

## UNDECIDABILITY II: REDUCTIONS

TUESDAY OCT 4

$A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \}$   
 $A_{TM}$  is undecidable: (constructive proof & subtle)

Assume machine H semi-decides  $A_{TM}$

$$H(M,w) = \begin{cases} \text{Accept} & \text{if } M \text{ accepts } w \\ \text{Rejects or loops} & \text{otherwise} \end{cases}$$

Construct a new TM  $D_H$  as follows: on input M, run H on (M,M) and output the "opposite" of H whenever possible.

$$D_H(D_H) = \begin{cases} \text{Reject} & \text{if } D_H \text{ accepts } D_H \\ & \text{(i.e. if } H(D_H, D_H) = \text{Accept}) \\ \text{Accept} & \text{if } D_H \text{ rejects } D_H \\ & \text{(i.e. if } H(D_H, D_H) = \text{Reject}) \\ \text{loops} & \text{if } D_H \text{ loops on } D_H \\ & \text{(i.e. if } H(D_H, D_H) \text{ loops)} \end{cases}$$

**Note:** There is no contradiction here!

$D_H$  loops on  $D_H$

We can effectively construct an instance which does not belong to  $A_{TM}$  (namely,  $(D_H, D_H)$ ) but H fails to tell us that.

That is:

Given any semi-decision machine H for  $A_{TM}$  (and thus a potential decision machine for  $A_{TM}$ ), we can effectively construct an instance which does not belong to  $A_{TM}$  (namely,  $(D_H, D_H)$ ) but H fails to tell us that.

So H cannot be a decision machine for  $A_{TM}$

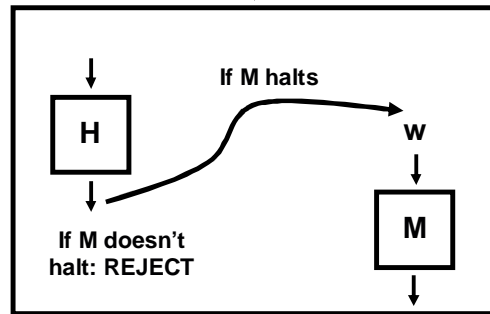
### THE HALTING PROBLEM

$HALT_{TM} = \{ (M,w) \mid M \text{ is a TM that halts on string } w \}$

Theorem:  $HALT_{TM}$  is undecidable

Proof: Assume, for a contradiction, that TM H decides  $HALT_{TM}$

$(M,w)$



### THE HALTING PROBLEM

$HALT_{TM} = \{ (M,w) \mid M \text{ is a TM that halts on string } w \}$

Theorem:  $HALT_{TM}$  is undecidable

Proof: Assume, for a contradiction, that TM H decides  $HALT_{TM}$

We use H to construct a TM S that decides  $A_{TM}$

On input  $(M,w)$ , S runs H on  $(M,w)$

If H rejects, reject

If H accepts, run M on w until it halts:

Accept if M accepts and reject if M rejects

In most cases, we will show that a language is undecidable by showing that if it is decidable, then so is  $A_{TM}$

We reduce deciding  $A_{TM}$  to deciding the language in question

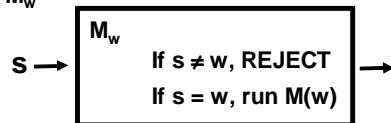
$E_{TM} = \{ M \mid M \text{ is a TM and } L(M) = \emptyset \}$

Theorem:  $E_{TM}$  is undecidable

Proof: Assume, for a contradiction, that TM Z decides  $E_{TM}$

Algorithm for  $A_{TM}$ : On input  $(M,w)$ :

1. Create  $M_w$

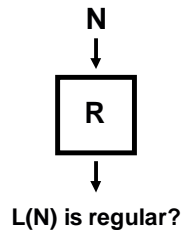
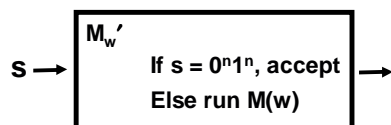


2. Run Z on  $M_w$

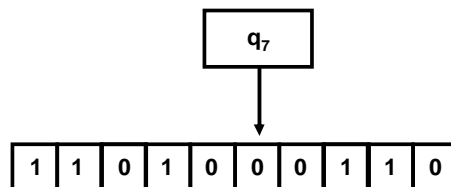
$REGULAR_{TM} = \{ M \mid M \text{ is a TM and } L(M) \text{ is regular} \}$

Theorem:  $REGULAR_{TM}$  is undecidable

Proof: Assume, for a contradiction, that TM R decides  $REGULAR_{TM}$



**CONFIGURATIONS**  
**11010q<sub>7</sub>00110**



## COMPUTATION HISTORIES

An accepting computation history is a sequence of configurations  $C_1, C_2, \dots, C_k$ , where

1.  $C_1$  is the start configuration,
2.  $C_k$  is an accepting configuration,
3. Each  $C_i$  follows from  $C_{i-1}$

An rejecting computation history is a sequence of configurations  $C_1, C_2, \dots, C_k$ , where

1.  $C_1$  is the start configuration,
2.  $C_k$  is a rejecting configuration,
3. Each  $C_i$  follows from  $C_{i-1}$

**M accepts w if and only if there exists an accepting computation history that starts with  $C_1 = q_0 w$**

$ALL_{PDA} = \{ P \mid P \text{ is a PDA and } L(P) = \Sigma^* \}$

Theorem:  $ALL_{PDA}$  is undecidable

Proof: Assume, for a contradiction, that TM A decides  $ALL_{PDA}$

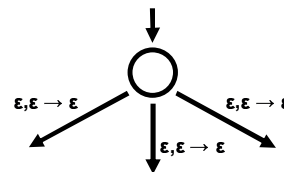
We use A to decide  $A_{TM}$

On input  $(M, w)$ , construct a PDA P that accepts  $\Sigma^*$  if and only if M does not accept w

P will recognize all strings that are NOT accepting computation histories for M on w

P will recognize all strings (read as sequences of configurations) that:

1. Do not start with  $C_1$
2. Do not end with an accepting configuration
3. Where some  $C_i$  does not properly yield  $C_{i+1}$



P recognizes all strings except:  
 $C_1, C_2^R, C_3, C_4^R, C_5, C_6^R, \dots, C_k$

$ALL_{PDA} = \{ P \mid P \text{ is a PDA and } L(P) = \Sigma^* \}$

**Theorem:**  $ALL_{PDA}$  is undecidable

**Proof:** Assume, for a contradiction, that TM A decides  $ALL_{PDA}$

We use A to decide  $A_{TM}$

On input  $(M,w)$ , construct a PDA P that accepts  $\Sigma^*$  if and only if M does not accept w

P will recognize all strings that are NOT accepting computation histories for M on w

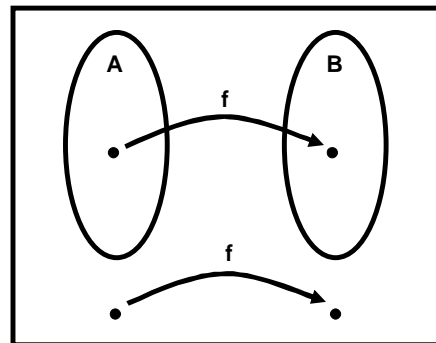
### MAPPING REDUCIBILITY

$f : \Sigma^* \rightarrow \Sigma^*$  is a computable function if some Turing machine M, on every input w, halts with just  $f(w)$  on its tape

A language A is mapping reducible to language B, written  $A \leq_m B$ , if there is a computable function  $f : \Sigma^* \rightarrow \Sigma^*$ , where for every w,

$$w \in A \Leftrightarrow f(w) \in B$$

f is called a reduction from A to B



**Theorem:** If  $A \leq_m B$  and B is decidable, then A is decidable

**Proof:** Let M decide B and let f be a reduction from A to B

We build a machine N that decides A as follows:

On input w:

1. Compute  $f(w)$
2. Run M on  $f(w)$

**All undecidability proofs from today can be seen as constructing an f that reduces  $A_{TM}$  to the proper language**

**WWW.FLAC.WS**

Read chapter 5.1 of the book for next time