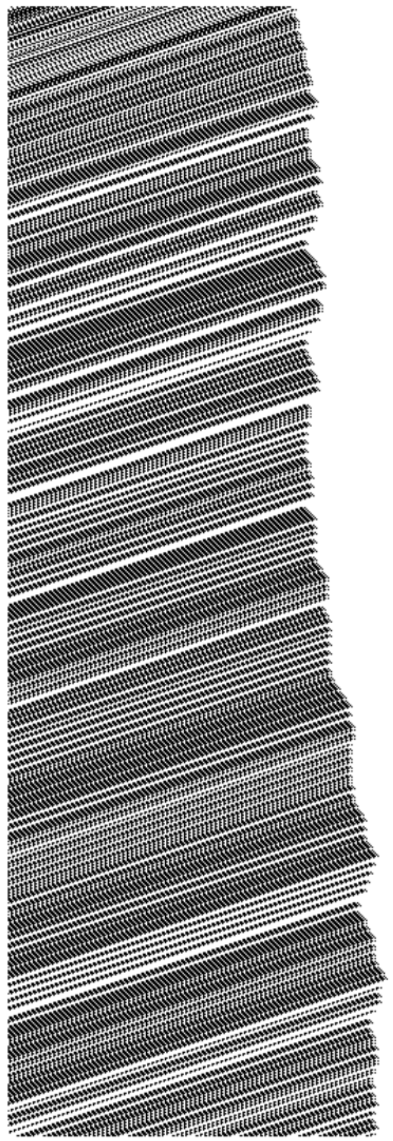
$\Rightarrow$ Church, 1936, three arguments: argument by confluence, argument by example and so-called step-by-recursive-step argument ("Each rule of procedure must be a recursive operation"): **no proper analysis of very notion of effective calculability!**

$\Rightarrow$ **Theoretical constructions vs. practical form:** [Turing's] has the advantage of making the identification with effectiveness in the ordinary (not explicitly defined) sense **evident immediately** – i.e. without the necessity of proving preliminary theorems. [Recursiveness and λ-definability] have the advantage of **suitability for embodiment in a system of symbolic logic**. (Church, 1936)
"Turing's definition of computability was **intrinsically plausible**, whereas with the other two [recursive functions and λ-definability], a person became convinced only after he investigated and found, much by surprise, **how much could be done with the definition**." (Kleene in an interview with Aspray, 1985)

$\Rightarrow$ (**Historically speaking, there is no such thing as CTT!**)

Post's two theses

Thesis I: Generating sequences and limits of the computable



Post's ambitions in the 20s

⇒ Various documents: (PhD, *Account of an anticipation*, *Note on a fundamental problem in postulate theory*)

⇒ **Purpose?** Research in foundations: "*[T]o obtain theorems about all [mathematical] assertions*" – proof consistency, completeness and decidability of propositional logic

⇒ **Approach?** Development of a "*general form of symbolic logic*" as an "*instrument of generalization*" characterized by the "*method of combinatory iteration*" which "*eschews all interpretation*" – modeling (processes of) symbolic logic ($\sim$ Lewis' "mathematics without meaning"):

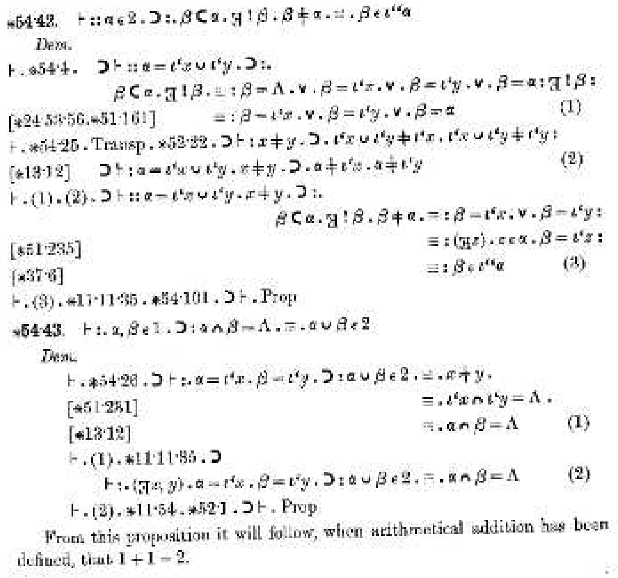
> [T]he method of combinatory iteration **completely neglects [...] meaning**, and considers the entire system purely from the symbolic standpoint as one in which both the enunciations and assertions are groups of symbols or symbol-complexes [....] and where these symbol assertions are obtained by starting with certain initial assertions and repeatedly applying certain rules for obtaining new symbol-assertions from old.

⇒ **1920-21**: Deciding the "*finiteness problem*" for first-order logic

> "*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.*" (Davis, 1994)

Post’s method at work: Generalization through formalization

Principia



$*54{\cdot}42.\ \vdash :: \alpha \in 2 . \supset :. \beta \subset \alpha . \exists ! \beta . \beta \neq \alpha . \equiv . \beta \in \iota `` \alpha$

Dem.

$\vdash . *54{\cdot}4 . \quad \supset \vdash :: \alpha = \iota ` x \cup \iota ` y . \supset :.$

$\qquad \beta \subset \alpha . \exists ! \beta . \equiv : \beta = \Lambda . \vee . \beta = \iota ` x . \vee . \beta = \iota ` y . \vee . \beta = \alpha : \exists ! \beta :$

$[*24{\cdot}53{\cdot}56 . *51{\cdot}161] \qquad \equiv : \beta = \iota ` x . \vee . \beta = \iota ` y . \vee . \beta = \alpha \qquad (1)$

$\vdash . *54{\cdot}25 . \mathrm{Transp} . *52{\cdot}22 . \supset \vdash : x \neq y . \supset . \iota ` x \cup \iota ` y \neq \iota ` x . \iota ` x \cup \iota ` y \neq \iota ` y :$

$[*13{\cdot}12] \quad \supset \vdash : \alpha = \iota ` x \cup \iota ` y . x \neq y . \supset . \alpha \neq \iota ` x . \alpha \neq \iota ` y \qquad (2)$

$\vdash . (1) . (2) . \supset \vdash :: \alpha = \iota ` x \cup \iota ` y . x \neq y . \supset :.$

$\qquad \beta \subset \alpha . \exists ! \beta . \beta \neq \alpha . \equiv : \beta = \iota ` x . \vee . \beta = \iota ` y :$

$[*51{\cdot}235] \qquad \equiv : (\exists z) . z \in \alpha . \beta = \iota ` z :$

$[*37{\cdot}6] \qquad \equiv : \beta \in \iota `` \alpha \qquad (3)$

$\vdash . (3) . *11{\cdot}11{\cdot}35 . *54{\cdot}101 . \supset \vdash . \mathrm{Prop}$

$*54{\cdot}43.\ \vdash :. \alpha , \beta \in 1 . \supset : \alpha \cap \beta = \Lambda . \equiv . \alpha \cup \beta \in 2$

Dem.

$\vdash . *54{\cdot}26 . \supset \vdash :. \alpha = \iota ` x . \beta = \iota ` y . \supset : \alpha \cup \beta \in 2 . \equiv . x \neq y .$

$[*51{\cdot}231] \qquad \equiv . \iota ` x \cap \iota ` y = \Lambda .$

$[*13{\cdot}12] \qquad \equiv . \alpha \cap \beta = \Lambda \qquad (1)$

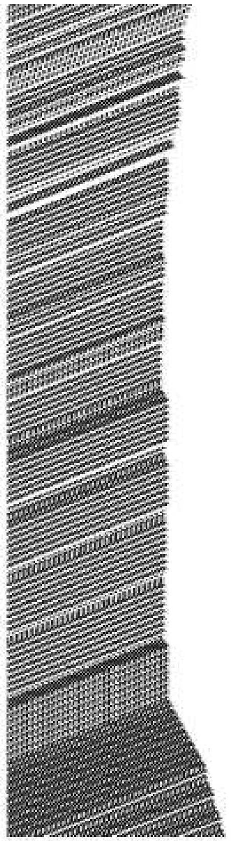
$\vdash . (1) . *11{\cdot}11{\cdot}35 . \supset$

$\qquad \vdash :. (\exists x , y) . \alpha = \iota ` x . \beta = \iota ` y . \supset : \alpha \cup \beta \in 2 . \equiv . \alpha \cap \beta = \Lambda \qquad (2)$

$\vdash . (2) . *11{\cdot}54 . *52{\cdot}1 . \supset \vdash . \mathrm{Prop}$

From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

Canonical Form
A&B



Tag Systems

Generalization I: Systems in canonical form A

	Propositional Logic	**Canonical form A**
I.	If p is an elementary proposition than so is $\sim p$ If p and q are elementary propositions than so is $p \vee q$	If $p_1, \dots, p_{m_1}$ are elementary propositions than so i $f_1(p_1, \dots, p_{m_1})$ $\vdots$ If $p_1, \dots, p_{m_\mu}$ are elementary propositions than so i $f_\mu(p_1, \dots, p_{m_\mu})$
II.	The assertion of a function involving a variable p produces the assertion of any function found from the given one by substituting for p any other variable q, or $\sim q$, or $(q \vee r)$[a]	The assertion of a function involving a variable p pro- duces the assertion of any function found from the given one by subsituting for p any other variable q, o $f_1(q_1, \dots, q_{m_1})$, or $f_\mu(q_1, \dots, q_{m_\mu})$
III.	$\vdash P$ $\vdash \sim P \vee Q$ produce $\vdash Q$	$\vdash g_{11}(P_1, \ldots, P_{k_1}) \ldots \vdash g_{r_1}(P_1, \ldots, P_{r_r})$ $\vdots$ $\vdash g_{1r_1}(P_1, \ldots, P_{r_1}) \ldots g_{rr_r}(P_1, \ldots, P_{r_r})$ produce produce $\vdash g_1(P_1, \ldots, P_{k_1}) \ldots \vdash g_r(P_1, \ldots, P_{r_r})$

[a]This corresponds to substitution

Continued on next page

Table 1 – continued from previous page		
	Propositional Logic	**Canonical form A**
IV.	Postulates:	Postulates:
	$\vdash\sim(p \vee p) \vee p$	$\vdash h_1(p_1, p_2, \ldots, p_{l_1})$
	$\vdash\sim(p \vee (q \vee r)) \vee (q \vee (p \vee r))$	$\vdash h_2(p_1, p_2, \ldots, p_{l_2})$
	$\vdash\sim q \vee (p \vee q)$	$\ldots$
	$\vdash\sim(\sim q \vee r) \vee (\sim(p \vee q). \vee (p \vee r))$	$\ldots$
	$\vdash (p \vee q) \vee (q \vee p)$	$\vdash h_\lambda(p_1, p_2, \ldots, p_{l_\lambda})$

Generalization II: Tag systems

"The general problem [of **determining for any two expressions in any system in canonical form A, what substitutions would make those expressions identical**] proving intractable, successive simplifications thereof were considered, one of the last being [the] problem of "tag." Again, after the finiteness problem for systems in canonical form A involving primitive functions of only one argument was solved, an attempt to **solve the problem for systems going, it seemed, but a little beyond this one argument case**, led once more essentially to the selfsame problem of "tag." The solution of this problem thus appeared as a vital stepping stone in any further progress to be made."

Generalization II: Tag systems

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

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

⇒ Definition of a *class* of symbolic logics according to a form

⇒ Very much in the spirit of the method of combinatory iteration – pure symbol manipulators without meaning.

⇒ Study of two decision problems (finiteness problems) for tag systems: the halting and reachability problem starting from the simplest case to the more ‘complex’ ones ($\mu = 1, 2, 3, ..., v = 1, 2, 3...$ – unpublished manuscript)

Generalization II: Tag systems

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

$A_0 = 10111011101000000$ ⇒ Primitive assertion

~~101~~110111010000001101

~~110~~1110100000011011101

~~111~~01000000110111011101

~~010~~0000011011101110100

~~000~~001101110111010000

~~001~~$\underbrace{10111011101000000}_{A_0}$ ⇒ Periodicity!

⇒ Definition of a *class* of symbolic logics according to a form

⇒ Very much in the spirit of the method of combinatory iteration – pure symbol manipulators without meaning.

⇒ Study of two decision problems (finiteness problems) for tag systems: the halting and reachability problem starting from the simplest case to the more 'complex' ones ($\mu = 1, 2, 3, ..., v = 1, 2, 3...$ – unpublished manuscript)

Generalization II: Tag systems

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

$A_0 = 10111011101000000 \Rightarrow$ Primitive assertion

~~101~~110111010000001101

~~110~~1110100000011011101

~~111~~01000000110111011101

~~010~~0000011011101110100

~~000~~001101110111010000

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

$\Rightarrow$ Definition of a *class* of symbolic logics according to a form

$\Rightarrow$ Very much in the spirit of the method of combinatory iteration – pure symbol manipulators without meaning.

$\Rightarrow$ Study of two decision problems (finiteness problems) for tag systems: the halting and reachability problem starting from the simplest case to the more 'complex' ones ($\mu = 1, 2, 3, ..., v = 1, 2, 3...$ – unpublished manuscript)

Generalization II: Tag systems

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

$A_0 = 10111011101000000 \Rightarrow$ Primitive assertion

~~101~~110111010000001101

~~110~~1110100000011011101

~~111~~01000000110111011101

~~010~~0000011011101110100

~~000~~001101110111010000

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

$\Rightarrow$ Definition of a *class* of symbolic logics according to a form

$\Rightarrow$ Very much in the spirit of the method of combinatory iteration – pure symbol manipulators without meaning.

$\Rightarrow$ Study of two decision problems (finiteness problems) for tag systems: the halting and reachability problem starting from the simplest case to the more 'complex' ones ($\mu = 1, 2, 3, ..., v = 1, 2, 3...$ – unpublished manuscript)

Generalization II: Tag systems

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

$A_0 = 10111011101000000 \Rightarrow$ Primitive assertion

~~101~~110111010000001101

~~110~~1110100000011011101

~~111~~01000000110111011101

~~010~~0000011011101110100

~~000~~001101110111010000

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

$\Rightarrow$ Definition of a *class* of symbolic logics according to a form

$\Rightarrow$ Very much in the spirit of the method of combinatory iteration – pure symbol manipulators without meaning.

$\Rightarrow$ Study of two decision problems (finiteness problems) for tag systems: the halting and reachability problem starting from the simplest case to the more 'complex' ones ($\mu = 1, 2, 3, ..., v = 1, 2, 3...$ – unpublished manuscript)

Generalization II: Tag systems

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

$A_0 = 10111011101000000$ ⇒ Primitive assertion

~~101~~11011101000000**1101**

~~110~~111010000001101**1101**

~~111~~0100000011011101**1101**

~~010~~00000110111011101**00**

~~000~~00011011101110100**00**

~~001~~$\underbrace{10111011101000000}_{A_0}$ ⇒ Periodicity!

⇒ Definition of a *class* of symbolic logics according to a form

⇒ Very much in the spirit of the method of combinatory iteration – pure symbol manipulators without meaning.

⇒ Study of two decision problems (finiteness problems) for tag systems: the halting and reachability problem starting from the simplest case to the more 'complex' ones ($\mu = 1, 2, 3, ..., v = 1, 2, 3...$ – unpublished manuscript)

Generalization II: Tag systems

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

$A_0 = 10111011101000000 \Rightarrow$ Primitive assertion

~~101~~1101110100000001101

~~110~~11101000000011011101

~~111~~010000000110111011101

~~010~~00000011011101110100

~~000~~001101110111010000

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

$\Rightarrow$ Definition of a *class* of symbolic logics according to a form

$\Rightarrow$ Very much in the spirit of the method of combinatory iteration – pure symbol manipulators without meaning.

$\Rightarrow$ Study of two decision problems (finiteness problems) for tag systems: the halting and reachability problem starting from the simplest case to the more 'complex' ones ($\mu = 1, 2, 3, ..., v = 1, 2, 3...$ – unpublished manuscript)

Generalization II: Tag systems

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

$A_0 = 10111011101000000$ ⇒ Primitive assertion

~~101~~110111010000001101

~~110~~1110100000011011101

~~111~~01000000110111011101

~~010~~0000011011101110100

~~000~~001101110111010000

~~001~~$\underbrace{10111011101000000}_{A_0}$ ⇒ Periodicity!

⇒ Definition of a *class* of symbolic logics according to a form

⇒ Very much in the spirit of the method of combinatory iteration – pure symbol manipulators without meaning.

⇒ Study of two decision problems (finiteness problems) for tag systems: the halting and reachability problem starting from the simplest case to the more 'complex' ones ($\mu = 1, 2, 3, ..., v = 1, 2, 3...$ – unpublished manuscript)

The frustrating problem of "Tag" and the reversal of Post's programme

$\Rightarrow$ **Exploring tag systems: pencil-and-paper computations and "observations"**

- "Observation" of three classes of behavior: periodicity, halt, unbounded growth.
- Three decidable classes ($v = 1$;$\mu = 1$; $\mu = v = 2$) (Wang, 1963; De Mol, 2010) – the proof involved "*considerable labor*"
- Infinite class with $\mu = 2, v = 3$: "intractable" (Minsky, 1967; De Mol, 2011)
- Infinite class with $\mu > 2, v = 2$: a zoo of TS of "bewildering complexity"

$\Rightarrow$ *Principia* vs. Lewis-like Abstract form ("mathematics without meaning") $\longrightarrow$ shift to an analysis of the behavior $\longrightarrow$ limitations of Lewis' ideal mathematics

$\Rightarrow$ **The reversal rooted in experience** "[T]he general **problem of "tag" appeared hopeless, and with it our entire program of the solution of finiteness problems.** This *frustration* [my emphasis], however, was largely based on the assumption that "tag" was but a minor, if essential, stepping stone in this wider program." (Post,1965)

After nine months of tagging ....

⇒ Development of two more forms: **canonical form** C (Post production systems):

$g_{11}P_{i_1^1}g_{12}P_{i_2^1}\dots g_{1m_1}P_{i_{m_1}^1}g_{1(m_1+1)}$	$abaP_1abP_2b$	$abaaaaaabababbaaaaab$
$g_{21}P_{i_1^2}g_{22}P_{i_2^2}\dots g_{2m_2}P_{i_{m_2}^2}g_{2(m_2+1)}$	$\underline{a}P_3\underline{bb}P_4\underline{b}$	$\underline{a}baaaaaababa\underline{bb}ababbaaaaa\underline{b}$
....	....	....
$g_{k1}P_{i_1^k}g_{k2}P_{i_2^k}\dots g_{km_k}P_{i_{m_k}^k}g_{k(m_k+1)}$		
produce	**produce**	**produce**
$g_1P_{i_1}g_2P_{i_2}\dots g_mP_{i_m}g_{(m+1)}$	$abaP_3b$	$ababaaaaaababab$

.... and **Normal form:**

g_iP_i	$1101P_i$	~~1101~~11011101000000
	produces	
$P_ig_{i'}$	P_i001	11011101000000001

... towards Post's thesis I

⇒ Insight that apparent simplicity does not imply 'real' simplicity: Proof of *"the most beautiful theorem in mathematics*" (Minsky, 1961)

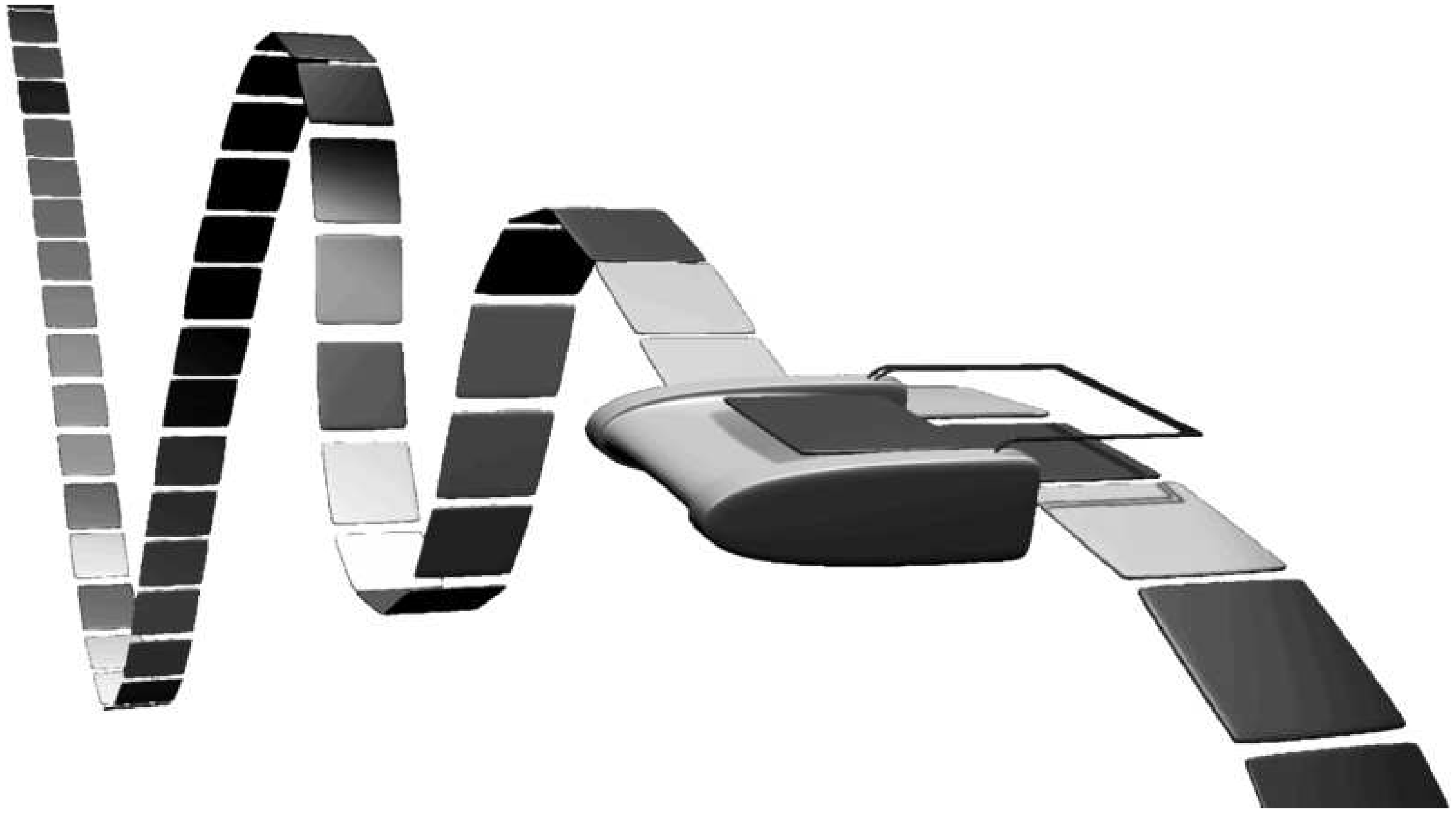
⇒ Idea that the whole PM can be reduced to normal form

> [F]or if the meager formal apparatus of our final normal systems can wipe out all of the additional vastly greater complexities of canonical form [...], the more complicated machinery of [the canonical form] should clearly be able to handle formulations correspondingly more complicated than itself.

⇒ **Post's thesis I** – any set of sequences that can be "generated" by some finite process can also be "generated" by the "primitive" normal form

⇒ The finiteness problem for normal form is unsolvable

Thesis II: Solvability and the realm of the computable



Taking into account the human factor in generating sets....

Problem with thesis I: Post's believe in thesis I rooted in his own experiences and interaction with his forms ($\sim$ Church and Kleene) $\Rightarrow$ less convincing for people not familiar with these forms (see e.g. correspondence Church-Post: "*while it is clear that every generated set in your sense is lambda-enumerable (recursively enumerable), I can see no way of proving the converse of this, and at the moment, therefore,* **it seems to me possible that the notion of a generated set is less general**." (Church to Post, June 26, 1936)

Post's analysis: "*[for the thesis to obtain its full generality] an analysis should be made of all the possible ways* **the human mind** *can set up finite processes to generate sequences.*" ($\sim$ Turing's "What are the possible processes which can be carried out in computing a number?")

> "[E]stablishing this universality [of the characterization of generated set of sequences in terms of normal form] is not a matter for mathematical proof, but of **psychological analysis of the mental processes involved in combinatory mathematical processes**.

$\Rightarrow$ **Post's solution**: Identification between Solvability and Formulation 1 (almost identical to Turing machines)

... opposition against definitional character of Church's thesis

A working hypothesis "*Its purpose is not only to present a system of a certain logical potency but also, [...] of psychological fidelity*"

> [T]o 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.

"The writer cannot overemphasize the fundamental importance to mathematics of the existence of absolutely unsolvable combinatory problems. True, with a specific criterion of solvability under consideration, say recursiveness [...], the unsolvability in question [...] becomes merely unsolvability by a given set of instruments. **[The] fundamental new thing is that for the combinatory problems the given set of instruments is in effect the only humanly possible set.**" (Post, 1965)

Post's new programme – Towards a natural law In search of wider and wider formulations and to prove that all these are logically reducible to the original formulation 1

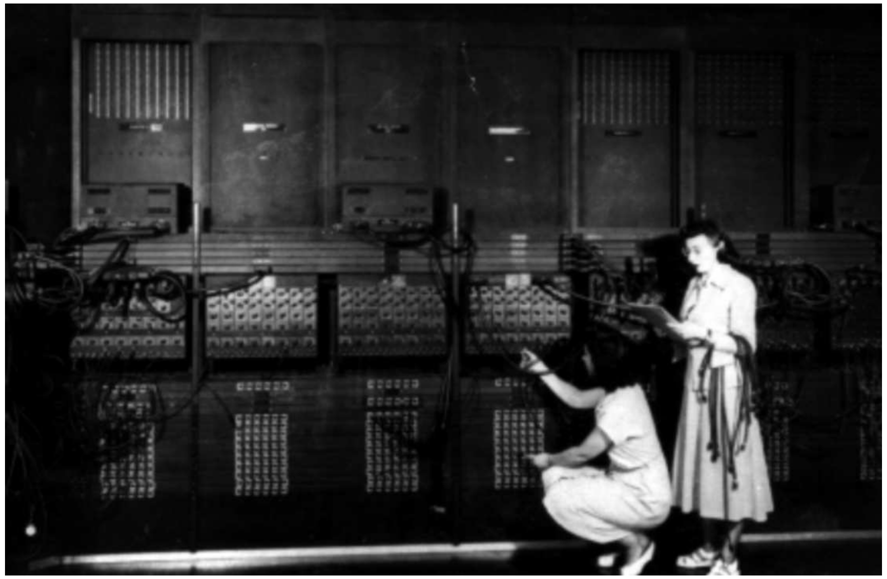
$\Leftrightarrow$ "In sharp contrast, Turing attempts to give an analytic argument for the claim that these simple processes are sufficient to capture all human mechanical calculations." (Sieg, 2005)

“When the bubble of symbolic logic finally burst” The rise of computer science

THE WORLD'S ONLY RELIABLE NEWSPAPER
COMPUTER VIRUS SPREADS TO HUMANS!
BAR GLASSES HELP YOU SEE STRAIGHT WHEN YOU'RE DRUNK!

“....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. (Post, 1965)

The rise of *computer* science...



- Late 40s: development of electronic and programmable machine to “relieve th[e] bottleneck of slow manual and analogue computation
- It is the machine’s speed that requires the introduction of logic into the machine: “[C]ontemplate the prospect of locking twenty people for two years during which they would be steadily performing computations. And you must give them such explicit instructions at the time of incarceration that at the end of two years you could return and obtain the correct result for your lengthy problem! This dramatizes the necessity for high planning, foresight, and consideration of the logical nature of computation. **This integration of logic in the problem is a consequence of the high speed.** ” (Von Neumann, 1948)
- Steady introduction of “formal methods” in CS practices

⇒ Leading question *Why* does CS-as-a-practice require formal methods? What kind of methods are suitable to the purpose?

⇒ Rediscovery of formal work on computability in context of programming and machine design – different roles for Church, Post and Turing

The significance of Turing's model for practices of CS

- ??The stored-program?? – an afterthought
- The (U)TM as a model for real machines (automata studies and real machines)

  "Let us imagine the operations performed by the computer to be split up into simple operations which are so elementary that it is not easy to imagine them further divided." (Turing, 1936)

  "[It is not] economically feasible to use a machine to perform complicated operations because of the extreme slowness and fairly large amount of memory required. [...] [It is suggested] that **it may be possible to reduce the number of components required for logical control** purposes, particularly if any cheap memory devices are developed." (Moore, 1952)

  "In this article will be described the logical principles of an electronic digital computer which has been **simplified to the utmost practical limit** at the sacrifice of speed." (van der Poel, 1952)

- ... and as a means to express the **fundamental interchangeability of instructions and data** which allow to express the notion of a general-purpose machine in a formal manner
- "**An impossible program**", Strachey, 1965: "A well-known piece of folklore among programmers holds that it is impossible to write a program which can examine any other program and tell, in every case, if it will terminate or gte into a closed loop when it is run"

$\Rightarrow$ Turing's UTM as a tool to reflect on the theoretical characterization of computers and programs (not about CTT however) $\Rightarrow$ But (!) "**programs unclear and conceptually not helpful**" (Backus, 1978)

The significance of Post's model for practices of CS

- **Backus-Naur form** (Notation and programming!): "It was only in trying to describe ALGOL 58 that I realized that there was trouble about syntax description. *It was obvious that Post's productions were just the thing* [m.i.], and I hastily adapted them to that use." (Backus, 1981)

  $<$letter string$>$ ::= $<$letter$>$ | $<$letter string$><$letter$>$

  $<$letter$>$ ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | Y | W | X | Y | Z

- Patterson and Burroughs company: "We are interested in your "productions" and your theory of canconical languages, which *appeal to us as possibly the most natural approach to the theory of computabiliy from the standpoint of one interested in syntactical machines*." (letter to Post, Dec. 17, 1952)

- **Chomsky and formal language theory**: Post's canonical form C = Post production systems: "I was interested at the time in automata theory and possible applications to linguistics. I'd studied standard versions of recursive function theory (Kleene, etc.), but when I came across Post's work (in Davis) *it was obvious that this was a good framework for systems of the sub-recursive hierarchy that could be adapted to the study of language*, specifically context-sensitive and context free grammars" (personal communication, 2005) $\Rightarrow$ formal languages in compiler design – accepting a particular string with the grammar formalizing the PL

$\Rightarrow$ Post's influence mostly on the **syntactical level of programs** – development of devices which generate sequences as a model for "formal" reasoning ideal for the study of "rewriting" A to B

The significance of Church’s model for practices of CS

Use of λ-calculus to develop and reason about **functional languages**: pure and impure functional languages (eg pure LISP vs. Scheme)

“*A program is a function of one variable*” (Rosser, 1984): $a + b$ is really $\lambda x(\lambda y.x + y)ab$

λ-calculus as thé programming language ($\neq$ TMs), viz. trh Church-Landin thesis: “*Programming languages are λ-calculus sweetened with syntactic sugar*” (Traktenbrot, 1988) – ”*Syntactic sugar causes cancer of the semicolon*” (Perlis) $\sim$ the search for the ultimate PL (the ALGOl program)

The λ-calculus as a calculus for reasoning about the meaning of programs (denotational semantics): ”*The problem of explaining [...] equivalences of expressions (whether in the same or different languages) is one of the tasks of semantics and is much too important to be left to syntax alone/ Besides, the mathematical concepts are required for the proof that the various equivalences have been correctly described*” (Scott and Strachey, 1971)
Denotation maps input into output; function from environment to a denotation

Discussion – afterthoughts

Discussion – afterthoughts: Value historical study of early models?

⇒ Different questions result in different models. And the differences do matter!!

- (Post) Finding the ultimate form without much meaning results in a syntax which makes possible automatic treatment exactly because it is so formal
- (Church) The development of a system which is basically a calculus for functions and focusdes on the notion of variable *is* fundamental to develop theories of programming
- (Turing) The top-down analysis of Turing of the notion of computability results in a model that allows to reflect on the limitations and characterization of computers as abstract programmable devices

⇒ The differences between the old models *do* matter retrospectively – CTT as a generic term hides this nuance

⇒ In focusing too much on one hero of a discipline one sees less clearly a fundamental characteristic of practices of CS which require domain-specific developments

⇒ (in retrospect) Models that go “beyond” Turing? The Wegner mistake and a missed chance