Computation after Turing

Jan van Leeuwen

Information and Computing Sciences
Utrecht University, The Netherlands

Children interacting with a machine

Thinking and acting

Learning

Having fun

```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2


C000                    ORG     ROM+$0000 BEGIN MONITOR
C000 8E 00 70  START    LDS     #STACK

                        ***************************************
                        * FUNCTION: INITA - Initialize ACIA
                        * INPUT: none
                        * OUTPUT: none
                        * CALLS: none
                        * DESTROYS: acc A

0013            RESETA  EQU     %00010011
0011            CTLREG  EQU     %00010001

C003 86 13      INITA   LDA A   #RESETA   RESET ACIA
C005 B7 80 04           STA A   ACIA
C008 86 11              LDA A   #CTLREG   SET 8 BITS AND 2 STOP
C00A B7 80 04           STA A   ACIA

C00D 7E C0 F1           JMP     SIGNON    GO TO START OF MONITOR

                        ***************************************
                        * FUNCTION: INCH - Input character
                        * INPUT: none
                        * OUTPUT: char in acc A
                        * DESTROYS: acc A
                        * CALLS: none
                        * DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04  INCH     LDA A   ACIA      GET STATUS
C013 47                 ASR A             SHIFT RDRF FLAG INTO CARRY
C014 24 FA              BCC     INCH      RECIEVE NOT READY
C016 B6 80 05           LDA A   ACIA+1    GET CHAR
C019 84 7F              AND A   #$7F      MASK PARITY
C01B 7E C0 79           JMP     OUTCH     ECHO & RTS

                        ***************************************
                        * FUNCTION: INHEX - INPUT HEX DIGIT
                        * INPUT: none
                        * OUTPUT: Digit in acc A
                        * CALLS: INCH
                        * DESTROYS: acc A
                        * Returns to monitor if not HEX input

C01E 8D F0      INHEX   BSR     INCH      GET A CHAR
C020 81 30              CMP A   #'0       ZERO
C022 2B 11              BMI     HEXERR    NOT HEX
C024 81 39              CMP A   #'9       NINE
C026 2F 0A              BLE     HEXRTS    GOOD HEX
C028 81 41              CMP A   #'A
C02A 2B 09              BMI     HEXERR    NOT HEX
C02C 81 46              CMP A   #'F
C02E 2E 05              BGT     HEXERR
C030 80 07              SUB A   #7        FIX A-F
C032 84 0F      HEXRTS  AND A   #$0F       CONVERT ASCII TO DIGIT
C034 39                 RTS

C035 7E C0 AF  HEXERR   JMP     CTRL      RETURN TO CONTROL LOOP
```

A machine interacting with children

Thinking and acting

Learning

Having f…

Computation…

- Classically: ‘a method to determine a thing *effectively*’
  - Counting and record keeping
  - Constructing mathematical objects
  - Calculating functions
  - Solving equations (‘models’)
  - Decision making (‘optimization’, ‘forecasting’)

- Computational view of other domains began ages ago (e.g. in philosophy)

Outline

- What Turing (and others) got us thinking about.
- Computational models
- The rise of algorithmics
- Some philosophy of computation
- Computational perspective in science
- What would Turing be thinking about `today'?

A need to know ..

- In Mathematics: D. Hilbert (*ICM Bologna*, 1928) following up on his famous millennium problems:
  - Is mathematics (complete, consistent, and) decidable?

- In Logic: D. Hilbert and W. Ackermann, *Grundzüge der theoretischen Logik*, 1. Auflage, Berlin (1928):
  - "The *Entscheidungsproblem* is solved when we know a procedure that allows for any given logical expression to decide by finitely many operations its validity (*Allgemeingultigkeit*) or satisfiability. […] "
  - "We like to make it clear that the problem would be solved […] even if the laboriousness (*Umständlichkeit*) of the procedure could make the practical implementation (*Durchführung*) illusory."
- […] in all areas: `*deren Satze uberhaupt einer logischen Entwickel-barkeit aus endlich vielen Axiomen fahig sind*'.

What is computability?

- Early 1930's: understanding the informal notion of 'effective calculability'

K. Gödel (1906-1978)

E.L. Post (1897-1954)

- Typical approach through functions:
  - λ-definable functions (Church)
  - recursive functions (Gödel)
  - general recursive functions (Kleene)
  - through symbol manipulation (Post)
  - discovery of unsolvability

A. Church (1903-1995)

J.B Rosser (1907-1989)

- Church (1936):
  - settles the Entscheidungsproblem
  - (later): "an effective method of computation, or algorithm, is one for which it would be possible to build a computing machine. This idea is developed into a rigorous definition by …"

S.C. Kleene (1909-1994)

1932-35: Quantum mechanics, probability, logic.
1936: Turing machine, computability, universal machine.
1936-38: Logic, algebra, number theory.
1938-39: Study of Enigma cipher machine.
1939-40: The Bombe, machine for Enigma decryption.
1939-42: Breaking of the Enigma code.
1943-45: Chief US-UK cryptography consultant.
1946: Computer (ACE) and software design.
1947-48: Programming, neural nets and artificial intell.
1949: First serious mathematical use of a computer.
1950: Turing Test for machine intelligence.
1951: Non-linear theory of biological growth.
1953-54: Unfinished work in biology and physics.

Cf. A. Hodges, The Alan Turing Home Page

A.M. Turing (1912-1954)

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 […]

[…]

In a recent paper Alonzo Church has introduced an idea of "effective calculability", which is equivalent to my "computability", but is very differently defined. Church also reaches similar conclusions about the Entscheidungsproblem. The proof of equivalence between "computability“ and "effective calculability" is outlined in an appendix to the present paper.

[…]

Computation : a stepwise process as carried out by a (human) computer.

Turing's proof

"Corresponding to each computing machine *M* we construct a formula **Un**(*M*) and we show that, if there is a general method for determining whether **Un**(*M*) is provable, then there is a general method for determining whether *M* ever prints 0."

(p. 47)

Turing: The real question is: "What are the possible processes which can be carried out in computing [a number]?"

Church-Turing thesis

Turing's variety of machines

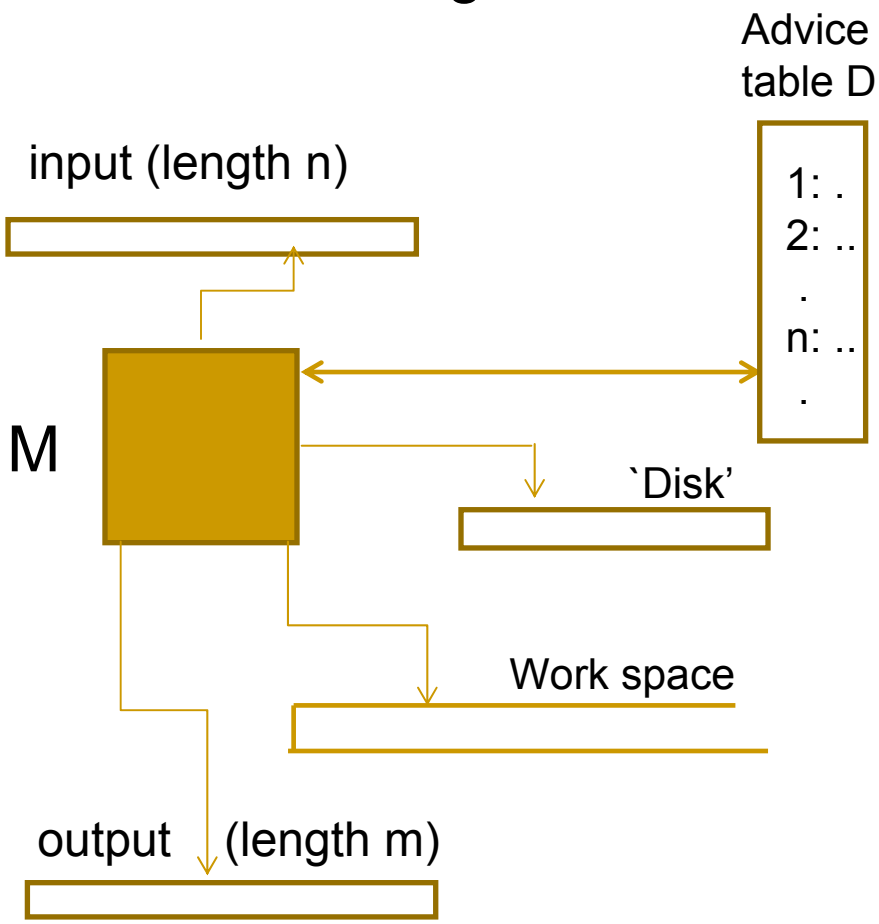
- Mimicking (understanding) computation:
  - Automatic machines (a-machines, 1936) (logical computing machines, 1948)
  - Choice machines (c-machines, 1936)
  - With intuition: Oracle machines (o-machines, 1938)
  - Partially random machines (r-machines, 1948), machines random element (1950)
  - Modifiable machines incl interference with machines (1950)
  - Practical computing machines (1948)
  - Continuous machines (1948)
  - No problem with potentially infinite computations and storage
- Mimicking (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 might `learn'
  - Machines that can play games (e.g. chess)
  - Can machines think (-> the imitation game aka the Turing test, 1950)
- Mimicking (understanding) nature as …
  - Morphogenesis (1950 – 1954)

What is an algorithm?

- Algorithm as TM program antiquated
  - Straight-line programs, circuits
  - Architectures now: parallel (multicore), distributed, self-organizing, amorphous, …
  - Adequate reflection needed for analysis, design, and verification
- D.E. Knuth
  - Section 1.1, Volume I (1969)
- S. Smale, L. Blum, K. Weihrauch, F. Cucker, …
  - Computability over the reals, complexity of real functions
- Y.N. Moschovakis
  - Algorithms are ‘recursors’, implementations are ‘iterators’
- Y. Gurevich
  - Algorithms are ‘abstract state machines’
  - Working on ‘world models’ (i.e. first order structures)
- JvL, J. Wiedermann, …
  - Algorithms as abstractions of ‘what […] machines do’
  - of `what … do’
- Other (e.g. process models)

Typical computational models (1)

- TM with learning



Input length fixed at: n
Clear `disk'

Repeat

Take input

Process it

Allow set-up advice D(n)

What remains on disk (only) carries over ('persists') to next run.

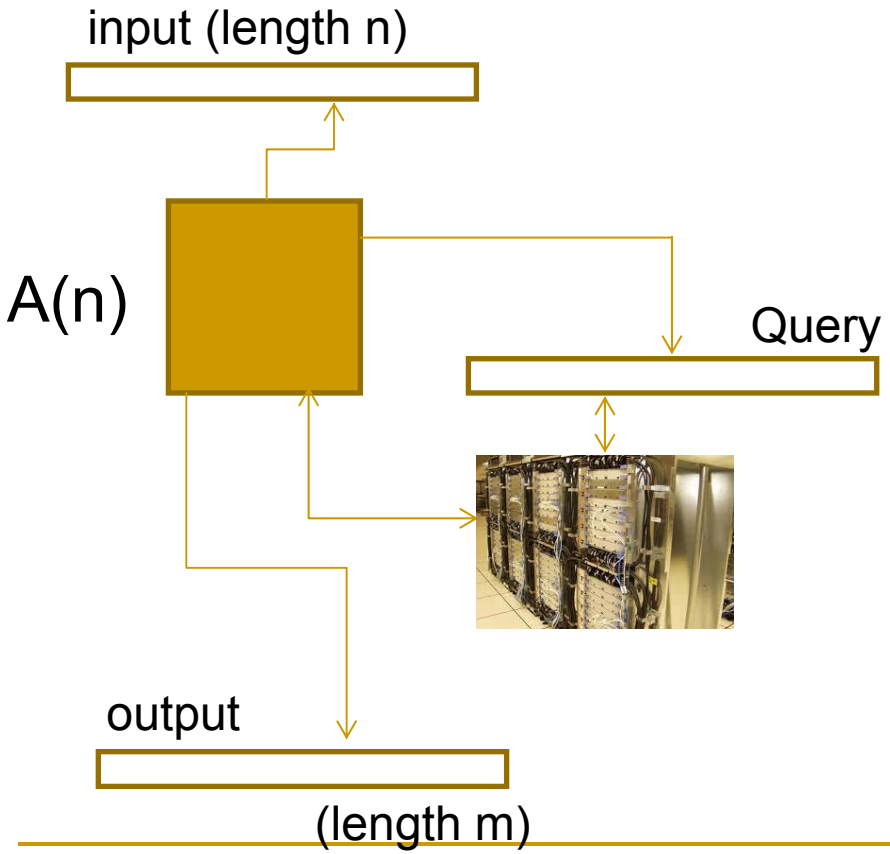
Output depends on input and what is learned from the past

Mapping

Until new n

Typical computational models (2)

- Smart automata with (TM-based) oracle



Input length fixed at: n
Retrieve automaton: A(n)

Repeat

Take input

Process it (finite-state) and
Build up query string

Ask oracle when A wants it

Oracle reset after every run
but *not* A(n)

Output depends on input and
previous `state of mind'

Mapping

Until new n

Typical computational models (3)

- Impact of resource bounds
- **Theorem**:

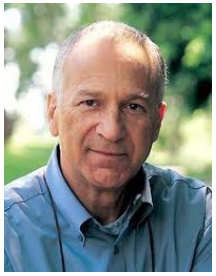
  Consider (cont'd) translations with input of size *n* and output size bounded by `polynomial bound' *p(n).* The following are equivalent for translations F:

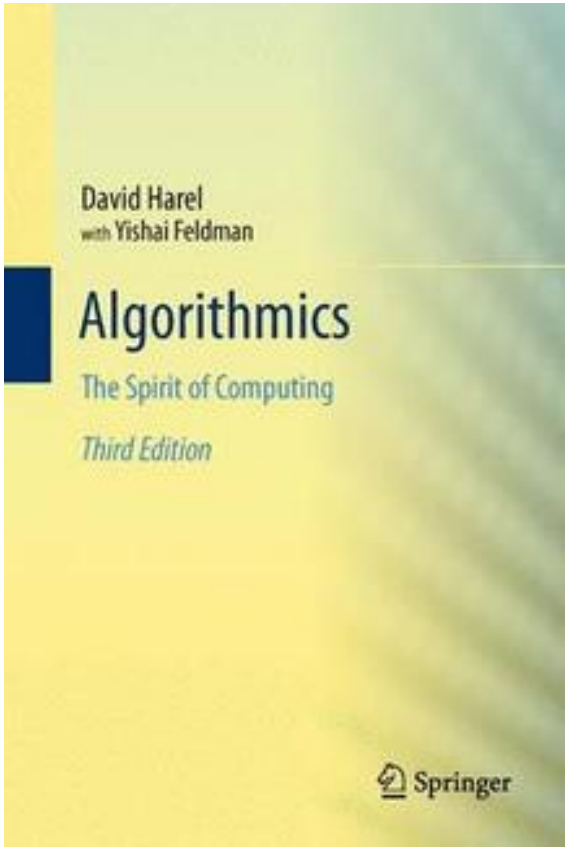
  - F is realized by a `learning' TM with *log p(n)*-bounded disk and *p(n)*-bounded advice.
  - F is realized by a family of *p(n)*-size bounded smart automata with *p(n)*-bounded queries to a TM-oracle.

- Models return later..

After Turing …

- Universal machine' concept
  - Computer revolution after WO II (not only Turing!)
  - Scientific (Turing: practical) computing, data processing
- Programming as major concern
  - Subroutines (Turing) -> levels of abstraction
  - Virtual machines
  - Program proving (Turing: by assertions, 1949)
  - Languages
- Design as major concern
  - Topdown programming (Turing: Programmer's Handbook, Manchester, 1951)
  - Design methodologies
- Algorithms as major concern
  - Design liberated from the `lower level' (virtual) machines
  - From computability to (computational) complexity: e.g. NC vs P, P vs NP, …
  - *Algorithmic (computational) thinking*
  - *Natural information processes*

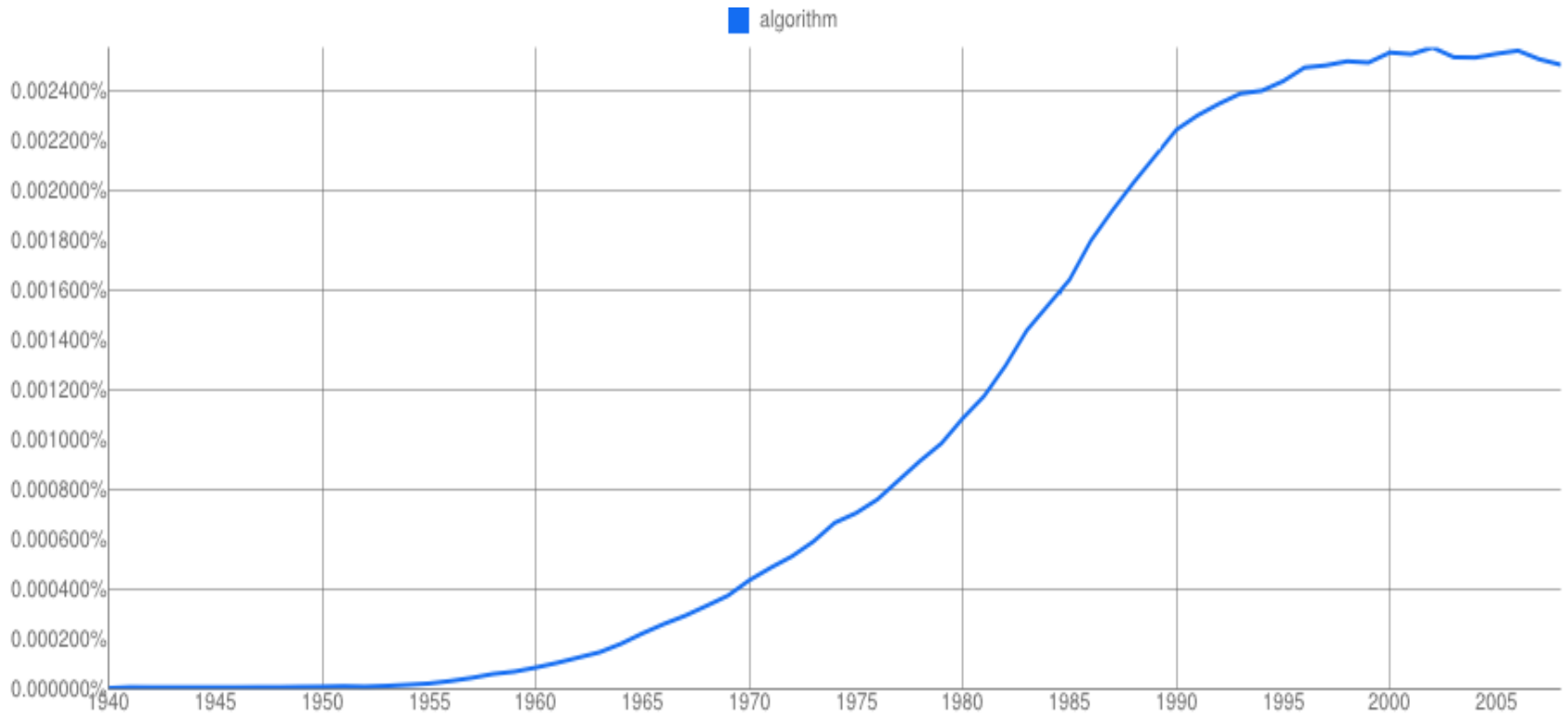




New printing for the Turing Year

- PART I. PRELIMINARIES
- 1. Introduction And Historical Review or, What's It All About? 2. Algorithms And Data or, Getting It Done 3. Programming Languages and Paradigms or, Getting It Done by Computer
- PART II. METHODS AND ANALYSIS 4. Algorithmic Methods or, Getting It Done Methodically 5. The Correctness of Algorithms or, Getting It Done Right 6. The Efficiency of Algorithms or, Getting It Done Cheaply
- PART III. LIMITATIONS AND ROBUSTNESS 7. Inefficiency and Intractability or, You Can't Always Get It Done Cheaply 8. Noncomputability and Undecidability or, Sometimes You Can't Get It Done At All! 9. Algorithmic Universality and Its Robustness or, The Simplest Machines That Get It Done
- PART IV. RELAXING THE RULES 10. Parallelism, Concurrency and Alternative Models or, Getting Lots Of Stuff Done at Once 11. Probabilistic Algorithms or, Getting It Done by Tossing Coins 12. Cryptography and Reliable Interaction or, Getting It Done in Secret
- PART V. THE BIGGER PICTURE 13. Software Engineering or, Getting It Done When It's Large 14. Reactive Systems or, Getting It to Behave Properly Over Time 15. Algorithmics And Intelligence or, Are They Better at It Than Us?

Algorithms: a household word



Google Ngram Viewer

Typical (CS-)algorithms of today

John MacCormick (2012),

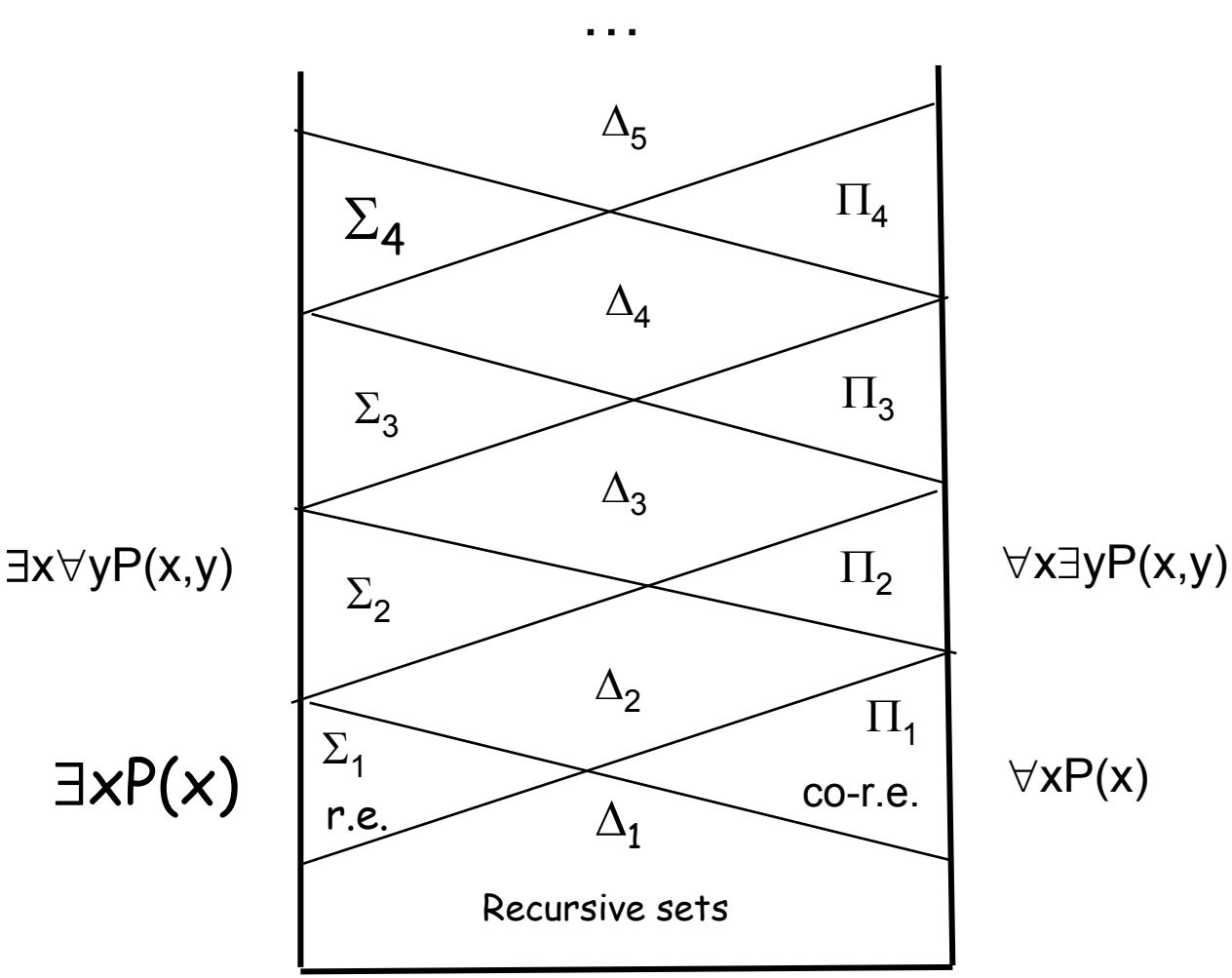
*Nine **Algorithms** That Changed the Future: The Ingenious Ideas That Drive Today's Computers*

- *Search Engine Indexing*: Finding Needles in the World's Biggest Haystack
- *PageRank*: The Technology That Launched Google
- *Public Key Cryptography*: Sending Secrets on a Postcard
- *Error-Correcting Codes*: Mistakes That Fix Themselves
- *Pattern Recognition*: Learning from Experience
- *Data Compression*: Something for Nothing
- *Databases*: The Quest for Consistency
- *Digital Signatures*: Who Really Wrote This Software?
- *What Is Computable*?

Some philosophy of computation

- Shift to computational processes
  - Understanding natural and man-made systems
  - Self-adjusting nature (unpredictable modification, evolution)
  - Cognitive abilities (learning, consciousness, intuition, …)
- Observed/created/embedded in context
  - Interaction with external agents (interactive, reactive, …)
- Unbounded operation
  - Always on, persistent data
  - Escaping the lower level of undecidability (limit notions)
  - Complexity (functions of time, energy, other measures)

Arithmetical hierarchy (cf. Kleene, Mostovski)



Typical computational models (4)

- `Multi-process' TM
  - Inputs of finite length *n*
  - `Red-green' computation, deterministic/non-deterministic
  - Related to Turing's *non-terminating circular a-machines*.
  - Complexity: (finite) number of mind changes.
- Results (JvL, J. Wiedermann)
  - RG machines recognize Σ_2 sets, accept Δ_2 sets.
  - There is no computable function *f* s.t. if L recognized by a *nondet.* red-green TM with *k* mind changes, then L can be recognized by a *det.* red-green TM with *f(k)* mind changes.
  - $P^{MIND} \neq NP^{MIND}$
- Other models for higher levels arithmetical hierarchy
  - Burgin, Gold, Hintikka, Hogarth, Putnam, vL-W, …

VOL. LIX. NO. 236.] [October, 1950

M I N D

A QUARTERLY REVIEW
OF
PSYCHOLOGY AND PHILOSOPHY

I.—COMPUTING MACHINERY AND INTELLIGENCE

BY A. M. TURING

1. *The Imitation Game.*

I PROPOSE to consider the question, ' Can machines think ? ‘ This should begin with definitions of the meaning of the terms ' machine ' and' think '. The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous. If the meaning of the words ' machine‘ and' think ' are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, ' Can machines think ?' is to be sought in a statistical survey such as a Gallup poll.

Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words. The new form of the problem can be described in terms of a game which we call the ' imitation game '.

[Objections’]

Computation after Turing

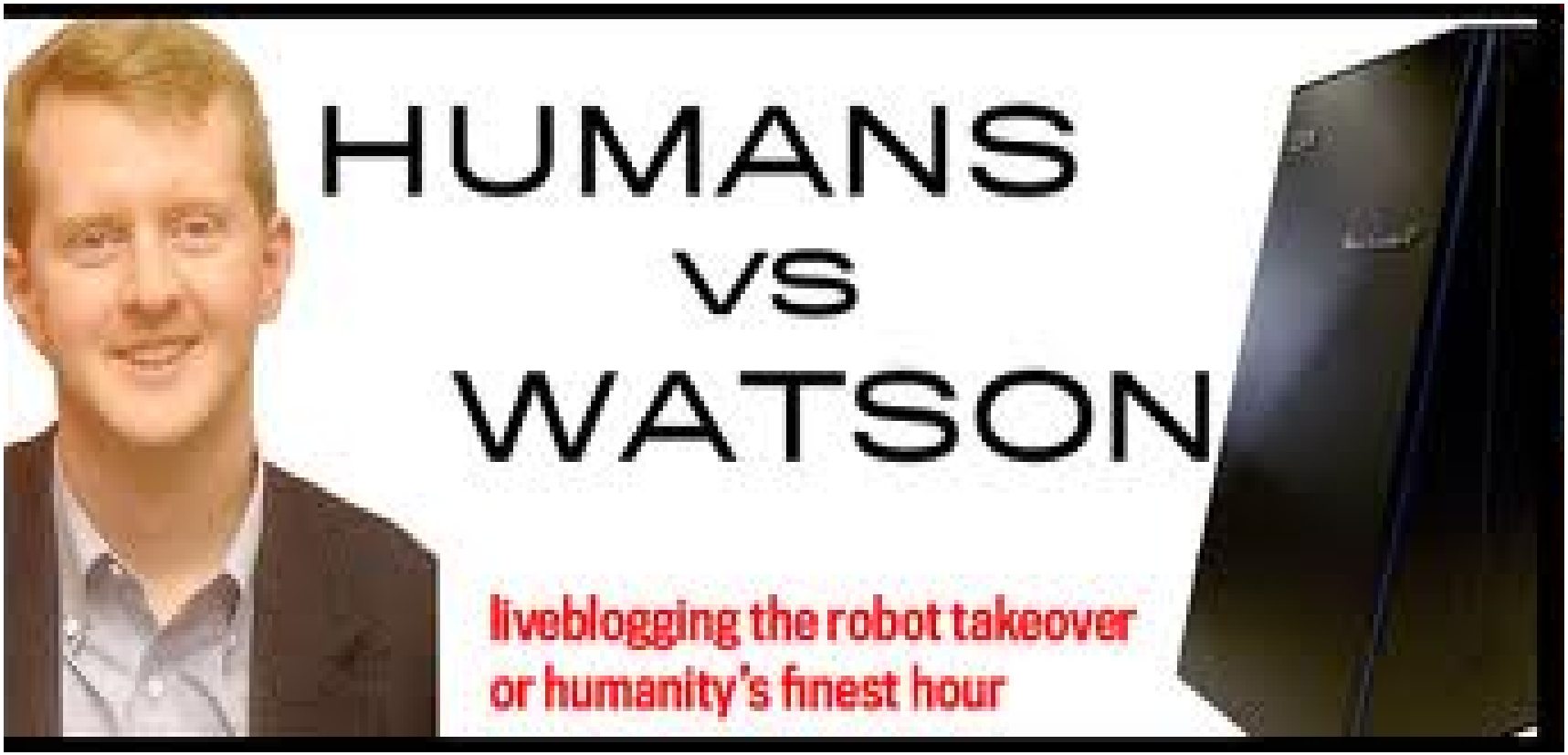
"The original question ' Can machines think' I believe to be too meaningless to deserve discussion. Nevertheless I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted."

Turing (1950)

Cf. Discussion by Searle (`Is the brain a digital computer', 1990).


$1,000
KEN
$2,000
WATSON
$400
BRAD

"They figured out algorithms in a way that they can actually do what the brain does." (Ken Jennings, 2011)



"Watson can process 500 gigabytes, the equivalent of *a million books*, per second.

The content was stored in Watson's RAM for the game because data stored on hard drives are too slow to access. "

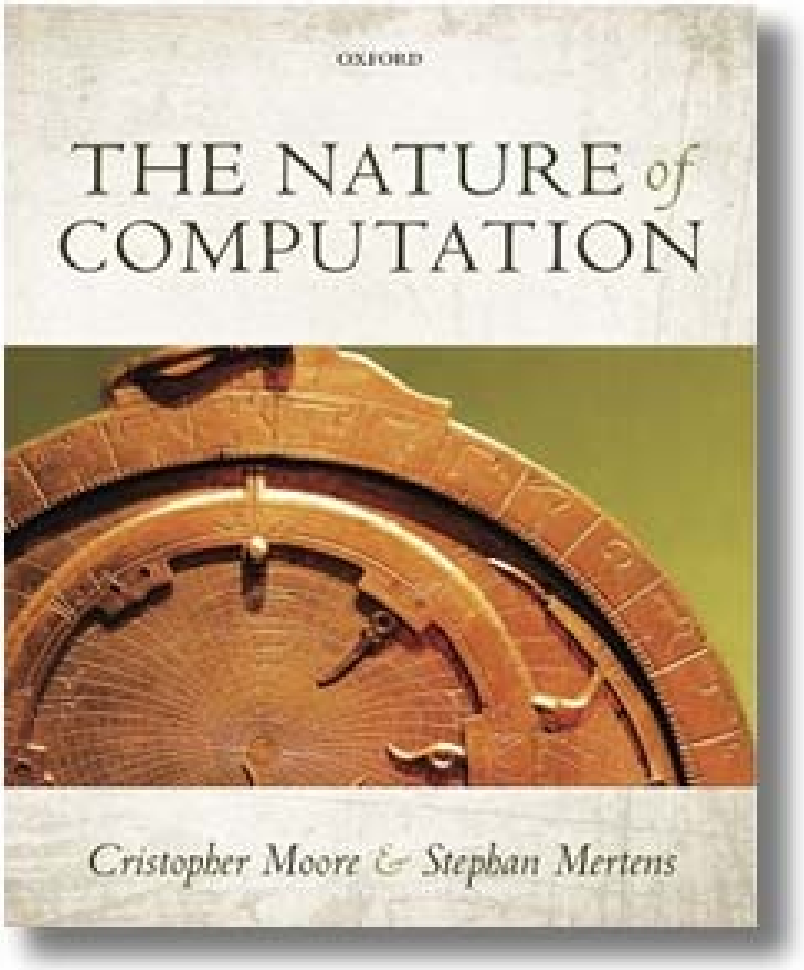
Watson is a workload optimized system designed for complex analytics, made possible by integrating massively parallel POWER7 processors and the IBM DeepQA software to answer *Jeopardy* questions in under three seconds. **Watson is made up of a cluster of ninety IBM Power 750 servers** (plus additional I/O, network and cluster controller nodes in 10 racks**) with a total of 2880 POWER7 processor cores and *16 Terabytes of RAM***. Each Power 750 server uses a 3.5 GHz POWER7 eight core processor, with four threads per core. (Cf. *IBM, Wikipedia*) (Speed characterizes intelligence..)

Computation after Turing …

- *Design and complexity*
  - From computation to computational processes
  - Exact, approximate, practical algorithms
  - Understanding of `complexity', not of `computers'
  - Push for algorithmic thinking
- *The algorithmic lens*
  - New way of doing/understanding science
  - Popularized by R.M. Karp ten years ago
  - Reality as an algorithmic phenomenon
  - Computation as natural phenomenon
  - E-Science
- *A Kuhnian revolution in science*?
  - Science is making the transition from modeling by equations to modeling by algorithms
  - Concept permeates all fields (life sciences, sustainability, ..)

Computational fundaments for science

"Anyone who truly wants to understand how the world works can no more ignore computation than they can ignore relativity or evolution."



What would Turing think of now…

- The

- Creating new realities (i.e. our experience of it)

Project Glass
Google

- Games

- Or … ?

THE ALAN TURING YEAR
A Centenary Celebration of the Life and Work of Alan Turing

2012

- June 23, 2012, is the Centenary of Alan Turing's birth in London. During his relatively brief life, Turing made a unique impact on the history of computing, computer science, artificial intelligence, developmental biology, and the mathematical theory of computability.
- 2012 will be a celebration of Turing's life and scientific impact, with a number of major events taking place throughout the year. Most of these will be linked to places with special significance in Turing's life, such as Cambridge, Manchester and Bletchley Park.
- The Turing Year is coordinated by the Turing Centenary Advisory Committee (TCAC), representing a range of expertise and organisational involvement in the 2012 celebrations. Organisations and individuals wanting to contribute ideas or support for the Turing Year are invited to contact any of the current TCAC members.
- http://www.mathcomp.leeds.ac.uk/turing2012/