

## UNIT 14C

### The Limits of Computing: Non-computable Functions

15110 Principles of Computing, Carnegie  
Mellon University - CORTINA

1

## Problem Classifications

- **Tractable Problems**
  - Problems that have reasonable, polynomial-time solutions
- **Intractable Problems**
  - Problems that may have no reasonable, polynomial-time solutions
- **Noncomputable Problems**
  - Problems that have no algorithms at all to solve them

15110 Principles of Computing, Carnegie  
Mellon University - CORTINA

2

## The Barber Paradox

Suppose there is a town with one male barber; and that every man in the town keeps himself clean-shaven: some shave themselves and some are shaved by the barber. Only the barber can shave another man. The barber shaves all and only those men who do not shave themselves.



Does the barber shave himself?

15110 Principles of Computing, Carnegie Mellon University - CORTINA

3

## Program Termination

- Can we determine if a program will terminate given a valid input?

- Example:

```
def mystery1(x)
  while (x != 1) do
    x = x - 2
  end
end
```

- Does this algorithm terminate when  $x = 15$ ?
- Does this algorithm terminate when  $x = 110$ ?

15110 Principles of Computing, Carnegie Mellon University - CORTINA

4

## Another Example

```
def mystery2(x)
  while (x != 1) do
    if x % 2 == 0 then
      x = x / 2
    else
      x = 3 * x + 1
    end
  end
end
```

- Does this algorithm terminate when  $x = 15$ ?
- Does this algorithm terminate when  $x = 110$ ?
- Does this algorithm terminate for any positive  $x$ ?

15110 Principles of Computing, Carnegie  
Mellon University - CORTINA

5

## The Halting Problem

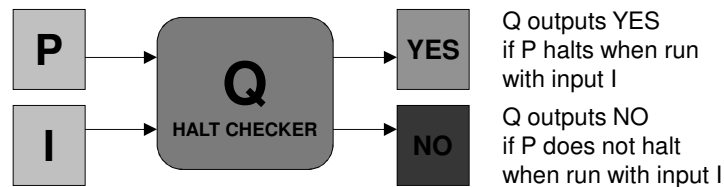
- Does a universal program  $Q$  exist that can take any program  $P$  and any input  $I$  for program  $P$  and determine if  $P$  terminates/halts when run with input  $I$ ?
- Alan Turing showed that such a universal program  $Q$  cannot exist.
  - This is known as the Halting Problem.

15110 Principles of Computing, Carnegie  
Mellon University - CORTINA

6

## Proof by Contradiction

- Assume a program Q exists that requires a program P and an input I.
  - Q determines if program P will halt when P is executed using input I.



- We will show that Q cannot exist by showing that if it did exist we would get a logical contradiction.

15110 Principles of Computing, Carnegie Mellon University - CORTINA

7

## Compilers

- A compiler is a program that takes as its input a program that needs to be translated from a high-level language (e.g. Ruby) to a low-level language (e.g. machine language).
  - In general, a program can process any data, so it can have a program as its input to process.
- Can a compiler compile itself?

15110 Principles of Computing, Carnegie Mellon University - CORTINA

8

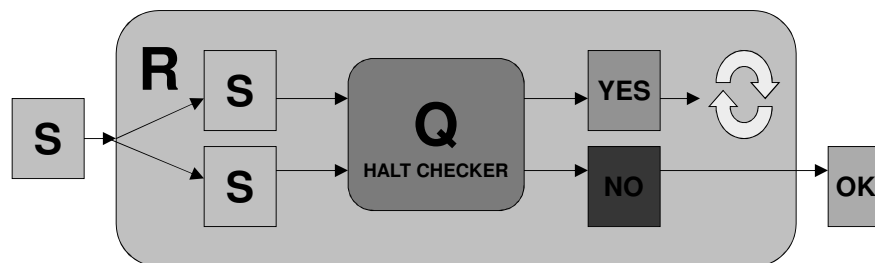
## Proof (cont'd)

- Let R be a program that takes input S, where S is a program.
- R asks the halt checker Q what happens if S runs with itself as input?
- If Q answers that S will halt if it runs with itself as input, then R goes into an infinite loop (and does not halt).
- If Q answers that S will not halt if it runs with itself as input, then R halts.

15110 Principles of Computing, Carnegie Mellon University - CORTINA

9

## How R Works

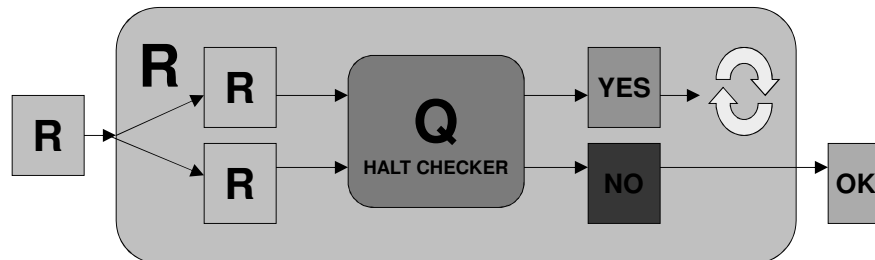


15110 Principles of Computing, Carnegie Mellon University - CORTINA

10

## R gets evil

- What happens if R tests itself?
  - If Q answers yes (R halts), then R goes into an infinite loop and does not halt.

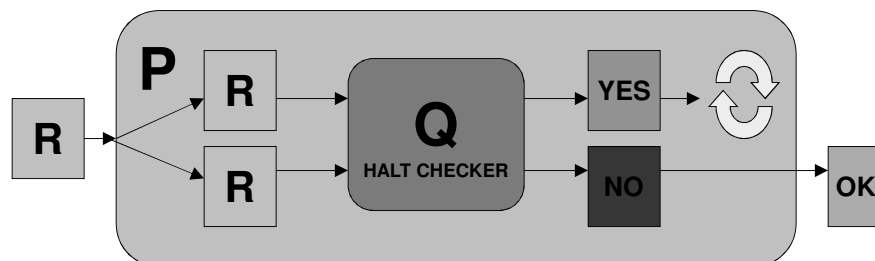


15110 Principles of Computing, Carnegie Mellon University - CORTINA

11

## R gets evil

- What happens if R tests itself?
  - If Q answers no (R does not halt), then R halts.



15110 Principles of Computing, Carnegie Mellon University - CORTINA

12

## Contradiction

- No matter what Q answers about R, R does the opposite, so Q can never answer the halting problem for the specific program R.
  - Therefore, a universal halting checker Q cannot exist.
- We can never write a computer program that determines if ANY program halts with ANY input.
  - It doesn't matter how powerful the computer is.
  - It doesn't matter how much time we devote to the computation.

## Contradiction in Real Life

