DIS 4A

Sunday, July 8, 2018 9:29 PM

Topics: Self-Reference and Computability

The halting problem

```
def TestHalt(P,X):
    if P(x) halts:
        return True
    else:      P(x) loops forever
        return False

def Turing(P):
    if TestHalt(P,P) == True:
        loop forever
    else:
        halt
```

Turing(Turing):

Case 1: halts ⇒ TestHalt(Turing, Turing) returns "no"
⇒ Turing(Turing) doesn't halt

Case 2: loops ⇒ TestHalt(Turing, Turing) returns "yes"
⇒ Turing(Turing) halts

programs are bit strings with finite length. They're countable and can be passed in as an argument.

	P_1	P_2	P_3	
P_1	H	H	L	
P_2	L	H	H	
P_3	H	L	L	
			⋱	▨ ← Turing(Turing)

So Turing does not exist on the list

⇒ TestHalt does not exist

Remark: the problem is about TestHalt. We use Turing to show TestHalt is not computable.

- **Reduction**: If you can use B to solve A...

  A → B. A reduces to B. B is harder.

- **Uncomputability proof**:

  Show P is not computable by using P to solve TestHalt.