Computability and Complexity

Winter 2009
Prof. Anna Karlin
TA: Thach Nguyen

What is this course about?

- Amazing, foundational, blow-your-mind kind of ideas
- It won't be obvious how this will help you with your job, but I promise that it will help you improve your thinking skills.
- It will expand you intellectually.
- And I sincerely hope you will have fun.

- Warning: some of the material is hard and you may not get it right away. Don't give up too easily!!

Acknowledgements

I have taken many of these slides and specific thoughts included here from my brilliant colleagues at other universities, including Scott Aaronson, Sanjeev Arora, Paul Beame, Bernard Chazelle, and the team of CMU's 15-251 course (which includes Anupam Gupta, Luis von Ahn and Stephen Rudich).

Most of today's slides are taken from

CMU course 15-251: Great Ideas in Theoretical Computer Science

Apologies for inconsistency in fonts/colors/styles/animation.

Heads up: we'll be using the board more and more as time goes on.

Humble observation

Contributions from complexity theory in the last 30 years rival those of any field.

I think some of them could shatter your vision of the universe.

Here are some examples:

IP=PSPACE

- Suppose an alien came to earth and said "I can play perfect chess". He (it?) can prove it to you.
- To be convinced of the proof, we would not have to spend billions of years analyzing one move sequence after another. We'd engage in a short conversation with the alien about the sums of certain polynomials over finite fields.

- Courtesy of Scott Aaronson

The Riemann Hypothesis

- Considered by many mathematicians to be the most important unresolved problem in pure mathematics
- Conjecture about the distribution of zeros of the Riemann zeta - function
- 1 Million dollar prize offered by Clay Institute

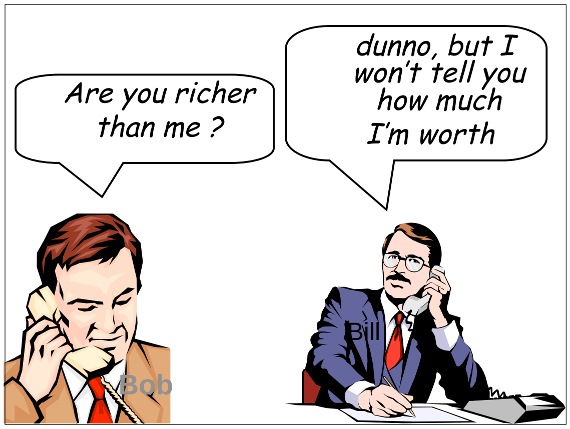
3D Bin Packing is NP-Complete

- There is a finite and not unimaginably large set of boxes, such that if we knew how to pack those boxes into the trunk of your car, then we'd also know a proof of the Riemann Hypothesis. Indeed, *every* formal proof of the Riemann Hypothesis with at most (say) a million symbols corresponds to some way of packing the boxes into your trunk, and vice versa. Furthermore, a list of the boxes and their dimensions can be feasibly written down.
- Courtesy of Scott Aaronson

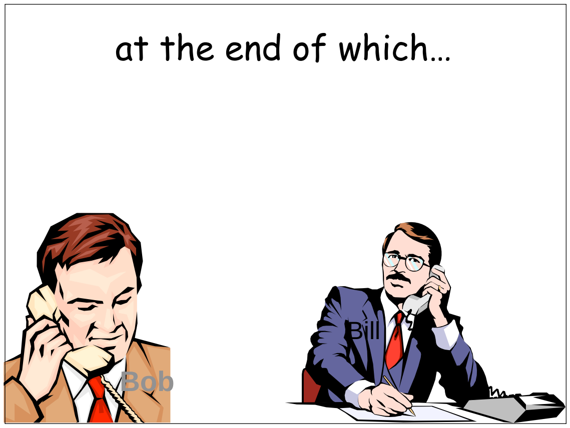
Zero-Knowledge Proofs, PCP Theorem

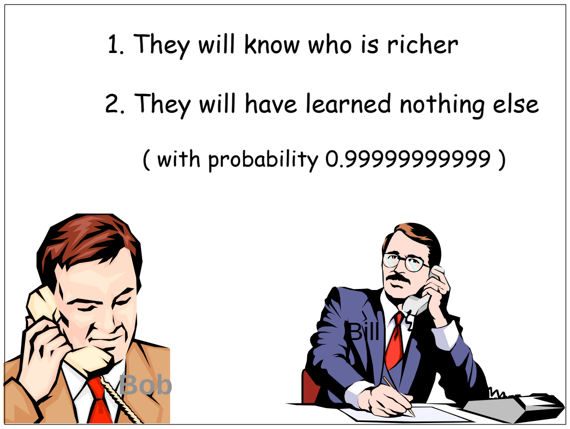
- Suppose you do prove the Riemann Hypothesis. Then it's possible to convince someone of that fact, *without revealing anything other than the fact that you proved it.*
- It's also possible to write the proof down in such a way that someone else could verify it, with very high confidence, *having only seen 3 bits of the proof.*
- Courtesy of Scott Aaronson

Zero Knowledge

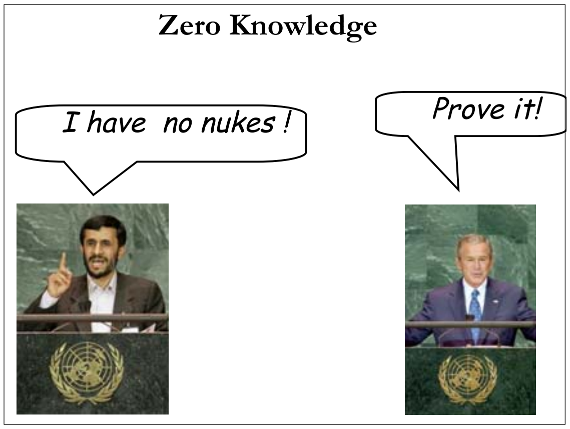


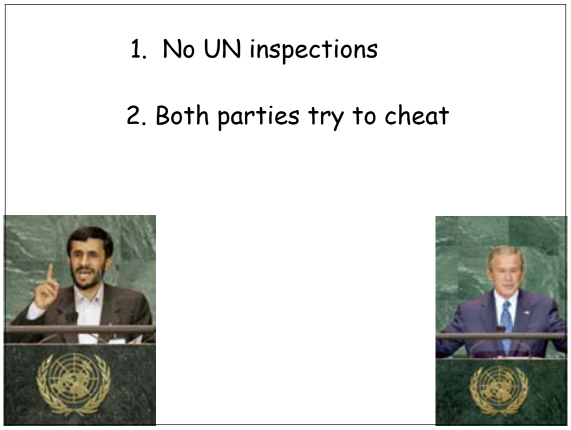
There exists a dialogue…


at the end of which…

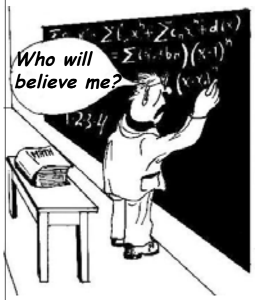

1. They will know who is richer
2. They will have learned nothing else

(with probability 0.99999999999)


Zero Knowledge


1. No UN inspections
2. Both parties try to cheat

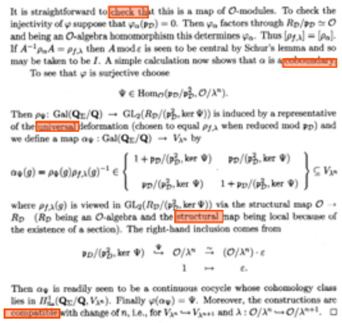

PCP



Step 1 write proof in special format

Step 2 verifier will pick 5 random words

Your Proof of Riemann's Hypothesis



It is straightforward to check that this is a map of $\mathcal{O}$-modules. To check the injectivity of φ suppose that $\varphi_\alpha(\mathfrak{p}_D) = 0$. Then φ_α factors through $R_D/\mathfrak{p}_D \simeq \mathcal{O}$ and being an $\mathcal{O}$-algebra homomorphism this determines φ_α. Thus $[\rho_{f,\lambda}] = [\rho_\alpha]$. If $A^{-1}\rho_\alpha A = \rho_{f,\lambda}$ then $A \bmod \varepsilon$ is seen to be central by Schur's lemma and so may be taken to be I. A simple calculation now shows that α is a coboundary.

To see that φ is surjective choose

$$\Psi \in \mathrm{Hom}_{\mathcal{O}}(\mathfrak{p}_D/\mathfrak{p}_D^2, \mathcal{O}/\lambda^n).$$

Then $\rho_\Psi\colon \mathrm{Gal}(\mathbf{Q}_\Sigma/\mathbf{Q}) \to \mathrm{GL}_2(R_D/(\mathfrak{p}_D^2, \ker\Psi))$ is induced by a representative of the universal deformation (chosen to equal $\rho_{f,\lambda}$ when reduced mod $\mathfrak{p}_D$) and we define a map $\alpha_\Psi : \mathrm{Gal}(\mathbf{Q}_\Sigma/\mathbf{Q}) \to V_{\lambda^n}$ by

$$\alpha_\Psi(g) = \rho_\Psi(g)\rho_{f,\lambda}(g)^{-1} \in \left\{ \begin{matrix} 1 + \mathfrak{p}_D/(\mathfrak{p}_D^2, \ker\Psi) & \mathfrak{p}_D/(\mathfrak{p}_D^2, \ker\Psi) \\ \mathfrak{p}_D/(\mathfrak{p}_D^2, \ker\Psi) & 1 + \mathfrak{p}_D/(\mathfrak{p}_D^2, \ker\Psi) \end{matrix} \right\} \subseteq V_{\lambda^n}$$

where $\rho_{f,\lambda}(g)$ is viewed in $\mathrm{GL}_2(R_D/(\mathfrak{p}_D^2, \ker\Psi))$ via the structural map $\mathcal{O} \to R_D$ (R_D being an $\mathcal{O}$-algebra and the structural map being local because of the existence of a section). The right-hand inclusion comes from

$$\begin{matrix} \mathfrak{p}_D/(\mathfrak{p}_D^2, \ker\Psi) & \xrightarrow{\Psi} & \mathcal{O}/\lambda^n & \xrightarrow{\sim} & (\mathcal{O}/\lambda^n)\cdot\varepsilon \\ & & 1 & \mapsto & \varepsilon. \end{matrix}$$

Then α_Ψ is readily seen to be a continuous cocycle whose cohomology class lies in $H^1_{\Sigma}(\mathbf{Q}_\Sigma/\mathbf{Q}, V_{\lambda^n})$. Finally $\varphi(\alpha_\Psi) = \Psi$. Moreover, the constructions are compatible with change of n, i.e., for $V_{\lambda^n} \hookrightarrow V_{\lambda^{n+1}}$ and $\lambda : \mathcal{O}/\lambda^n \hookrightarrow \mathcal{O}/\lambda^{n+1}$. □

I see something fishy.
I say you're a fool or a liar!

Verifier

Everything looks fine.

I say you're a genius!

In either case, V will be right 0.999999999 of the time

Verifier

Everything looks fine.

I say you're a genius!

If your 2000-page proof is wrong in only one step, how can verifier spot an error in 5 random words?

Verifier

Everything looks fine.

I say you're a genius!

How does verifier know you proved Riemann's hypothesis and not 2+2=4 ?

Verifier

Course Outline (tentative)

- Computability - Turing machines, universality, undecidability
  - Arora, Barak - Chapter 1
  - Sipser -- Chapters 3-5
- NP-completeness
  - Arora, Barak - Chapter 2
  - Sipser -- Chapter 7
- Space Complexity - PSPACE completeness
  - Arora, Barak, Chapter 3
  - Sipser -- Chapter 8
- Randomized computation
  - Arora, Barak, Chapter 7
  - Sipser -- Section 10.2
- Interactive Proof Systems - IP=PSPACE, zero-knowledge proofs
  - Arora, Barak, Chapter 9
  - Sipser - Section 10.4
- Probabilistically Checkable Proofs, hardness of approximation
  - Arora, Barak, Chapter 11
- The Bright Side of Hardness - cryptography
  - Sipser -- Section 10.6
  - Arora, Barak, Chapter 10.

Administrivia

- Course web -- sign up for mailing list.
- Sipser book is highly recommended
- Disconnect between some lectures and the book
- Office hours right before class 5:30 -- 6:30
- Weekly written homeworks, posted on Wednesdays, due 9 days later - 70% of grade
- Turn in by mail to ncthach@cs.washington.edu on Fridays.
- Anonymous feedback

Project

- Short (~10 mins) oral presentation during final 2 weeks of quarter - 30% of grade
  - Either pick one theorem to prove for the class or pick a relevant pop-science/historical book, read it and present some interesting aspects of what you read.
- Example books:
  - The Universal Computer: From Leibniz to Turing
  - Alan Turing: The Enigma
  - The Proof and Paradox of Kurt Godel
  - The Mystery of the Aleph: Mathematics, the Kabbalah, and the Search for Infinity
  - The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography
- Project must be approved no later than March 1.
- Project scheduling, week of March 16.

Computability Theory

The HELLO assignment

Write a JAVA program to output the words "Hello World!" on the screen and halt.

Space and time are not an issue.
The program is for an "ideal" computer, meaning with unlimited memory.

PASS for any working HELLO program, no partial credit.

Grading Script

The grading script G must be able to take any Java program P and grade it.

$$G(P)=\begin{cases}\text{Pass, if P prints only the word "Hello World!" and halts.}\\ \text{Fail, otherwise.}\end{cases}$$

How exactly might such a script work?

It's got to be able to handle programs like this….

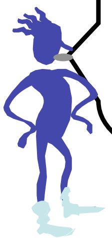
```
_(__,___,____){___/__<=1?_(__,___+1,_
___):!(___%__)?_(__,___+1,0):___%__=
=___/
__&&!____?(printf("%d\t",___/__),_(__,__
_+1,0)):___%__>1&&___%__<___/__?_(
__,1+
___,____+!(___/__%(___%__))):___<__*
__?_(__,___+1,____):0;}main(){_(100,0,0
);}
```

What kind of program could a student who hated his/her TA hand in?

Nasty Program

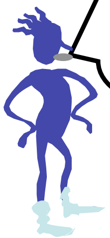
```
n:=0;
while (n is not a counter-example
     to the Riemann Hypothesis) {
     n++;
}
print “Hello World!”;
```

The nasty program is a PASS if and only if the Riemann Hypothesis is false.



A TA nightmare: Despite the simplicity of the HELLO assignment, there is no program to correctly grade it!

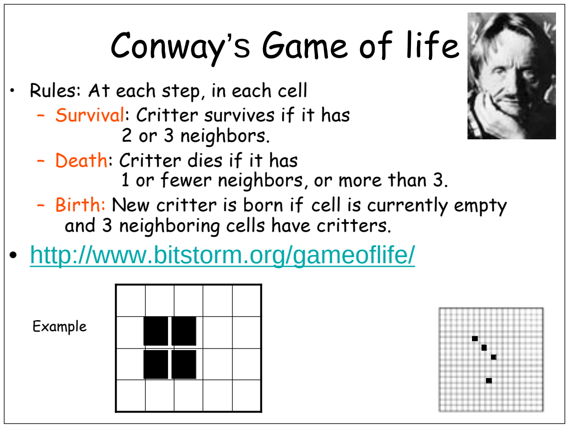
And we will prove this.



The theory of what can and can’t be computed by an ideal computer is called **Computability Theory** or **Recursion Theory**.

Computability

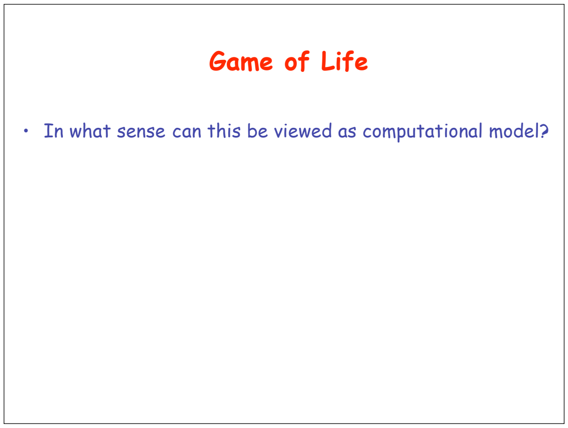
- What is computation?
- Later: Given a computational model, what can we compute and what is impossible to compute?
- And even later: How do we design our computations so they are efficient?



Conway's Game of life

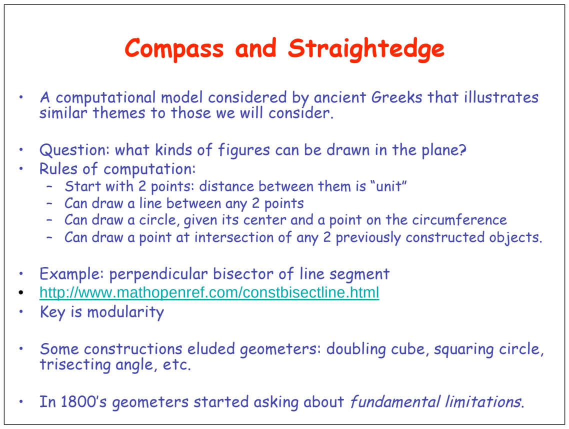
- Rules: At each step, in each cell
  - Survival: Critter survives if it has 2 or 3 neighbors.
  - Death: Critter dies if it has 1 or fewer neighbors, or more than 3.
  - Birth: New critter is born if cell is currently empty and 3 neighboring cells have critters.
- http://www.bitstorm.org/gameoflife/

Example



Game of Life

- In what sense can this be viewed as computational model?



Compass and Straightedge

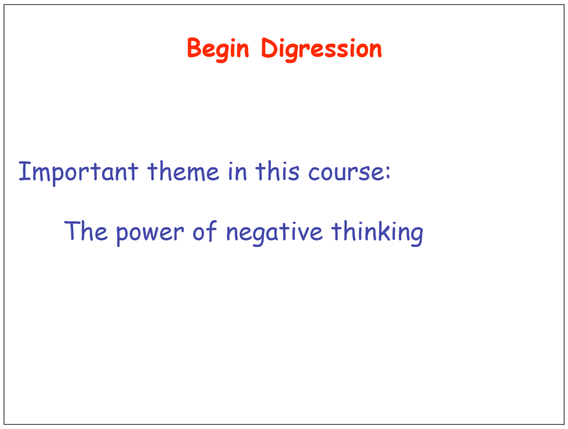
- A computational model considered by ancient Greeks that illustrates similar themes to those we will consider.

- Question: what kinds of figures can be drawn in the plane?
- Rules of computation:
  - Start with 2 points: distance between them is "unit"
  - Can draw a line between any 2 points
  - Can draw a circle, given its center and a point on the circumference
  - Can draw a point at intersection of any 2 previously constructed objects.

- Example: perpendicular bisector of line segment
- http://www.mathopenref.com/constbisectline.html
- Key is modularity

- Some constructions eluded geometers: doubling cube, squaring circle, trisecting angle, etc.

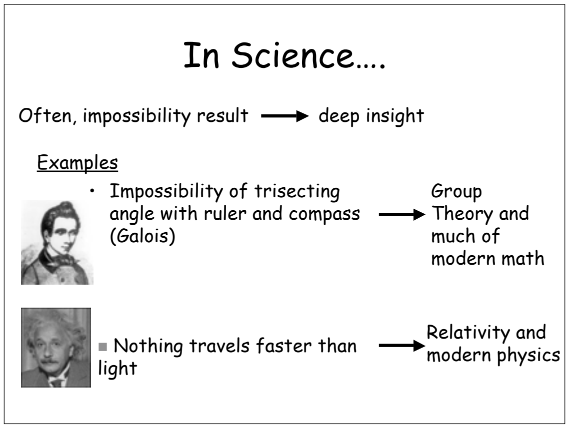
- In 1800's geometers started asking about *fundamental limitations*.



Begin Digression

Important theme in this course:

The power of negative thinking



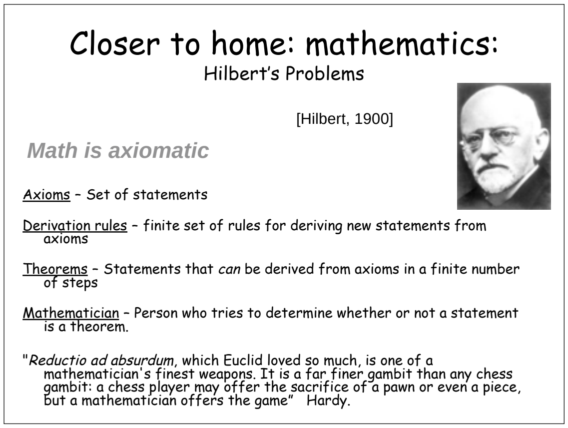
In Science….

Often, impossibility result ⟶ deep insight

Examples

- Impossibility of trisecting angle with ruler and compass (Galois) ⟶ Group Theory and much of modern math

- Nothing travels faster than light ⟶ Relativity and modern physics



Closer to home: mathematics:

Hilbert's Problems

[Hilbert, 1900]

Math is axiomatic

Axioms – Set of statements

Derivation rules – finite set of rules for deriving new statements from axioms

Theorems – Statements that *can* be derived from axioms in a finite number of steps

Mathematician – Person who tries to determine whether or not a statement is a theorem.

"*Reductio ad absurdum*, which Euclid loved so much, is one of a mathematician's finest weapons. It is a far finer gambit than any chess gambit: a chess player may offer the sacrifice of a pawn or even a piece, but a mathematician offers the game" Hardy.

Hilbert's Program

- The goal of Hilbert's program was to provide a secure foundation for all mathematics. This should include:
  - A formalization of all mathematics
  - Completeness: a proof that all true mathematical statements can be proved in the formalism
  - Soundness: a proof that no contradiction can be obtained in the formalism
  - Computability: there should be an algorithm for deciding the truth or falsity of any mathematical statement

Godel's Incompleteness Theorems

- Stunned the mathematical world by showing that most of the goals of Hilbert's program were impossible to achieve.
- First Incompleteness Theorem: In any system of logic that is consistent (can't prove a contradiction) and computable (application of rules is mechanical), there are true statements about integers that can't be proved or disproved within that system .
- Second Incompleteness Theorem: No consistent, computable system of logic can prove its own consistency.

End Digression

Back to models of computation

Turing develops a model of computation

- Wanted a model of human calculation.
- Wanted to strip away inessential details.
- What are the important features?

  - Paper (size? shape?)
  - The ability to read or write what's on the paper.
  - The ability to shift attention to a different part of the paper
  - The ability to have what you do next depend on what part of the paper you are looking at and on what your state of mind is
  - Limited number of possible states of mind.

Let's get our hands a bit dirty…

- Formal model of Turing Machine
- Examples:
  - Palindromes
  - Adding, multiplying, etc.
  - In his original paper, Turing showed how to compute binary representation of e and π, among other things.
- Turing Machine programming techniques
- Details don't matter:
  - Multiple tapes
  - Tape infinite in both directions
  - Size of alphabet

Turing Machine ≡ Ideal C Program

- Ideal C/C++/Java programs
  - Just like the C/C++/Java you're used to programming with, except no bound on amount of memory.
    - **No overflow**
    - **No out of memory errors**
- Equivalent to Turing machines except a lot easier to program !
  - Henceforth, we'll interchangeably talk about programs in your favorite programming language and Turing machines.

Church-Turing Thesis

- *Anything "computable" is computable by Turing machine.*
- *Any "reasonable, physically realizable" model of computation can be simulated on Turing machine with only polynomial slowdown.*
  - *Program in C++, Pascal, Lisp, pseudocode*
  - *Game of Life*
  - *The brain?*
- *Not a theorem. Just a belief, borne out by computational models we know about. Powerful idea.*

Turing's next great insight: duality between programs and data

- Notation:
  - We'll write **<P>** for the code of program **P** and **<P,x>** for the pair of the program code and an input **x**
  - i.e. **<P>** is the program text as a sequence of ASCII symbols and **P** is what actually executes
  - We'll write P(x) to denote the output when we run program P on input x.
- <P> can be viewed as data -- can be input to another program!

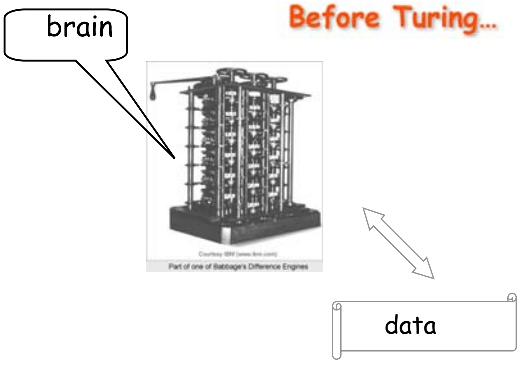
Which leads to Universality!

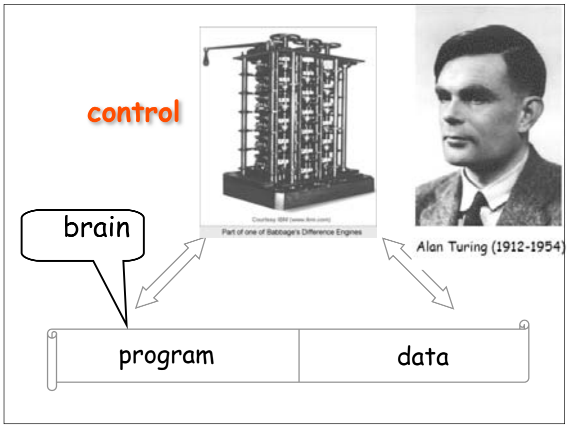
- A Turing machine interpreter **U**
  - On input **<P>** and its input **x**, **U** outputs the same thing as **P** does on input **x**
  - At each step it decodes which operation **P** would have performed and simulates it.
- One Turing machine, the Universal TM, is enough!
  - Basis for modern stored-program computer
    - Von Neuman studied Turing's UTM design

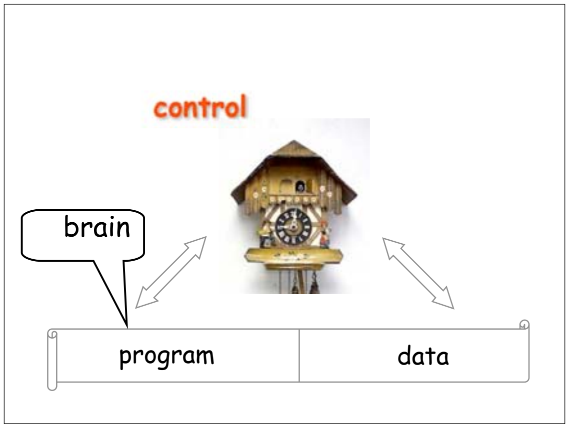
- *"existence of software industry lemma" --Scott Aaronson*

Universality

Before Turing...





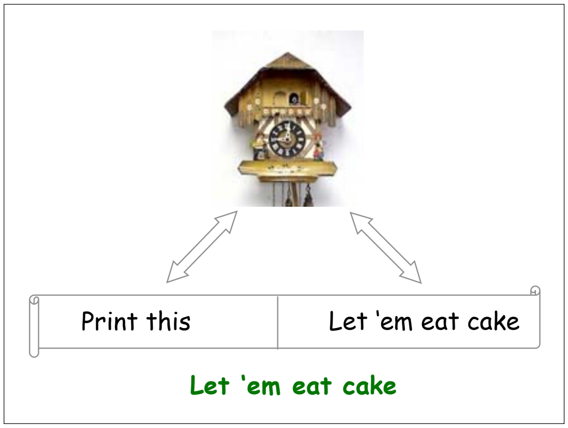


control

brain

program

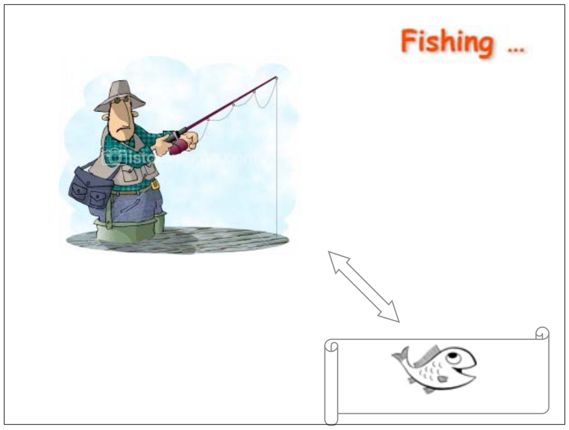
data



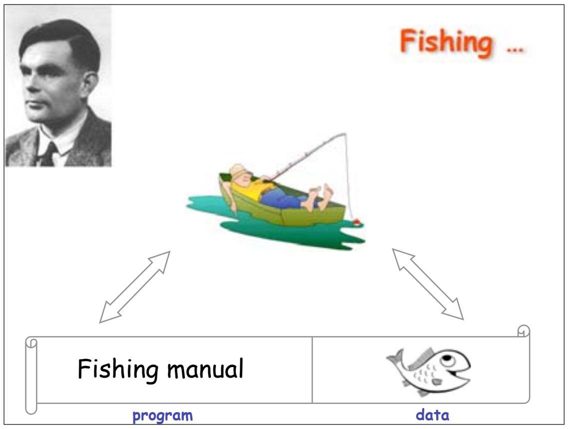
Print this

Let 'em eat cake

Let 'em eat cake



Fishing …

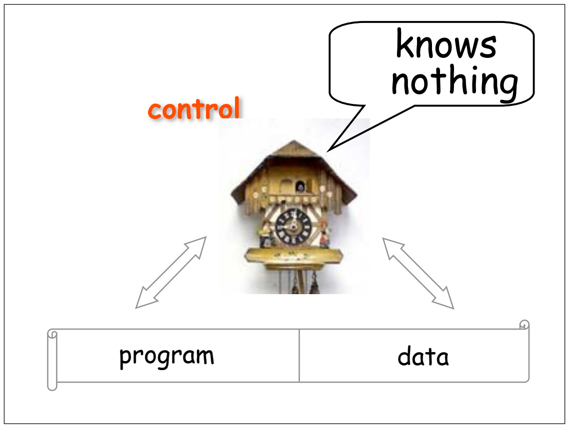


Fishing …

Fishing manual

program

data

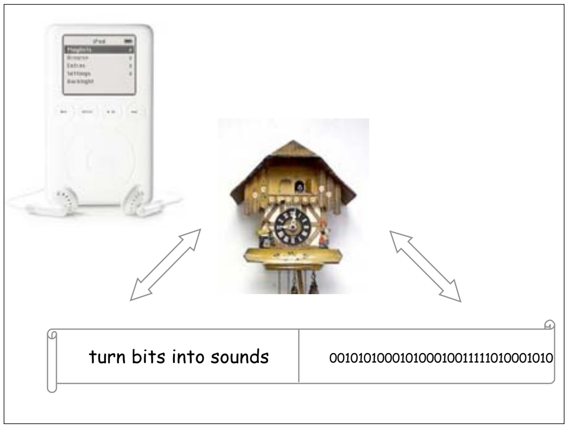


knows nothing

control

program

data

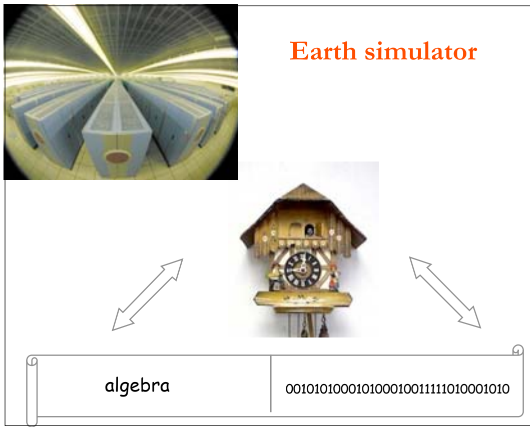


turn bits into sounds

001010100010100010011111010001010

display/organize email

0010101000101000100111111010001010



Earth simulator

algebra

0010101000101000100111111010001010

Finally: some problems can't be solved on computers

We will show that there is no algorithm for solving the "halting problem".

Reminder : P(P) is shorthand for P(<P>), the output obtained when we run P on the text of its own source code

K = { programs P | P(P) halts }

The Halting Problem

Is there a program HALT such that:

HALT(P) = yes, if P(P) halts
HALT(P) = no, if P(P) does not halt

THEOREM: There is no program to solve the halting problem (Alan Turing 1937)

We'll use a "Proof by contradiction"

"When something's not right, it's wrong."

Bob Dylan

THEOREM: There is no program to solve the halting problem (Alan Turing 1937)

Suppose a program HALT existed that solved the halting problem.

HALT(P) = yes, if P(P) halts
HALT(P) = no, if P(P) does not halt

We will call HALT as a subroutine in a new program called CONFUSE.

CONFUSE

```
CONFUSE(P)
{ if (HALT(P))
      then loop forever;          // i.e., we don't halt
  else exit;                      // i.e., we halt

}
```

Does CONFUSE(CONFUSE) halt?

CONFUSE

```
CONFUSE(P)
{ if (HALT(P))
      then loop forever;          // i.e., we dont halt
  else exit;                      // i.e., we halt
}
```

Suppose CONFUSE(CONFUSE) halts:
then HALT(CONFUSE) = TRUE
⇒ CONFUSE will loop forever on input CONFUSE

Suppose CONFUSE(CONFUSE) does not halt
then HALT(CONFUSE) = FALSE
⇒ CONFUSE will halt on input CONFUSE

CONTRADICTION

Alan Turing (1912-1954)

Theorem: [1937]

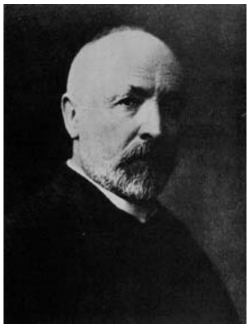
There is no program to solve the halting problem

Detour Through Infinity

∞

What does it mean to say that two sets have the same size?

Georg Cantor (1845-1918)



Cantor's Definition (1874)

Two sets are defined to have the same size, or cardinality, if and only if they can be placed into bijection

Bijection: 1-to-1, onto correspondence

Do N and E have the same cardinality?

N = { 0, 1, 2, 3, 4, 5, 6, 7, … }

E = { 0, 2, 4, 6, 8, 10, 12, … }
The even, natural numbers.

How can E and N have the same cardinality? E is a proper subset of N with plenty left over.

The attempted correspondence
f(x) = x does not take E onto N.

E and N do have the same cardinality!

N = 0, 1, 2, 3, 4, 5, …
E = 0, 2, 4, 6, 8,10, …

f(x) = 2x is a bijection

Lesson:

Cantor's definition only requires that some one-to-one correspondence between the two sets is also onto (i.e., a bijection), not that all one-to-one correspondences are bijections!

This distinction never arises when the sets are finite

Do N and Z have the same cardinality?

N = { 0, 1, 2, 3, 4, 5, 6, 7, … }

Z = { …, -2, -1, 0, 1, 2, 3, … }

N and Z do have the same cardinality!

N = 0, 1, 2, 3, 4, 5, 6 …
Z = 0, 1, -1, 2, -2, 3, -3, ….

$$f(x) = \begin{cases} \lceil x/2 \rceil & \text{if } x \text{ is odd} \\ -x/2 & \text{if } x \text{ is even} \end{cases}$$

A Useful Transitivity Lemma

Lemma:
If
f: A→B is a bijection, and
g: B→C is a bijection.
Then h(x) = g(f(x)) defines a function
h: A→C that is a bijection

Hence, N, E, and Z all have the same cardinality.

Onto the Rationals!

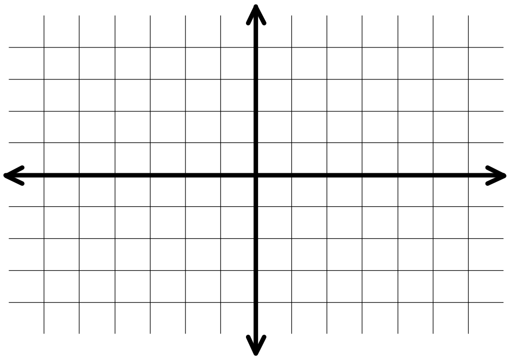
Do N and Q have the same cardinality?

N = { 0, 1, 2, 3, 4, 5, 6, 7, …. }

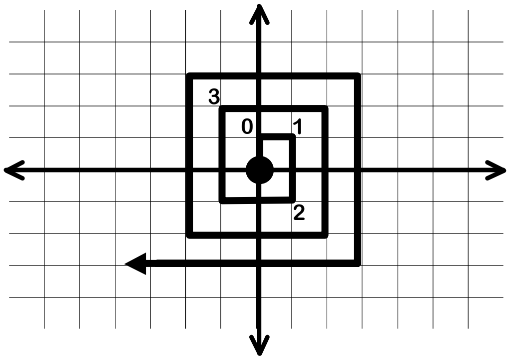
Q = The Rational Numbers

How could it be????

The rationals are dense: between any two there is a third. You can’t list them one by one without leaving out an infinite number of them.



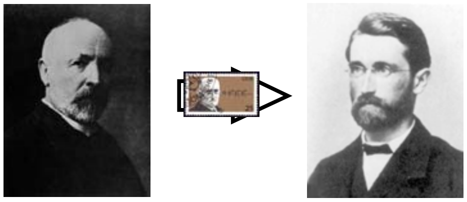
The point at x,y represents x/y



The point at x,y represents x/y

Cantor's 1877 letter to Dedekind:

"I see it, but I don't believe it! "



Countable Sets

We call a set countable if it can be placed into a bijection with the natural numbers N

Hence N, E, Z, Q are all countable

Do N and R have the same cardinality?
I.e., is R countable?

N = { 0, 1, 2, 3, 4, 5, 6, 7, … }

R = The real numbers

Theorem: The set $R_{[0,1]}$ of reals between 0 and 1 is not countable

Proof: (by contradiction)

Suppose $R_{[0,1]}$ is countable

Let f be a bijection from N to $R_{[0,1]}$

Make a list L as follows:

0: decimal expansion of f(0)
1: decimal expansion of f(1)
…
k: decimal expansion of f(k)
…

Position after decimal point

Index

L	0	1	2	3	4	…
0						
1						
2						
3						
…						

Position after decimal point

Index

L	0	1	2	3	4	…
0	3	3	3	3	3	3
1	3	1	4	1	5	9
2	1	2	4	8	1	2
3	4	1	2	2	6	8
…						

L	0	1	2	3	4	…
0	d_0					
1		d_1				
2			d_2			
3				d_3		
…					d_4	

L	0	1	2	3	4
0	d_0 2				
1		d_1 6			
2			d_2 5		
3				d_3 1	
…					… 3

Define the following real number
$\text{Confuse}_L = 0.C_0C_1C_2C_3C_4C_5$ …

$$C_k = \begin{cases} 5, \text{ if } d_k=6 \\ 6, \text{ otherwise} \end{cases}$$

Diagonalized!

By design, Confuse_L can't be on the list L!

Indeed, note that Confuse_L differs from the kth element on the list L in the kth position.

This contradicts the assumption that the list L is complete; i.e., that the map f: N to $R_{[0,1]}$ is onto.

The set of reals is uncountable!
(Even the reals between 0 and 1)

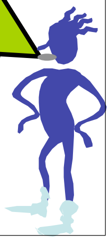
Sanity Check

Why can’t the same argument be used
to show that the set of rationals Q
is uncountable?

End detour through infinity:

What does all this have to do with Turing machines and the Halting problem?

Turing’s argument is essentially the reincarnation of Cantor’s Diagonalization argument that we just saw.



Standard Notation

Σ = Any finite alphabet

Example: {a,b,c,d,e,…,z}

Σ^* = All finite strings of symbols from Σ including the empty string ε

Theorem: Every infinite subset S of Σ^* is countable

Proof:

Sort S first by length and then alphabetically

Map the first word to 0, the second to 1, and so on…

Some infinite subsets of Σ^*

Σ = The symbols on a standard keyboard

For example:

The set of all possible Java programs is a subset of Σ^*

The set of all possible Turing machines is a subset of Σ^*

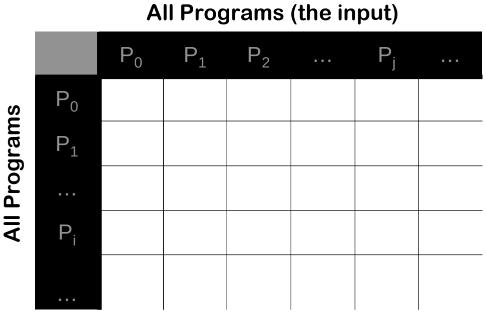
The set of all possible finite pieces of English text is a subset of Σ^*

Thus:

The set of all possible Java programs is countable.

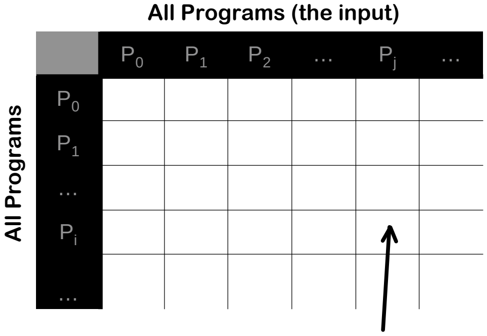
The set of all possible Turing machines is countable.

The set of all possible finite length pieces of English text is countable.

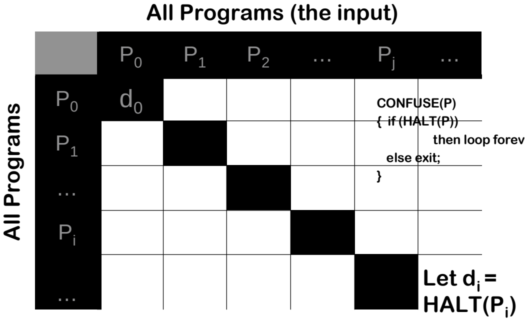


Programs (computable functions) are countable, so we can put them in a (countably long) list

---



YES, if $P_i(P_j)$ halts
No, otherwise



```
CONFUSE(P)
{ if (HALT(P))
            then loop forever;
   else exit;
}
```

Let d_i = HALT(P_i)

CONFUSE(P_i) halts iff $d_i = 0$
(The CONFUSE function is the negation of the diagonal.)

Hence CONFUSE cannot be on this list.

One final interesting digression about infinities …

We know there are at least 2 infinities. (The number of naturals, the number of reals.)

Are there more?

Definition: Power Set

The power set of S is the set of all subsets of S.

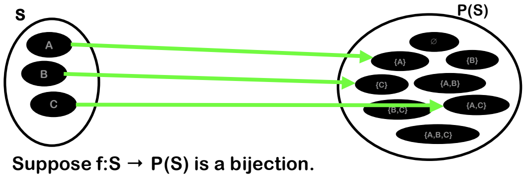
The power set is denoted as P(S)

Proposition:

If S is finite, the power set of S has cardinality $2^{|S|}$

How do sizes of S and P(S) relate if S is infinite?

Theorem: S can't be put into bijection with P(S)



Suppose f:S → P(S) is a bijection.

Let $\text{CONFUSE}_f = \{ x \mid x \in S, x \notin f(x) \}$

Since f is onto, exists $y \in S$ such that $f(y) = \text{CONFUSE}_f$.

Is y in CONFUSE_f?

YES: Definition of CONFUSE_f implies no

NO: Definition of CONFUSE_f implies yes

For any set S (finite or infinite),
the cardinality of P(S)
is strictly greater than
the cardinality of S.

This proves that there are at least a countable number of infinities.

Indeed, take any infinite set S. Then P(S) is also infinite, and its cardinality is a larger infinity than the cardinality of S.

This proves that there are at least a countable number of infinities.

The first infinity is the size of all the countable sets. It is called:

$$\aleph_0$$

$$\aleph_0, \aleph_1, \aleph_2, \dots$$

Cantor wanted to show that there is no set whose size is strictly between $\aleph_0$ and $\aleph_1$

Cantor called his conjecture the "Continuum Hypothesis."

However, he was unable to prove it. This helped fuel his depression.

The Continuum Hypothesis can't be proved or disproved from the standard axioms of set theory!

This has been proved!

Adding CH=T to set theory doesn't create inconsistency. Neither does adding CH=F.

Consistent: can't prove a contradiction

End of digression…

Next: proving undecidability.
The crucial notion of a reduction.

Computability Theory: Vocabulary Lesson

We call a set $S \subseteq \Sigma^*$ decidable or recursive if there is a program P such that:

$P(x)$ = yes, if $x \in S$
$P(x)$ = no, if $x \notin S$

We already know: the halting set K is undecidable

Decidable and Computable

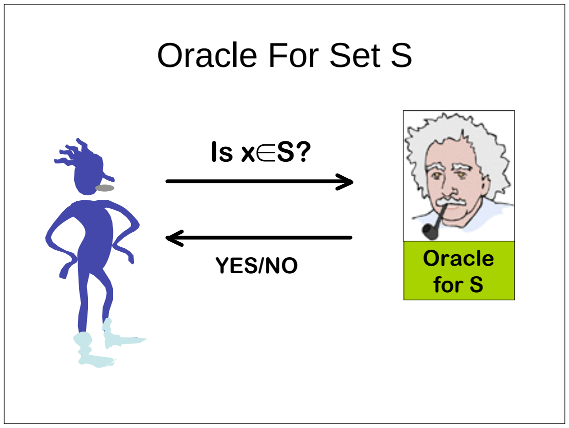
Subset S of Σ^* ⇔ Function f_S

x in S ⇔ $f_S(x) = 1$
x not in S ⇔ $f_S(x) = 0$

Set S is decidable ⇔ function f_S is computable

Sets are "decidable" (or undecidable), whereas functions are "computable" (or not)

Oracles and Reductions

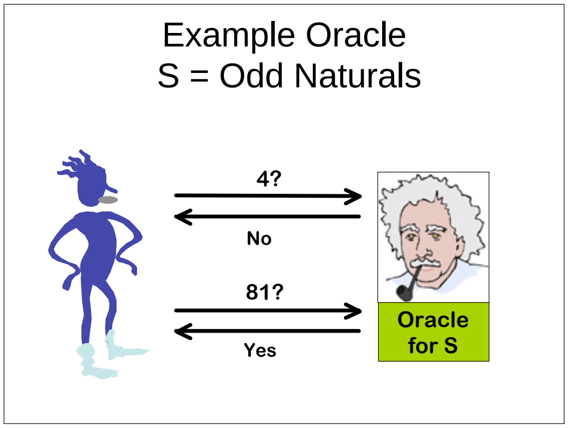


Oracle For Set S

Is x∈S?

YES/NO

Oracle for S



Example Oracle
S = Odd Naturals

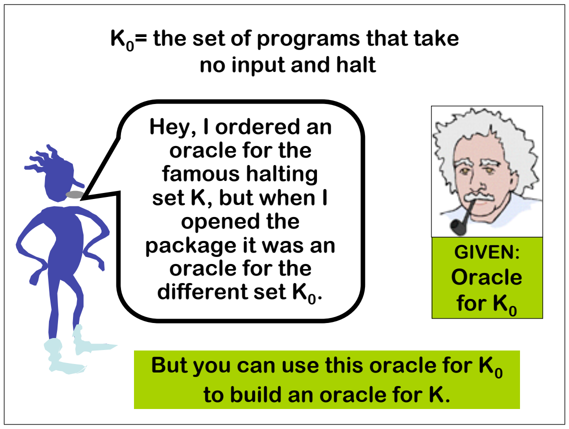
4?

No

81?

Yes

Oracle for S

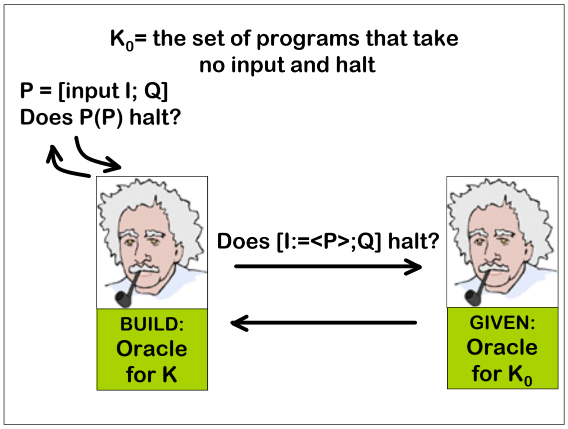


K_0= the set of programs that take no input and halt

Hey, I ordered an oracle for the famous halting set K, but when I opened the package it was an oracle for the different set K_0.

GIVEN: Oracle for K_0

But you can use this oracle for K_0 to build an oracle for K.



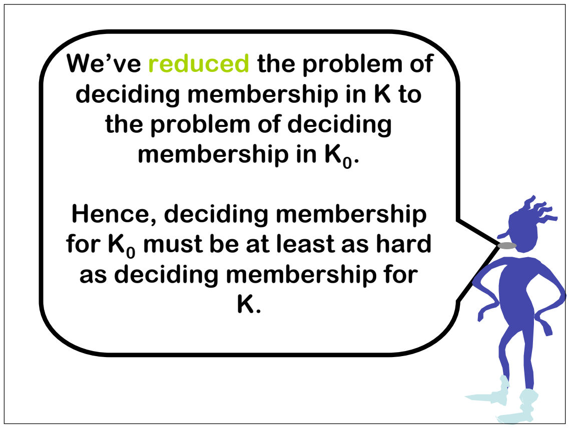
K_0= the set of programs that take no input and halt

P = [input I; Q]
Does P(P) halt?

BUILD: Oracle for K

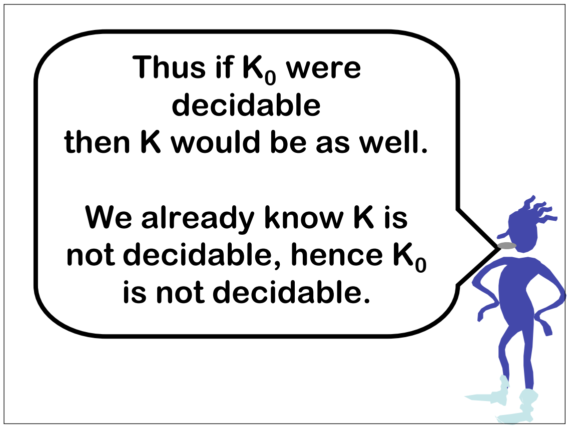
Does [I:=<P>;Q] halt?

GIVEN: Oracle for K_0



We've reduced the problem of deciding membership in K to the problem of deciding membership in K_0.

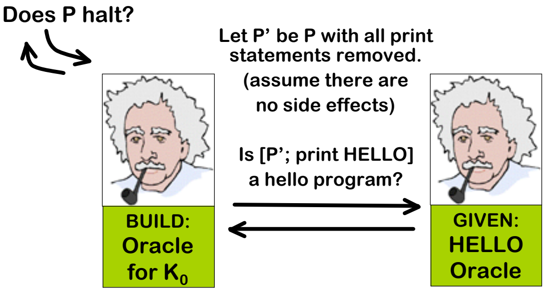
Hence, deciding membership for K_0 must be at least as hard as deciding membership for K.



Thus if K_0 were decidable then K would be as well.

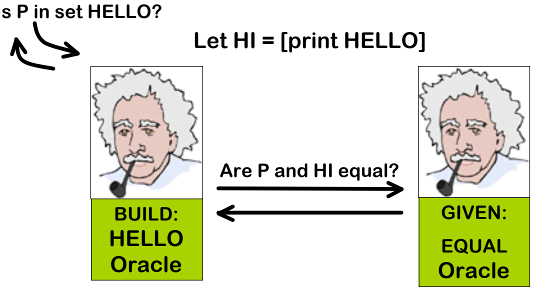
We already know K is not decidable, hence K_0 is not decidable.

HELLO = the set of programs that print hello and halt



Hence, the set HELLO is not decidable.

EQUAL = All <P,Q> such that P and Q have identical output behavior on all inputs



Halting with input, Halting without input, HELLO, and EQUAL are all undecidable.

Diophantine Equations

Does polynomial $4X^2Y + XY^2 + 1 = 0$ have an integer root? I.e., does it have a zero at a point where all variables are integers?

D = {multivariate integer polynomials P | P has a root where all variables are integers}

Famous Theorem: D is undecidable

This is the solution to Hilbert's 10th problem]

Hilbert

Resolution of Hilbert's 10th Problem: Dramatis Personae

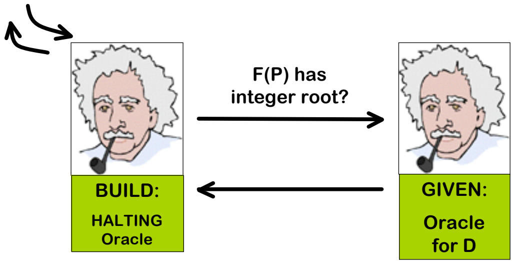
Martin Davis, Julia Robinson, Yuri Matiyasevich (1982)

Polynomials can Encode Programs

There is a computable function
F: Java programs that take no input → Polynomials over the integers

Such that
program P halts ⇔ F(P) has an integer root

D = the set of all integer polynomials with integer roots

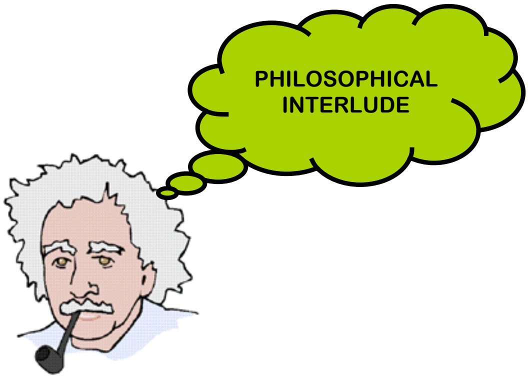


Does program P halt?

F(P) has integer root?

BUILD:
HALTING Oracle

GIVEN:
Oracle for D



PHILOSOPHICAL INTERLUDE

CHURCH-TURING THESIS

Any well-defined procedure that can be grasped and performed by the human mind and pencil/paper, can be performed on a conventional digital computer with no bound on memory.

The Church-Turing Thesis is NOT a theorem. It is a statement of belief concerning the universe we live in.

Your opinion will be influenced by your religious, scientific, and philosophical beliefs…

…mileage may vary

Empirical Intuition

No one has ever given a counter-example to the Church-Turing thesis. I.e., no one has given a concrete example of something humans compute in a consistent and well defined way, but that can't be programmed on a computer. The thesis is true.

Mechanical Intuition

The brain is a machine. The components of the machine obey fixed physical laws. In principle, an entire brain can be simulated step by step on a digital computer. Thus, any thoughts of such a brain can be computed by a simulating computer. The thesis is true.

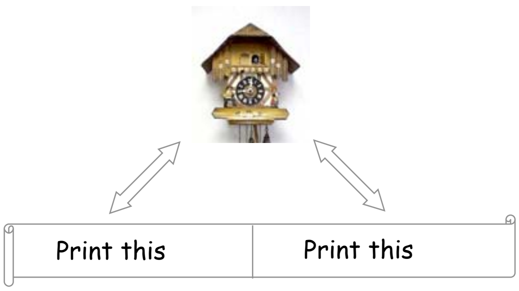
Quantum Intuition

The brain is a machine, but not a classical one. It is inherently quantum mechanical in nature and does not reduce to simple particles in motion. Thus, there are inherent barriers to being simulated on a digital computer. The thesis is false. However, the thesis is true if we allow quantum computers.

Some of the big ideas we've seen so far

- The Turing Machine model and the Church-Turing thesis
- Universality via duality
- Some problems can't be solved on computers
- Diagonalization and the different types of infinity
- Notion of reduction.

Self-reference

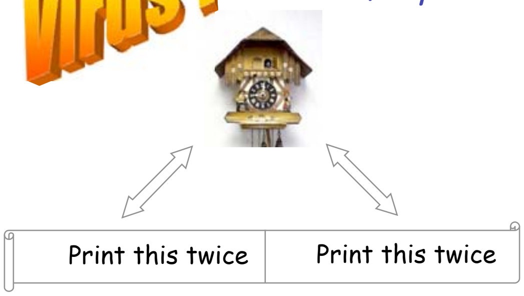


Print this | Print this

Print this

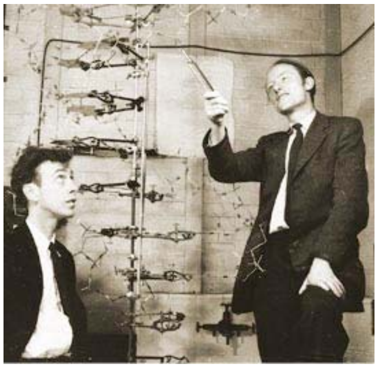
Virus !

Self-replication



Print this twice | Print this twice

Print this twice **Print this twice**



James Watson – Francis Crick, 1953