

# Alan Turing, the Birth of Computing, and the Power of Mathematics

Rod Downey  
Victoria University  
Wellington  
New Zealand

Victoria University October 2012

# Plan

- ▶ In this lecture I will talk about **some** of Turing's contributions to the theory and practice of computation.
- ▶ A couple of developments from this legacy.
- ▶ I will also mention a few of his other contributions.
- ▶ View them as a case study on how from the mid 20th Century, we have entered the age of mathematics.
- ▶ First a “fly by” or Turing's work.

# The Scope of Turing's Work

- ▶ Turing worked famously on the **Entscheidungsproblem** (=decision problem, more later)
- ▶ And this had the key idea of stored program computers via universal machines....ACE
- ▶ Ideas in cryptography both breaking cryptosystems and making them for voice.
- ▶ Word problems in cancellation semigroups.
- ▶ Cryptography and Statistics.
- ▶ "Checking a Large Routine" symbolic model checking and program verification. His thesis was in this "Systems of logic based on ordinals" and looked at transfinite methods of verification.
- ▶ "Local Programming Methods and Conventions," programming methodology.
- ▶ "Rounding-off Errors in Matrix Processes" Ill-posed problems and "the other" theory of computation.
- ▶ Intelligent Machinery and the Turing test
- ▶ Computer chess (before stored program computers)

# Plan

- ▶ Clearly, I cannot discuss all of this here. For more refer to the archive <http://www.turing.org.uk/sources/biblio.html> and various upcoming Turing volumes.
- ▶ Also the Copeland-Proudfoot article in *The Rutherford Journal* <http://www.rutherfordjournal.org/article040101.html>.
- ▶ I will try to do some of these in some detail and use a broad sweep for others.
- ▶ I will begin with the birth of the digital computer, and the Turing Machine.

- ▶ What research is useful/important?
- ▶ It is prerry clear that it is hard for even the experts to anticipate what will prove to be important.
- ▶ We see a couple of examples in this talk.
- ▶ I realize that most research is “targetted” for outcomes that are easily seen to be important and practical.
- ▶ Here in New Zealand, for instance....



# Born of logic

- ▶ Thanks to Moshe Vardi for this and the next quote (my highlighting).
- ▶ Cosma R. Shalizi, Santa Fe Institute (A famous US think-tank).

*If, in 1901, a talented and sympathetic outsider had been called upon (say by a **granting agency**) to survey the sciences and name a branch that would be the **least fruitful** in the century ahead, his choice might well have settled upon **mathematical logic**, and exceedingly recondite field whose practitioners could all have fit into a small auditorium. It had no practical applications, and not even that much mathematics to show for itself: its crown was an exceedingly obscure definition of cardinal numbers.*

## More recently

- ▶ Martin Davis (1988) Influences of mathematical Logic on Computer Science.

*When I was a student, even the topologists regarded mathematical logicians as living in **outer space**. Today the connections between logic and computers are a matter of **engineering practice** at every level of computer organization.*

- ▶ Yuri Gurevich (Microsoft) quoted as saying engineers need logic not calculus!
- ▶ Read a somewhat dated but wonderful collection in the Bulletin of Symbolic Logic: **On the Unusual Effectiveness of Logic in Computer Science** (Halpern, Harper, Immerman, Kolaitis, and Vardi).
- ▶ Echoes Wigner's 1960 article "The unreasonable effectiveness of mathematics in the natural sciences," and Galileo's "The book of nature is writ in the language of mathematics."



# Entscheidungsproblem

- ▶ The most famous mathematician of his generation, David Hilbert, famously asked for a **decision procedure** for number theory.
- ▶ This was born of 19th century determinism which imagined the universe as a big machine whose path was completely determined.
- ▶ Of course this “universe” version is **still open**: is the universe mechanical; **can the universe produce incomputability?**
- ▶ Still others ask **can the universe produce anything that is computable, or is everything random?**
- ▶ This question is too big for my brain and I will stick with questions in formal logic and number theory.
- ▶ Notice a weaker question is can everything that is **true** of some **formal system** be **proved** in such a system? (**completeness**)

# A history of impossibility proofs

- ▶ You are taught at school that you can solve the quadratic  $ax^2 + bx + c = 0$ .
- ▶ **Ingredients:** numbers,  $+$ ,  $-$ ,  $\times$ , division,  $\sqrt{\quad}$ , maybe cube roots, powers etc.
- ▶ Operations : Combine in sensible ways.
- ▶ Can we do the same for **degree 3**, the “cubic”

$$ax^3 + bx^2 + cx + d = 0?,$$

what about degree 4, etc.

- ▶ This was one of the many questions handed to us by the Greeks.
- ▶ The answer is yes for degree 3 and degree 4; there are such formulae, such “algorithms” using these ingredients for solving them.

# The Sorry Tale

- ▶ For degree 3 this was first proven by Ferro (1500).
- ▶ Ferro left it to his son-in-law Nave and pupil Fiore.
- ▶ (In those days, maths was like a trade, and information was secret; a bit like much modern research supported by drug companies.)
- ▶ Fiore challenged Tartaglia (in 1535) who then re-discovered the solution with a few days to spare, leaving Foire in ignomy.
- ▶ Tartaglia also kept it secret, but told Cardano, who promised by his Christian faith to keep it secret, but....
- ▶ in 1545 Cardano **published it** in his great text **Ars Magna**
- ▶ Additionally Cardano published how to extend to degree 4, being discovered by a student Ferrari.

- ▶ Finally, in 1823, a young Norwegian mathematician, Abel proved that there is **no** recipe using the given ingredients for the degree 5 case, the quintic.
- ▶ (The paper was called “Memoir on algebraic **purifications**...” rather than “Memoir on algebraic equations...” due to a typesetting error.)
- ▶ (My favourite error in one of my own papers referred to a journal “Annals of Mathematical Logic” as “**Animals** of Mathematical Logic.” It made me think of some of my colleagues!)
- ▶ Nobody believed Abel, for a long time. (There had been an earlier announcement by Ruffini, which contained “gaps”.)

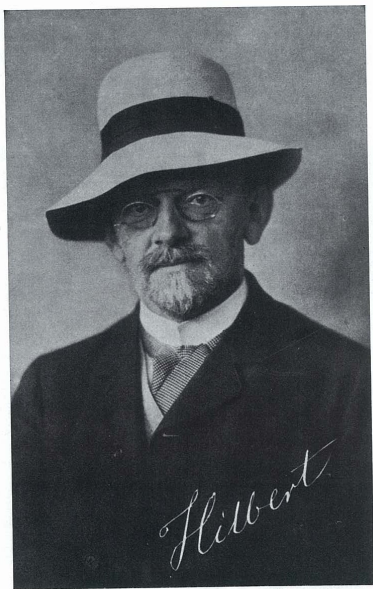
- ▶ Evariste Galois (1811-32) eventually gave a general methodology for deciding if a given degree  $n$  equation admits a solution with the ingredients described.
- ▶ This work laid the basis for **group** theory. Of course this is one of the cornerstones of mathematics, as it is the calculus of symmetries.
- ▶ Galois method is to associate a “group” (a mathematical structure) with each equation, so that the equation is solvable in terms of the given ingredients (arithmetic operations and radicals) **iff** the group has a certain structure on its subgroups. This is one of the gems of mathematics.

# Changing Ingredients

- ▶ It is **not** true to say that the quintic has **no** solution, just none with the given ingredients.
- ▶ We can add some new ingredients : “elliptic functions” and show that there is a method of solving the general degree  $n$  equation.
- ▶ These operations are “mechanical” **so there is a an algorithm for solving all such equations.**
- ▶ Showing the Entscheidungsproblem undecidable asks for the defeat of *any* mechanical method.

# Hilbert

- ▶ David Hilbert, 1900, working from a background of 19th century determinism basically asked the question of whether mathematics could be finitely “mechanized”.



David Hilbert, 1912 — one of a group of portraits of professors  
which were sold as postcards in Göttingen



- ▶ David Hilbert, 1900, working from a background of 19th century determinism basically asked the question of whether mathematics could be finitely “mechanized”.
- ▶ Can we create an algorithm, a machine, into which one feeds a statement about mathematics or at least in a reasonable “formal system” and from the other end a decision emerges: true or false.
- ▶ Or, for a given formal system, can we eventually produce proofs all the “truths” of that system.
- ▶ Hilbert also proposed that we should prove the consistency of various formal systems of mathematics.

# Formal System

- ▶ Leibnitz's dream, the first order logic Frege 1879.
- ▶ It is not important what this is, save to say the type envisioned would be a bunch of axioms, saying things like
- ▶ for all numbers  $x$ ,  $x+1$  exists,
- ▶ for all numbers  $x$  and  $y$   $x + y = y + x$ ,
- ▶ and other "obvious truths."
- ▶ plus rules of inference, like "if whenever  $P$  is true then  $Q$  is true, and whenever  $Q$  is true then  $R$  is true; then whenever  $P$  is true  $R$  is true."
- ▶ induction.

- ▶ Hilbert's dreams were forever shattered by a young mathematician, Kurt Gödel.



- ▶ Hilbert's dreams were forever shattered by a young mathematician, Kurt Gödel.
- ▶ He prove the two **incompleteness** theorems.
- ▶ The first incompleteness theorem says that any sufficiently rich formal system has statements
  - ▶ expressible in the system
  - ▶ true of the system, but
  - ▶ cannot be proven in the system.
- ▶ Secondly no sufficiently rich formal system can prove its own consistency.
- ▶ The collective intuition of a generation of mathematicians was wrong.
- ▶ Of course, Tarski proved that some rich systems like Euclidean Geometry are decidable.

# The confluence of ideas in 1936

- ▶ First Church, then Turing and Post proposed models for computation. In retrospect, it is clear that since the models are equivalent that Church and Kleene first showed that that Entscheidungsproblem is undecidable (by **lambda definable functions**).
- ▶ Church-Kleene's proof filter's through the inner workings of Gödel's proof, whereas nowadays we would work the other way.
- ▶ Church proposed his thesis that these modeled all effectively computable processes.
- ▶ Church  $\lambda$ -definable functions. Herbrand-Gödel general recursive functions (proved the same by Kleene in 1936).
- ▶ Post a Turing machine like model.
- ▶ Turing : Turing machine.

# Turing Machine

- ▶ Machine is a **Box** with a finite number of **Internal States** (i.e. mental states)
- ▶ Reads/writes on a two way potentially infinite tape.
- ▶ Action : can move **Left, Right**, or **(over-)Print a symbol**,
- ▶ Depending on (state, symbol)
- ▶ Now a TV commercial from our sponsors.

# Why Turing?

- ▶ The arguments by Church for the acceptance of  $\lambda$ -definability as capturing were (i) by example (i) confluence (ii) step by step arguments echoing logical proof systems and (iii) failure of diagonalization.
- ▶ These were **not accepted** at the time. See e.g. Davis, Gandy 1995, Soare 2012, Kleene 1995.
- ▶ First and foremost Turing has a **conceptual analysis** giving what many regard as a **proof** of the thesis that TM's capture what is computable.
- ▶ This **analysis** is the **fundamental contribution** of Turing's paper.
- ▶ See "The Universal Turing Machine: A Half Century Survey" R. Herken (ed) Springer 1995 (2nd Ed).



# Turing's analysis

- ▶ He considers an **abstract human computer**
- ▶ By limitations of sensory and mental apparatus we have
  - (i) fixed bound for the symbols.
  - (ii) fixed bound for number of squares
  - (iii) fixed bound to the number of actions at each step
  - (iv) fixed bound on the movement.
  - (v) fixed bound on the number of states.
- ▶ This justifies TM's
- ▶ Also examples like Bessel functions.
- ▶ Gandy, Soare (and others) argue that Turing **proves** any function calculable by an **abstract human** is computable by a TM.
- ▶ For instance, Turing's analysis was the only thing that convinced Gödel.

► Gandy (1995):

*What Turing did, by his analysis of the processes and limitations of calculations of human beings, was to clear away, with a single stroke of his broom, this dependency on contemporary experience, and produce a characterization-within clearly perceived limits- which will stand for all time..... What Turing also did was to show that calculation can be broken down into the iteration (controlled by a “program”) of extremely simple concrete operations; so concrete that they can easily be described in terms of (physical) mechanisms. (The operations of  $\lambda$ -calculus are much more abstract.)*

# The Universal Machine

- ▶ The other **major** contribution was the notion of a **universal machine**, a **compiler**.
- ▶ Turing has the first **universal** machine. The **idea** that there could be a single machine which interpreted programs to emulate any other machine.
- ▶ Church-Kleene's ingenious solution did **not** use the “halting problem” encoded, except implicitly (this is a slightly tricky point and you need to (try to) read the original papers. They are difficult reads.

Turing said in a lecture of 1947 with his design of ACE (automated computing engine)

*The special machine may be called the universal machine; it works in the following quite simple manner. When we have decided what machine we wish to imitate we punch a description of it on the tape of the universal machine... . The universal machine has only to keep looking at this description in order to find out what it should do at each stage. Thus the complexity of the machine to be imitated is concentrated in the tape and does not appear in the universal machine proper in any way... . [D]igital computing machines such as the ACE ... are in fact practical versions of the universal machine.*

# Prehistory and posthistory

- ▶ Babbage said of his **Analytical Engine** (not a stored program machine) “it could do anything except compose country dances.” (quoted in Huskey and Huskey 1980, p 300)
- ▶ Actually now computers do compose country dances.
- ▶ The idea that a computer could be universal was a long time penetrating.
- ▶ Howard Aitken (1956), a US computer expert of the time:

*If it should turn out that the basic logics of a machine designed for numerical solution of differential equations coincide with the logics of a machine intended to make bills for a department store, I would regard this as the most amazing coincidence that I have ever encountered.*
- ▶ Read more on this in Martin Davis' or Herken's books.

# The birth of computers

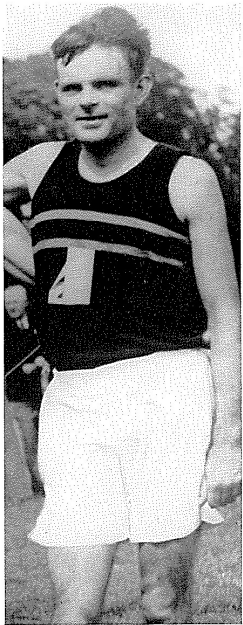
- ▶ Turing was aware of the **possibilities** of using stored program machines.
- ▶ The war intervened and Turing famously was involved in Hut 8. Identified as one of the key codebreakers.
- ▶ Bletchley park had 10,000 members during the war, but it was widely regarded that the people in Hut 8 and, in particular Hilton, Alexander, Tutte, Foss, and notably Turing (who ran the hut) were key players.
- ▶ For example, the **Bombe** modified from Polish ideas, kind of running the Enigma machine backwards
- ▶ Incidentally, I learned only recently that the codes used in the war were actually secure, but brought down by humans, who provided the ingresses for the cracking.

- ▶ It is widely reported that Hut 8, and Bletchley park shortened the war by at least 2 years. Perhaps saving 2 million people.
- ▶ For example, until the 4 rotor Enigma was broken, the UK was starving. German U-boats were sinking supplies from the US routinely. In the month after it was broken, no ships sunk.
- ▶ After brilliant work of Tutte, the *Schüsselzusatz SZ40*, the “tunny”, 12 rotor machine was broken, thereby allowing the allies access to Hitler’s messages to his generals.
- ▶ Solved using **Colossus** machine. A several tonne valve machine, the first large scale (2400 valve) and electronic computer. (NB This is **not** what you read in texts, but now known after declassified documents.)
- ▶ Invented by a cockney, Tommy Flowers, who was **never** allowed to reveal this. Took nine months of his life, working 10 hours a day.
- ▶ The British government gave Flowers £1000, and Turing an OBE.
- ▶ At least it saved Turing later from prison, he only received “treatment” for his “crime” (of homosexuality). In spite of a huge petition this year, no pardon, only an apology.



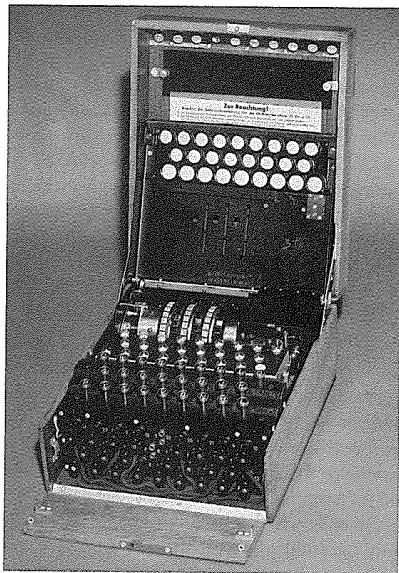
Aged 5





After a successful race. May, 1950





The Enigma Machine, employed by the Germans to encrypt classified and sensitive messages during World War II. (HultonArchive/Getty Images)



John von Neumann, Princeton, 1932

# Crypto and Coding Theory

- ▶ Actually these are remarkable case studies in applied “pure” maths.
- ▶ *Coding Theory* allows us to send messages through noisy channels and figure out what was sent.
- ▶ Invented by Hamming in the mid-20th century.
- ▶ **Modern life would be impossible without it.** (Think bar codes, internet, digital-anything (CD's,DVD's,TV's,cell phones, internet)
- ▶ These things are miracles of engineering, but miracles of mathematics!
- ▶ Cryptography: sending message **without** a third party figuring out what you sent.
- ▶ Modern “public key” invented in the 1977 although earlier by the British secret service in 1973, but never released
- ▶ **Modern life would be impossible without it.** Banking, cell phones, internet, anything involving security.
- ▶ Uses **number theory, group theory, algorithmic randomness, complexity theory.**
- ▶ Famous quote of Hardy: “Nothing I do will every be used...” (A mathematician's apology)

# The Birth of Computers

- ▶ Turing learnt of the possibilities for large scale computers through the work of Tommy Flowers on the **Colossus** machine.
- ▶ McCulloch and Pitt used Turing ideas to show the control mechanism for a TM could be simulated by a finite collection of gates with delays. (1943)
- ▶ Von Neumann knew of Turing's ideas and with two other co-authors proposed a practical architecture for stored program machines. He uses the McCulloch and Pitt ideas. (1945) EDIAC.
- ▶ Later ENVAC.

Stanley Frankel (friend of von Neumann)

*von Neumann was well aware of the fundamental importance of Turing's paper of 1936 'On computable numbers ...', which describes in principle the 'Universal Computer'.... Many people have acclaimed von Neumann as the 'father of the computer' (in a modern sense of the term) but I am sure that he would never have made that mistake himself. He might well be called the midwife, perhaps, but he firmly emphasized to me, and to others I am sure, that the fundamental conception is owing to Turing*

- ▶ Turing proposed ACE (automated computing engine), Never built due to all kinds of things. Architecture very influential.
- ▶ For example, Huxley's G15 computer, the first PC (about the size of a fridge) was based on it, with about 400 sold worldwide, and remaining in use until 1970(!).
- ▶ However, first stored program computer in Manchester, in lab run by Turing's lifetime friend Max Newmann.
- ▶ Turing wrote the (first) programming manual.

## A Basic Undecidable Question

- ▶ Using the fact that all Turing machines can be enumerated we can use a beautiful argument of Cantor about differing sizes of infinite sets(!) to show that there is no algorithm to decide the following question.
- ▶ INPUT Turing machine number  $x$  and an input  $y$ .  
QUESTION Does the machine  $x$  halt on input  $y$ .



- ▶ (Proof. Suppose that we could decide this algorithmically. We can then use the decision procedure to construct a machine  $M$  that halts on input  $n$  if  $T_n$  does not halt on input  $n$ , and our machine  $M$  does *not* halt if machine  $T_n$  does halt on input  $n$ . Then  $M$  would be some machine  $T_m$ , but then  $T_m(m)$  halts if and only if  $M(m)$  halts iff  $T_m(m)$  does not halt....)
- ▶ We **code** this problem into others.

## Example-Conway's Theorem

- ▶ Collatz-type functions.  $f(x) = \frac{x}{2}$  if  $x$  is even, and  $f(x) = 3x + 1$  if  $x$  odd.
- ▶ e.g.  $f(3) = 10$   $f(f(3)) = 5$ , get the sequence, 3,10,5,16,8,4,2,1
- ▶ Do you always get to 1? (Still open)
- ▶ General type of question : e.g  $g(x) = 1/2x$  if  $x$  divisible by 4,  $g(x) = 5x - 1$  if  $x$  has remainder 1 when divided by 4, etc.
- ▶ John Conway (1980's) showed that there is no general algorithm to decide  
INPUT A system like the above, and a number  $x$ .  
QUESTION Does  $x$  get back to 1?



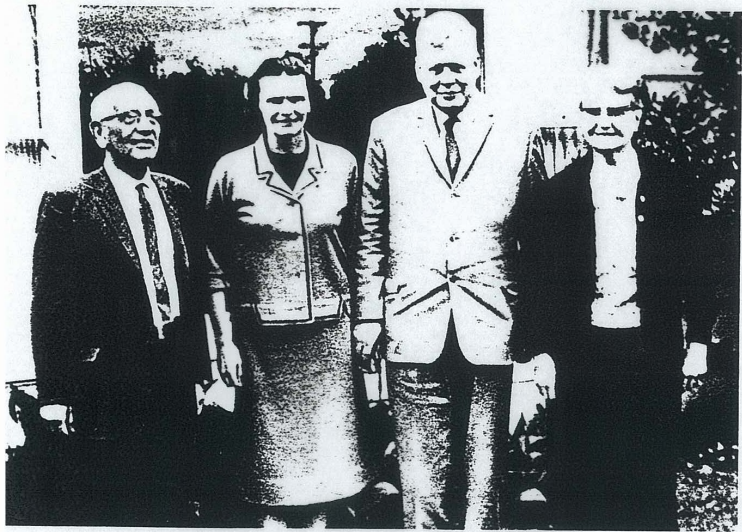
# Wang Tiles

- ▶ INPUT a set of square coloured tiles of the same size. Only same colour borders next to one another.  
QUESTION Can an initial configuration be extended to colour the plane?
- ▶ Wang in the 60's showed that there is no algorithm to decide this.

- ▶ This is very closely related to DNA.
- ▶ Early work on cellular automata.
- ▶ DNA can be thought of as programming languages.
- ▶ Recent “self assembly” modelled by tiles where certain tiles can “bond” (in a nondeterministic way) with others. (Winfree, only verified by experiment 15 years later)
- ▶ Currently going back to construct nano-scale molecular shapes (e.g. “smiley faces”) at will.

# Hilbert's 10th problem

- ▶ INPUT A rational polynomial  $P$  in variables  $x_1, \dots, x_n$   
QUESTION Is there a positive solution to the equation  $P = 0$ ?
- ▶ Matijasevich, after Julia Robinson in the 70's showed there is no algorithm to decide such questions.
- ▶ But there is now a polynomial whose only positive rational zeroes are the primes!



This shows myself, Julia Robinson, Raphael Robinson, and my wife.

$$\begin{aligned}
Q(a, \dots, z) = & (k+2)\{1 - [wz + h + j - q]^2 \\
& - [(gk + 2g + k + 1)(h + j) + h - z]^2 \\
& - [2n + p + q + z - e]^2 - [16(k+1)^3(k+2)(n+1)^2 + 1 - f^2]^2 \\
& - [e^3(e+2)(a+1)^2 + 1 - o^2]^2 - [(a^2 - 1)y^2 + 1 - x^2]^2 \\
& - [16r^2y^4(a^2 - 1) + 1 - u^2]^2 \\
& - [((a + u^2(u^2 - a))^2 - 1)(n + 4dy)^2 + 1 - (x + cu)^2]^2 \\
& - [n + 1 + v - y]^2 \\
& - [(a^2 - 1)l^2 + 1 - m^2]^2 - [ai + k + 1 - l - i]^2 \\
& - [p + l(a - n - 1) + b(2an + 2a - n^2 - 2n - 2) - m]^2 \\
& - [q + y(a - p - 1) + s(2ap + 2a - p^2 - 2p - 2) - x]^2 \\
& - [z + pl(a - p) + t(2ap - p^2 - 1) - pm]^2\}.
\end{aligned}$$



# Hilbert's 10th problem

- ▶ INPUT A polynomial  $P$  in variables  $x_1, \dots, x_n$   
QUESTION Is there a positive solution to the equation  $P = 0$ ?
- ▶ Matijasevich, after Julia Robinson in the 70's showed there is no algorithm to decide such questions.
- ▶ But there is now a polynomial whose only rational zeroes are the primes!

## Recent Examples

- ▶ Recently it was shown by Braverman and Yampolsky (STOC, 2007) that Julia sets can be noncomputable, any halting problem being codable. (Also Blum-Smale-Shub, but that's another story.)
- ▶ Julia set:  $z \mapsto z^2 + \alpha z$ , where  $\alpha = e^{2\pi i \theta}$ .
- ▶ Nabutovsky and Weinberger (Geometriae Dedicata, 2003) showed that basins of attraction in differential geometry faithfully emulated certain computations. Refer to Soare Bull. Symbolic Logic.
- ▶ Remark: Earlier and ignored work by Lee Rubel on universal PDE's.

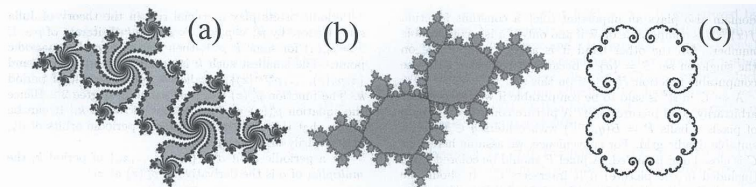


Figure 1: Examples of quadratic Julia sets  $J_p$  (black), and filled Julia sets  $K_p$  (gray); orbits that originate at white points escape to  $\infty$ ; note that on picture (c)  $K_p = J_p$ , since  $K_p$  has empty interior

- ▶ The answer is inescapable: these diverse mathematical objects, tiles, Conway sequences, and polynomials can be used to simulate computations.

## But does it matter?

- ▶ Most problems **in real life** seem to be tractable.
- ▶ For example: why do sat solvers work so well on real problems?
- ▶ Why is genomic sequencing possible?
- ▶ For example: word problems are generically decidable.
- ▶ For example: big hardware is routinely verified (see below).
- ▶ **Your challenge:** explain this.

## Other work of Turing

- ▶ Lots of technical work in logic.
- ▶ Proofs of equivalence of the models. (JSL papers)
- ▶ Undecidability of the **word problem** for cancellation semigroups.
- ▶ Proposed methods for **symbolic verification** of programs. Symbolic verification has grown into modern model checking, though not really using Turing's ideas.
- ▶ Proposed methods of logically constructing programs.
- ▶ First computer chess program (1950). See the webcast of Kasparov's talk in Manchester, Turing 100 conference.

# Model checking

- ▶ The world is full of hardware and algorithmic processes.
- ▶ It is **good** to know that they don't have catastrophic failures.
- ▶ Turing proposed methods for **symbolic verification** of programs. This has grown into modern model checking.
- ▶ Idea : represent processes by **symbols**, and have a **transition "calculus"** and then verify by calculation.
- ▶ Began with things like Hoare logic ("logic is the calculus of computer science").
- ▶ **Modern life would be impossible without it.** It is very bad if any of the embedded hardware in e.g. a plane fails.
- ▶ Works well for hardware, still in development for software.
- ▶ Can be applied to e.g. **industrial processes**.

# Machine Intelligence

- ▶ Famous unpublished paper on this from a sabbatical at Cambridge.
- ▶ His boss (Charles Darwin) thought it was a “schoolboy paper”. Would not let it be published. Now it is regarded as a classic.
- ▶ Later famously posed the **Turing Test**.
- ▶ Often mis-quoted as saying machine intelligence by the end of the 20th century. Actual quote (from a radio discussion with Max Newmann) “at least 100 years.”
- ▶ Emphasized **optimization** as a key strategy for artificial intelligence, and realized in his chess program.



# Machine learning

- ▶ Huge numbers of things are now modelled by machine learning.
- ▶ Huge databases exist and are being “mined”.
- ▶ **Modern life would be impossible without it.** Modern medicine, weather prediction, internet, DNA analysis, evolution of things like language, etc
- ▶ Uses **optimization**, **statistics combinatorics** etc.
- ▶ Currently one of the most important areas of research in CS.
- ▶ We have shown that expert systems are readily modelable, WATSON, etc. This the future of many things in e.g. medicine.

## Some Other Things Left Out

- ▶ "Rounding-off Errors in Matrix Processes" Ill-posed problems and "the other" theory of computation.
- ▶ He was the first to properly study complexity of matrix algorithms like determinant computations when dividing by near zero quantities.
- ▶ This was centered in **numerical analysis**
- ▶ Morphogenesis: How do leopards get their spots?
- ▶ Suggests a simple mechanism based on partial differential equations.
- ▶ 20 years(!) before experimental verification.
- ▶ diffusion/reaction equations.
- ▶ Basically stable, but under perturbation creates a feedback loop.

# Partial differential equations

- ▶ These are equations which model continuous processes
- ▶ Modern life would be impossible without it.
- ▶ E.g. Any scanning device, any modelling in physics, modelling continuous industrial processes, computer graphics (e.g. Avatar etc), electronics, materials science, etc.

# Conclusions

- ▶ The most profound changes in human history, perhaps, came from small beginnings in logic.
- ▶ Mathematics **permeates** modern life. It is **essential**.
- ▶ Almost all recent developments in applied science have maths underpinning them.
- ▶ A consequence of this is that to understand modern life and not treat it as magic, you should understand and appreciate mathematics.
- ▶ Turing had great vision anticipating modern developments.
- ▶ **Maths Rocks**
- ▶ **Turing Rocks**

Thank You