The Halting Problem

I want a ~~method~~ computer program (originally, "mechanistic procedure". this was in 1930s before actual computers) that will tell me whether ~~some~~ any computer program given to it as input will eventually halt.

Thm: (Turing) This is impossible.

"**Pf:**" Suppose for contradiction there is such a computer program, packaged as a C function:

```
boolean haltp(char* program,
              char* input)
        :
        :
        :
end
```

I can write a new function:

```
void breaker(char* x) {
  if haltp(x, x)
      go into infinite loop
  else
      terminate
}
```

Call breaker(this string as input)

Suppose haltp returns true when input green stuff, green stuff, then when breaker is called w/ green stuff, it doesn't halt. This means haltp lied.

Suppose haltp returns false when input green stuff, greenstuff, then when breaker is called w/ 〰, it halts, This means haltp lied.

Conclusion: there is no such program.

Why do we have another 15 weeks?

"Theorem:" Every positive integer can be described in less than 1000 ~~words~~ letters (in English)

"Pf": Suppose not. Then there exists a smallest positive integer that can't be described with less than 1000 letters. We can describe this integer as "The smallest positive integer that cannot be described with less than one thousand ~~ch~~ letters" which is a description with <1000 letters

---

~~Why~~ How ~~this~~ can this happen? It turns out ~~we~~ once we make precise what it means to "describe," this paradox goes away.

This semester: Talk about mathematically precise notions of "computer program," "input," "output" "halt" etc. to make sure this argument holds up.

Outline for semester (main concepts)

① Mathematical (and informal) definitions of several kinds of automata - formal mathematical models of computers w/ varying levels of complexity and capabilities.

(singular: automaton)

② Automata will take a string (w/ a defined alphabet) as input and output either "true" or "false"

③ Given an automaton, we can talk about the language of the automaton - the set of strings it says "true" to.

Given a particular type of automaton, this will imply some properties of its language.

how the strings in language are related to each other,

NOT ~~properties the individual strings have~~

Example language: {strings of a's and b's where the number of a's is divisible by the number of b's}

baaa ~~abbb~~ is in language

abaaba ~~babbab~~ is in language

abbaa is not

---

(4) Think about a language in terms of rules for generating all the possible strings in the language.

grammar - set of rules for generating all strings in lang.

type of automation

↕

restrictions on language
(properties)

↕

how complicated our grammar
needs to be.