# The Halting Problem

Joseph Paul Cohen

## What is the halting problem?

Given some program and some input, determine if the program will halt on the given input.

$HALT(\#_p, x)$ = Does the program $\#_p$ halt on input $x$

$$HALT(\#_p, x) = \begin{cases} TRUE & \text{When } \#_p \text{ is run on } x \text{ it ends} \\ FALSE & \text{When } \#_p \text{ is run on } x \text{ it does not end} \end{cases}$$

## What is a program?

Some set of instructions that operate on some input

```
1 f(x) :
2 while x < 3 do
3     print x
4     x = x + 1
```

```
1 f(x) :
2 while x < 3 do
3     print x
4     x = x + 1
```

$$f(0) \longrightarrow 0, 1, 2$$

$$f(-1) \longrightarrow -1, 0, 1, 2$$

$$f(5) \longrightarrow \text{NOTHING PRINTED}$$

## What is a program?

Every program can be represented as some number

$f(x)$ can be represented as some number $\#_f$, such as 239847238

$$f(x) \longrightarrow \#_f$$

```
1 f(x) :
2 while x < 3 do
3     print x
4     x = x + 1
```

What does this return?

$$HALT(\#_f, 2)$$

```
1 f(x) :
2 while x < 3 do
3     print x
4     x = x + 1
```

What does this return?

$HALT(\#_f, 2)$

TRUE

```
1 t(x) :
2 while x > 3 do
3     print x
4     x = x + 1
```

What does this return?

$HALT(\#_t, 2)$

```
1 t(x) :
2 while x > 3 do
3     print x
4     x = x + 1
```

What does this return?

$HALT(\#_t, 2)$

TRUE

```
1 t(x) :
2 while x > 3 do
3     print x
4     x = x + 1
```

What does this return?

$HALT(\#_t, 5)$

```
1 t(x) :
2 while x > 3 do
3     print x
4     x = x + 1
```

What does this return?

$HALT(\#_t, 5)$

FALSE

There can never be a program for the the function *HALT*

There can never be a program for the the function *HALT*

This is a big deal

## Outline for Proof

We assume that a program for *HALT* exists. (There is a $\#_{HALT}$)
We write a program with $\#_{HALT}$
We use that program and come to a contradiction

$$g(x) = \begin{cases} FALSE & \text{If } x \text{ run on } x \text{ never halts} \\ \uparrow & \text{Never returns if } x \text{ run on } x \text{ halts} \end{cases}$$

```
1 g(x) :
2 if HALT(x,x) == TRUE then
3     while TRUE do
4         print "Never Returns"
5 else
6     return FALSE
```

$$g(x) = \begin{cases} FALSE & \text{If } x \text{ run on } x \text{ never halts} \\ \uparrow & \text{Never returns if } x \text{ run on } x \text{ halts} \end{cases}$$

```
1 g(x) :
2 if HALT(x,x) == TRUE then
3     while TRUE do
4         print "Never Returns"
5 else
6     return FALSE
```

$$HALT(\#_p, x) = \begin{cases} TRUE & \text{If when } \#_p \text{ is run on } x \text{ it ends} \\ FALSE & \text{If when } \#_p \text{ is run on } x \text{ it does not end} \end{cases}$$

$g$ is a program so it has a number $\#_g$

What happens when we run $\#_g$ on itself?

$$g(\#_g)$$

```
1 g(x) :
2 if HALT(x,x) == TRUE then
3 |   while TRUE do
4 |   |   print "Never Returns"
5 else
6 |   return FALSE
```

Lets look at what happens when: $g$ halts when run on $g$

$$\Longrightarrow HALT(\#_g, \#_g) == FALSE$$

$$\Longrightarrow g \text{ does not halt on } g$$

```
1 g(x) :
2 if HALT(x,x) == TRUE then
3 |   while TRUE do
4 |   |   print "Never Returns"
5 else
6 |   return FALSE
```

How about: $g$ does not halt when run on $g$

$$\Longrightarrow HALT(\#_g, \#_g) == TRUE$$

$$\Longrightarrow g \text{ halts when run on } g$$

We have a contradiction so there can be no program for *HALT*