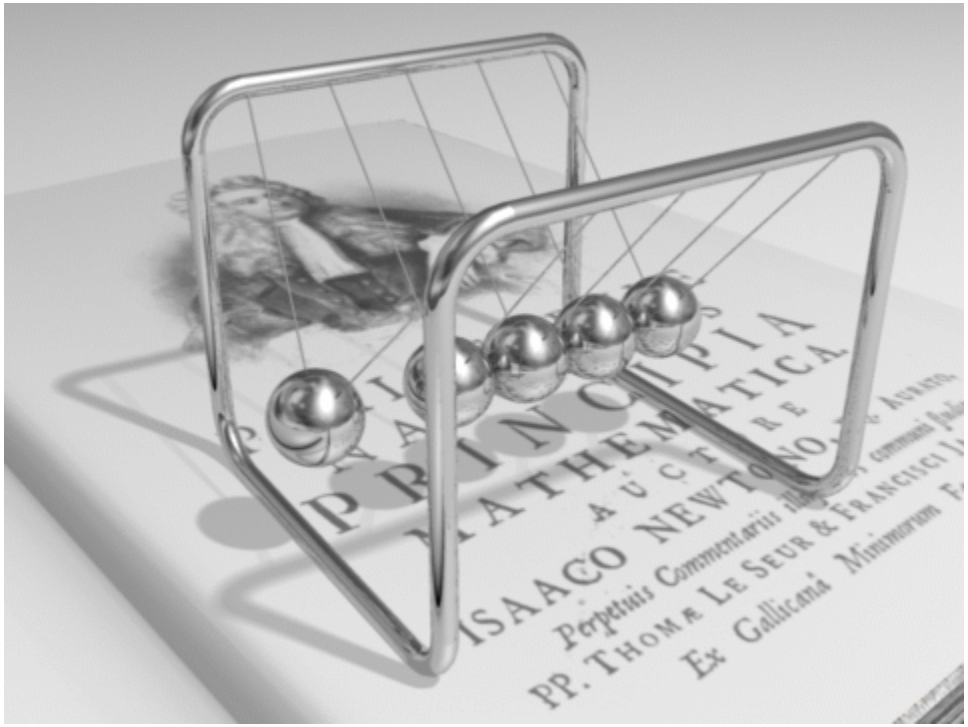
Jean-Louis Giavitto
IBISC, CNRS & Université d'Evry

Ça se calcule, c'est un calcul, *parce que* ça calcule ça



Plan

A. Nasses, filets et lacets : définir le calcul

B. Qu'est ce qu'on a voulu attraper, qu'est ce qu'on a effectivement attrapé : les limitations internes de la « calculabilité *effective* »

C. Ça se calcule : Computational Science :

D. C'est un calcul ou la Nature comme machine (où la « naturalisation » comme une mécanisation)

E. Parce que ça calcule ça : calcul et causation

Contents

A. Nasses, filets et lacets : la « curieuse définition » du calcul
   1. 1930-1940 : la chasse est ouverte : des définitions concurrentes avec des visées différentes
      a. les fonctions récursives de Kleene (1935)
      b. la machine de Post (1936)
      c. le lambda-calcul de Church (1936)
      d. la machine de Turing (1937)
   2. la thèse de Church-Turing pour « universaliser » ce qu'on a attrapé
      a. équivalence par simulation
      b. la machine universelle
      c. de la simulation à la traduction

B. Les limitations internes de la « calculabilité *effective* »
   1. Il y a des choses qu'on ne peut pas calculer
      a. le problème de l'arrêt
      b. calculer sur les réels
   2. Ce qu'on évite : l'infini actuel
   3. ce dont ne veut pas vraiment parler (l'implémentation finale) : une approche *fonctionnaliste*
   4. Ce dont on veut parler : les 700 prochains langages de programmation, opérations primitives, oracle et contrôle, non-déterminisme
   5. Ce dont on veut parler peut être fade : non-déterminisme, interaction, aléatoire

C. Computational Science : ça se calcule
   1. Exemple : computational physics (mécanique newtonienne avec le livre de Sussman)
   2. Physical Computation : calculer avec des processus physiques

Contents

D. La Nature comme machine (où la « naturalisation » comme une mécanisation) : c'est un calcul
   1. la « naturalisation » comme une mécanisation
   2. Descartes (le vivant comme machine et la « preuve » de Von Neumann)
   3. l'Allemand,
   4. Fredkin, Deutsch et les autres…
   5. un précédent : penser et calculer
      a. la thèse (Hobbes, Leibniz, Boole et les autres)
      b. les opposants : Searle,

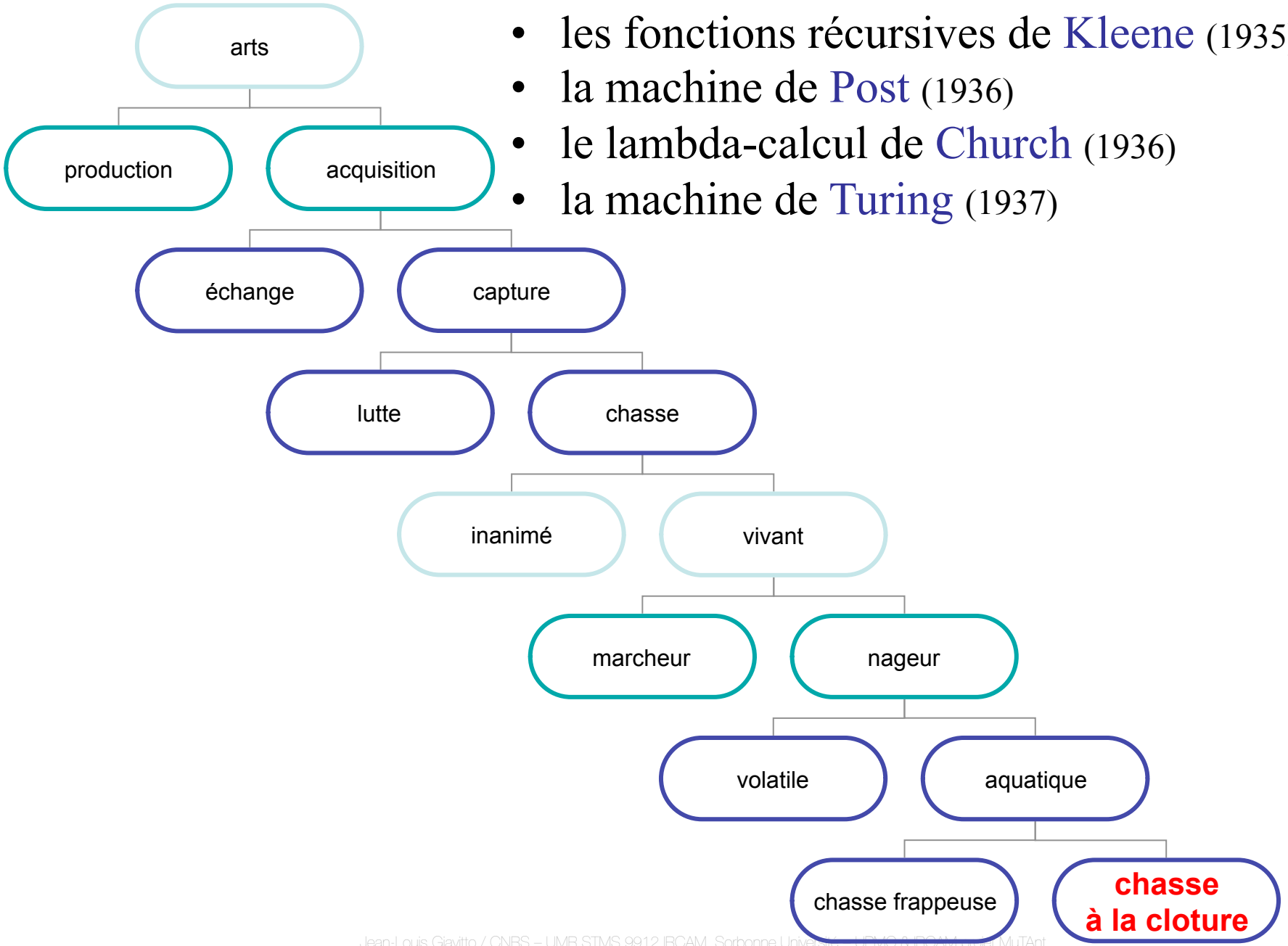
E. Hyper-computation
   1. Y a t'il des systèmes physiques qui calculent plus que ce qu'on calcule effectivement? exhiber un pb. de décision non Turing solvable ! (pas seulement un speed-up)
   2. Des tentatives peu convaincantes
      a. des machines qui accélèrent
      b. des machines analogiques
   3. Mais si la Nature ne calcule pas *alors* on pourra étendre la notion de calcul

F. Parce que ça calcule ça : calcul et causation
   1. explication et causalité
   2. calcul comme intelligibilité comme causation : relier des entrées à des sorties, systèmes dynamiques, une logique de l'observable, de l'équation à la résolution
   3. les problèmes auxquels cela ne répond pas
      a. Lois de la Nature et calcul : comment privilégier un point de vue
      b. Calcul et complexité et prédiction (La vie rêvée des maths) et prédiction
      c. Les boas ouverts et les boas fermés : au-delà, il y a le magma

Nasses, filets et lacets :
la « curieuse définition » du calcul

Cerner le concept

- les fonctions récursives de Kleene (1935)
- la machine de Post (1936)
- le lambda-calcul de Church (1936)
- la machine de Turing (1937)



Cerner le concept

arts

General recursive functions of natural numbers[1].

Von

S. C. Kleene in Madison (Wis., U.S.A.).

The substitution

1) $$\varphi(x_1, \ldots, x_n) = \theta(\chi_1(x_1, \ldots, x_n), \ldots, \chi_m(x_1, \ldots, x_n)),$$

and the ordinary recursion with respect to one variable

(2) $$\varphi(0, x_2, \ldots, x_n) = \psi(x_2, \ldots, x_n)$$
$$\varphi(y+1, x_2, \ldots, x_n) = \chi(y, \varphi(y, x_2, \ldots, x_n), x_2, \ldots, x_n),$$

where $\theta, \chi_1, \ldots, \chi_m, \psi, \chi$ are given functions of natural numbers, are examples of the definition of a function φ by equations which provide a step by step process for computing the value $\varphi(k_1, \ldots, k_n)$ for any given set $k_1, \ldots, k_n$ of natural numbers. It is known that there are other definitions of this sort, e. g. certain recursions with respect to two or more variables simultaneously, which cannot be reduced to a succession of substitutions and ordinary recursions[2]). Hence, a characterization of the notion of recursive definition in general, which would include all these cases, is desirable. A definition of general recursive function of natural numbers was suggested by Herbrand to Gödel, and was used by Gödel with an important modification in a series of lectures at Princeton in 1934. In this paper we offer several observations on general recursive functions, using essentially Gödel's form of the definition.

The definition will be stated in § 1. It consists in specifying the form of the equations and the nature of the steps admissible in the computation of the values, and in requiring that for each given set of arguments the computation yield a unique number as value. The operations on symbols which occur in the computation have a similarity to ordinary recursive operations on numbers. This similarity will be utilized, by the Gödel method of representing formulas by numbers, to prove that every (general) recursive function is expressible in the form $\psi(\varepsilon y[\varrho(x_1, \ldots, x_n, y) = 0])$ where ψ and ϱ are ordinary or „primitive"

[1]) Presented to the American Mathematical Society, September 1935.

[2]) W. Ackermann, Zum Hilbertschen Aufbau der reellen Zahlen, Math. Annalen 99 (1928), S. 118—133; Rózsa Péter, Konstruktion nichtrekursiver Funktionen, Math. Annalen 111 (1935), S. 42—60.

les fonctions récursives de Kleene (1935)

qu'est qu'une définition récursive acceptable pour une fonction arithmétique

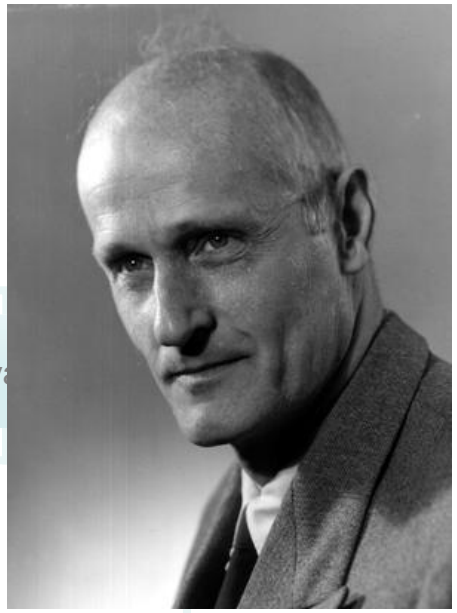
ture

chasse

imé

viva

marcheur



volatile

aquatique

chasse frapeuse

chasse à la cloture

Cerner le concept

arts

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By Alonzo Church.

1. Introduction. There is a class of problems of elementary number theory which can be stated in the form that it is required to find an effectively calculable function f of n positive integers, such that $f(x_1, x_2, \cdots, x_n) = 2$ [2] is a necessary and sufficient condition for the truth of a certain proposition of elementary number theory involving $x_1, x_2, \cdots, x_n$ as free variables.

An example of such a problem is the problem to find a means of determining of any given positive integer n whether or not there exist positive integers x, y, z, such that $x^n + y^n = z^n$. For this may be interpreted, required to find an effectively calculable function f, such that $f(n)$ is equal to 2 if and only if there exist positive integers x, y, z, such that $x^n + y^n = z^n$. Clearly the condition that the function f be effectively calculable is an essential part of the problem, since without it the problem becomes trivial.

Another example of a problem of this class is, for instance, the problem of topology, to find a complete set of effectively calculable invariants of closed three-dimensional simplicial manifolds under homeomorphisms. This problem can be interpreted as a problem of elementary number theory in view of the fact that topological complexes are representable by matrices of incidence. In fact, as is well known, the property of a set of incidence matrices that it represent a closed three-dimensional manifold, and the property of two sets of incidence matrices that they represent homeomorphic complexes, can both be described in purely number-theoretic terms. If we enumerate, in a straightforward way, the sets of incidence matrices which represent closed three-dimensional manifolds, it will then be immediately provable that the problem under consideration (to find a complete set of effectively calculable invariants of closed three-dimensional manifolds) is equivalent to the problem, to find an effectively calculable function f of positive integers, such that $f(m, n)$ is equal to 2 if and only if the m-th set of incidence matrices and the n-th set of incidence matrices in the enumeration represent homeomorphic complexes.

Other examples will readily occur to the reader.

[1] Presented to the American Mathematical Society, April 19, 1935.

[2] The selection of the particular positive integer 2 instead of some other is, of course, accidental and non-essential.

345

le lambda-calcul de Church (1936)

qu'est qu'une fonction définie intentionnellement par des règles de calcul

chasse

vivant

marcheur

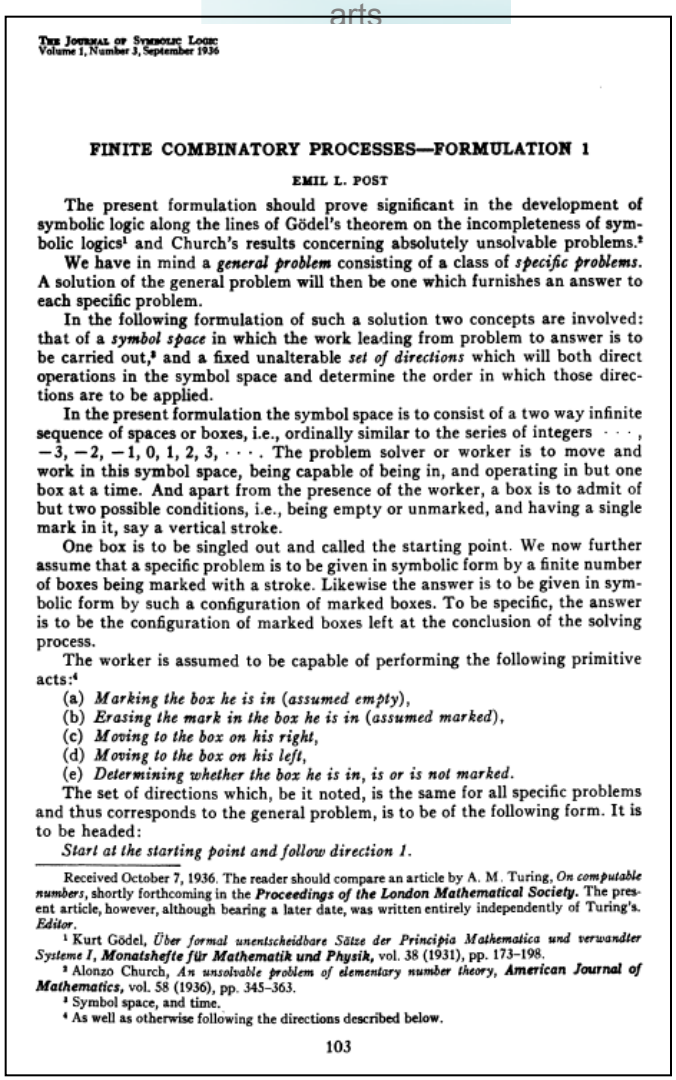
volatile

aquatique

chasse frapeuse

chasse à la cloture

Cerner le concept



THE JOURNAL OF SYMBOLIC LOGIC
Volume 1, Number 3, September 1936

FINITE COMBINATORY PROCESSES—FORMULATION 1

EMIL L. POST

The present formulation should prove significant in the development of symbolic logic along the lines of Gödel's theorem on the incompleteness of symbolic logics[1] and Church's results concerning absolutely unsolvable problems.[2]

We have in mind a *general problem* consisting of a class of *specific problems*. A solution of the general problem will then be one which furnishes an answer to each specific problem.

In the following formulation of such a solution two concepts are involved: that of a *symbol space* in which the work leading from problem to answer is to be carried out,[3] and a fixed unalterable *set of directions* which will both direct operations in the symbol space and determine the order in which those directions are to be applied.

In the present formulation the symbol space is to consist of a two way infinite sequence of spaces or boxes, i.e., ordinally similar to the series of integers $\cdots, -3, -2, -1, 0, 1, 2, 3, \cdots$. The problem solver or worker is to move and work in this symbol space, being capable of being in, and operating in but one box at a time. And apart from the presence of the worker, a box is to admit of but two possible conditions, i.e., being empty or unmarked, and having a single mark in it, say a vertical stroke.

One box is to be singled out and called the starting point. We now further assume that a specific problem is to be given in symbolic form by a finite number of boxes being marked with a stroke. Likewise the answer is to be given in symbolic form by such a configuration of marked boxes. To be specific, the answer is to be the configuration of marked boxes left at the conclusion of the solving process.

The worker is assumed to be capable of performing the following primitive acts:[4]

(a) *Marking the box he is in (assumed empty),*
(b) *Erasing the mark in the box he is in (assumed marked),*
(c) *Moving to the box on his right,*
(d) *Moving to the box on his left,*
(e) *Determining whether the box he is in, is or is not marked.*

The set of directions which, be it noted, is the same for all specific problems and thus corresponds to the general problem, is to be of the following form. It is to be headed:

Start at the starting point and follow direction 1.

Received October 7, 1936. The reader should compare an article by A. M. Turing, *On computable numbers*, shortly forthcoming in the *Proceedings of the London Mathematical Society*. The present article, however, although bearing a later date, was written entirely independently of Turing's. *Editor.*

[1] Kurt Gödel, *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I*, *Monatshefte für Mathematik und Physik*, vol. 38 (1931), pp. 173–198.

[2] Alonzo Church, *An unsolvable problem of elementary number theory*, *American Journal of Mathematics*, vol. 58 (1936), pp. 345–363.

[3] Symbol space, and time.

[4] As well as otherwise following the directions described below.

103

La machine de Post (1936)

mécaniser le raisonnement

arts

apture

chasse

animé

vi

marcheur

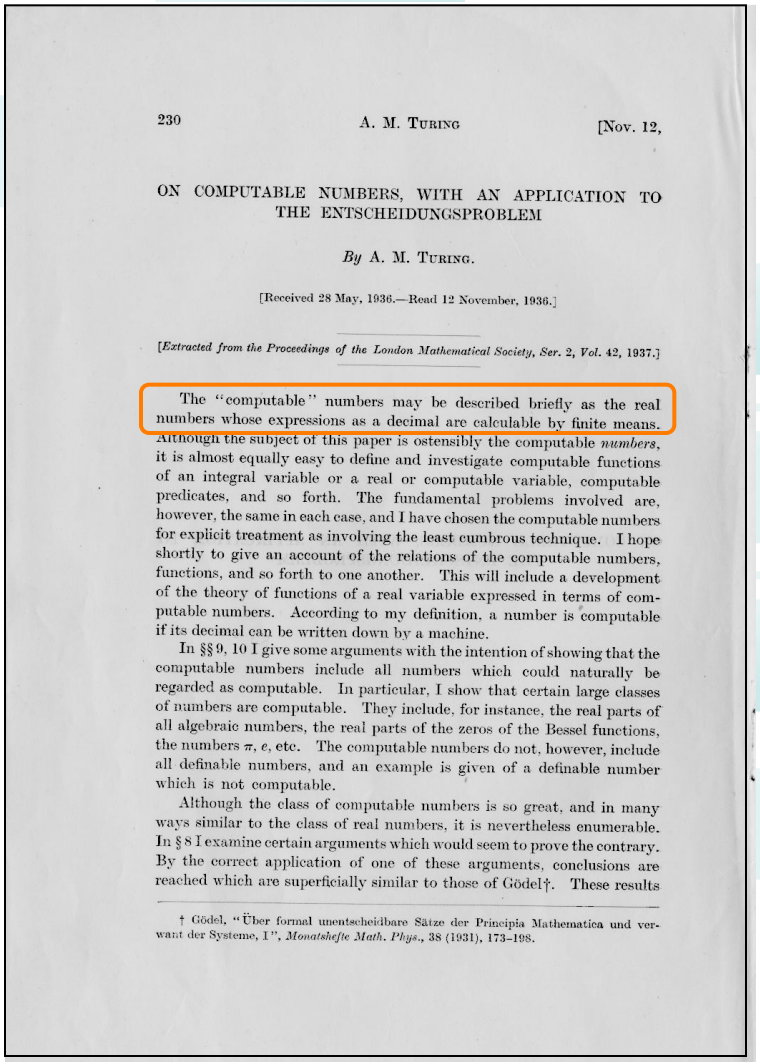
volatile

aquatique

chasse frapeuse

chasse à la cloture

Cerner le concept



230 A. M. TURING [Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

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

[Extracted from the Proceedings of the London Mathematical Society, Ser. 2, Vol. 42, 1937.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

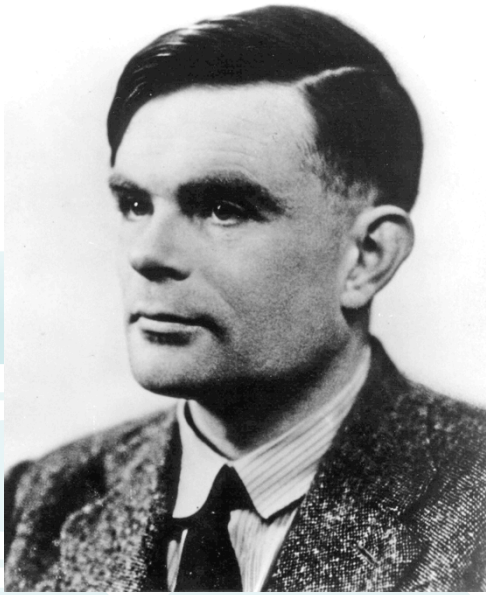
In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwant der Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.

la machine de Turing (1937)

qu'est qu'un nombre calculable,
un calcul défini par
« *some purely mechanical process* »



arts

chasse

vivant

marcheur

volatile

aquatique

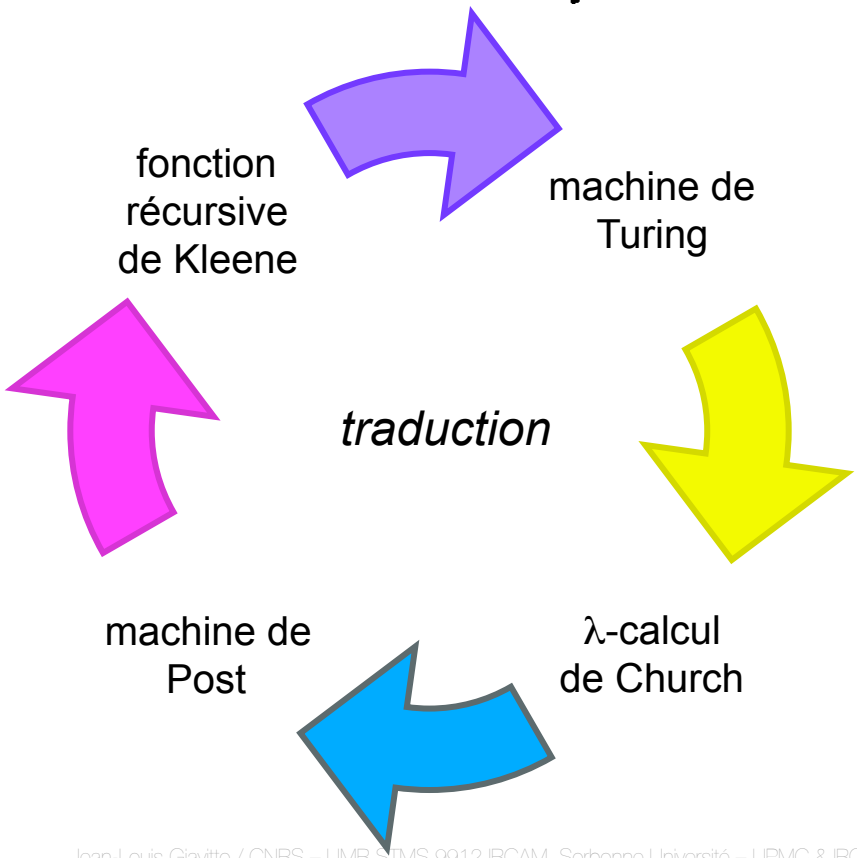
chasse frapeuse

chasse à la cloture

Comment rendre la notion de calcul « universelle » ?

The writer expects the present formulation to turn out to be logically equivalent to recursiveness in the sense of the Gödel-Church development.[7] Its purpose, however, is not only to present a system of a certain logical potency but also, in its restricted field, of psychological fidelity. In the latter sense wider and wider formulations are contemplated. On the other hand, our aim will be to show that all such are logically reducible to formulation 1. We offer this conclusion at the present moment as a *working hypothesis*. And to our mind such is Church's identification of effective calculability with recursiveness.[8] Out of this

E. Post



Effectivement calculable = mécaniquement calculable = récursif

It was stated above that « **a function is effectively calculable if its values can be found by some purely mechanical process ». We may take this statement literally, understanding by a purely mechancial process one which could be carried out by a machine**. It is possible to give a mathematical description, in a certain normal form, of the structures of these machines. The development of these ideas leads to **the author's definition of a computable function**, and to **an identification of computability † with effective calculability**.

"† We shall use the expression "computable function" to mean a function calculable by a machine, and we let "effectively calculable" refer to the intuitive idea without particular identification with any one of these definitions. We do not restrict the values taken by a computable function to be natural numbers; we may for instance have computable propositional functions."

Turing, *Systems of Logic Based on Ordinals, thèse* sous la direction de Church, qui paraît en 1939.

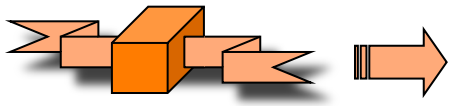
Une machine universelle

It can be shown that a single special machine of that type can be made to do the work of all. It could in fact be made to work as a model of any other machine. The special machine may be called the universal machine.

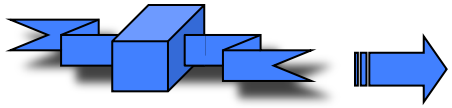
A. Turing.

- l'idée est de « coder » la table des actions par des données sur le ruban
- considéré comme la préfiguration des architectures de Von Neumann ("the stored-program computer »)
- le codage de la table des actions sur le ruban permet en principe à une machine de Turing de répondre à une question sur le comportement d'une autre machine de Turing. Exemple : le problème de l'arrêt.
- malheureusement, beaucoup de propriétés sont indécidables
    - Le théorème de Rice-Myhill-Shapiro nous dit que pour toute propriétés non triviales des fonctions partielles (i.e. qui n'est pas vrai pour toutes les fonctions partielles ou aucunes), décider si un algorithmme calcule une fonction partielle ayant cette propriété, est indécidable.
    - Mais le théorème de Rice ne nous dit rien sur les propriétés exhibées par une machine : il porte sur les fonctions partielles. Par exemple, décider si une machine exécute plus de 100 étapes sur une entrée donnée, ou bien si une machine comporte plus de 5 états, sont des propriétés non-triviales décidables.

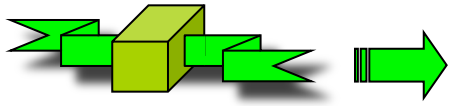
Simulation *versus* traduction



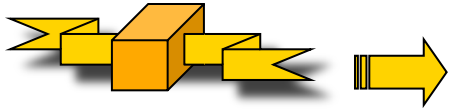
λx.x(…)



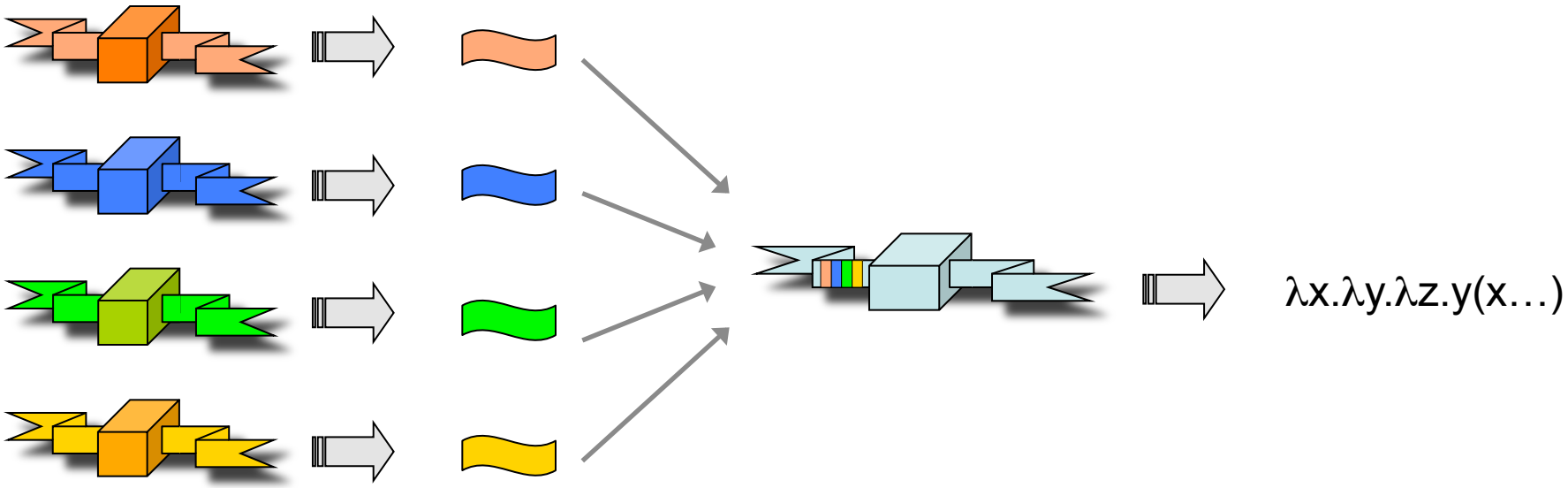
λx.λy.y(…)



λx. (…)x



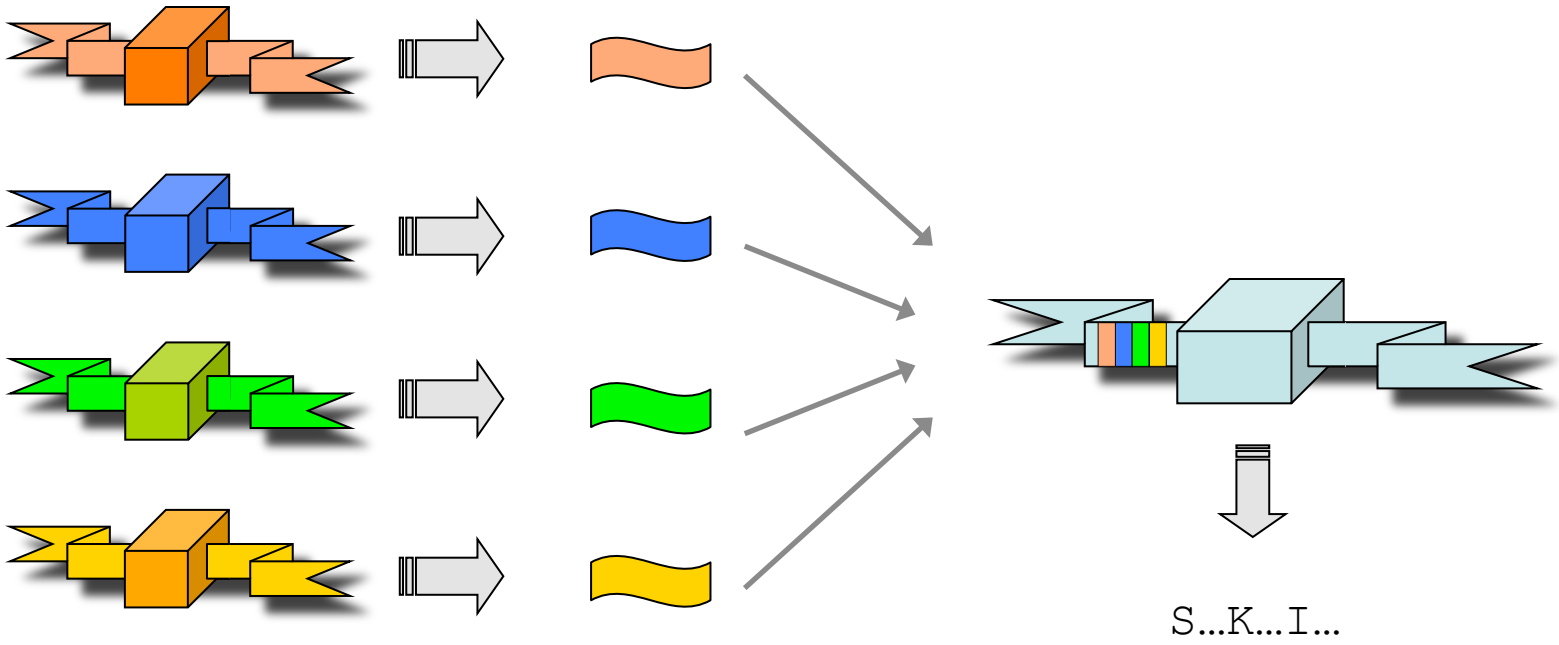
λx.λy.λz.y(x…)



λx.λy.λz.y(x…)

Simulation *versus* traduction

L'existence d'une machine universelle de Turing donne un statut plus fort à la possibilité de traduire chaque machine en une fonction récursive ou λ-définissable : il suffit d'une seule traduction, celle de la machine universelle pour être capable de traduire toute les autres.



Qu'est ce qu'on a voulu capturer

> Turing's work gives an analysis of the concept of "mechanical procedure" (alias "algorithm" or "computation procedure" or "finite combinatorial procedure"). This concept is shown to be equivalent to that of a "Turing machine". A formal system can simply be defined to be any mechanical procedure for producing formulas, called provable formulas ... the concept of formal system, whose essence it is that **reasoning is completely replaced by mechanical operations on formulas**. (Note that the question of whether there exist finite *non-mechanical* procedures ... not equivalent with any algorithm, has nothing whatsoever to do with the adequacy of the definition of "formal system" and of "mechanical procedure".
>
> *Gödel 1964*

Les propriétés attendues d'une « procédure mécanique »

1. la « procédure mécanique » consiste en un *nombre fini* d'instructions *déterministes* (i.e., déterminant un unique pas d'exécution de la procédure) et qui commande l'exécution d'un *nombre fini* d'*opérations primitives* pour un *nombre fini* de fois.
2. Les instructions sont données par une spécification *non-ambigüe* de taille *finie*.
3. La procédure ne requière *pas d'intuition* du domaine (e.g. pas d'intuitions sur les nombres et leur propriétés), pas d'invention, pas d'ingéniosité ni de devinettes.
4. Pour chaque argument de la fonction à calculer, la procédure de calcul est la même (*uniformité*).
5. La procédure produit la valeur *correcte* de la fonction pour chaque argument (*quand la procédure termine*).

La thèse de Robin Gandy:

Toute fonction qui peut être calculée par un dispositif mécanique discret déterministe est Turing-calculable

***Thèse :* Une machine de Gandy**
= dispositif mécanique discret déterministe

- **Principe I: Un système dynamique déterministe discret**
  - l'état est pris dans un ensemble héréditairement fini HF(U) construit au-dessus d'un ensemble de symboles U et stable par permutation de U
  - la fonction d'évolution F: HF(U) → HF(U) est stable pour l'automorphisme de HF(U) induit par toute permutation de HF(U)
- **Principle II: Une description de profondeur bornée *a priori***
  Chaque état est composé d'atome et d'autres (sous-)états. Mais la hiérarchie de composition est bornée : il n'y a pas de chaîne infinie décroissante de sous-états.
- **Principle III: Une composition à partir d'un répertoire fini et donné *a priori***
  L'état d'une machine de Gandy est une réunion d'états (de profondeur fini) pris dans un ensemble fini fixé.

- **Principe IV : une causalité *locale***

un état F(x) peut se définir comme une fonction d'un nombre fini de parties de x :

- *il y a une limite inférieur à la taille des composants élémentaires de la machine*
- *chaque composants a un nombre fini de voisins*
- *il y a une limite supérieur à la propagation des changements*

A priori plus général qu'une machine de Turing : permet des changements dans un nombre arbitraire de composants (contrairement à Turing où l'on ne change qu'une seule case du ruban). *Mais*

***Théorème :* Gandy calculable = Turing calculable**

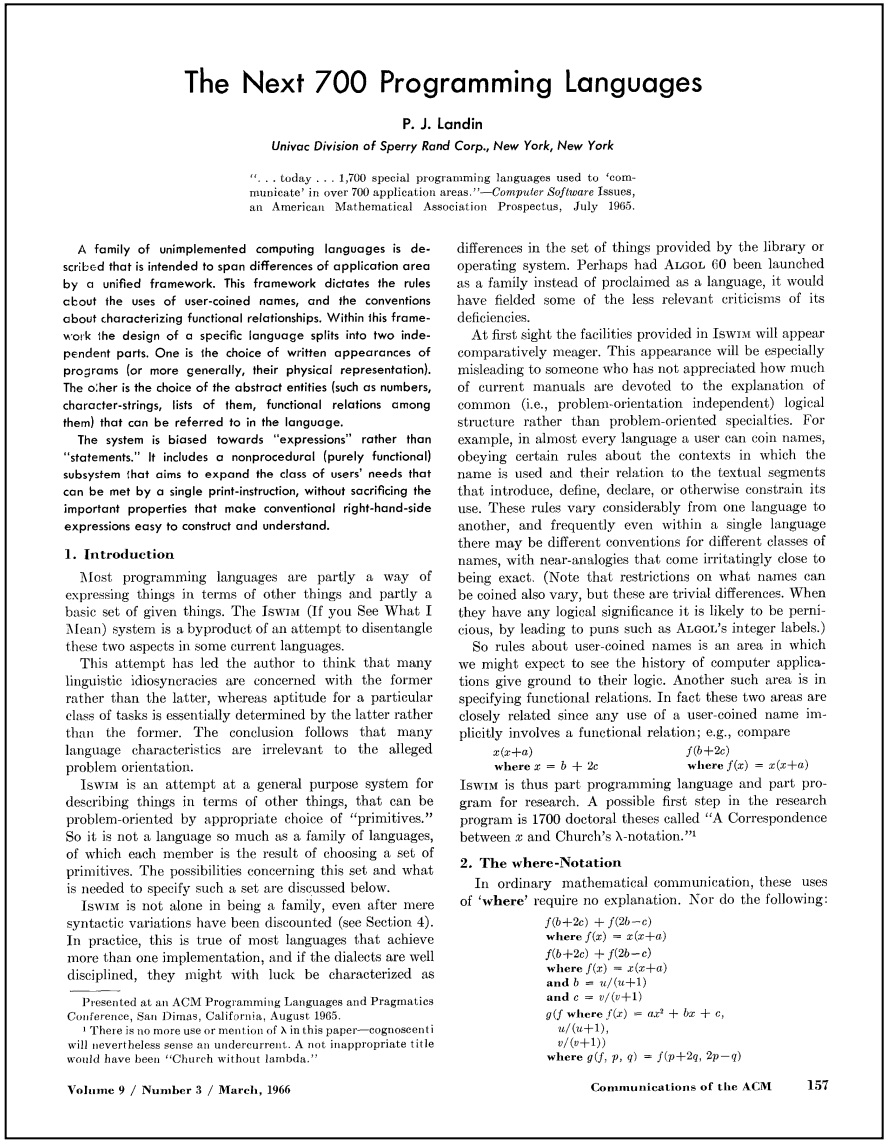
Les limitations internes de la « calculabilité effective »

Les limites

- Il y a des choses qu'on ne peut pas calculer
  - le problème de l'arrêt
  - calculer sur les réels
  - …
- Ce qu'on évite : l'infini actuel
- Ce dont ne veut pas vraiment parler : la concrétisation finale de la machine. En fait, on veut parler des programme, pas du calculateur. L'approche est *fonctionnaliste*.

  Unlike programs, computers must obey the laws of physics. (A. & S.)
- Ce dont on peut parler peut être fade vis-à-vis de ce qui devient intéressant :
  - interaction, systèmes ouverts, calcul en continu…
  - non-déterminisme
  - aléatoire

Les limites : ce dont on veut parler



The Next 700 Programming Languages

P. J. Landin

Univac Division of Sperry Rand Corp., New York, New York

"... today ... 1,700 special programming languages used to 'communicate' in over 700 application areas."—*Computer Software* Issues, an American Mathematical Association Prospectus, July 1965.

A family of unimplemented computing languages is described that is intended to span differences of application area by a unified framework. This framework dictates the rules about the uses of user-coined names, and the conventions about characterizing functional relationships. Within this framework the design of a specific language splits into two independent parts. One is the choice of written appearances of programs (or more generally, their physical representation). The other is the choice of the abstract entities (such as numbers, character-strings, lists of them, functional relations among them) that can be referred to in the language.

The system is biased towards "expressions" rather than "statements." It includes a nonprocedural (purely functional) subsystem that aims to expand the class of users' needs that can be met by a single print-instruction, without sacrificing the important properties that make conventional right-hand-side expressions easy to construct and understand.

1. Introduction

Most programming languages are partly a way of expressing things in terms of other things and partly a basic set of given things. The ISWIM (If you See What I Mean) system is a byproduct of an attempt to disentangle these two aspects in some current languages.

This attempt has led the author to think that many linguistic idiosyncracies are concerned with the former rather than the latter, whereas aptitude for a particular class of tasks is essentially determined by the latter rather than the former. The conclusion follows that many language characteristics are irrelevant to the alleged problem orientation.

ISWIM is an attempt at a general purpose system for describing things in terms of other things, that can be problem-oriented by appropriate choice of "primitives." So it is not a language so much as a family of languages, of which each member is the result of choosing a set of primitives. The possibilities concerning this set and what is needed to specify such a set are discussed below.

ISWIM is not alone in being a family, even after mere syntactic variations have been discounted (see Section 4). In practice, this is true of most languages that achieve more than one implementation, and if the dialects are well disciplined, they might with luck be characterized as differences in the set of things provided by the library or operating system. Perhaps had ALGOL 60 been launched as a family instead of proclaimed as a language, it would have fielded some of the less relevant criticisms of its deficiencies.

At first sight the facilities provided in ISWIM will appear comparatively meager. This appearance will be especially misleading to someone who has not appreciated how much of current manuals are devoted to the explanation of common (i.e., problem-orientation independent) logical structure rather than problem-oriented specialties. For example, in almost every language a user can coin names, obeying certain rules about the contexts in which the name is used and their relation to the textual segments that introduce, define, declare, or otherwise constrain its use. These rules vary considerably from one language to another, and frequently even within a single language there may be different conventions for different classes of names, with near-analogies that come irritatingly close to being exact. (Note that restrictions on what names can be coined also vary, but these are trivial differences. When they have any logical significance it is likely to be pernicious, by leading to puns such as ALGOL's integer labels.)

So rules about user-coined names is an area in which we might expect to see the history of computer applications give ground to their logic. Another such area is in specifying functional relations. In fact these two areas are closely related since any use of a user-coined name implicitly involves a functional relation; e.g., compare

$x(x+a)$ **where** $x = b + 2c$ $\qquad$ $f(b+2c)$ **where** $f(x) = x(x+a)$

ISWIM is thus part programming language and part program for research. A possible first step in the research program is 1700 doctoral theses called "A Correspondence between x and Church's λ-notation."[1]

2. The where-Notation

In ordinary mathematical communication, these uses of '**where**' require no explanation. Nor do the following:

$f(b+2c) + f(2b-c)$
where $f(x) = x(x+a)$

$f(b+2c) + f(2b-c)$
where $f(x) = x(x+a)$
and $b = u/(u+1)$
and $c = v/(v+1)$

$g(f$ **where** $f(x) = ax^2 + bx + c,$
$u/(u+1),$
$v/(v+1))$
where $g(f, p, q) = f(p+2q, 2p-q)$

Presented at an ACM Programming Languages and Pragmatics Conference, San Dimas, California, August 1965.

[1] There is no more use or mention of λ in this paper—cognoscenti will nevertheless sense an undercurrent. A not inappropriate title would have been "Church without lambda."

Distinguer entre

- un ensemble de valeurs et d'opérations primitives
- et comment on exprime des choses (valeurs, calculs) « en fonction d'autres choses indépendamment de la nature de ces choses »
  - comment on fait référence aux objets (mécanisme de liaisons)
  - expressions fonctionnelles
  - structure de contrôle, modularisation, ...

Exemple : l'algèbre combinatoire

- On se donne un ensemble de fonctions primitives f_i. Un calcul c'est une expression $\Phi(x_1, \ldots, x_n)$ construit à partir des f_i et des variables x_i

- le processus de construction de Φ doit être lui-même un calcul (une fonction) :

  une algèbre (A = $\{f_1, \ldots\}$, .) est combinatoirement complète ssi pour toute expression $\Phi(x_1, \ldots, x_n)$ il existe un élément F de A tel que

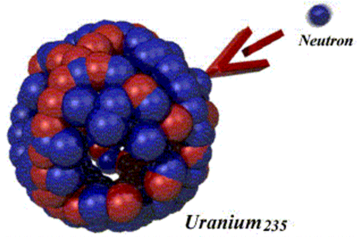
  $$F.a_1.a_2 \ldots a_n = \Phi(a_1, \ldots, a_n)$$

  pour tout $a_1, \ldots, a_n$ de A

- Un informaticien s'intéresse au moins autant (sinon plus) à F qu'aux f_i (?)

Computational Science :
ça se calcule

L’expérience numérique



Computational science

This book presents classical mechanics from an unusual perspective. It focuses on understanding motion rather than deriving equations of motion. It weaves recebt diiscoveries in non linear dynamics throughout the presentation, rather than presenting them as an afterhought. It uses functionnal mathematical notation that allos precise understanding of fundamental properties of classical mechanics. It uses computation to constrain notation, to capture and formalize methods, for simulation, and for symbolic analysis.

[…]

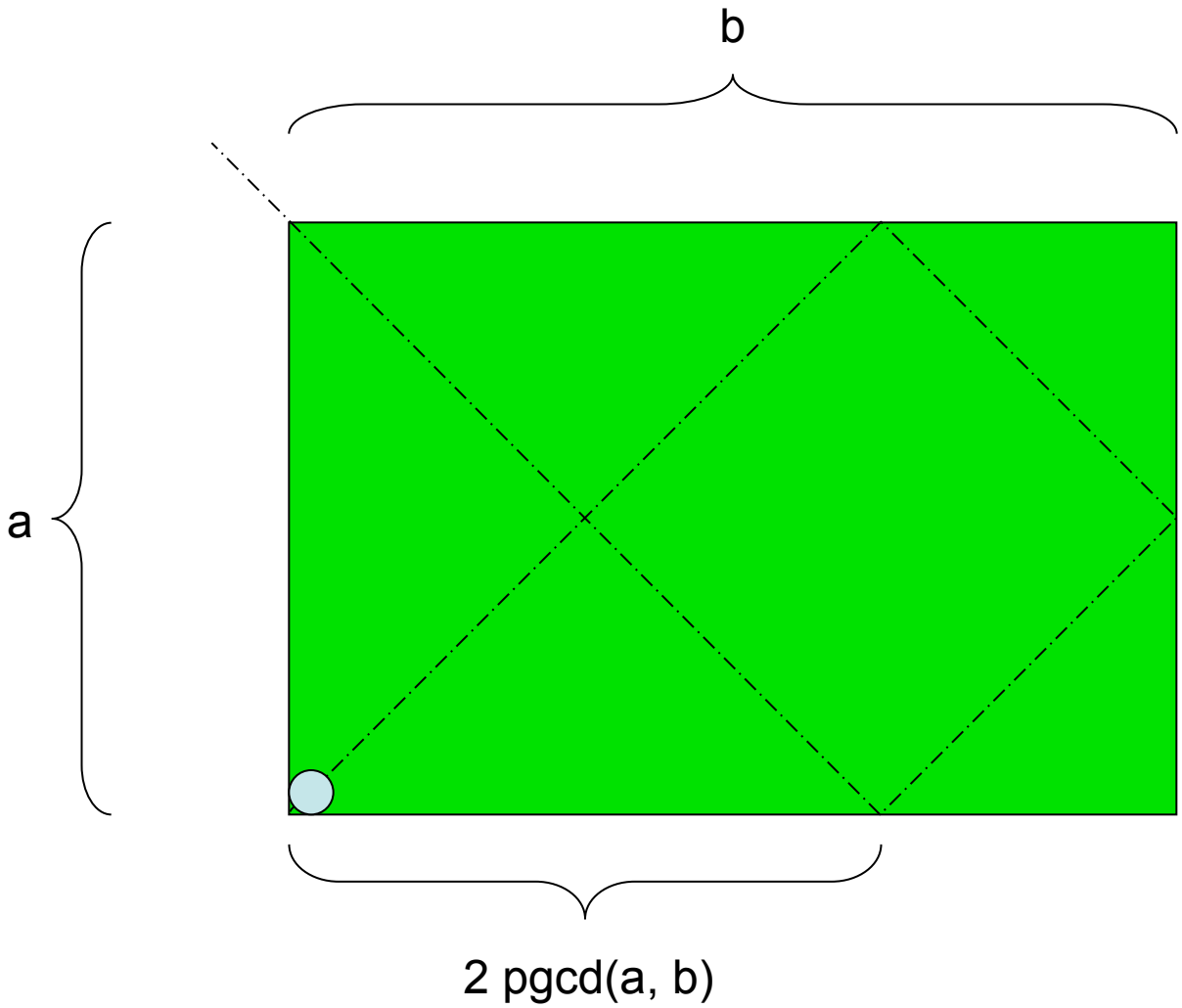
We learned a great deal about both mechanics and computation by this process.

`http://www-swiss.ai.mit.edu/~gjs/6946/sicm-html/index.html`
`http://mitpress.mit.edu/sicp/full-text/book/book.html`

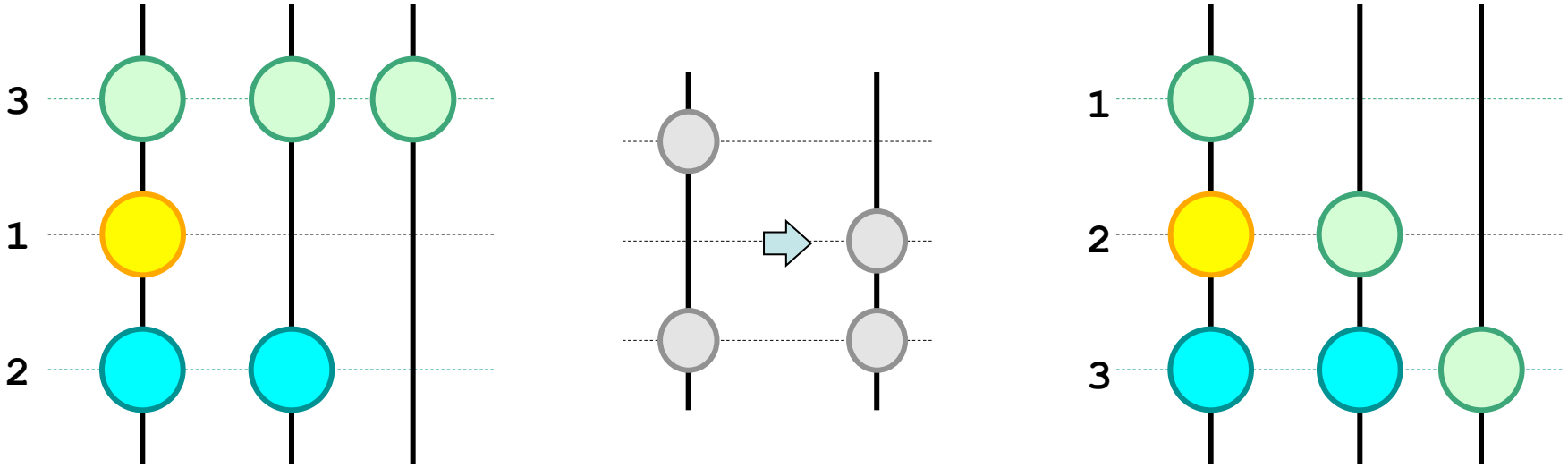
Incidemment...

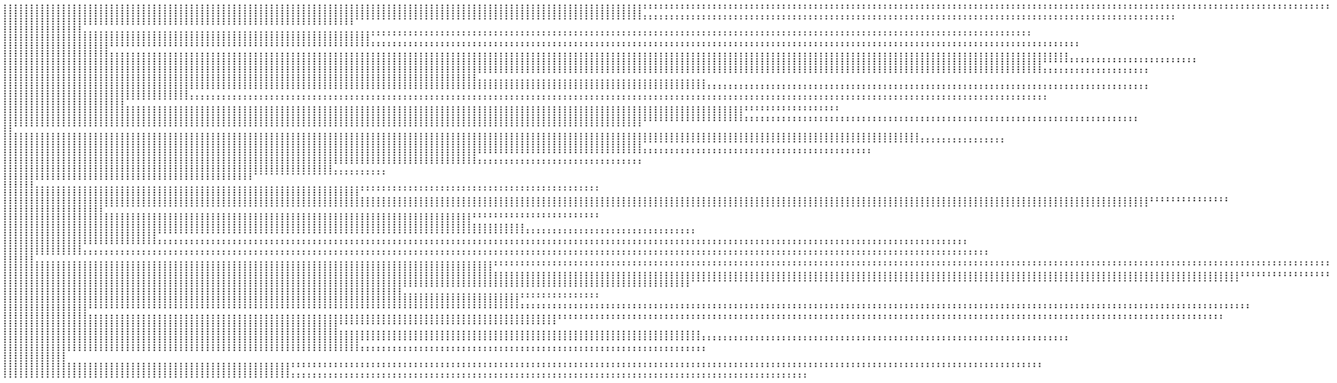
- Computational Physics *versus* Physical Computation
- Computational Biology *versus* Biological Computation
- Computational Chemistry *versus* Chimical Computation
- etc.

Calculer un PGCD avec un billard



Trier avec un boulier et la gravité





Peux t'on simuler tous les processus naturels ?

- la simulation peux avoir un coût prohibitif
- simulation et approximation
- faut-il admettre (et combien) d'opérations primitives qui échappent à une machine de Gandy ?
- faut-il des structures de contrôle non discrète ?
- *Unlike programs, computers must obey the laws of physics.* […] *Of course the computer itself can be so modeled. Think of it: the behavior of the smallest physical switching element is modeled by quantum mechanics described by differential equations whose detailed behavior is captured by numerical approximations represented in computer programs executing on computers composed of …!* (A. & S.)

La Nature comme machine

(où la « naturalisation » comme une mécanisation) :

c'est un calcul

De la thèse de Church-Turing logique à la thèse de Church-Turing physique

- Thèse CT physique forte

  tout ce qui peut être accompli par un système physique est Turing-calculable

- Thèse CT physique faible

  toute fonction qui peut être calculée par un système physique est Turing-calculable

CT physique forte

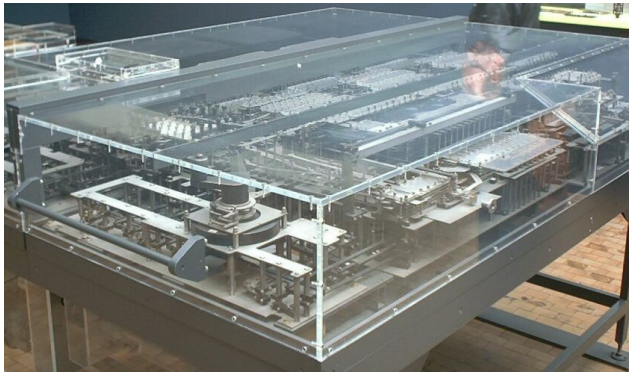
- Pas très précis. On peut préciser :
  1. Any physical process can be simulated by some TM (e.g., Deutsch 1985, Wolfram 1985, Pitowsky 2002).
  2. Any function over strings that is computable by processes that manipulate arbitrary real numbers (in the sense defined by Blum et al. 1998) is Turing-computable.
  3. Any system of equations describing a physical system gives rise to solutions that are Grzegorczyk-computable (Earman 1986).
  4. For any physical system S and observable W, there is a Turing-computable function f: $\mathbf{N} \rightarrow \mathbf{N}$ such that for all $t \in \mathbf{N}$, $f(t)=W(t)$ (Pitowsky 1990).

- Falsifiable

  Exemple (2) : le nombre de fonctions qu'on peut spécifier n'est pas dénombrable

- Mais est-ce que la nature utilise vraiment des objets infinis ?

Zuse, Fredkin, Deutsch, Wolfram…
des automates cellulaires à la physique digitale



"Rechnender Raum"
(Computing Cosmos / Calculating Space)

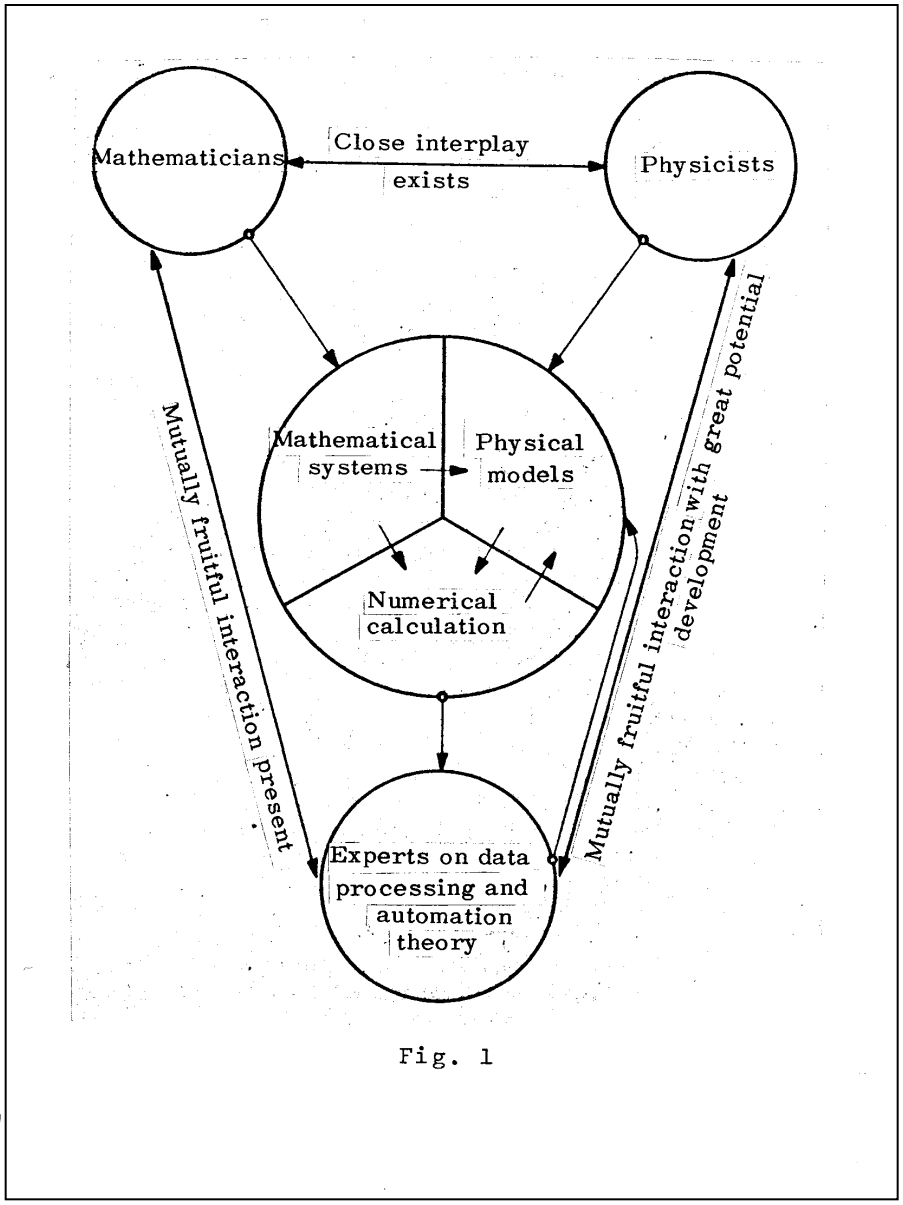
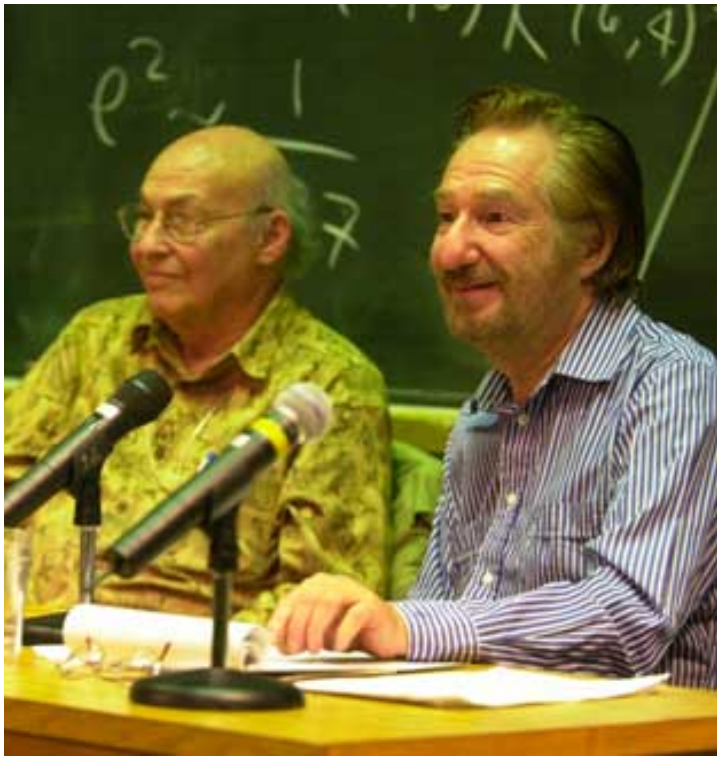


Fig. 1

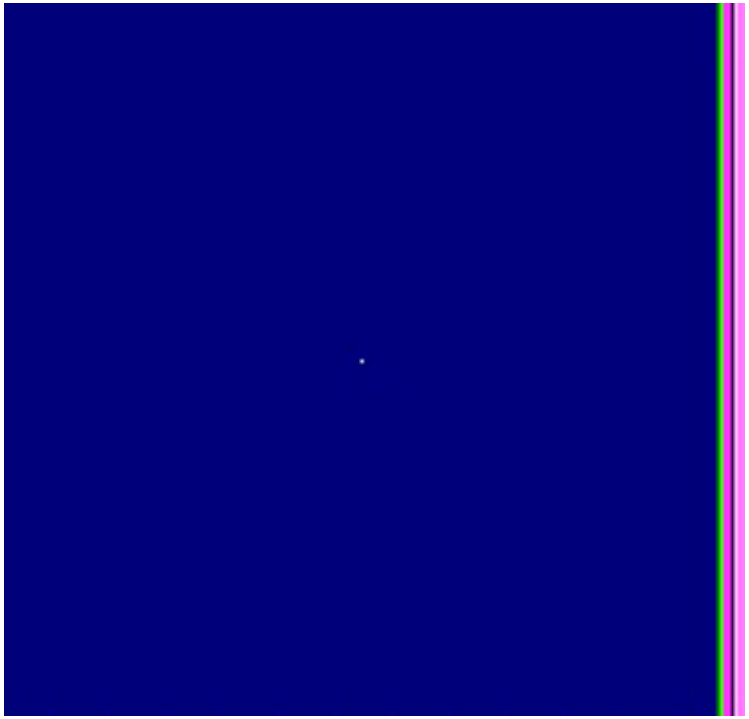
Fredkin et la conservation des invariants



Marvin Minsky Edward Fredkin

Si l'univers est un AC, alors il faut qu'il respecte les symétries qu'on observe dans les lois de la nature.

Par exemple, il faut qu'il soit réversible.



La nature ne peut pas être une calculette

- Il est intéressant de rapprocher cette question des stratégies de réfutations développées dans le débat « le cerveau comme ordinateur ».
- Les opposants

Hyper-computation

la Nature sait faire plus que des machines de Turing, et on peut s'en servir pour calculer

(ou bien : les machines de Turing ne sont pas assez puissantes pour simuler le réel)

Le zoo des extensions…

- Il y a beaucoup de tentative pour dépasser le Turing-calculable (hypercomputation, super-Turing, etc).
- Beaucoup de ces tentatives évitent d'exhiber un problème de décision solvable dans un nouveau cadre et qui ne soit pas Turing-calculable.
- Souvent, ces tentatives font appel à des calculs infinis ou bien à des fonctions continues qui me semblent poser des problèmes physiques
- Un bestiaire :
  - les machines de Zenon
  - les ordinateurs newtoniens
  - les calculateurs analogiques

Les machines de Zenon

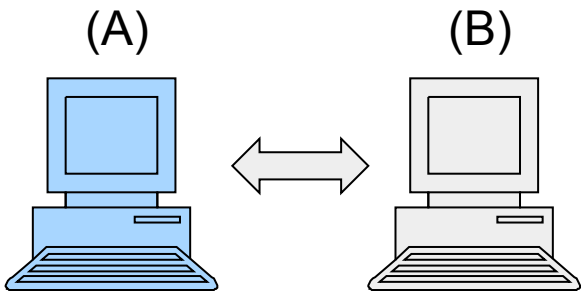
- C'est une machine de Turing qui accélère :
  - la 1er instruction s'exécute en 1 seconde
  - la 2ème instruction s'exécute en 0.5 seconde
  - …
  - la $n$ème instruction s'exécute en 2^{-n} seconde
- Au temps 2+ε on a donc effectué une infinité d'opérations

- Des machines de Zenon existent (par exemple une flèche qui passe par une infinité de point en un temps fini).

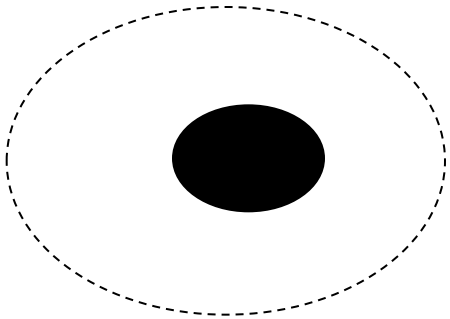
- Une telle machine peut décider si un programme de Turing P stoppe ou pas : c'est un programme H qui effectue P et si P s'arrête avant le temps 2, alors P stoppe et sinon, au temps 2+ε, H peut conclure que P boucle.

- Problèmes :
  - La même instruction peut s'exécuter en un temps arbitrairement court. La densité d'énergie ou la propagation des signaux devient donc infinie.

La machine et le trou noir



- A communique avec B qui est entrain de tomber vers l'horizon d'un trou noir.
- A mesure que B s'approche du trou noir, le temps de B est ralenti par rapport à celui de A.
- A peut par exemple tester toutes les formules de ZFC pour savoir lesquelles sont des théorèmes et communiquer le résultat à B
- Problèmes :
  - le destinataire du calcul est au bord d'un trou noir
  - A doit fonctionner un temps infini (donc ne pas s'user et disposer d'une source d'énergie infinie)
  - B travaille avec une horloge. Considérons le dernier cycle d'horloge avant que B ne croise l'horizon du trou noir. Avant le début de ce cyle, A n'a effectué qu'une partie de la recherche. Durant le derrnier cycle d'horloge de B , A parcourt une infinité d'assertion, mais le message envoyé à B doit être traité en moins d'un cycle d'horloge, ce qui n'est pas possible.



Les calculateurs analogiques

1956, Heathkit Analog Computer H1, 70 tubes, 15 amplificateurs

- Calcul analogique sur des variables continues représentant des réels
- Domaine actif jusqu'à la fin des années 60
- Deux problèmes au moins :
  - Si on utilise un courant électrique : il y a un nombre entier d'électrons
  - Problèmes de précisions :
    - des mesures
    - fluctutations aléatoires (bruit thermique, rayonnement, …)
    - erreurs systématique due à l'incertitude des paramètres des composants
    - distorsions due aux non-linéarités des amplificateurs utilisés

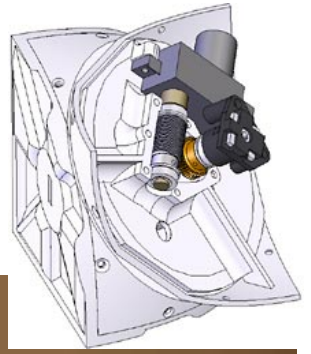
Errors as small as 0.1% are hard to achieve;
error of 1% are not unusual, and they are sometimes as large as 5%-10%.
This is in striking contrast to digital machines

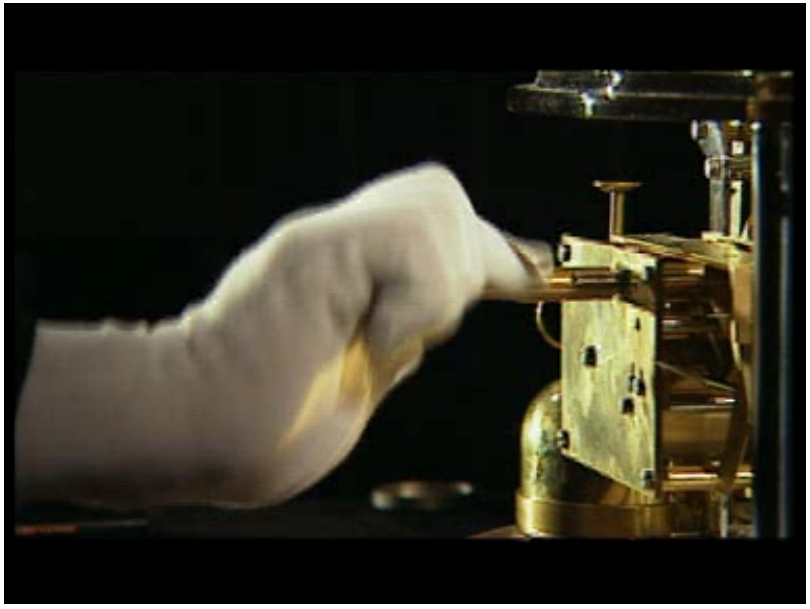
Le débat est peut-être ailleurs

Une naturalisation

Expliquer la nature par elle-même ou
la nature se suffit à elle-même pour être intelligible.

- Descartes : le corps machine
- La Reine Christine : *j'y croirais quand les horloges auront des petits*
- Vaucanson : les anatomies mouvantes
- Von Neumann :
  l'automate auto-reproducteur (logique)
- Hod Lipson :
  un automate auto-reproducteur (physique)



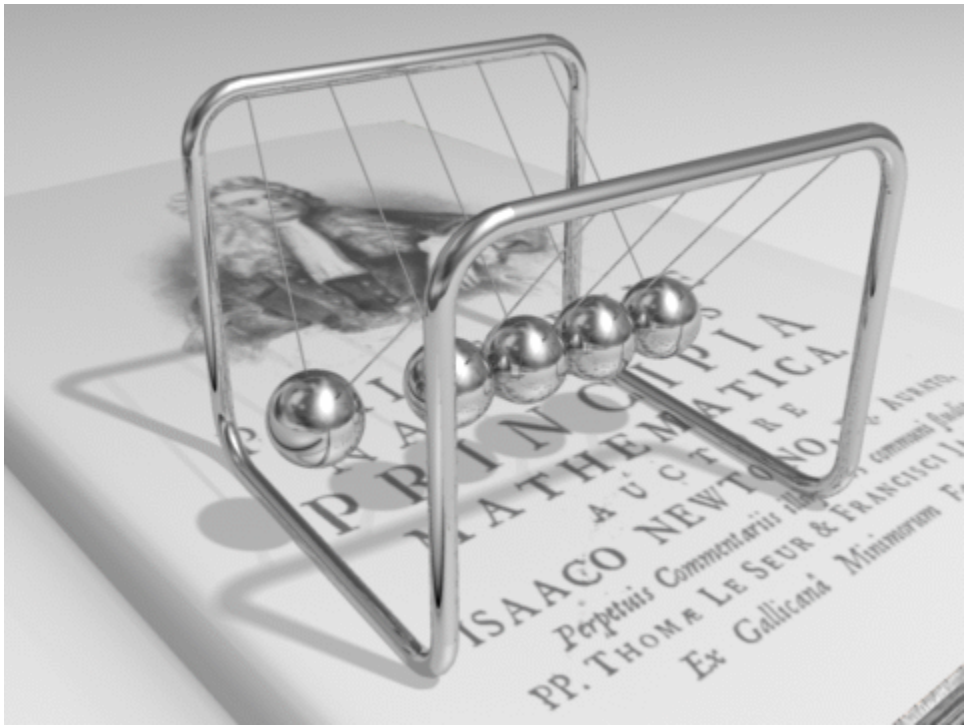


un canard digérateur par Frédéric Vidoni

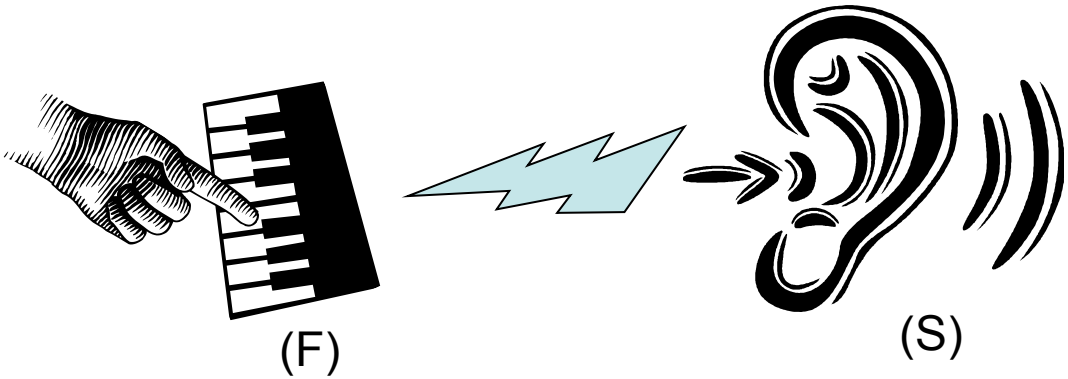
Hod Lipson, may 2005, Cornell University
http://www.news.cornell.edu/stories/may05/selfrep.ws.html

Parce que ça calcule ça :

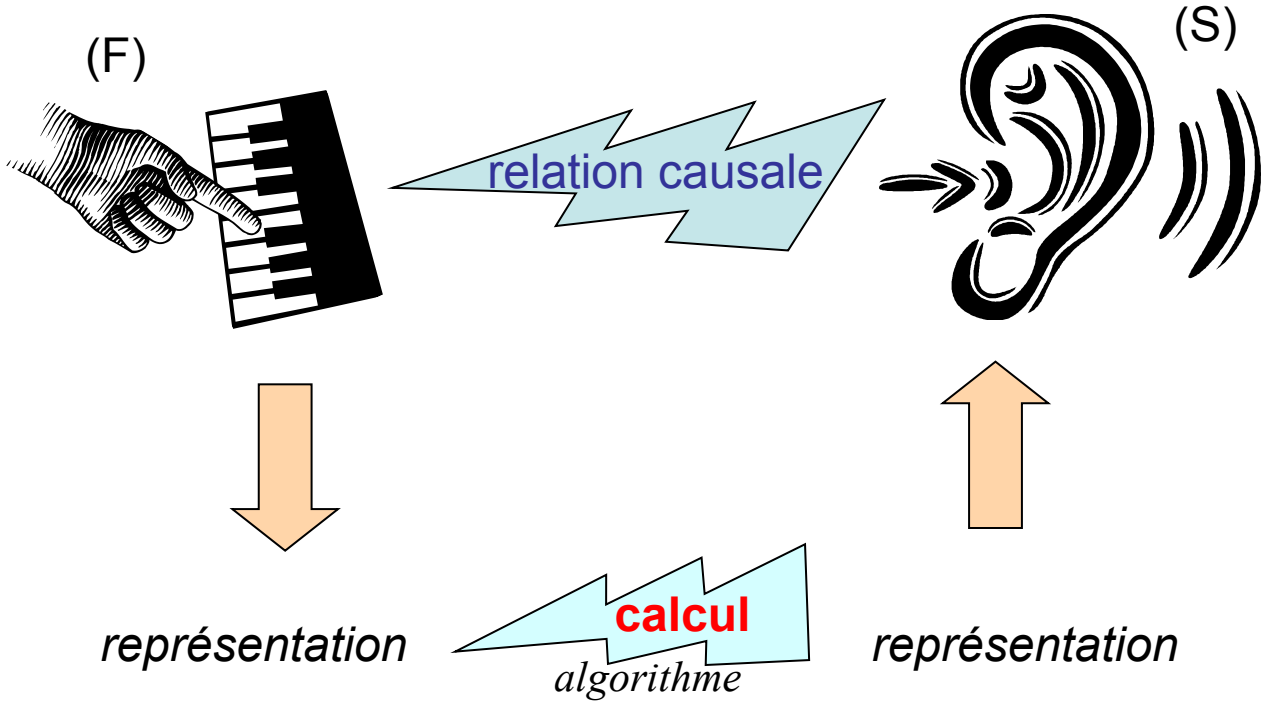
calcul et causation



The Incredible Machine (TIM)



La relation causale comme calcul



Trois avantages :

1. sociologiquement, le concept de calcul s'est diffusé partout et la métaphore computationelle est déjà couramment utilisée,

2. Les contraintes que Turing impose à un processus pour être un calcul, impose une forme d'intelligibilité particulièrement désirable,

3. Concevoir la relation causale comme un calcul entre la cause et ses effets est un point de vue particulièrement opératoire.

Le calcul, un concept nomade et envahissant

- Every computer program is a model, hatched in the mind, of a real or mental process (A. & S.)

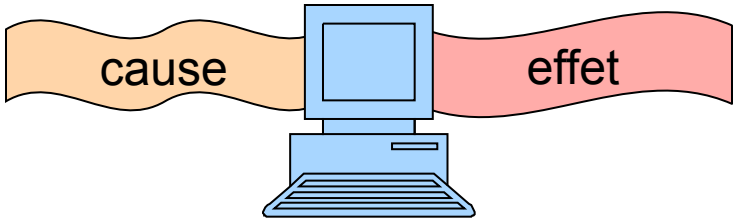
- … chez ces mutants, les cellules embryonnaires interprètent mal leur position et engagent des programmes morphogénétiques inapropriés… (A. Prochiants, Leçon Inaugurale)

- Il faut changer le logiciel du Parti Socialiste (?)

.

- …

Une forme d'intelligibilité désirable



- un algorithme est une description finie,
- déterministe
- composé à partir d'opérations primitives en nombre fini
- non ambigüe
- uniforme
- explicite (pas d'intuition, ni de devinettes…)
- …
- un algorithme est plus précis qu'une fonction (extensionnelle)
- …
- *qui se traduit dans le référentiel (la machine) de l'autre dès que ce référentiel (sa machine) est assez puissante (universelle)*

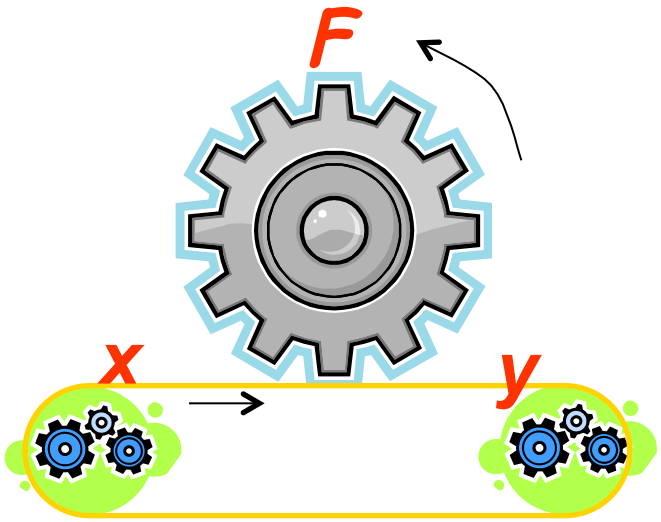
→ **c'est une explication mécaniste qui répond à des critères très sévères d'objectivité tout en permettant sa transmission intersubjective**

Et particulièrement opératoire

- les algorithmes ça s'analysent, ça se composent, se décomposent, se comparent, etc…

- c'est mieux qu'une équation : c'est « orienté », i.e. on sait sur quoi agir

versus



Un point de vue heuristique

Deux exemples :

- il y a un lien entre « calculable » et « une « logique des observations finies » :

- Le théorème de Löwenheim-Skolem : si une théorie du premier ordre admet dénombrable, alors elle admet des modèles de toute cardinalité infinie.
  Je préfère le sens :
  Si une théorie du premier ordre admet un modèle (quelconque) alors elle admet un modèle dénombrable.

Une différence fondamentale