Disc 4a

Andy

UC Berkeley

July 9, 2018

Minilecture: Computability

Definition

Computability asks can a problem be solved via algorithm (computers)?

1. Can we have a program that tells us if another program halts?

Minilecture: Computability

Definition

Computability asks can a problem be solved via algorithm (computers)?

1. Can we have a program that tells us if another program halts?
2. Is there a perfect antivirus?

Minilecture: Computability

Definition

Computability asks can a problem be solved via algorithm (computers)?

1. Can we have a program that tells us if another program halts?
2. Is there a perfect antivirus?
3. Perfect compression algorithm?

Minilecture: Computability

What is a computer?

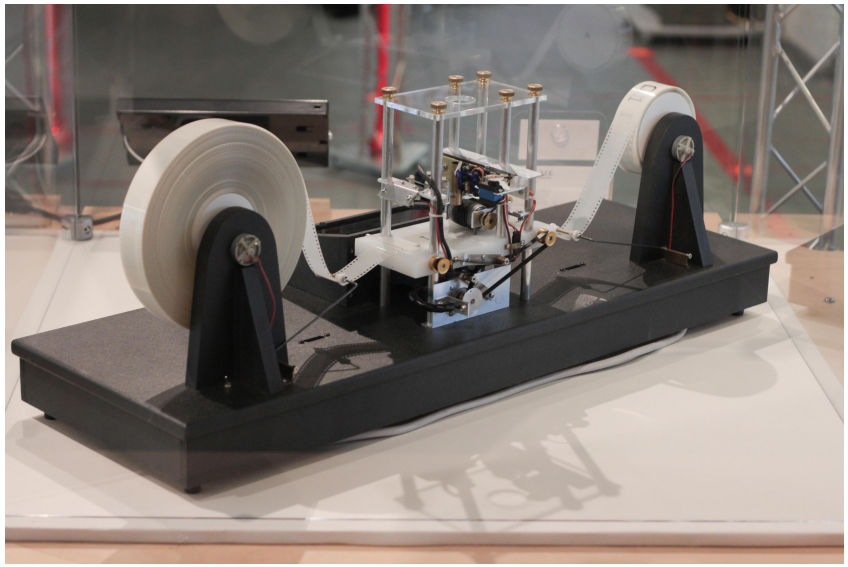

Figure: Turing Machine

1. Has memory. Ideally infinite memory.

Minilecture: Computability

What is a computer?

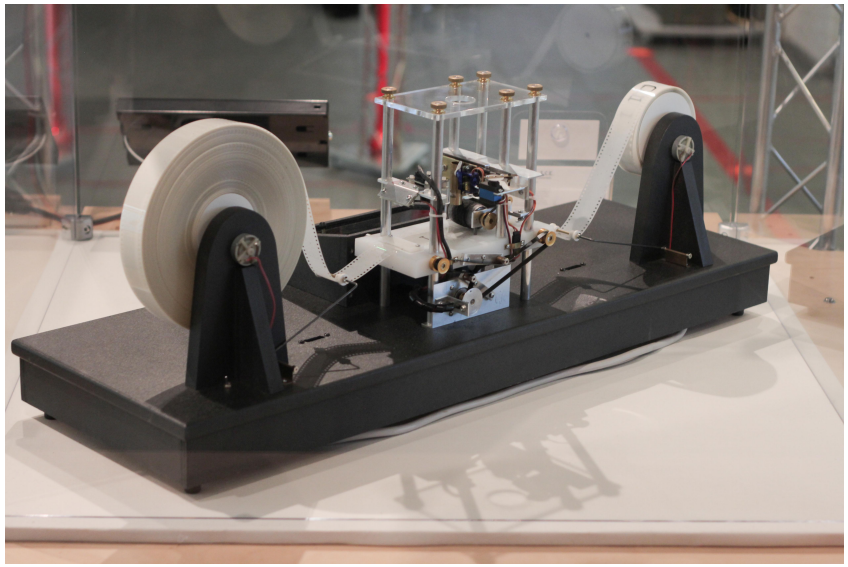

Figure: Turing Machine

1. Has memory. Ideally infinite memory.
2. Has a head that can read memory

Minilecture: Computability

What is a computer?

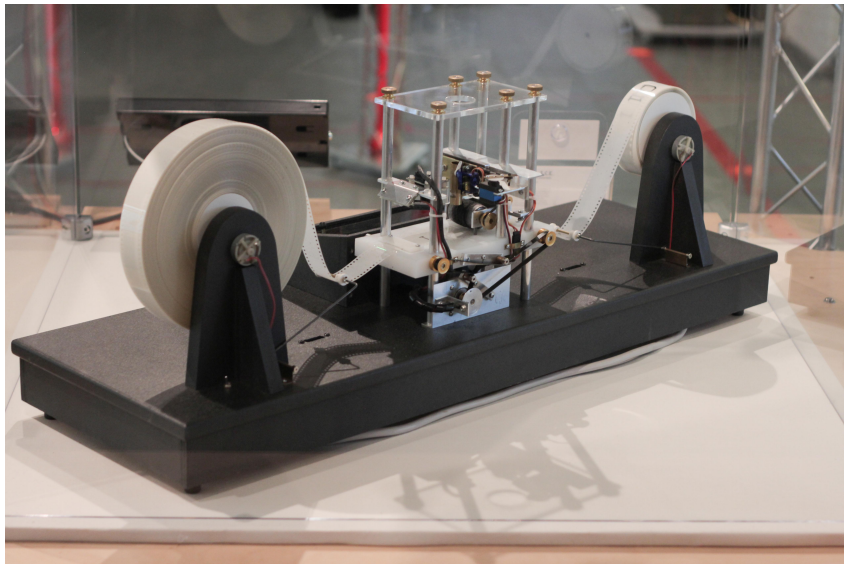

Figure: Turing Machine

1. Has memory. Ideally infinite memory.
2. Has a head that can read memory
3. Head can write 1 or 0 to memory

Minilecture: Computability

What is a computer?

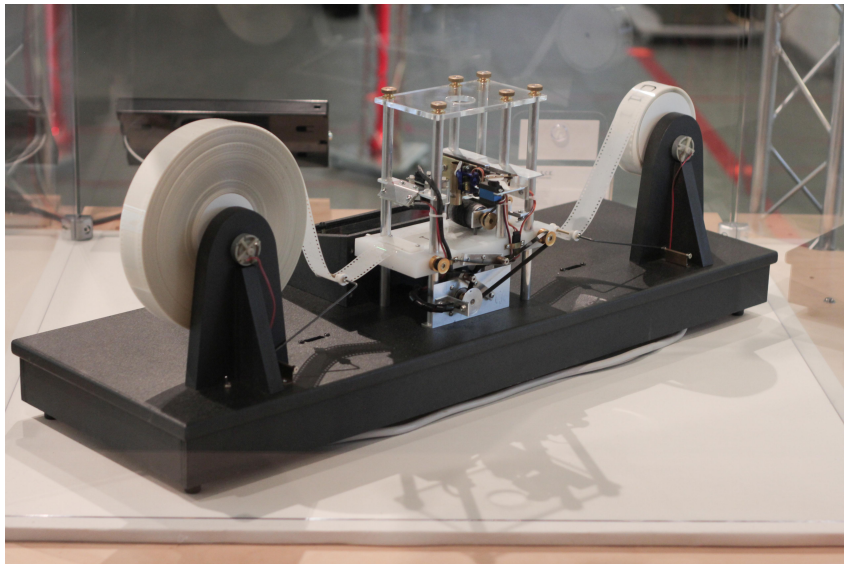

Figure: Turing Machine

1. Has memory. Ideally infinite memory.
2. Has a head that can read memory
3. Head can write 1 or 0 to memory
4. Head decides what to do based on memory.

Assumptions

1. Computers have infinite compute
2. Computers have infinite memory
3. But computation still takes time

Halting Problem

1. "This statement is false"

Halting Problem

1. "This statement is false"
2. Can we have a program that determines if another program halts?

Halting Problem

1. "This statement is false"
2. Can we have a program that determines if another program halts?
3. Assume we have a program, TestHalt, that takes in P and x, and will tell you if P(x) halts

Including Code

```
def Turing(P):
  if TestHalt(P, P):
    loopforever();
  else:
    halt();
```

1. We run Turing(Turing)

Including Code


```
def Turing(P):
  if TestHalt(P, P):
    loopforever();
  else:
    halt();
```

1. We run Turing(Turing)
2. If Turing(Turing) halts, then Turing(Turing) loops...

Including Code


```
def Turing(P):
  if TestHalt(P, P):
    loopforever();
  else:
    halt();
```

1. We run Turing(Turing)
2. If Turing(Turing) halts, then Turing(Turing) loops...
3. If Turing(Turing) loops, then Turing(Turing) halts...

Including Code


```
def Turing(P):
  if TestHalt(P, P):
    loopforever();
  else:
    halt();
```

1. We run Turing(Turing)
2. If Turing(Turing) halts, then Turing(Turing) loops...
3. If Turing(Turing) loops, then Turing(Turing) halts...
4. Paradox

Including Code


```
def Turing(P):
  if TestHalt(P, P):
    loopforever();
  else:
    halt();
```

1. We run Turing(Turing)
2. If Turing(Turing) halts, then Turing(Turing) loops...
3. If Turing(Turing) loops, then Turing(Turing) halts...
4. Paradox
5. We cannot have a paradox. If you can build a Testhalt -¿ paradox

Including Code


```
def Turing(P):
  if TestHalt(P, P):
    loopforever();
  else:
    halt();
```

1. We run Turing(Turing)
2. If Turing(Turing) halts, then Turing(Turing) loops...
3. If Turing(Turing) loops, then Turing(Turing) halts...
4. Paradox
5. We cannot have a paradox. If you can build a Testhalt -¿ paradox
6. Anything that allows you to build a TestHalt program is impossible