Lecture 12

Why TM's?
Programs are OK too

Fix Σ = printable ASCII

Programming language with ints, strings & function calls

"Computable function" = always returns something

"Decider" = computable function always returning 0 / 1

"Acceptor" = accept if return 1; reject if $\neq 1$ or loop

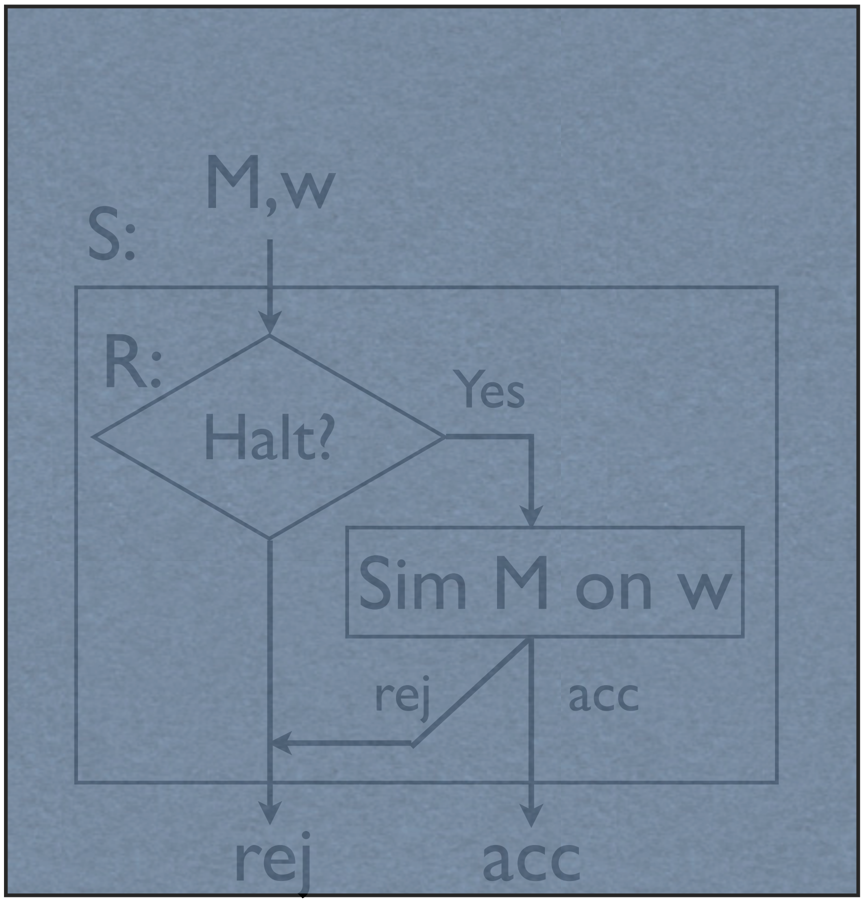
A_{Prog} = {<P,w> | program P returns 1 on input w }

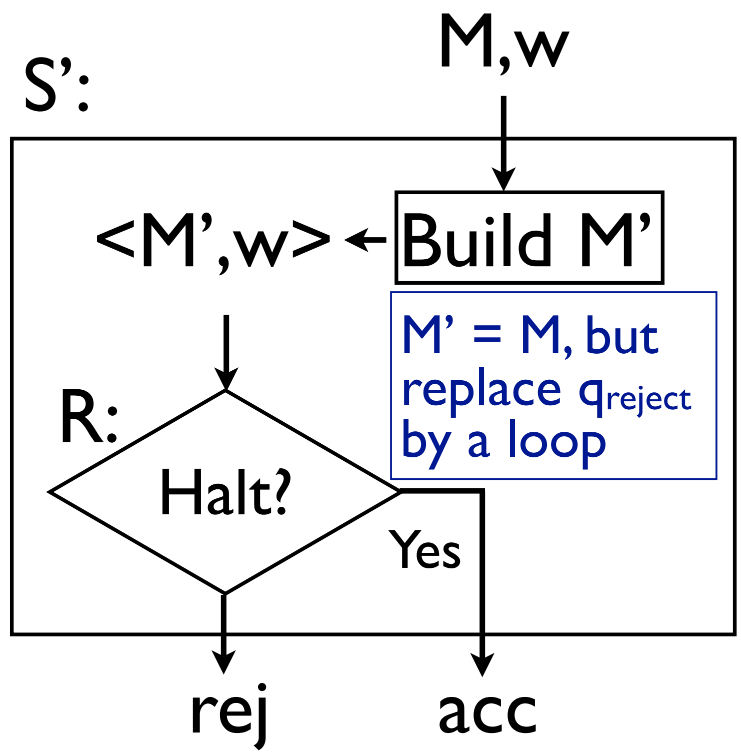
$HALT_{Prog}$ = {<P,w> | prog P returns *something* on w }

...

A_{TM} ($\le_T$ vs $\le_m$) $HALT_{TM}$

f(<M,w>) = <M',w>





From Lecture 07

$A_{Prog} \leq_m HALT_{Prog}$

$f(\langle P,w \rangle) = \langle P',w \rangle$

```
sub f(P,w){
  // build P'
  pn = ...//(find P's name)
  pp = "sub " + pn + "prime(x){"
  pp += P
  pp += "if "+pn+"(x) return 1;"
  pp += "while True {;}}"
  val = "<" + pp + "," + w + ">"
  return val
```

```
sub Pprime(x){
    sub P(y){
    ...                  (copy of P)
    }
    if P(x) return 1;
    while True { ; }
}
```

Programs vs TMs

Everything we've done re TMs can be rephrased re programs

From the Church-Turing thesis (hopefully made concrete in earlier HW) we know they are equivalent.

Above example shows some things are easier with programs.

Others get harder (e.g., "Universal TM" is a Java interpreter written in Java; "configurations" and "computation histories" are much messier)

TMs are convenient to use here since they strike a good balance

But I hope you can mentally translate between the two; decidability/ undecidability of various properties of programs are obviously more directly relevant.

Mapping Reducibility

Defn: A is *mapping reducible* to B ($A \leq_m B$) if there is computable function *f* such that $w \in A \Leftrightarrow f(w) \in B$

A special case of $\leq_T$:
Call subr only once; its answer is *the* answer

Theorem:

$A \leq_m B$ & B decidable (recognizable) $\Rightarrow$ A is too

$A \leq_m B$ & A *un*decidable (*un*recognizable) $\Rightarrow$ B is too

$A \leq_m B$ & $B \leq_m C \Rightarrow A \leq_m C$

Most reductions we've seen were actually $\leq_m$ reductions.

Other Examples of $\leq_m$

$A_{TM} \leq_m REGULAR_{TM}$ $f(<M,w>) = <M_2>$

Build M_2 so $L(M_2) = \Sigma^* / \{ 0^n 1^n \}$, as M accept/rejects w

$EMPTY_{TM} \leq_m EQ_{TM}$ $f(<M>) = <M, M_{reject}>$

$L(M_{reject}) = \varnothing$, so equiv to M iff $L(M) = \varnothing$

$A_{TM} \leq_m MPCP$

$MPCP \leq_m PCP$

} 5.2

$A_{TM} \leq_m \overline{EMPTY_{TM}}$ $f(<M,w>) = <M_1>$

Build M_1 so $L(M_1) = \{w\} / \varnothing$, as M accept/rejects w

EMPTY_{TM} is undecidable

EMPTY_{TM} = { <M> | M is a TM s.t. $L(M) = \varnothing$ }

Pf: To show: $A_{\text{TM}} \leq_{\text{T}} \text{EMPTY}_{\text{TM}}$

On input <M,w> build M' :

M' on input x:

1. erase x
2. write w
3. run M on w
4. if M accepts w, then accept x
5. otherwise, reject x

Do *not* run M or M'. (That whole "halting thing" means we might not learn much if we did.) But note that L(M') is/is not empty exactly when M does not/does accept w, so knowing whether $L(M') = \varnothing$ answers whether <M,w> is in A_{TM}. And our hypothetical "EMPTY_{TM}" subroutine applied to M' tells us just that. I.e., $A_{\text{TM}} \leq_{\text{T}} \text{EMPTY}_{\text{TM}}$

$$L(M') = \begin{cases} \Sigma^*, & \text{if M accepts w} \\ \varnothing, & \text{if M rejects w} \end{cases}$$

NB: it shows $A_{\text{TM}} \leq_{\text{m}} (\text{EMPTY}_{\text{TM}})^{C}$

From Lecture 07

$REGULAR_{TM}$ is undecidable

$REGULAR_{TM}$ = { <M> | M is a TM s.t. L(M) is regular }

Pf: To show: $A_{TM} \leq_T REGULAR_{TM}$

On input <M,w> build M' :

M' on input x:

1. if $x \in \{0^n1^n | n \geq 0\}$, accept x
2. otherwise, erase x
3. write w
4. run M on w
5. if M accepts w, then accept x
6. otherwise, reject x

Do *not* run M or M'. (That whole "halting thing" ...) But note that L(M') is/is not regular exactly when M does/does not accept w, so knowing whether L(M') is regular answers whether <M,w> is in A_{TM}. The hypothetical "$REGULAR_{TM}$" subroutine applied to M' tells us just that. I.e., $A_{TM} \leq_T REGULAR_{TM}$

$$L(M') = \begin{cases} \Sigma^*, \text{ if M accepts w} \\ \{0^n1^n | n \geq 0\}, \text{ otherwise} \end{cases}$$

Exercise: Is it $A_{TM} \leq_m REGULAR_{TM}$? If not, could it be changed?

From Lecture 07

More on $\leq_T$ vs $\leq_m$

Theorem: For *any* L, $L \leq_T \overline{L}$

The same is not true of $\leq_m$:

Theorem: L recognizable and $L \leq_m \overline{L} \Rightarrow$ L is decidable.

Proof: on input x, dovetail recognizers for $x \in L$ & $f(x) \in L$

Corr: $A_{TM} \leq_T \overline{A_{TM}}$ but *not* $A_{TM} \leq_m \overline{A_{TM}}$

Theorem: $A \leq_m B$ iff $\overline{A} \leq_m \overline{B}$

Theorem: If L is not recognizable and both $L \leq_m B$ and $L \leq_m \overline{B}$, then neither B nor $\overline{B}$ are recognizable

EQ_{TM} is neither recognizable nor co-recognizable

M_0: on any input x, reject x. $L(M_0) = \varnothing$

M_1: on any input x, accept x. $L(M_1) = \Sigma^*$

For any <M,w>, let h(<M,w>) = M_2 be the TM that, on input x,

1. runs M on w
2. if M accepts w, then accept x.

Claim: $L(M_2) = \Sigma^*$ (if $\langle M,w\rangle \in A_{TM}$), else $= \varnothing$ & h computable

Then $\overline{A_{TM}} \leq_m EQ_{TM}$ via $g(\langle M,w\rangle) = \langle M_0,M_2\rangle$

And $\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$ via $f(\langle M,w\rangle) = \langle M_1,M_2\rangle$ (& $A_{TM} \leq_m EQ_{TM}$)