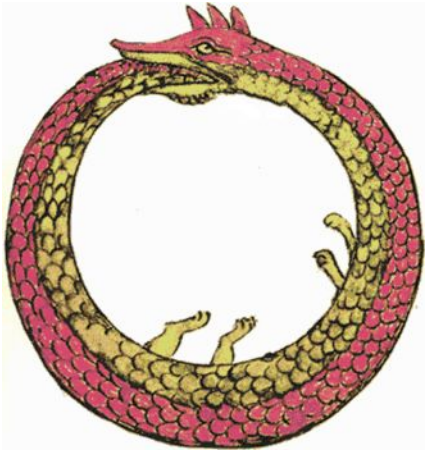
COMP112 in Week 12, 2022

much of this talk is based on one by
Peter Andreae (a.k.a. “Pondy”)

Halting

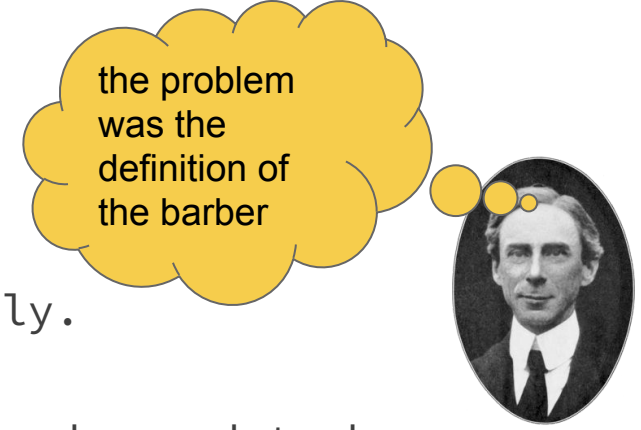
don’t stop

Self-reference



Eubulides the Megarian

A PARADOX



Imagine a town with a strict law on shaving.
Every adult human male is required to shave daily.
But it's not obligatory to shave yourself.
For those who don't want to, there's the **barber**, decreed to be:

"The barber is the "one who shaves all those, and those only, who do not shave themselves"."

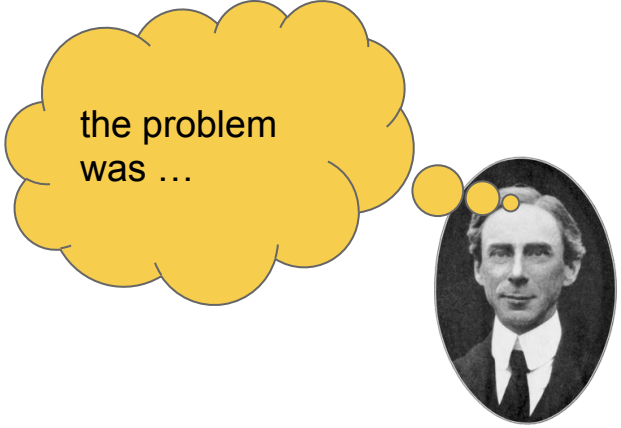
This definition seems fine… but who will shave the barber?

- if he shaves himself, he can't be the barber
- if he *doesn't* shave himself, he must shave himself

Another one

Some books reference themselves

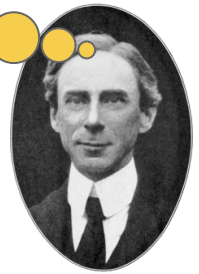
e.g. a dictionary



'Reviews of this book', a book containing only reviews of itself

Another one

there's a problem with sets - ie. at the very heart of logic!



Some sets contain themselves:

e.g. the set of ideas *is an idea*

But not all:

e.g. a set of birds *is not a bird*

Consider the set of *sets that contain themselves*,

and **the set of *sets that don't***

does it contain itself?

- if it does... it's one of those that doesn't
- but if it doesn't... then it has to

Preventing Bugs in Programs

Programs have bugs

- What problems does this cause?

How can we get rid of bugs?

- Testing
- Verifying - proving the code is right
- Using safer languages,
  - e.g. “buffer overflow” not possible in Java, vs C

- Automated checking in the compiler
  - e.g. type checking in Java, vs Python
  - e.g. assertions and annotations

- How good could we get?

Perfect checking isn't possible

Fundamental results of Theoretical Computer Science!

Can prove that some things we would like to do are not possible

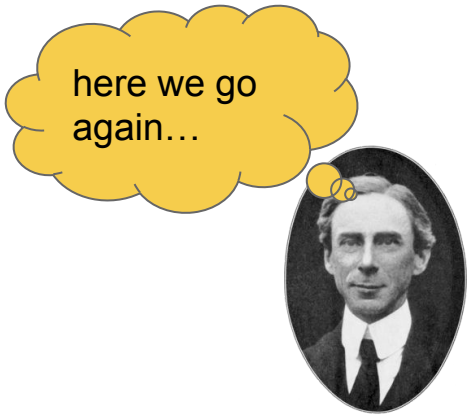
```
/* lunar lander in game: smooth descent */
double height = 1000;   // start 1000 meters up
double speed = 200;
while (height > 0){
   moveDown(speed);           //  move down for 1 second
   speed = speed * 0.75;      // slow down a bit
}
UI.println("Landed");
```

Will it stop?

Will it halt?

```
public void simulate(int rabbits, int foxes){
   while (rabbits + foxes > 0) {
      if (rabbits >  foxes) {
          rabbits = rabbits - foxes/2;
      }
      else {
          foxes = foxes - rabbits/10;
      }
      if (foxes < 100) { foxes++; }
      if ( rabbits < 10)  { rabbits = rabbits*2; }
   }
}
```

Could you *always* detect if a program will halt?



No! proof by contradiction:

- Suppose you gave me a perfect loop checker:

  checker(program, input)

  - "OK" **if** program(input) halts
  - "Loops" **if** program(input) loops forever

- Then I could make a new program that uses your program:

```
tricky (program)
    if ( checker(program, program) == "OK" )  { while (true) {}  }
    if ( checker(program, program) == "Loops")  { return; }
```

TRICKY TRICKY

```
tricky (program)
   if ( checker (program, program) == “OK” )  { while (true) {} }
   if ( checker (program, program) == “Loops”)  { return; }
```

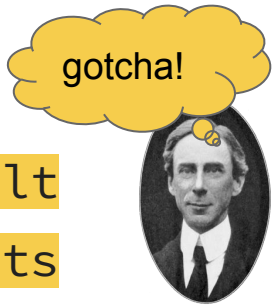
What does **tricky(tricky)** do?

if checker(tricky, tricky) == “OK” then ☐ loops forever

ie, if tricky(tricky) halts, then ☐ loops forever

if checker(tricky, tricky) == “Loops” then ☐ halts

ie, if tricky(tricky) loops forever then ☐ halts



if tricky(tricky) halts, then tricky(tricky) doesn’t halt

if tricky(tricky) doesn’t halt, then tricky(tricky) halts

The Halting problem

- Contradictions aren't possible.

  Therefore the assumption must be false.

  - You can’t make a checker that always tells whether an arbitrary program will halt.

- The Halting Problem is just one example of non-computable (undecidable) problems.

- Gödel’s Theorems: You cannot make a theorem prover that could prove all the true statements in some logical system.

What does it mean for preventing bugs

- You can make a useful compiler/program checker that can identify lots of problems, but you cannot make a perfect one.

  Don’t waste time trying to make it perfect, just make it better

- You can write theorem provers that will help you prove that a program is correct, but you can’t make a perfect one.

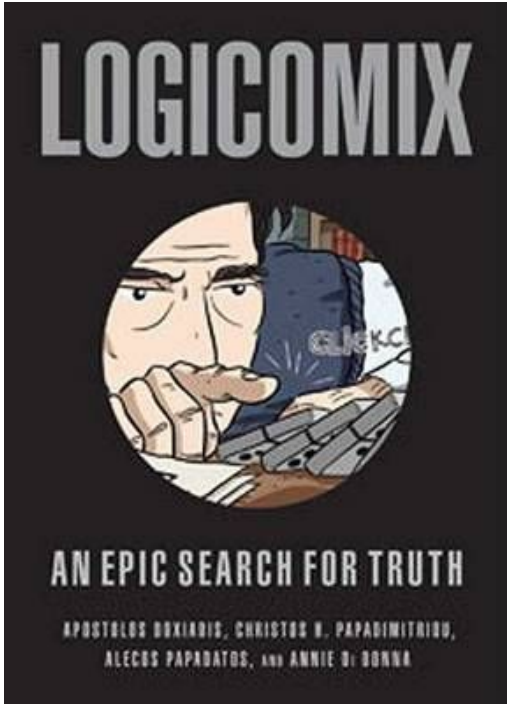
- Designing languages, compilers, automated provers, is important for improving software, but there are fundamental limitations.

Kurt Gödel and so on

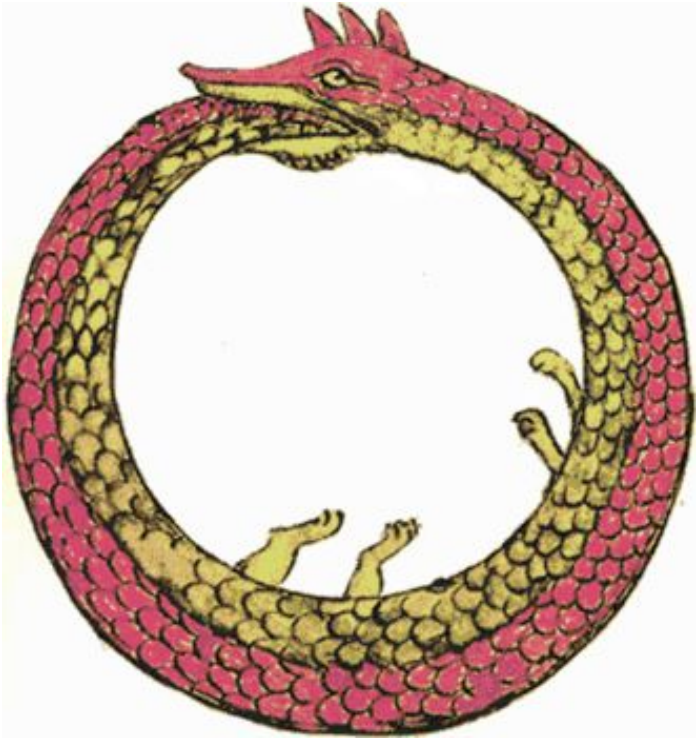
Is a correctly formulated mathematical question *necessarily answerable?*

i.e. is every mathematical statement **proveable**, either the statement itself, or - if it states something false - its opposite?

No. There will *always* be unanswerable questions.



Many Thanks



please don't forget

https://en.wikipedia.org/wiki/Self-reference