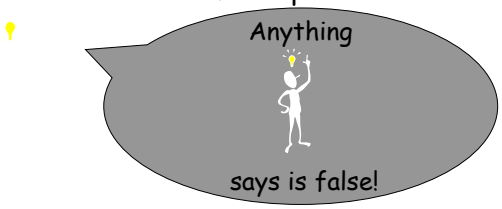
Great Theoretical Ideas In Computer Science

S. Rudich
V. Adamchik

CS 15-251 Spring 2006

Lecture 26 Apr 25, 2006 Carnegie Mellon University

Review: Cantor's Legacy
Turing's Legacy:
The Limits Of Computation.



Review: infinite sets

Cantor's Definition (1874):
Two sets are defined to have the
same size
if and only if they can be
placed into 1-1 onto correspondence.

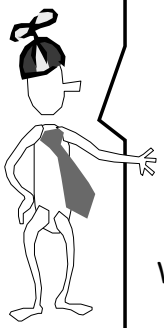
Review: infinite sets

Cantor's Definition (1874):
Two sets are defined to have the
same cardinality
if and only if they can be
placed into 1-1 onto correspondence.

Continuum Hypothesis

$\aleph_1 = 2^{\aleph_0}$

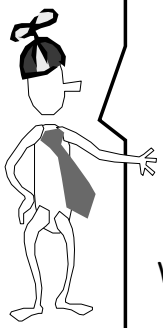
There are no infinite sets between $\mathbb{N}$ and $\mathbb{R}$



Warm-up Problem

Consider all polynomials with integer coefficients.

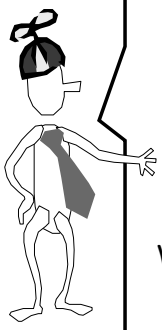
What is the cardinality of this set?



Warm-up Problem

Consider all polynomials with rational coefficients.

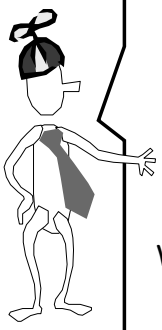
What is the cardinality of this set?



Problem

Consider the algebraic numbers.

What is the cardinality of this set?



Problem

Consider the transcendental numbers.

What is the cardinality of this set?

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

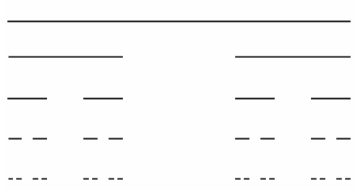
$\mathbb{R}^n$ can be put in 1-1 correspondence with [0,1].

Cantor's set

Tiny sets (measure zero) with uncountably many points

Cantor's set

Cantor Set is formed by repeatedly cutting out middle thirds of a line segment:



Cantor's set

What remains is called the Cantor set

How much did we remove?

What is the size of the Cantor set?

Cantor's set

How much did we remove?

Cantor's set

Thinking of the size as a length, we removed everything.

Therefore, the Cantor set is very tiny.

Cantor's set

On the other hand, the Cantor set is not empty, since we did not remove the end points

0, 1, 1/3, 2/3,…

Cantor's set

We will show that the Cantor set is the big as the whole interval (0,1).

Cantor's set

We remove all the ternary decimals with 1 in the decimal place.

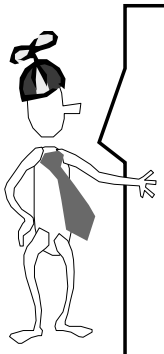
$$\left(\frac{1}{3},\frac{2}{3}\right)\Rightarrow(0.1_3, 0.2_3)$$

$$\left(\frac{1}{9},\frac{2}{9}\right)\Rightarrow(0.01_3, 0.02_3)$$

$$\left(\frac{7}{9},\frac{8}{9}\right)\Rightarrow(0.21_3, 0.22_3)$$

Cantor's set

The Cantor set is a set of numbers whose ternary decimal representations consist entirely of 0's and 2's.



Problem

Does 1/12 belong to the Cantor set?

Cantor's set

Can you find a 1-1 map between {0,1} and the Cantor set?

Cantor's set

The one-to-one map between {0,1} and the Cantor set is called the "Devil's Staircase" .

To see this bijection, take a number from the Cantor set in ternary notation, divide its digits by 2, and you get all coefficients in binary notation.

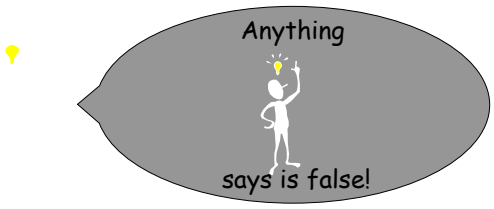
A little bit further

Mathematically speaking the base is not necessarily an integer

Cantor's dust $\leftrightarrow \sum_{k=1}^{\infty} \frac{c_k}{3^k}$

Generalization $\leftrightarrow \sum_{k=1}^{\infty} c_k b^k$

Turing's Legacy:



Turing's Legacy:

A **problem** is a yes/no question.

An algorithm is a **solution** to a problem if it correctly provides the appropriate yes/no answer to the problem.

A problem is **decidable** if it has a solution.

Turing's Legacy:

Are all problems decidable?

Decidable and Computable

Subset S of Σ^*	$\Leftrightarrow$	Function f_S
x in S	$\Leftrightarrow$	$f_S(x) = 1$
x not in S	$\Leftrightarrow$	$f_S(x) = 0$

Set S is decidable $\Leftrightarrow$ function f_S is computable

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

The HELLO assignment

Write a JAVA program to output the word "HELLO" on the screen and halt.

Space and time are not an issue.
The program is for an ideal computer.

PASS for any working HELLO .
No partial credit.

Grading Script

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

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

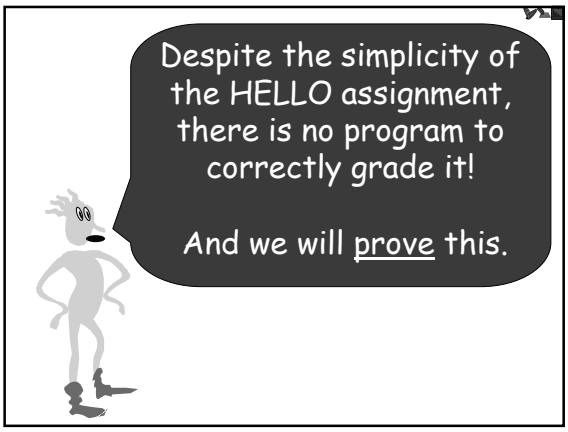
How exactly might such a script work?

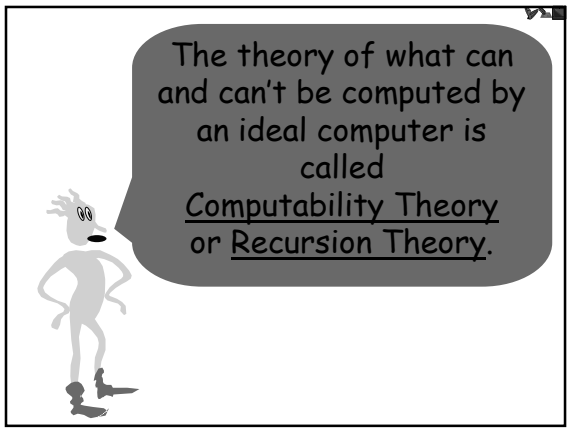
What kind of program could a student who hated his/her TA hand in?

Nasty Program

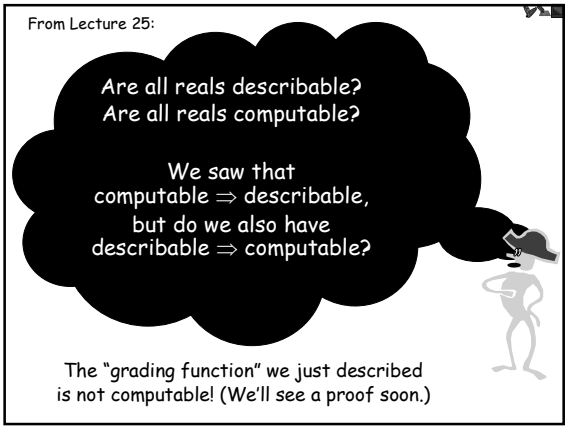
```
n:=0;
while (n is not a counter-example
    to the Riemann Hypothesis) {
    n++;
}
print "Hello";
```

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





From Lecture 25:



The "grading function" we just described is not computable! (We'll see a proof soon.)

Computable Function

Fix any finite set of symbols, Σ.
Fix any precise programming language, e.g., Java.

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

A program is any finite string of characters that is syntactically valid.

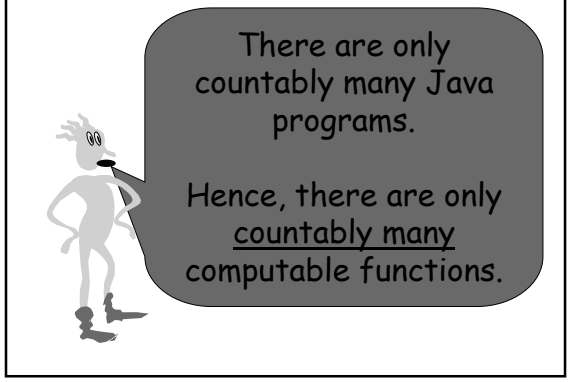
A function $f : \Sigma^* \to \Sigma^*$ is computable if there is a program P that when executed on an ideal computer, computes f.

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

Proof:

Sort S by first by length and then alphabetically.

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



Uncountably many functions

The functions $f: \Sigma^* \to \{0,1\}$ are in 1-1 onto correspondence with the subsets of Σ^* (the powerset of Σ^*).

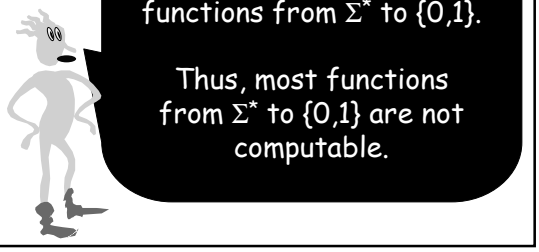
Subset S of Σ^*	$\Leftrightarrow$	Function f_S
x in S	$\Leftrightarrow$	$f_S(x) = 1$
x not in S	$\Leftrightarrow$	$f_S(x) = 0$

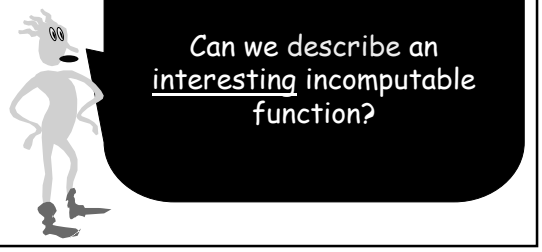
Uncountably many functions

The functions $f: \Sigma^* \to \{0,1\}$ are in 1-1 onto correspondence with the subsets of Σ^* (the powerset of Σ^*).

Hence, the set of all $f: \Sigma^* \to \{0,1\}$ has the same size as the power set of Σ^*.

And since Σ^* is countably infinite, its power set is uncountably infinite.





Notation And Conventions

Fix a single programming language (Java)

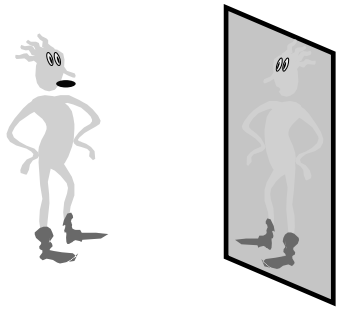
When we write program P we are talking about the text of the source code for P

P(x) means the output that arises from running program P on input x, assuming that P eventually halts.

The meaning of P(P)

It follows from our conventions that P(P) means the output obtained when we run P on the text of its own source code.

P(P) … So that's what I look like



The Halting Set K

Definition:
K is the set of all programs P such that P(P) halts.

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

The Halting Problem
K = {P | P(P) halts }

Is there a program HALT such that:

HALT(P) = yes, if P∈K
HALT(P) = no, if P∉K

HALT decides whether or not any given program is in K.

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

Suppose a program HALT existed that solved the halting problem.

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

CONFUSE

```
boolean CONFUSE(P)
{
  if (HALT(P) == True)
     then loop forever;
  else return True;
}
```

Does CONFUSE(CONFUSE) halt?

Does CONFUSE(CONFUSE) halt?

```
boolean CONFUSE(P)
{
  if (HALT(P) == True)
     then loop forever;
  else return True;
}
```

Consider both cases

1. CONFUSE(CONFUSE) halts then (by def.) HALT(CONFUSE) is True.

But then CONFUSE will loop forever.

Does CONFUSE(CONFUSE) halt?

```
boolean CONFUSE(P)
{
  if (HALT(P) == True)
     then loop forever;
  else return True;
}
```

Consider both cases

2. CONFUSE(CONFUSE) runs forever then (by def.) HALT(CONFUSE) is False.

But then CONFUSE halts.

Alan Turing (1912-1954)

Theorem: [1937]

There is no program to solve the halting problem

Turing's argument is essentially the reincarnation of Cantor's Diagonalization argument that we saw in the previous lecture.

All Programs (the input)

All Programs	P_0	P_1	P_2	…	P_j	…
P_0						
P_1						
…						
P_i						
…						

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

All Programs (the input)

All Programs	P_0	P_1	P_2	…	P_j	…
P_0						
P_1						
…						
P_i						
…						

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

All Programs (the input)

All Programs	P_0	P_1	P_2	…	P_j	…
P_0	d_0					
P_1		d_1				
…			…			
P_i				d_i		
…					…	

Let $d_i = HALT(P_i)$

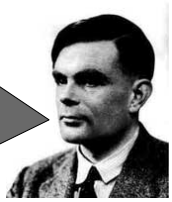
CONFUSE(P_i) halts iff d_i = no
(The CONFUSE function is the negation of the diagonal.)

Hence CONFUSE cannot be on this list.

From last lecture:

Is there a real number that can be described, but not computed?

Consider the real number R_K whose binary expansion has a 1 in the j^{th} position iff $P_j \in K$ (i.e., if the j^{th} program halts).



Proof that R_K cannot be computed

Suppose it is, and program FRED computes it. then consider the following program:

```
MYSTERY(program text P)
  for j = 0 to forever do {
    if (P == Pj)
       then use FRED to compute jth bit of RK
    return YES if (bit == 1), NO if (bit == 0)
  }
```

MYSTERY solves the halting problem!

Self-Reference Puzzle

Write a program that prints itself out as output. No calls to the operating system, or to memory external to the program.

HW: Auto Cannibal Maker

Write a program AutoCannibalMaker that takes the text of a program EAT as input and outputs a program called $SELF_{EAT}$.

When $SELF_{EAT}$ is executed, it should output $EAT(SELF_{EAT})$

Suppose HALT with no input was programmable in JAVA.

Write a program AutoCannibalMaker that takes the text of a program EAT as input and outputs a program called $SELF_{EAT}$.

When $SELF_{EAT}$ is executed it should output $EAT(SELF_{EAT})$

Let EAT(P) = halt, if P does not halt
loop forever, otherwise.

What does $SELF_{EAT}$ do?

Contradiction! Hence EAT does not have a corresponding JAVA program.