CS121 Lecture 9: Oct 1, 2019

Previously: Finite Functions

Today:

Thm 1: Exists a TM U s.t. $U(M,x) = M(x)$ for every TM M and $x \in \{0,1\}^*$

Thm 2: Exists $F: \{0,1\}^* \to \{0,1\}$ that is uncomputable

Thm 3: Exist infinitely F's that are uncomputable.

Reductions: "If pigs could whistle then horses can fly"

proof by contradiction

Universal TM

Lemma 1: Can represent a TM as a string

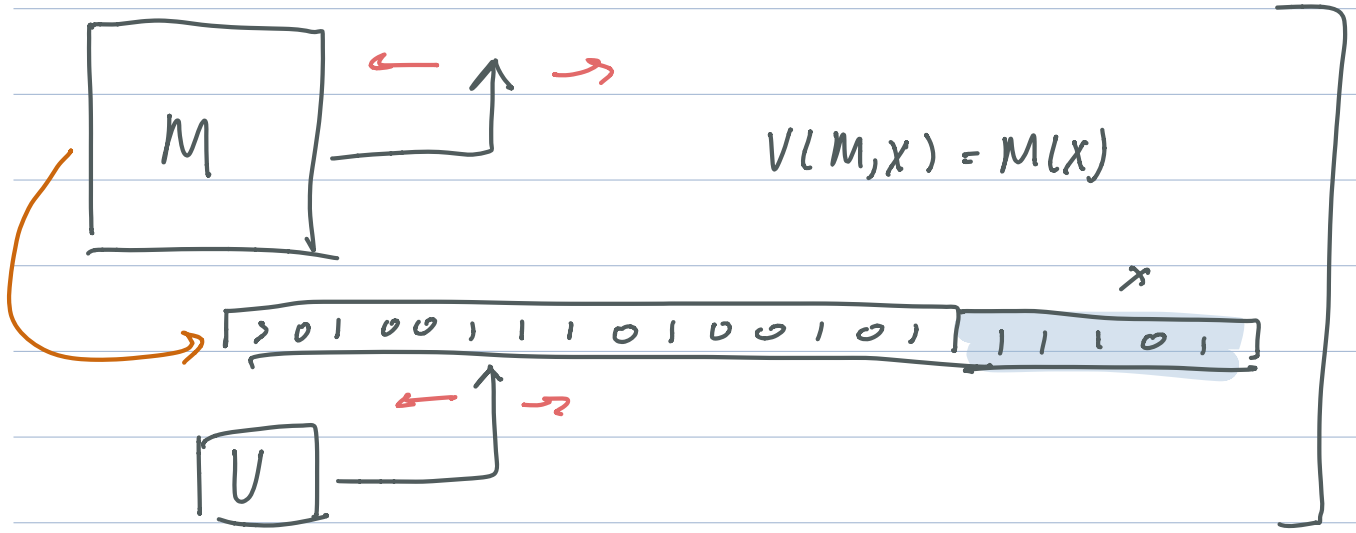
Proof: With transition table

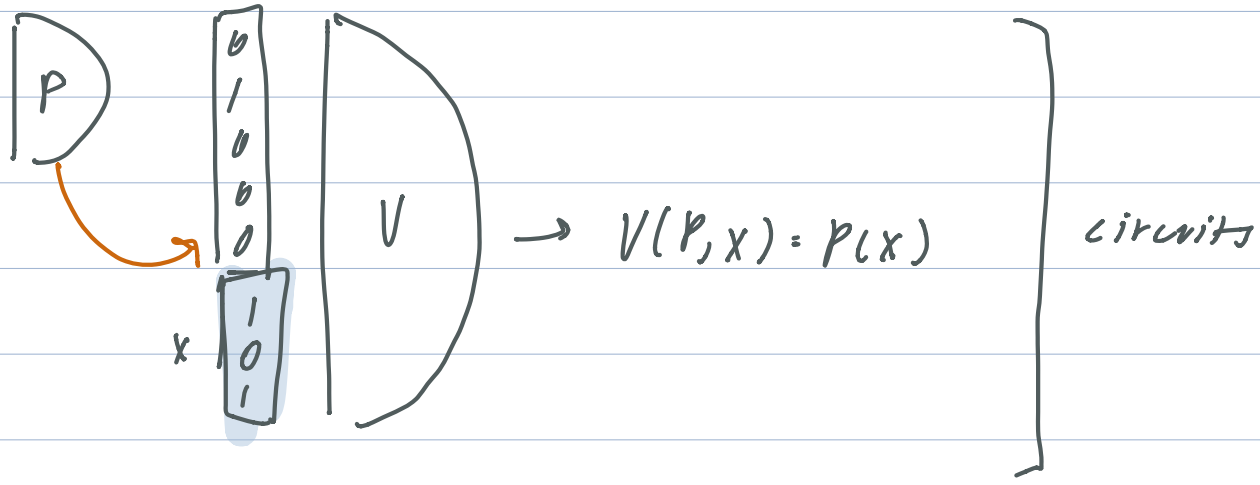
Thm 1: Exists TM U s.t. for every TM M and $x \in \{0,1\}^*$, $U(M,x) = M(x)$

i.e., for all x, if M halts on x then $U(M,x) = M(x)$

Otherwise $U(M,x) = \perp$.

aka interpreter for TM





Proof of Thm 1: By VLEVCAHIT, enough to come up w/ NAND-RAM program for doing so or even Python, etc.

The Halting problem

$HALT: \{0,1\}^* \to \{0,1\}$

$$HALT(M,x) = \begin{cases} 1, & M(x) \neq \perp \\ 0, & M(x) = \perp \end{cases}$$

← M halts on input x

← M enters inf loop on input x

Ex 1: What is HALT(P, 1024)? 1, because breaks for powers of 2

```
P = def mystery(n):
        i = 1
        while True:
            if i == n: break
            i = i * 2
```

Ex 2: What is HALT(A, 1021)? 0, 1021 is prime

```
A = def mystery2(n):
        j = 2
        while True:
            if j == n: j = 2
            if j % n == 0: break
            j += 1
```

Thm 2: HALT is uncomputable

For every TM (or program) H there is a pair M, x such that H(M,x) ≠ HALT(M,x)

- To prove: Pretend adversary came up w/ program H, show you can come up with a pair M, x s.t. H(M,x) ≠ HALT(M,x)

$\forall H \; \exists (M, x)$

↑ Adversary ↖ us

```
int boaz_test(char *input) {
    int h;
    h = eddie_halt(input, input);
    if(h) while(1);
    return 0;
}
```

boaz_test(boaz_test) halts

→ eddie_halt(boaz_test, boaz_test) = 1

→ boaz_test(boaz_test) doesn't halt

OR

boaz_test(boaz_test) doesn't halt

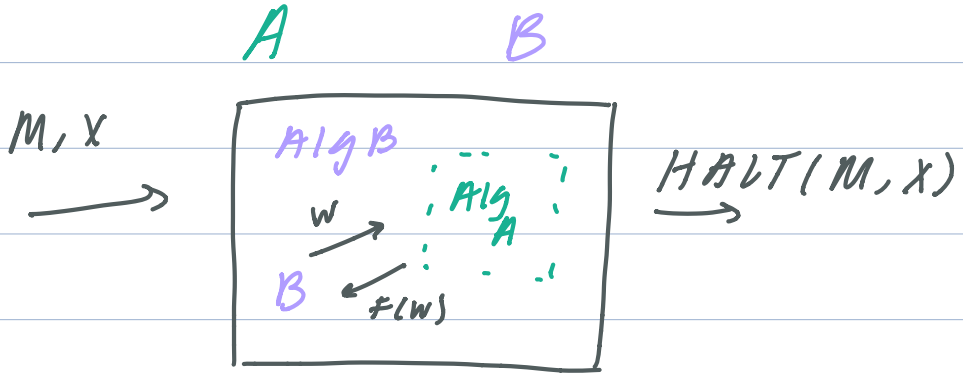
→ eddie_halt(boaz_test, boaz_test) = 0

→ boaz_test(boaz_test) halts

Reductions

Thm: F is uncomputable

proof: ∃ Alg A for F ⇒ ∃ Alg B for HALT



1. Description of B: Via calls to "magic box" computing F

2. Analysis of B: If box computes on B

Aside

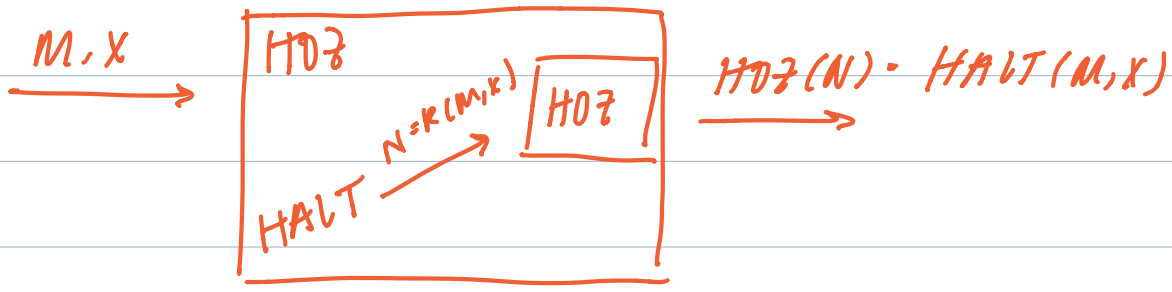
Q: Give function hardwire(code, m) that takes code of a python function f(n) and number m, and returns the code of Python function g() s.t. g() returns f(m)

```
def hardwire(code, m):
    return code.replace("f(n):\n", f"g():\n    n = {m}\n")
```

Lemma: There is a computable function hardwire: $\{0,1\}^* \to \{0,1\}^*$ s.t. for every TM M and $x \in \{0,1\}^*$, hardwire$(M,x) = N$ s.t. $N(w) = M(x)$ for every $w \in \{0,1\}^*$

Ex Reduction #1: Let $HOT(N) = \begin{cases} 1, & N(0) \neq \perp \\ 0, & N(0) = \perp \end{cases}$

Prove that HOT is uncomputable



Assume toward contradiction that A computes HOT.
Show B that computes HALT by showing computable R satisfying $HOT(R(M,x)) = HALT_{(M,x)}$ for every TM and input x.

```
X = 01101101000,
def f(x):                    ⟹    def g(m):
    ....                               X = 01101101000,
                                       ....
```

if f(x) halts on x, g(m) halts on m=0.

Rice's Theorem

if $F: \{\text{Turing Machines}\} \to \{0,1\}$ has the property that F(M) only depends on M's functionality, then F is either trivial or uncomputable.

Def: M and M' are functionally equivalent if for every $x \in \{0,1\}^*$, $M(x) = M'(x)$.
Notation: $M \cong M'$

Def. $F: \{0,1\}^* \to \{0,1\}$ is semantic if for every $M \cong M'$, $F(M) = F(M')$

Q: Let one: $\{0,1\}^* \to \{0,1\}$ be constant one function ($one(w) = 1$ for every w). Then one is semantic.

Rice: For every $F: \{0,1\}^* \to \{0,1\}$, if F is semantic then either $F = one$ or $F = zero$ or F is uncomputable.