Logic, Maths and Modern Society

Rod Downey
Victoria University
Wellington
New Zealand

2019

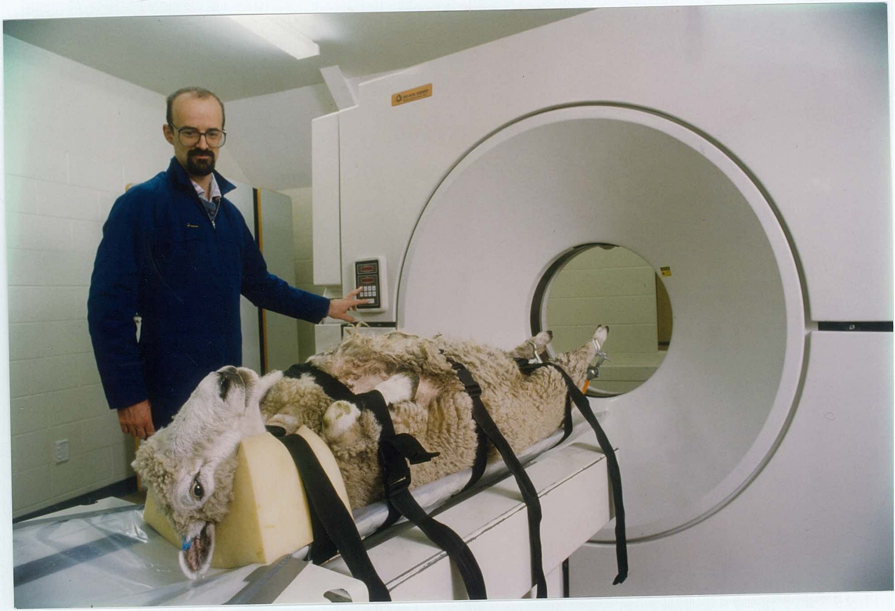
"The book of nature is writ in the language of mathematics."-Galileo

Plan

- So what do mathematicians do?
- So what is logic and why do I care?
- Where did it come from?
- The backbone of modern society.
- Miscellaneous examples.
- A couple of pointers as to what kinds of stuff I do.

Research

- What research is useful/important?
- It is pretty 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 “targeted” for outcomes that are easily seen to be important and practical.
- Here in New Zealand, for instance....



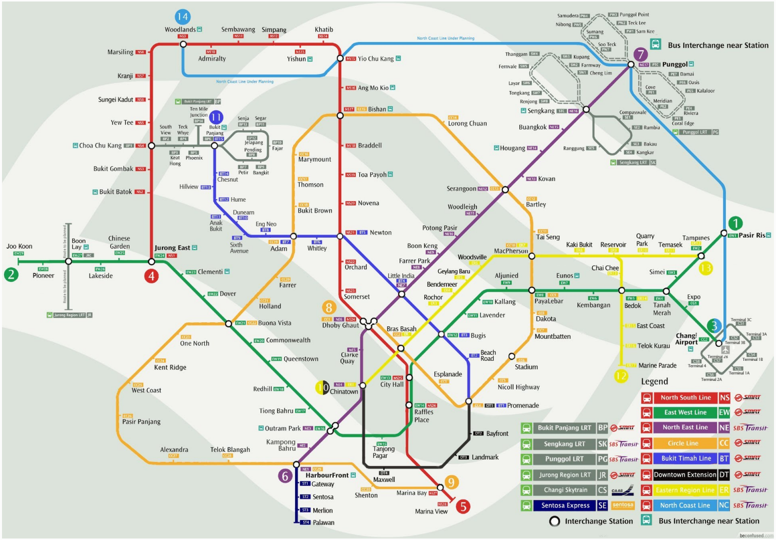
Mathematics

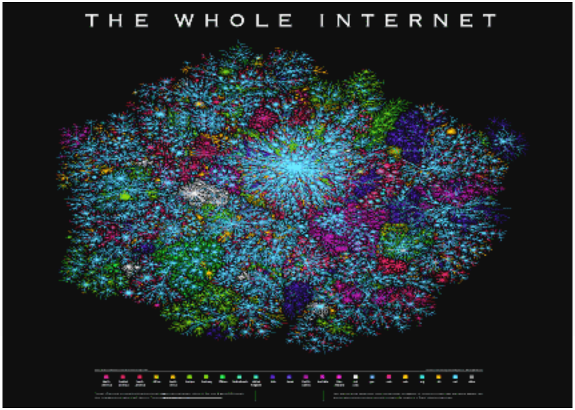
- So what do we do?
- Ultimately I think mathematicians build symbolic models of the world.
- Then manipulating them allows them to understand/predict/explore.
- The Egyptians/Babylonians/Greeks/Chinese/Incas invented geometry to help building and the motions of the cosmos etc.
- They and others invented methods of calculating interest rates etc to make money.
- Later from physics we invented differential equations which can be used to describe rates of change. Witness the CT scan above.

An Example

- From my own work.
- Graphs: These are abstract models consisting of points ("vertices") and lines ("edges") between them.
- Graphs are abstract models of many situations.
- For example, the points might be people and we might connect if they are friends.
- Maybe we might then "cluster edit" to find groups of mutual friends cliques.
- This is a high level view of how ad targeting in e.g. Google works.
- But the points might represent bits of DNA and we might be figuring out what causes a disease, etc.
- Or it might be bits of music and measuring similarity using "Kolmogorov complexity"

Some examples

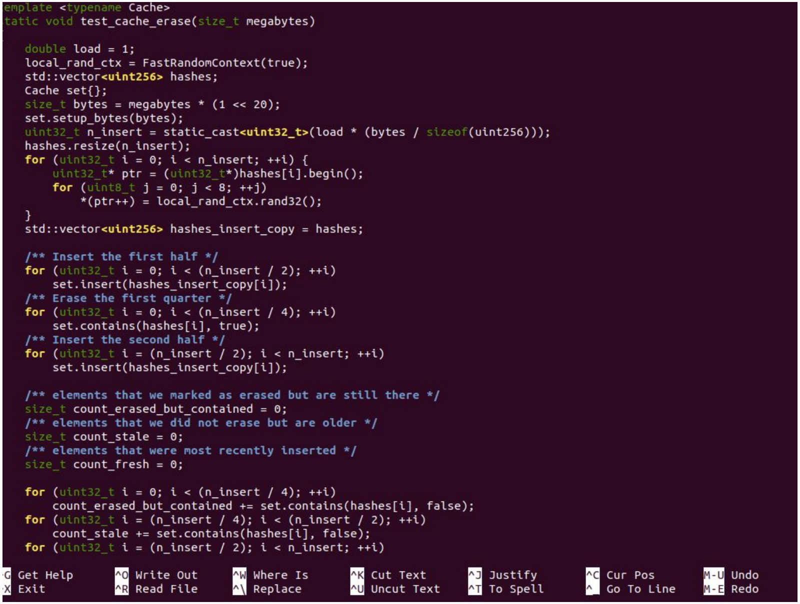



THE WHOLE INTERNET

Algorithms

- What is the point of this abstraction? The thing is that if we understand properties of types of graphs, then no matter what the application the properties will hold.
- In my work, this has applied to algorithm design.
- We gave an approach which gave methods (algorithms) for, e.g. cluster editing in certain kinds of graphs.
- More later, but algorithms here mean sequences of instructions which tell you how to do something.
- Baking a cake; working out your tax return; assembling flat furniture.

Or formal stuff



```
emplate <typename Cache>
tatic void test_cache_erase(size_t megabytes)

    double load = 1;
    local_rand_ctx = FastRandomContext(true);
    std::vector<uint256> hashes;
    Cache set{};
    size_t bytes = megabytes * (1 << 20);
    set.setup_bytes(bytes);
    uint32_t n_insert = static_cast<uint32_t>(load * (bytes / sizeof(uint256)));
    hashes.resize(n_insert);
    for (uint32_t i = 0; i < n_insert; ++i) {
        uint32_t* ptr = (uint32_t*)hashes[i].begin();
        for (uint8_t j = 0; j < 8; ++j)
            *(ptr++) = local_rand_ctx.rand32();
    }
    std::vector<uint256> hashes_insert_copy = hashes;

    /** Insert the first half */
    for (uint32_t i = 0; i < (n_insert / 2); ++i)
        set.insert(hashes_insert_copy[i]);
    /** Erase the first quarter */
    for (uint32_t i = 0; i < (n_insert / 4); ++i)
        set.contains(hashes[i], true);
    /** Insert the second half */
    for (uint32_t i = (n_insert / 2); i < n_insert; ++i)
        set.insert(hashes_insert_copy[i]);

    /** elements that we marked as erased but are still there */
    size_t count_erased_but_contained = 0;
    /** elements that we did not erase but are older */
    size_t count_stale = 0;
    /** elements that were most recently inserted */
    size_t count_fresh = 0;

    for (uint32_t i = 0; i < (n_insert / 4); ++i)
        count_erased_but_contained += set.contains(hashes[i], false);
    for (uint32_t i = (n_insert / 4); i < (n_insert / 2); ++i)
        count_stale += set.contains(hashes[i], false);
    for (uint32_t i = (n_insert / 2); i < n_insert; ++i)

G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos      M-U Undo
X Exit          ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell     ^  Go To Line   M-E Redo
```

Applications of my own work

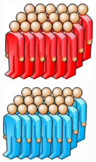
- Mike Langston and his University of Tennessee team.
- Prostate Cancer

APAC 2012

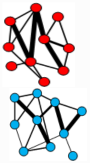
Application: Prostate Cancer

Case

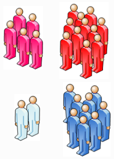
Control



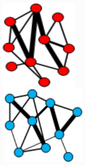
Genes Drive Person-Person Correlations



Select Thresholds, Extract Cliques

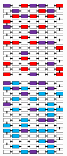


Classify Subtypes and Eliminate Outliers



Persons Drive Gene-Gene Correlations

Select Thresholds, Extract Cliques



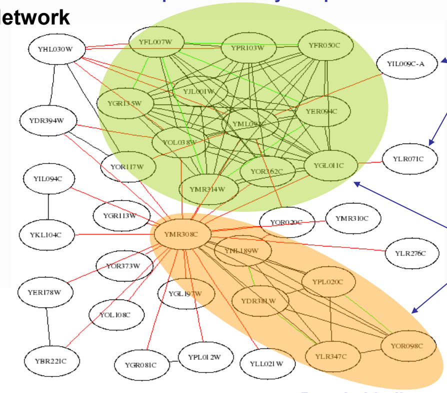
Perform Assorted Forms of Differential Analysis to Identify Network Differences

Sample Result: Putative Prostate Cancer Biomarkers
KLK3 = PSA
ETS1
MAZR
KROX
NFKB

Application, Protein Complex Prediction

Protein-Protein Interaction Network

Peptidase activity complex



yeast proteins

protein complexes

🟥 edge deleted

🟩 edge added

Protein binding complex

Recognize as Cluster Edit

Applications of my own work

- Peter Shaw, Faisal N. Abu-Khzam, Robyn Marsh, Heidi Smith-Vaughan
- Otitis Media (an ear infection) Northern Territory, Australia.
- 30% of aboriginal children are deaf
- 97.5% (!) of indigenous inmates.
- Not understood multi-pathogen disease, so Network Analysis.
- Multi-variable (parameterized) analysis combined with traditional statistical methods (which alone failed).

Clinical

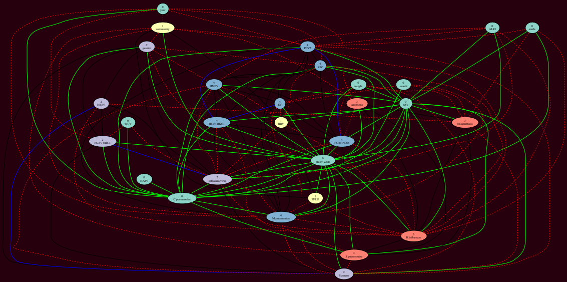


Figure: Pathway analysis of nasal pathogens reveals four distinct clusters using clique as a structural measure. (The diagram was produced using Graphvis by first weighting and coloring the graph based on the clusters found. [1]).

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 the 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."

Formal logic

- Logic studies principles of correct reasoning.
- We represent reasoning and knowledge by symbols in the same way we did graphs.
- Then manipulate the symbols using certain rules of inference (depending on the logics) to make conclusions.
- Logics include modal logics which are used to understand "possible worlds" $\diamond P$ means "P is possible", heavily used in various forms in program verification, quantum logics, fuzzy logics ("P is likely to happen"), threshold logics, which are used in neural nets, etc.
- It is the only part of mathematics that takes language seriously.

Propositional logics

- The simplest system. (This is a wee bit mathematical, sorry.)
- A proposition is a statement that is either true (1) or false (0). e.g. New Zealand has a national government; represented by N, say, and false.
- We analyse compound statements made up from connectives such as "and" $\wedge$, "or" $\vee$, "not" $\neg$, "implies" $\rightarrow$.
- we can define using truth tables

P	Q	$P \wedge Q$	$P \vee Q$	$\neg P$	$P \rightarrow Q$
1	1	1	1	0	1
1	0	0	1	0	0
0	1	0	1	1	1
0	0	0	0	1	1

Truth tables

- We can test statement/arguments in this logic.

  If Jane is a scout, then Fred is a pilot. Jane is a scout. Therefore Fred is a pilot.

- This is called modus ponens and would be represented by

  $J \to F;\ J$
  $\therefore F.$

- Note the argument below not true:
  If Jane is a scout, then Fred is a pilot. Fred is a pilot. Therefore Jane is a scout. This is a common fallacy of formal reasoning. ("Post hoc ergo propter hoc".)
- (Think of a cat thinking: If I am a dog then I have 4 legs. I have 4 legs. Therefore I am a dog.)

The L'Aquila Earthquake of 2009-Bad logic

- L'Aquila is in a seismically active part of Italy. Background earthquakes of magnitude < 2.5 are quite common.
  1. Six seismologists concluded:
     - L'Aquila is a high seismic risk area
     - But earthquake swarms are common there, rarely leading to large earthquakes
     - Nothing indicated that this swarm was different
  2. 6 Days Later: 5.8 magnitude earthquake hits; 380 people die, 1500 injured, and the worst earthquake in 30 years.
- Six seismologists + an engineer were put on trial
- Convicted (October 2012)
  of involuntary manslaughter - the judgment said that they had provided

  "an assessment of the risks that was incomplete, inept, unsuitable, and criminally mistaken

- Worldwide outcry (5000 Italian scientists signed a letter to the President, international scientific organisations objected)
- November 2014: the seismologists convictions were overturned
- November 2015: they were fully acquitted
- What was the misunderstanding of the logic by the court?
  1. About 50% of earthquakes are preceded by foreshocks. i.e. If you have an earthquake then you (probably) had a foreshock. $E \to F$.
  2. The judge concluded that since there was a foreshock, there would be an earthquake. He was using $E \to F; F, \therefore E$.
  3. However, there is very low probability that if there is a foreshock there is an earthquake. $F \to E$ is very unlikely.
  4. This fallacy of reasoning and lack of understanding of stats caused all the problems.
- Actually the above used a probability ("fuzzy") logic, which are used all the time in e.g. washing machines etc.

Algorithms for truth

- We can test by big truth tables. e.g. $(P \wedge (Q \vee R)) \wedge ((P \wedge Q) \wedge \neg(P \to R))$

P	Q	R	$(P \wedge (Q \vee R)) \wedge ((P \wedge Q) \wedge \neg(P \to R))$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

- This is called satisfiable as at at least one line is true.
- Note that every new variable doubles the size of the table.

Want to be rich and famous?

- the SAT problem asks for an efficient method for determining if a statement of propositional logic is satisfiable.
- But we have a method: truth tables!
- The problem is that if we had 100,000 variables (which happens in commercial applications) then generating the truth table would take more time than available in the universe.
- On the other hand we can guess a satisfying assignment of trues and falses and check easily. The $P \neq NP$ conjecture says that there is no efficient method taking e.g. $100,000^3$ steps. (A \$1,000,000 Clay Prize.)
- Why do we care? If $P = NP$ (reasonably) then all modern cryptosystems will be insecure. Modern banking would fail. But lots of algorithms such as scheduling, voice recognition, etc would become much much faster.

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 ever be used..." (A mathematician's apology)

Back to graphs

- The first explicit use of graphs was by Euler.

Königsberg Bridge Problem

Can I travel over all the bridges exactly once?

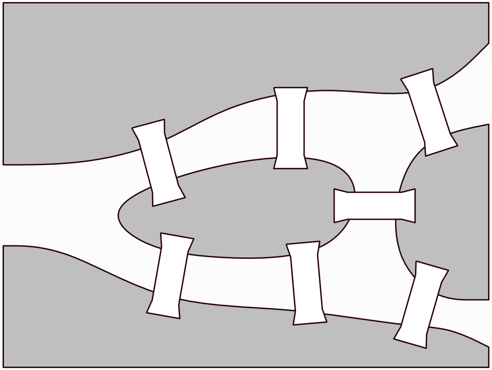
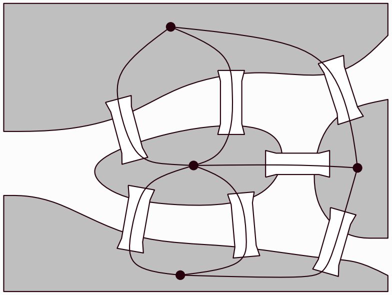


Figure: Königsberg Bridges.

Euler's Analysis

Euler realised that the route taken inside each landmass is completely irrelevant to the problem. So we may as well replace each of the four landmasses with a single vertex, and represent each bridge as an edge joining a pair of landmasses.



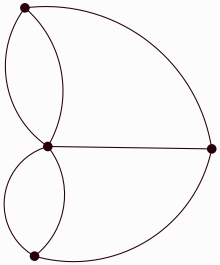


Figure: Representing Königsberg as a (multi-)graph.

Euler's Theorem

A path through all the edges exactly once returning to where you start is called an Euler Cycle.

Theorem (Euler)

A connected multi-graph has an Euler Cycle if and only if all the vertices have even degree (number of edges from them).

- So it is easy to figure out if any graph has an Euler cycle. This problem is in P.
- But we think that to figure out if the network has a path through each **Vertex** exactly once (A Hamilton Cycle) you have to try all possibilities.

Hamilton Cycle

Hamiltonian cycles are named after William Rowan Hamilton (1805–1865). He proposed (and sold!) a board game which involved finding such cycles in the graph (which is called the *dodecahedron graph*). The edges drawn with bold lines show a Hamiltonian cycle. (You may not be surprised to hear that the game was a commercial failure.)

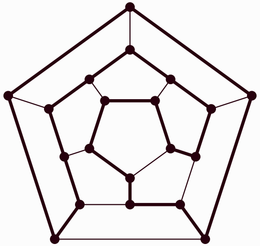


Figure: A Hamiltonian cycle in the dodecahedron graph.

The Core Problem in Complexity

- I know a way to do a task algorithmically.
- How can I prove this is the best, most efficient algorithm for the task.
- What do I mean by that anyway?
- How do I find better algorithms?

The Paradox

- The paradox: We can easily show that many many problems we really care about can be converted into instances of SAT. 50 years of research has generated modern SAT SOLVERS which work very well on instances which come from real data. E.g. NASA uses this for its robot navigation.
- We have no idea why they work. I love this question.
- If we understood this we'd be able to revolutionize algorithm design.

The need for deeply understanding when algorithms work (or not) has never been greater

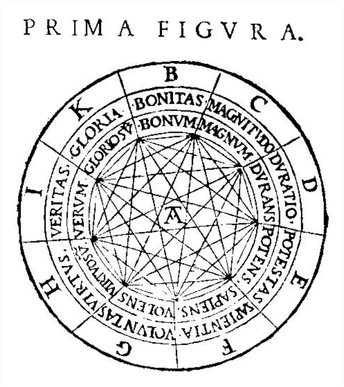
(T. Roughgarden-Beyond Worst Case Analysis-Communications Association for Computing Machinery-2019)

Some of my work

- Sometimes we can explain algorithms which work efficiently when they are not supposed to.
- This is what Mike Fellows and I did beginning in the 1990's. We asked "When is it the case that the only thing you know about a problem is it's size? Answer: Crypto by design."
- So we designed a method of algorithm design which specifically exploited known parameters.
- The paradigm is to have an evolving discourse with the problem to understand what is the cause of intractability.
- This actually occurred by chance. We were trying to understand a very esoteric bit of maths "well-quasi-ordering of finite graphs." From this pure bit of research a new area evolved with many applications.
- It is by logic that we prove, but by intuition that we discover (Henri Poincairé).

Llull

- Raymond Llull (14th C) was one the the first to try to mechanize reasoning.
- He had a reckoner to try to calculate the aspects of god.



Llull based this notion on the idea that there were a limited number of basic, undeniable truths in all fields of knowledge, and that everything about these fields of knowledge could be understood by studying combinations of these elemental truths.

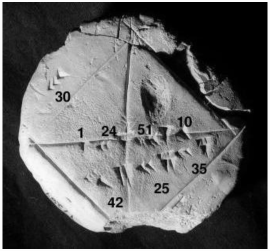
Leibnitz

- Leibnitz believed that much of human reasoning could be reduced to calculations.
- Invented calculus ratiocinator, which resembles symbolic logic,

  The only way to rectify our reasonings is to make them as tangible as those of the Mathematicians, so that we can find our error at a glance, and when there are disputes among persons, we can simply say: Let us calculate, without further ado, to see who is right. (1685)

Algorithms; Calculation

- The earliest examples of algorithms we know can be found from e.g.



- This came from an approximation to $\sqrt{2}$ by the Babylonians nearly 4,000 years ago.

$$1 + \frac{24}{60} + \frac{51}{60^2} + \frac{10}{60^3} \approx 1.414213.$$

- Archimedes had methods of calculating the area of a circle using "polygons". He used a 96-gon. That is, regular, and has 96 sides.
- Amazingly his ideas still influence us today (another talk!)

More algorithms

- Lots of calculation material such as logarithm tables (Napier).
- Famously Astrolabes and calculations of latitudes and longitude (also needs clocks).
- Abacuses.
- Most of historical mathematics was concerned with calculation.
- To my knowledge all calculating devices were task specific.

Back to Logic-History

- In
  the west began with the Greeks, earliest Aristotle via syllogisms such as:

  All humans are mortal.
  All Greeks are humans.
  Therefore, All Greeks are mortal.

- Interestingly, while the Greeks had no symbolic representation, they worked within a rich logic called predicate logic. Nowadays, we'd write this as:

  $\forall x(H_x \rightarrow M_x).$
  $\forall x(G_x \rightarrow H_x)$
  $\therefore \forall x(G_x \rightarrow M_x).$

- Some Greeks have children : $\exists x(G_x \wedge \exists y Cxy)$.
- This logic allows us to say all, or some, of the individuals have a property. The truth of H_x depends on which x is "interpreted".

Entscheidungsproblem

- The most famous mathematician of his generation, David Hilbert, famously asked for a decision procedure for predicate logic like we had with truth tables.
- This was born of 19th century determinism which imagined the universe as a big machine whose path was completely determined.
- Hilbert wanted a *formal system* where everything that is true be proved within the system? (completeness)
- (A proof is a special kind of algorithm, where we deduce statements from axioms and rules of inference.)

- Can we create an algorithm, a machine, into which one feeds a statement about mathematics or at least in a reasonable "formal system" i.e. formal logic, and from the other end a decision emerges: true or false.
- Or, for a given formal system, can we eventually produce proofs of all the "truths" of that system.
- Hilbert also proposed that we should prove the consistency of mathematics; i.e. know we can't prove a contradiction.

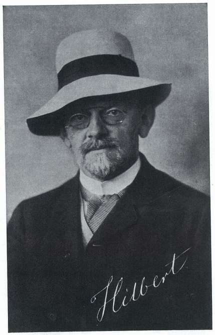
The Dark Lord

- Leibnitz' dreams and Hilbert's dreams were forever shattered by the ideas of a young mathematician, Kurt Gödel.

Gödel

- He proved 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.
- That is, we have no idea if our systems are consistent!
- The collective intuition of a generation of mathematicians was wrong.
- Of course, Tarski proved that some rich systems like Euclidean Geometry are decidable.

Maybe there is hope



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

- Showing the Entscheidungsproblem undecidable asks for the defeat of *any* mechanical method.

The confluence of ideas in 1936

- Gödel's work involved proofs, but the Entscheidungsproblem was still open. Proofs were special algorithms.
- To answer it we'd need to define what mechanical method actually means.
- Note that is a *philosophical* question.
- First Church, then Turing and Post proposed models for computation.
- Turing : Turing machine.

Turing Machine-Model Thought

- 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)

Why Turing?

- 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 computor
- 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.
- Gandy, Soare (and others) argue that Turing proves any function calculable by an abstract human is computable by a TM.

- Gandy (1995):

  What Turing did, by his analysis of the processes and limitations of calculations of human beings, ... 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 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.
- It is so easy now for us to think of everything as data, but these are the papers this idea came from!

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 post-history

- 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

- 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)

John von Neumann, Princeton, 1932

- 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.

- 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*

- Von Neumann, could be the most brilliant person ever, but
- Advocated *A*-bombing Kyoto (and not Nagasaki/Hiroshima) as it would have more effect on the Japanese soul.

  > *"If you say why not bomb them [the Russians] tomorrow, I say why not bomb them today? If you say today at five o'clock, I say why not one o'clock?" (1950)*

Modern algorithmics

- Computing has generated a huge number of questions for mathematicians.
- Many modern algorithms work by learning. Here we have some kind of measure which we try to optimize by using training sequences.
- The idea is to abandon thinking through in advance how to best do a task, but let some kind evolving process do it. The cleverness is the design of how this occurs.
- Goes back to Turing (Chess) and von Neumann.
- What is intelligence? I remember hearing Kasparov in 2012 say he felt that there was intelligence in the program he lost to Deep Blue (1997).
- Also AlphaGo defeated the world Go champion Lee Sendol.
- Later AlphaGo Free, defeated AlphaGo.
- Massive computational power, and neural networks and reinforcement learning.

Should we worry about AI?

- I get asked about this...Yes!
- Quite aside from the thigs we are currently worrying about with purpose built AI (Google Analytics etc) what about:
- Unless you are religious I think you have to conclude that consciousness must be mechanical. We seem determined to make this.
- Suppose that we actually model brains with neural nets. Some facts:
  1. Neurone information transfer speed: 200 Hz, Computer 2G Hz
  2. Axon speed: 100 metres/s Computer 300,000,000 m/s
  3. Maximum size: a very big head. Computer ??
- There has not been much historical evidence for stronger species looking after the weak.

Other methods

- We also use statistical learning, and probabilistic techniques.
- Some of my work involves algorithmic randomness, which is the mathematics of how to understand randomness.
- We live in the age of statistics, and yet understand them poorly.

  (H. G. Wells) Statistical thinking will one day be as necessary for efficient citizenship as the ability to read and write.

- But that's another talk.
- Currently, I am working on building a theoretical model of online computation, as there is none.
- These are algorithms which need to react to situations presented by the world. Think Triage Nurse.
- But that's another talk.

Summary

- We live in the most mathematical age of all time.
- Mathematical techniques lie in the heart of almost every device we interact with.
- I have given a brief account of some the history of how we got here.
- Logic was the mother of all of this.
- And a pointer to some of the some of the things I have contributed.
- And pointed out that all of this came from Blue Skies Research

- Thanks for Listening
- Thanks to the Marsden Fund for over 20 years support
- Thanks to Victoria University and my colleagues, all those postdocs and students.
- Thanks most of all to my wife Kristin, for years of tolerance.