CSCI 301 - Lecture 35: Computability; Reductions

Thm: HALT is not decidable

Proof: Suppose $H(\langle M, x\rangle)$ exists

Construct $Z(\langle M, w\rangle)$ that:

1. Simulates H on $\langle M, w\rangle$
   - Loop forever if $H(\langle M, w\rangle)$ accepts
   - Accept if $H(\langle M, w\rangle)$ rejects

Run $Z(\langle Z, Z\rangle)$

If Z accepts, then H said Z didn't terminate
doesn't terminate, H said it did!

Reduction

Reduce Halt to A_{TM} $\quad L(M_{TM}) = \{ \langle M, w \rangle :$ M accepts $w \}$

turing machine

Suppose A_{TM} decidable; so M_{TM} exists.

Construct $M_H(\langle M, x \rangle)$:

- Simulate M_{TM} on $\langle M, x \rangle$
  - if M_{TM} accepts, accept
- Otherwise construct M' to mirror M, except $q_{accept} \leftrightarrow q_{reject}$

Run M_{TM} on $\langle M', x \rangle$
if M_{TM} accepts, then M halted and rejected x; accepts
otherwise reject

if M accepts x
or M rejects x
return true
else,
return false

Suppose $M_{HI}(M)$ exists

Construct $H(\langle M, x \rangle)$ to solve HALT:

Construct $M'()$

Simulate M on input x, but

- put all output on 2nd tape
- before accept or reject, print HELLO to the 1st tape

return $M_{HI}(M')$

```
def H(M, x):
    def M'():
        Run M on x, except
            - output to 2nd tape
            - accept() -> Print(HELLO); accept()
            - reject() -> print(HELLO); reject()

    return M_HELLO(M')
```