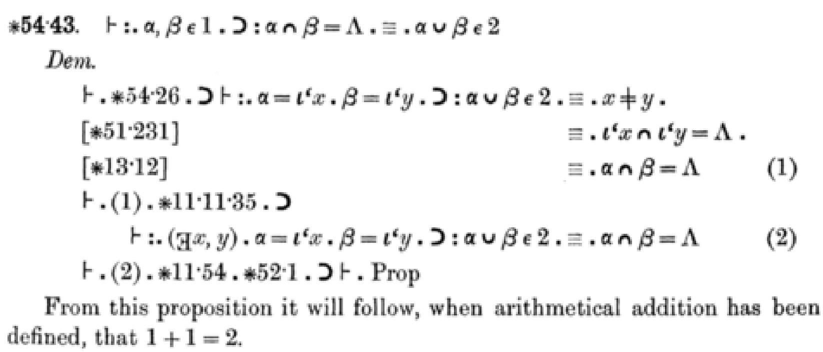
CSE 311: Foundations of Computing

Lecture 27: Uncountable Sets, Uncomputable Functions

Review Session Sunday

Midterm regrades are done
You can pick up your midterm now or in my office hour.
Final exam information on website
Catalyst survey you must fill out will open tomorrow.



$*54 \cdot 43.\quad \vdash :.\, \alpha, \beta \in 1 \,.\supset :\, \alpha \cap \beta = \Lambda \,.\equiv.\, \alpha \cup \beta \in 2$

Dem.

$\vdash .\, *54 \cdot 26 \,.\supset \vdash :.\, \alpha = \iota\text{‘}x \,.\, \beta = \iota\text{‘}y \,.\supset :\, \alpha \cup \beta \in 2 \,.\equiv.\, x \neq y \,.$

$[*51 \cdot 231] \qquad \equiv .\, \iota\text{‘}x \cap \iota\text{‘}y = \Lambda \,.$

$[*13 \cdot 12] \qquad \equiv .\, \alpha \cap \beta = \Lambda \qquad (1)$

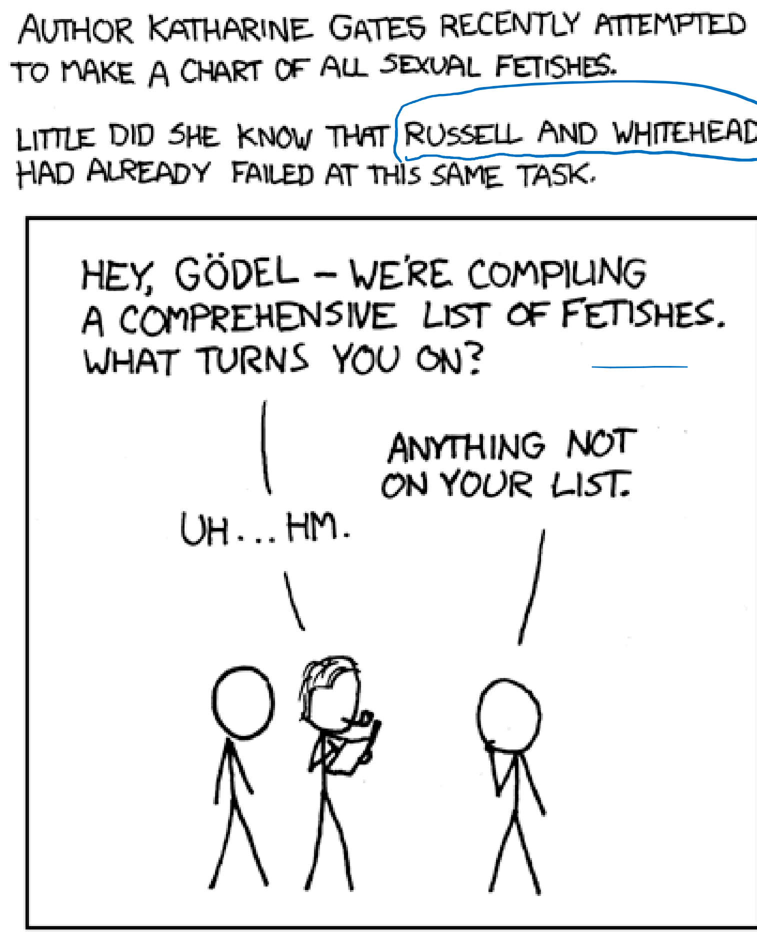
$\vdash .\, (1) \,.\, *11 \cdot 11 \cdot 35 \,.\supset$

$\vdash :.\, (\exists x, y) \,.\, \alpha = \iota\text{‘}x \,.\, \beta = \iota\text{‘}y \,.\supset :\, \alpha \cup \beta \in 2 \,.\equiv.\, \alpha \cap \beta = \Lambda \qquad (2)$

$\vdash .\, (2) \,.\, *11 \cdot 54 \,.\, *52 \cdot 1 \,.\supset \vdash .\,$ Prop

From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

[proved on page 86 of Volume II of Russell and Whitehead's "Principia Mathematica":
"The above proposition is occasionally useful."]



Last time: Countable sets

A set $\boldsymbol{S}$ is **countable** iff we can order the elements of $\boldsymbol{S}$ as

$$\boldsymbol{S} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \dots\}$$

Countable sets:

$\mathbb{N}$ - the natural numbers
$\mathbb{Z}$ - the integers
$\mathbb{Q}$ - the rationals
Σ^*- the strings over any finite Σ
The set of all Java programs

} Shown by "dovetailing"

Not every set is countable

Theorem [Cantor]:
The set of real numbers between 0 and 1 is not countable.

Proof will be by contradiction. Using a new method called diagonalization.

Real numbers between 0 and 1: $[0,1)$

Every number between 0 and 1 has an infinite decimal expansion:

1/2	=	0.50000000000000000000000...
1/3	=	0.33333333333333333333333...
1/7	=	0.14285714285714285714285...
π-3	=	0.14159265358979323846264...
1/5	=	0.19999999999999999999999...
	=	0.20000000000000000000000...

$10x = 1.99999\ldots$
$x = 0.1999\ldots$
$9x = 1.8$
$\therefore x = 0.2$

Representation is unique except for the cases that the decimal expansion ends in all 0's or all 9's.
We will never use the all 9's representation.

Proof that $[0,1)$ is not countable

Suppose, for the sake of contradiction, that there is a list of them:

		1	2	3	4	5	6	7	8	9	...
r_1	0.	5	0	0	0	0	0	0	0	...	...
r_2	0.	3	3	3	3	3	3	3	3	...	...
r_3	0.	1	4	2	8	5	7	1	4	...	...
r_4	0.	1	4	1	5	9	2	6	5	...	...
r_5	0.	1	2	1	2	2	1	2	2	...	...
r_6	0.	2	5	0	0	0	0	0	0	...	...
r_7	0.	7	1	8	2	8	1	8	2	...	...
r_8	0.	6	1	8	0	3	3	9	4	...	...
...	....	...	....	....	...	...	...	...	...	...	

Proof that $[0,1)$ is not countable

Suppose, for the sake of contradiction, that there is a list of them:

		1	2	3	4	5	6	7	8	9	...
r_1	0.	5	0	0	0	0	0	0	0	...	...
r_2	0.	3	3	3	3	3	3	3	3	...	...
r_3	0.	1	4	2	8	5	7	1	4	...	...
r_4	0.	1	4	1	5	9	2	6	5	...	...
r_5	0.	1	2	1	2	2	1	2	2	...	...
r_6	0.	2	5	0	0	0	0	0	0	...	...
r_7	0.	7	1	8	2	8	1	8	2	...	...
r_8	0.	6	1	8	0	3	3	9	4	...	...
...	....	...	....	....	...	...	...	...	...	...	

Proof that $[0,1)$ is not countable

Suppose, for the sake of contradiction, that there is a list of them:

Flipping rule:

Only if the other driver deserves it.

		1	2	3	4						
r_1	0.	5	0	0	0						
r_2	0.	3	3	3	3						
r_3	0.	1	4	2	8	5	7	1	4	...	...
r_4	0.	1	4	1	5	9	2	6	5	...	...
r_5	0.	1	2	1	2	2	1	2	2	...	...
r_6	0.	2	5	0	0	0	0	0	0	...	...
r_7	0.	7	1	8	2	8	1	8	2	...	...
r_8	0.	6	1	8	0	3	3	9	4	...	...
...	....	...	....	....	...	...	...	...	...	...	

Proof that $[0,1)$ is not countable

Suppose, for the sake of contradiction, that there is a list of them:

		1	2	3	4						
r_1	0.	5 [1]	0	0	0						
r_2	0.	3	3 [5]	3	3						
r_3	0.	1	4	2 [5]	8	5	7	1	4	...	...
r_4	0.	1	4	1	5 [1]	9	2	6	5	...	...
r_5	0.	1	2	1	2	2 [5]	1	2	2	...	...
r_6	0.	2	5	0	0	0	0 [5]	0	0	...	...
r_7	0.	7	1	8	2	8	1	8 [5]	2	...	...
r_8	0.	6	1	8	0	3	3	9	4 [5]	...	...
...	....	...	....	....	...	...	...	...	...	...	

Flipping rule:

If digit is 5, make it 1.

If digit is not 5, make it 5.

Proof that $[0,1)$ is not countable

Suppose, for the sake of contradiction, that there is a list of them:

		1	2	3	4						
r_1	0.	5 [1]	0	0	0						
r_2	0.	3	3 [5]	3	3						
r_3	0.	1	4	2 [5]	8	5	7	1	4	...	...
r_4	0.	1	4	1	5 [1]	9	2	6	5	...	...
r_5	0.	1	2	1	2	2 [5]	1	2	2	...	...
r_6	0.	2	5	0	0	0	0 [5]	0	0	...	...
r_7	0.	7	1	8	2	8	1	8 [5]	2	...	...

Flipping rule:

If digit is 5, make it 1.

If digit is not 5, make it 5.

If diagonal element is $\mathbf{0}.x_{11}x_{22}x_{33}x_{44}x_{55}\cdots$ then the flipped diagonal number call it $\boldsymbol{d} = \mathbf{0}.\hat{x}_{11}\hat{x}_{22}\hat{x}_{33}\hat{x}_{44}\hat{x}_{55}\cdots$ is also a real number in [0,1).

0.1551555 - -

Proof that $[0,1)$ is not countable

Suppose, for the sake of contradiction, that there is a list of them:

		1	2	3	4						
r_1	0.	5^{1}	0	0	0						
r_2	0.	3	3^{5}	3	3						
r_3	0.	1	4	2^{5}	8	5	7	1	4	...	...
r_4	0.	1	4	1	5^{1}	9	2	6	5	...	...
						2^{5}	1	2	2	...	...
						0	0^{5}	0	0	...	...
						8	1	8^{5}	2	...	...

Flipping rule:

If digit is 5, make it 1.

If digit is not 5, make it 5.

For every $n \geq 1$:

$r_n \neq d = 0.\hat{x}_{11}\hat{x}_{22}\hat{x}_{33}\hat{x}_{44}\hat{x}_{55}\cdots$

because the numbers differ on the n-th digit!

If diagonal element is $0.x_{11}x_{22}x_{33}x_{44}x_{55}\cdots$ then the flipped diagonal number call it $d = 0.\hat{x}_{11}\hat{x}_{22}\hat{x}_{33}\hat{x}_{44}\hat{x}_{55}\cdots$ is also a real number in [0,1).

Proof that $[0,1)$ is not countable

Suppose, for the sake of contradiction, that there is a list of them:

		1	2	3	4						
r_1	0.	5^{1}	0	0	0						
r_2	0.	3	3^{5}	3	3						
r_3	0.	1	4	2^{5}	8	5	7	1	4	...	...
r_4	0.	1	4	1	5^{1}	9	2	6	5	...	...
						2^{5}	1	2	2	...	...
						0	0^{5}	0	0	...	...
						8	1	8^{5}	2	...	...

Flipping rule:

If digit is 5, make it 1.

If digit is not 5, make it 5.

For every $n \geq 1$:

$r_n \neq d = 0.\hat{x}_{11}\hat{x}_{22}\hat{x}_{33}\hat{x}_{44}\hat{x}_{55}\cdots$

because the numbers differ on the n-th digit!

So the list is incomplete, which is a contradiction.

Thus the real numbers between 0 and 1 are **not countable**: “uncountable”

∎

A note on this proof

- The set of rational numbers in [0,1) also have ~~binary~~ decimal representations like this
  - The only difference is that rational numbers always have repeating decimals in their expansions 0.33333… or .25000000…
- So why wouldn't the same proof show that this set of rational numbers is uncountable?
  - Given any listing (even one that is good like the dovetailing listing) we could create the flipped diagonal number d as before
  - However, d would not have a repeating decimal expansion and so wouldn't be a rational #

    It would not be a "missing" number, so no contradiction.

The set of all functions $f : \mathbb{N} \to \{0, \ldots, 9\}$ is uncountable

	0	1	2	3	4	5	6	7	8	9	10	11	12	...
f	5	3	2	1	8	0	9	0	1	0	7	3	4	

The set of all functions $f : \mathbb{N} \rightarrow \{0, \ldots, 9\}$ is uncountable

Supposed listing of all the functions:

	1	2	3	4	5	6	7	8	9	...
f_1	5	0	0	0	0	0	0	0	...	...
f_2	3	3	3	3	3	3	3	3	...	...
f_3	1	4	2	8	5	7	1	4	...	...
f_4	1	4	1	5	9	2	6	5	...	...
f_5	1	2	1	2	2	1	2	2	...	...
f_6	2	5	0	0	0	0	0	0	...	...
f_7	7	1	8	2	8	1	8	2	...	...
f_8	6	1	8	0	3	3	9	4	...	...
...	...	....	....	...	...	...	...	...	...	

The set of all functions $f : \mathbb{N} \to \{0, \dots, 9\}$ is uncountable

Supposed listing of all the functions:

	1	2	3	4						
f_1	5^{1}	0	0	0						
f_2	3	3^{5}	3	3						
f_3	1	4	2^{5}	8	5	7	1	4	...	...
f_4	1	4	1	5^{1}	9	2	6	5	...	...
f_5	1	2	1	2	2^{5}	1	2	2	...	...
f_6	2	5	0	0	0	0^{5}	0	0	...	...
f_7	7	1	8	2	8	1	8^{5}	2	...	...
f_8	6	1	8	0	3	3	9	4^{5}	...	...
...	...	....	....	...	...	...	...	...	...	

Flipping rule:

If $f_n(n) = 5$, set $D(n) = 1$

If $f_n(n) \neq 5$, set $D(n) = 5$

The set of all functions $f : \mathbb{N} \to \{0, \ldots, 9\}$ is uncountable

Supposed listing of all the functions:

	1	2	3	4						
f_1	5^{1}	0	0	0						
f_2	3	3^{5}	3	3						
f_3	1	4	2^{5}	8	5	7	1	4	...	...
f_4	1	4	1	5^{1}	9	2	6	5	...	...
f_5	1	2	1	2	2^{5}	1	2	2	...	...
f_6	2	5	0	0	0	0^{5}	0	0	...	...
f_7	7	1	8	2	8	1	8^{5}	2	...	...

Flipping rule:

If $f_n(n) = 5$, set $D(n) = 1$

If $f_n(n) \neq 5$, set $D(n) = 5$

For all n, we have $D(n) \neq f_n(n)$. Therefore $D \neq f_n$ for any n and the list is incomplete! $\Rightarrow$ $\{f \mid f: \mathbb{N} \to \{0,1, \ldots, 9\}\}$ is **not** countable

Uncomputable functions

We have seen that:

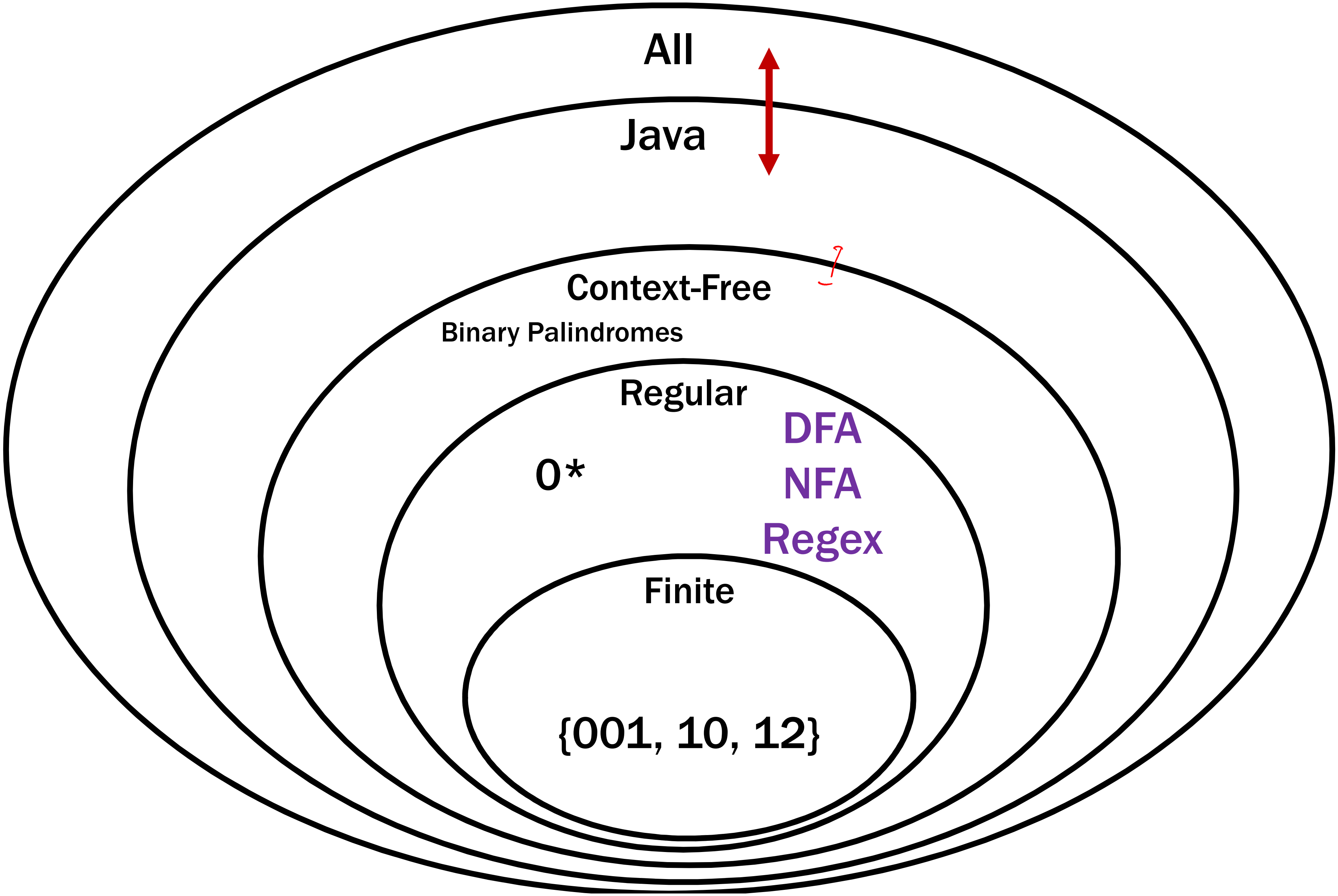
- The set of all (Java) programs is countable
- The set of all functions $f : \mathbb{N} \to \{0, \ldots, 9\}$ is not countable

So: There must be some function $f : \mathbb{N} \to \{0, \ldots, 9\}$ that is not computable by any program!

Interesting… maybe.

Can we come up with an explicit function that is uncomputable?

Recall our language picture



Some Notation

We’re going to be talking about *Java code*.

CODE(**P**) will mean “the code of the program **P**”

So, consider the following function:

```
public String P(String x) {
    return new String(Arrays.sort(x.toCharArray()));
}
```

What is **P**(CODE(**P**))?

“((()))..;AACPSSaaabceeggghiiiilnnnnnooprrrrrrrrrrsssttttttuuwxxyy{}”

The Halting Problem

Given: - CODE(**P**) for any program **P**

- input **x**

Output: **true** if **P** halts on input **x**

false if **P** does not halt on input **x**

It turns out that it isn't possible to write a program that solves the Halting Problem!

Proof by contradiction

- Suppose that **H** is a Java program that solves the Halting problem. Then we can write this program:

```
public static void D(x) {
    if (H(x,x) == true) {
        while (true);   /* don't halt */
    }
    else {
        return;          /*    halt    */
    }
}
```

- Does **D**(CODE(**D**)) halt?

Does **D**(CODE(**D**)) halt?

```
public static void D(x) {
    if (H(x,x) == true) {
        while (true); /* don't halt */
    }
    else {
        return;       /*   halt   */
    }
}
```

H solves the halting problem implies that
H(CODE(**D**),x) is **true** iff **D**(x) halts, **H**(CODE(**D**),x) is **false** iff not

D(x) doesn't halt!

Does D(CODE(D)) halt?

```
public static void D(x) {
    if (H(x,x) == true) {
        while (true); /* don't halt */
    }
    else {
        return;        /*    halt    */
    }
}
```

H solves the halting problem implies that

H(CODE(**D**),x) is **true** iff **D**(x) halts, **H**(CODE(**D**),x) is **false** iff not

```
public static void D(x) {
    if (H(x,x) == true) {
        while (true); /* don't halt */
    }
    else {
        return;          /*   halt   */
    }
}
```

Does D(CODE(D)) halt?

H solves the halting problem implies that

H(CODE(**D**),x) is **true** iff **D**(x) halts, **H**(CODE(**D**),x) is **false** iff not

Suppose that **D**(CODE(**D**)) **halts**.

Then, by definition of **H** it must be that

H(CODE(**D**), CODE(**D**)) is **true**

Which by the definition of **D** means **D**(CODE(**D**)) **doesn't halt**

Suppose that **D**(CODE(**D**)) **doesn't halt**.

Then, by definition of **H** it must be that

H(CODE(**D**), CODE(**D**)) is **false**

Which by the definition of **D** means **D**(CODE(**D**)) **halts**

Does D(CODE(D)) halt?

```
public static void D(x) {
    if (H(x,x) == true) {
        while (true); /* don't halt */
    }
    else {
        return;        /*   halt   */
    }
}
```

H solves the halting problem implies that

H(CODE(**D**),x) is **true** iff **D**(x) halts, **H**(CODE(**D**),x) is **false** iff not

Suppose that **D**(CODE(**D**)) **halts**.

Then, by definition of **H** it must be that

H(CODE(**D**), CODE(**D**)) is **true**

Which by the definition of **D** means **D**(CODE(**D**)) **doesn't halt**

Suppose that **D**(CODE(**D**)) **doesn't halt**.

Then, by definition of **H** it must be that

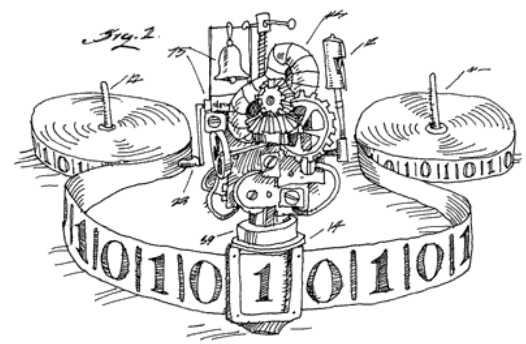
H(CODE(**D**), CODE(**D**)) is **false**

Which by the definition of **D** means **D**(CODE(**D**)) **halts**

Contradiction!

Done

- We proved that there is no computer program that can solve the Halting Problem.
  - There was nothing special about Java*
    [Church-Turing thesis]



- This tells us that there is no compiler that can check our programs and guarantee to find any infinite loops they might have.

Connection to diagonalization

Write <**P**> for **CODE(P)**

Some possible inputs **x**

All programs **P**

	<P_1>	<P_2>	<P_3>	<P_4>	<P_5>	<P_6>	....				
P_1	0	1	1	0	1	1	1	0	0	0	1 ...
P_2	1	1	0	1	0	1	1	0	1	1	1 ...
P_3	1	0	1	0	0	0	0	0	0	0	1 ...
P_4	0	1	1	0	1	0	1	1	0	1	0 ...
P_5	0	1	1	1	1	1	1	0	0	0	1 ...
P_6	1	1	0	0	0	1	1	0	1	1	1 ...
P_7	1	0	1	1	0	0	0	0	0	0	1 ...
P_8	0	1	1	1	1	0	1	1	0	1	0 ...
P_9	. . . .										
.	. . . .										
.											

(P,x) entry is **1** if program **P** halts on input **x** and **0** if it runs forever

Connection to diagonalization

Write <P> for CODE(P)

Some possible inputs **x**

All programs P

	$\langle P_1 \rangle$	$\langle P_2 \rangle$	$\langle P_3 \rangle$	$\langle P_4 \rangle$	$\langle P_5 \rangle$	$\langle P_6 \rangle$	....					
P_1	0^{1}	1	1	0	1							
P_2	1	1^{0}	0	1	0							
P_3	1	0	1^{0}	0	0							
P_4	0	1	1	0^{1}	1	0	1	1	0	1	0	...
P_5	0	1	1	1	1^{0}	1	1	0	0	0	1	...
P_6	1	1	0	0	0	1^{0}	1	0			1	...
P_7	1	0	1	1	0	0	0^{1}	0	0	0	1	...
P_8	0	1	1	1	1	0	1	1^{0}	0	1	0	...
P_9	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
.												

Behavior of program ***D*** would be like the flipped diagonal, so it can't be in the list of all programs.

Contradiction!

(P,x) entry is **1** if program **P** halts on input **x** and **0** if it runs forever