The Philosophy of Computation

(Selected slides)

Jan van Leeuwen

Department of Information and Computing Sciences
Utrecht University, The Netherlands

COMP UTAT ION

- 6th AISB Symp. Computing & Philosophy, Exeter, 2013.
- 7th AISB Symp. Computing & Philosophy, London, 2014.
- 2nd AISB Symp. Computat. Creativity, Canterbury, 2015.
- Submitted material, 2015.

- how has our conception of *computation* evolved?
- what *is* computation?

Outline

1. The question
2. Philosophy at the rescue
3. A view of computation
4. (Making it formal)
5. A theoretical model*
6. Conclusion

1. The question

A.M. Turing (1912-1954)

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.]

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 verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.

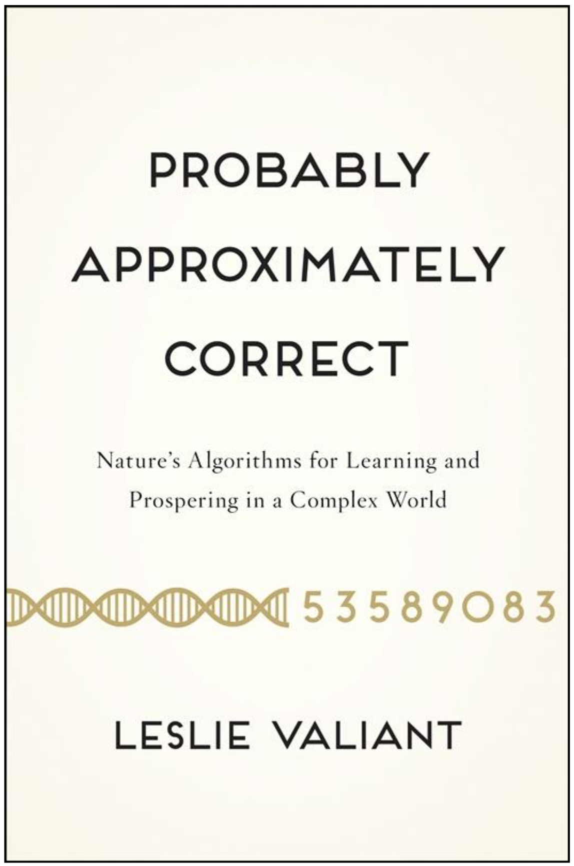
Turing's broader view: how do processes work … computationally?

- Understanding computation (as machines…):
  - *Automatic machines* (a-machines, 1936) (*logical computing machines*, 1948)
  - Choice machines (c-machines, 1936)
  - With intuition: Oracle machines (o-machines, 1938)
  - Partially random machines (r-machines, 1948), machines random element (1950)
  - Modifiable machines incl. interference with machines (1950)
  - Practical computing machines (1948)
  - Continuous machines (1948)
  - No problem with potentially infinite computations and storage

- Understanding the brain:
  - Unorganized machines (1948)
    - Organized: A-type, B-type (artificial neural nets, brain model)
    - Self-organizing: P-type (learning nets)
    - Educating: learning machines
  - Intelligent machines, how they `learn'
  - Machines that can play games (e.g. chess)
  - Can machines think (-> the imitation game aka the Turing test, 1950)
  - *Understanding (simulating) mental processes by computation .*

- Understanding the algorithms of ... ?
  - Morphogenesis (1950 – 1954)

Leslie Valiant (2013)



"Turing's concept [*of computation*] may enable us to understand human activity itself. […] since the beginning of life, the dominating force on Earth within all its life forms *was* computation. [..] These computations were weak but they were exceedingly good, however, at one enterprise: adaptation. These are the computations that I call *ecorithms* […]

- Understanding (all) natural processes in terms of, or even *as*, **computation**?

(Some) recent opinions:

J. Searle: "Computation does not name a machine process. It names an *abstract mathematical process that we have found ways to implement in machines*."

L.G. Valiant: "Computation [is] the execution of *step-by-step procedures* for *processing information*."

A. V. Aho: "Mathematical abstractions called *models of computation* are at the heart of computation and computational thinking. Computation is a process that is defined in terms of an underlying model."

J.S. Conery: "Computation is *symbol manipulation*. Computation is a discrete process, a sequence of distinct transitions".

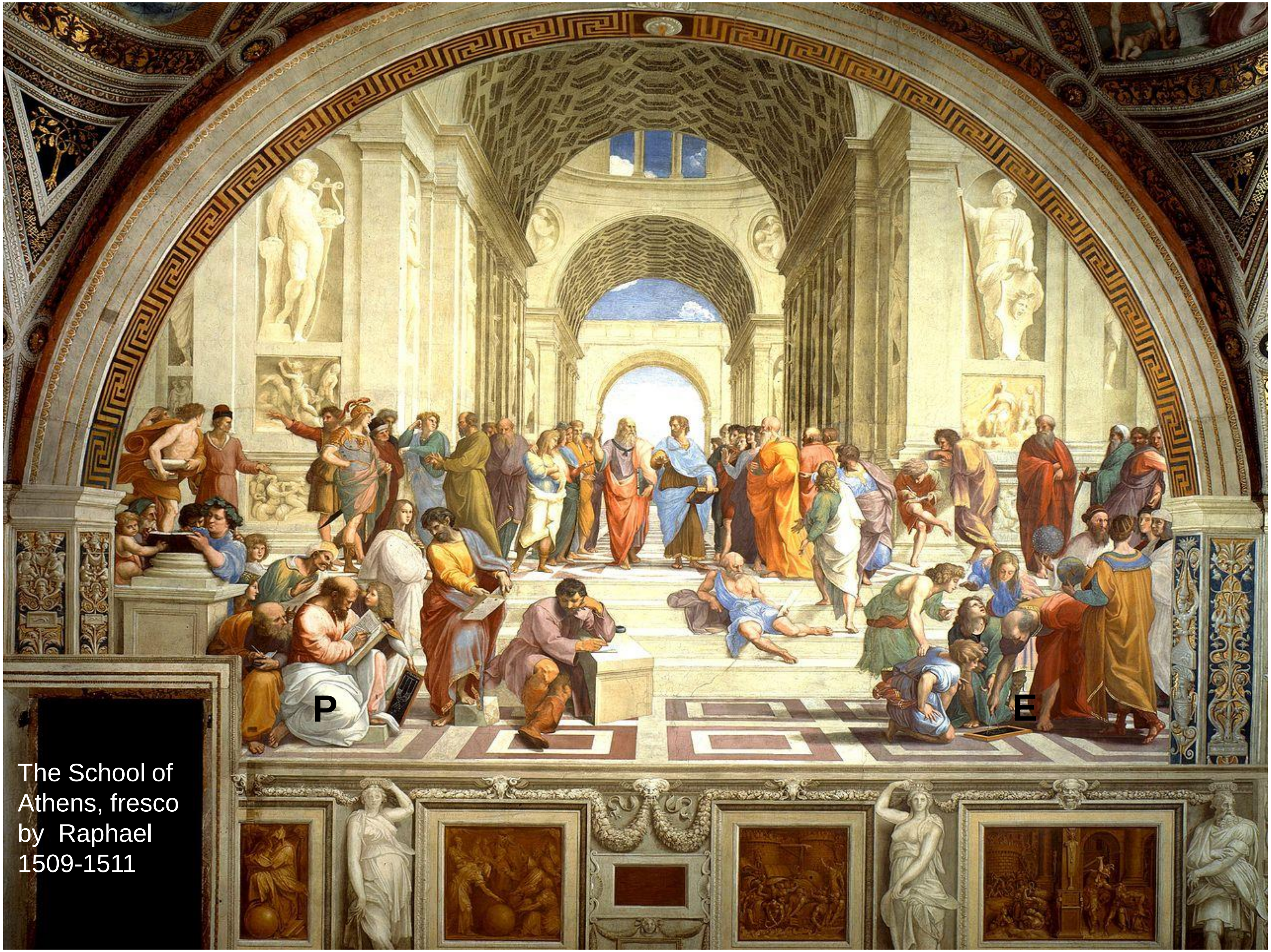
P.S. Rosenbloom: "Computation is *information transformation*."

D. Frailey: "Computation in its broadest sense is anything that happens (as opposed to things that are static). If so, then the principles of computation are, in fact, the principles of *processes*.

D. Deutsch: "A computation is a *physical* process in which physical objects like computers, or slide rules or brains are used to discover, or to demonstrate or to harness properties of abstract objects — like numbers and equations."

All models are wrong but some are useful
George EP Box

2. Philosophy of … Computing



The School of Athens, fresco by Raphael 1509-1511

Great issues in CS

- **Object(s) of study**
  - Natural science, engineering, design, backcasting, …
- **Development**
  - Historical lines, paradigms, is science or technology leading?
  - Centric views: logic, mathematics, programming, engineering, specific applications, etc.
  - Influences: scientific, technological, social, societal, political, …
  - \`From tool perspective to science perspective', why, what leads the transformation?
- **Philosophy of \`perspectives'**
  - Impact of centric views
    - Information-oriented, Computing-oriented, Communication-oriented, Cognition-oriented, Design-oriented, …
  - Sub-areas (AI, agents, information processing, gaming, …)
- **Concepts and mothods**
  - Ontology, epistemology, what is understanding in cs.
  - Fundamental notions: how to understand computation, information, programs (formally, o/w)
  - Abstraction, modeling, simulation, algorithmic mechanisms, languages, specification, design methods
  - Methodology: theory, experiment, *program, productize?*
- **Impact**
  - On our thinking
    - Algorithmic (computational) thinking, design science, …
    - Awareness of resources, complexity, \`think parallel' (Intel), …
  - Reviving "can machines think, have emotions, etc" (mimicking by mass computation).
  - Impact on our values.
  - Impact on our future.

Computation is ??

- **Early conceptions**
  - R. Lull (1230-1315): “mechanically generate all possible thoughts or ideas…”
  - Th. Hobbes (1588-1679) : “all thought is a form of computation.”
  - G.W. Leibniz (1646-1716): “reduce all reasoning to calculation.”
- **Evolution**
  - *Calculation* ⇒ Mathematical method to determine something effectively ⇒ discrete and scientific computing ⇒ …
  - *Record keeping* ⇒ business data processing ⇒ information technology ⇒ data & knowledge engineering ⇒ …
  - *Automating* ⇒ …
- **Now: Shift towards know-ledge-enriched processes**
  - Making all systems ‘intelligent’
  - Self-adjusting (self-modifying, interactive, evolving)
  - Cognitive abilities (learning, creativity, …)
- ***Anything that can be (or we want to be) understood as computation***
  - Everything virtual: simulation -> visualization -> interaction -> experience -> emotion -> …
  - The *Algorithmic Lens* (Karp) , explanatory power
  - New way of doing/understanding science (Kuhn)?
  - of *discovering …*?

3. A view of computation

The current views:

- tend strongly towards observer-*in*dependence,
- view computation as a mechanistic process,
- *define it by what an underlying machine (or model) is doing*, i.e. by how it is *realized,*
- not covering newer cases adequately.

But

- Our experience with computation points in another direction: we are primarily interested in ***WHAT* a computation ultimately does** for us.

!

Thesis:

Computation is a (repeatable) process of *knowledge generation,* in a suitable knowledge space.

Based on the initial data and an existing theory, a computation *reveals*, cq *makes knowledge (information) explicitly known* that is not yet included in the given knowledge base.

Observer-relative

Intrinsic to relevant theory

Example: process generating primes

Underlying theory: Peano axioms

Generated knowledge: $\mathbf{2^{57,885,161} - 1}$ **is prime.**

Justification: certificate

J. Wiedermann & JvL, 6th AISB Symp. on Computing and Philosophy, Exeter, 2013.

Examples

Program in a machine

Processing information

When/what does *it compute:*

- when producing knowledge to the observer (c.q. to another process)

Kowledge?

Repeatable?

Justifiable?

E.g. Primes.

Process in a neuron

Processing information

When/what does it *compute:*

- when producing knowledge to the observer (c.q. to another process)

Knowledge?

Repeatable?

Justifiable?

E.g. Trigger

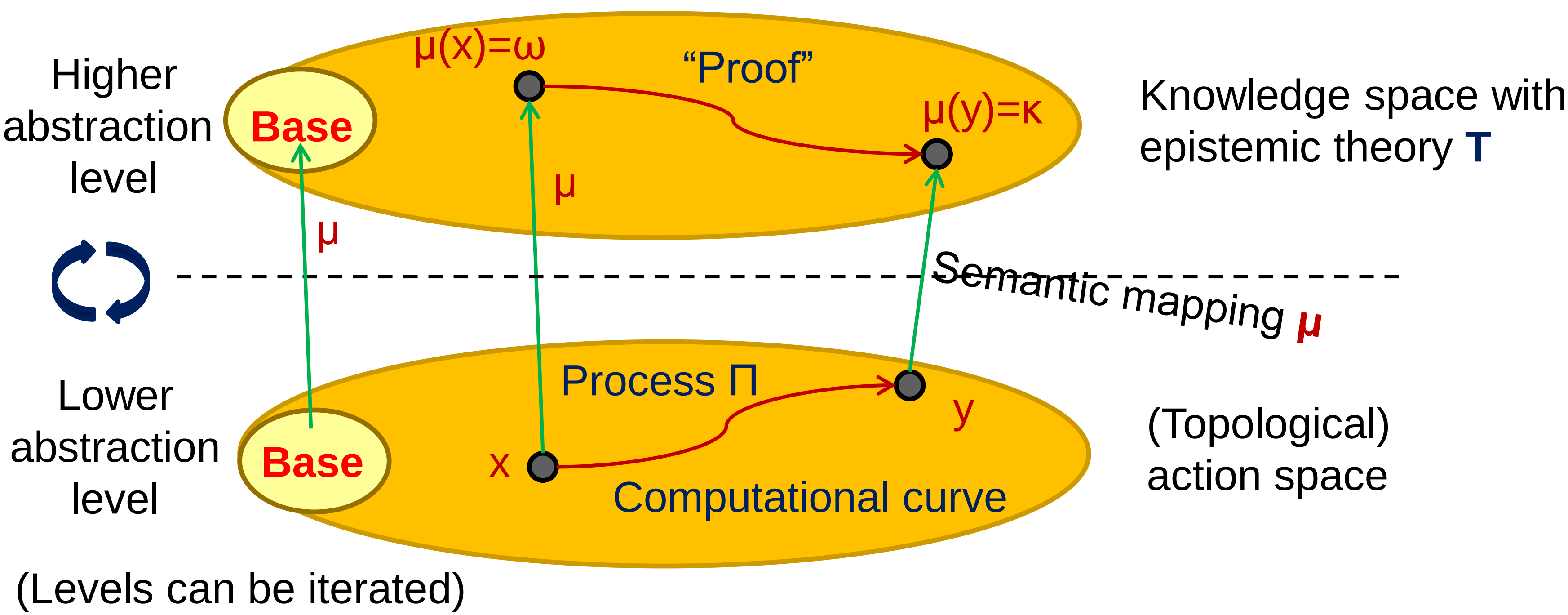
Challenges

- Does a rock compute?
- Experimentation
- Searle's Chinese Room
- Test for Computationality

- Criteria (Cantwell Smith, Piccinini)?
  - empirical, conceptual, cognitive soundness
- Observer-dependence unavoidable?
- Challenging the Central (Turing) Dogma…?

4/5. A theoretical model (in progress)

A model of computation



Postulate: Computational processes Π are

- Repeatable
- Justifiable: Π(x)=y if and only if (T, ω) ├ κ
- *Compositional*

This model of computation is

- machine independent
- algorithm free
- representation free

Exploring spaces by computation

- Knowledge *generation*
  - with bundle $B = \{c_i\}_{i \,\epsilon\, I}$, from knowledge base E_0
  - $e \,\epsilon\, E$ is producible by $c_1, \ldots, c_k$ from B (denoted $c_1, \ldots, c_k \vdash e$) if $c_1{}^{init} \,\epsilon\, E_0$, and $e \,\epsilon\, \delta(c)$ for some $c \,\epsilon\, c_1 \Delta \ldots \Delta\, c_k$
- K_B : all knowledge items that can become known using bundle B.
  - ***Proposition***: K_B is the least fixpoint of an operator G defined over 2^E, hence well-defined. If B is compositional, then $K_B = G(E_0)$.
- Notion:
  - A bundle B is universal for E if and only K_B = E.
- *Knowledge recognition*
  - Counterpart to generation
  - ***Proposition***. Given bundle B, a computational process can be designed which precisely `recognizes' all items from K_B , based on the computational mechanism underlying B.

6. Conclusion

Computation as knowledge generation

- Separates computing and non-computing objects
  - *relative to observer*
  - *test: specify spaces etc*
- Allows classification of computations
  - *wrt quality and quantity of produced knowledge*.
- Is independent of type of process
  - *Covers both known -and yet unknown?- instances of computation.*
- Allows considering computations at a high abstract level.
  - *`how to compute it', test for algorithmic mechanisms*
  - *the central dogma resp. ASMs are instances*
  - *new theory of computation*
- Resolves notorious problems in the scope of the classical definition.
  - *According to new definition*, cognition is computation.
  - *Ecorithms are computations*
- Focuses on the intrinsic meaning of computations
  - *Consequences in AI, philosophy, epistemology, methodology of science, cognitive sciences, …*
  - *Epistemic question: identify knowledge generation*
- Resolves Abramsky's questions:
  - *What do we compute, why.*
- New meaning to the computation-centric perspective
  - *Understanding science is explaining computations?*.

Computation is knowledge generation

"Creation of knowledge ... now has to be understood as one of the fundamental processes in nature; fundamental in the sense that one needs to understand them in order to understand the universe in a fundamental way."

D. Deutsch

"Computation ... now has to be understood as one of the fundamental processes in nature; fundamental in the sense that one needs to understand them in order to understand the universe in a fundamental way."

J. van Leeuwen & J. Wiedermann

The more we understand the Universe, the more computation (hence knowledge) we see around us.

DISCLAIMER

These slides are for use at IIT.SRC 2015 only. They are not meant for distribution or posting outside the realm of the conference.

All materials are intended for educational use only. Illustrations and other materials should be cited by their original sources, including our own.

Any violations of copyright will be removed once they become known to us.