

## CHAPTER 4: DECIDABILITY

EVERY QUESTION ABOUT REGULAR  
LANGUAGES IS DECIDABLE.

SOME QUESTIONS ABOUT CFG'S  
ARE DECIDABLE, BUT  
SOME ARE NOT.

THE "HALTING PROBLEM"  
IS ~~NOT~~ NOT DECIDABLE.

SOME LANGUAGES ARE NOT  
TURING RECOGNIZABLE.

MANY QUESTIONS ABOUT  
TURING MACHINES ARE NOT  
DECIDABLE AND SOME  
ARE NOT TURING RECOGNIZABLE.

## "THE HALTING PROBLEM"

- GIVEN A PROGRAM....  
WILL IT HALT?
- GIVEN THIS TURING MACHINE,  
WILL IT HALT WHEN RUN  
ON SOME GIVEN INPUT,  $x$ ?
- GIVEN THIS PROGRAM WRITTEN  
IN JAVA [OR ANY OTHER LANGUAGE],  
WILL IT EVER GET INTO AN  
INFINITE LOOP?

### ANSWER:

- IN GENERAL, WE CAN'T ALWAYS  
KNOW.
- THE BEST WE CAN DO IS  
RUN THE PROGRAM... BUT...
- FOR MANY PROGRAMS, WE CAN PROVE  
"IT WILL ALWAYS HALT" / "IT MAY LOOP"
- BUT FOR PROGRAMS IN GENERAL,  
THE QUESTION IS **UNDECIDABLE** 2

PROBLEM:

GIVEN A D.F.A. AND A STRING,  
WILL THE DFA ACCEPT?

IS THIS PROBLEM DECIDABLE?

LANGUAGES ARE DECIDABLE.

WE MUST EXPRESS THE PROBLEM  
IN TERMS OF LANGUAGES!

" X IS A MEMBER OF THE LANGUAGE "

IFF

" THE ANSWER TO THE PROBLEM IS **YES**."

## THEOREM

THE LANGUAGE...

$$A_{\text{DFA}} = \left\{ \langle B, w \rangle \mid B \text{ is a DFA THAT ACCEPTS STRING } w \right\}$$

... IS DECIDABLE.

## POSSIBLE CONFUSION

STRING  
 $w$

LANGUAGE

DFA  $\equiv$  REGULAR LANGUAGE

STRING  
 $\langle B, w \rangle$

LANGUAGE

$A_{\text{DFA}}$

NOT REGULAR,  
NOT CFG,  
BUT IT IS  
DECIDABLE

PROOF THAT

$$A_{\text{DFA}} = \left\{ \langle B, w \rangle \mid B \text{ is a DFA that} \right. \\ \left. \text{accepts string } w \right\}$$

IS DECIDABLE.

PROVIDE A TM THAT DECIDES IT.

THE TM IS GIVEN AS INPUT  $\langle B, w \rangle$   
A DFA AND A STRING  $w$ .

THE TM CHECKS TO MAKE SURE  
 $B$  IS A VALID REPRESENTATION.

THE TM THEN SIMULATES  $B$   
ON  $w$ .

IF  $B$  REACHES ~~THE~~ A FINAL  
STATE AT THE END OF  $w$ ,

THEN THE TM WILL ACCEPT.

OTHERWISE THE TM WILL REJECT.

THIS TM WILL ALWAYS HALT.

(WE COULD PROVE THAT THE TM  
WILL ALWAYS HALT, IF NECESSARY.)

CONSIDER THE QUESTION OF WHETHER  
A NONDETERMINISTIC FINITE STATE  
AUTOMATON ACCEPTS A GIVEN STRING.

THE LANGUAGE...

$$A_{\text{NFA}} = \left\{ \langle B, w \rangle \mid \begin{array}{l} B \text{ is an NFA that} \\ \text{accepts string } w \end{array} \right\}$$

... IS DECIDABLE.

PROOF

CONSTRUCT A TM THAT TAKES AS  
INPUT THE REPRESENTATION OF  
AN NFA<sup>B</sup> AND A CANDIDATE STRING<sup>w</sup>.

APPROACH 1: SIMULATE THE NFA ON  $w$ .

APPROACH 2:

CONVERT THE NFA TO A DFA

THIS ALGORITHM WAS DESCRIBED  
IN CHAPTER 1

WE COULD PROGRAM A TM TO DO IT.

SIMULATE THE DFA ON  $w$ .

WE COULD PROGRAM A TM TO DO IT.

[WE DID THAT ~~FOR~~ FOR  $A_{\text{DFA}}$ ]

Accept if the simulation accepts,  
otherwise REJECT.

GIVEN A REGULAR EXPRESSION  $R$   
AND A STRING  $w$ , CAN WE DECIDE  
WHETHER  $R$  GENERATES  $w$ ?

THE LANGUAGE...

$$A_{\text{REX}} = \left\{ \langle R, w \rangle \mid \begin{array}{l} R \text{ is a regular expression} \\ \text{that generates string } w \end{array} \right\}$$

...IS DECIDABLE.

HOW DO WE KNOW?

WE CAN BUILD A TM / WE CAN WRITE  
A PROGRAM THAT, GIVEN A REG.  
EXPRESSION  $R$  AND STRING  $w$  AS  
INPUT, WILL DETERMINE WHETHER  
 $\langle R, w \rangle$  IS IN  $A_{\text{REX}}$ .

AND

THAT ALGORITHM / TM WILL  
ALWAYS HALT.

HALTING?

OFTEN THIS IS SELF-EVIDENT.  
SOMETIMES WE MIGHT TO PROVE  
IT CAREFULLY.

MANY PROGRAMS CAN BE PROVEN TO  
ALWAYS HALT.

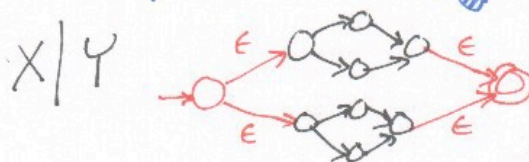
BUT THIS DOES NOT MEAN THE  
HALTING PROBLEM IS DECIDABLE!

$$A_{\text{REX}} = \{ \langle R, w \rangle \mid R \text{ is a regular expression that generates } w \}$$

### ALGORITHM / TM

CONVERT  $R$  INTO A NFA,  $B$ .

Recall the procedure ~~XXXXXX~~ from Chapter 1



Construct string  
 $\langle B, w \rangle$

Use the TM from the last theorem to check whether this is in

$$A_{\text{NFA}} = \{ \langle B, w \rangle \mid B \text{ is an NFA that accepts } w \}$$

~~XXXXXX~~



## ACCEPTANCE TESTING

IS STRING  $w$  IN THE LANGUAGE?

$$A_{DFA} = \{ \langle B, w \rangle \mid \dots \}$$

## EMPTINESS TESTING

IS THE LANGUAGE EMPTY?

$$L = \emptyset?$$

$$E_{DFA} = \{ \langle B \rangle \mid \dots \}$$

## EQUALITY TESTING

ARE TWO LANGUAGES THE SAME?

$$EQ_{DFA} = \{ \langle A, B \rangle \mid \dots \}$$

THE LANGUAGE...

$$E_{DFA} = \{ \langle A \rangle \mid A \text{ is a DFA AND } L(A) = \emptyset \}$$

... IS DECIDABLE.

ALGORITHM / TM TO DECIDE IT:

GIVEN A DFA "A", CAN WE  
GO FROM THE INITIAL STATE  
TO A FINAL STATE?

IF SO, THEN THE DFA COULD  
GENERATE SOME STRING.

⇒ IS ANY FINAL STATE  
REACHABLE FROM THE INITIAL  
STATE?

A GRAPH PROBLEM:

- MARK THE INITIAL STATE
- REPEAT UNTIL NO NEW STATES  
GET MARKED...
- MARK ANY STATE WHERE THERE  
IS A TRANSITION TO IT FROM  
A MARKED STATE
- CHECK TO SEE IF ANY FINAL  
STATE GOT MARKED.

## THEOREM

THE LANGUAGE

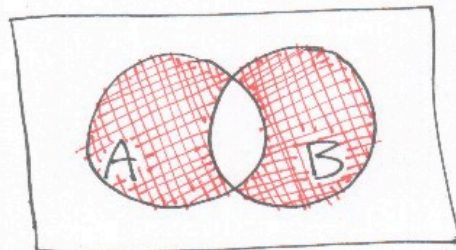
$$EQ_{DFA} = \{ \langle A, B \rangle \mid \left. \begin{array}{l} A \text{ and } B \text{ are} \\ \text{DFAs and} \\ L(A) = L(B) \end{array} \right\}$$

IS DECIDABLE.

## PROOF

Let  $C$  be the "SYMMETRIC DIFFERENCE" between  $A$  and  $B$ .

"ANYTHING IN  
A OR B BUT  
NOT BOTH."



$$C = (A \cap \bar{B}) \cup (\bar{A} \cap B)$$

NOTE

IF  $A = B$  THEN THE SYMMETRIC DIFFERENCE WILL BE  $\emptyset$ .

GIVEN...

A = DFA TO ACCEPT  $L(A)$

B = DFA TO ACCEPT  $L(B)$

... WE KNOW HOW TO COMBINE DFA'S.

$\overline{L(A)}$

$L(A) \cup L(B)$

$L(A) \cap L(B)$

---

BUILD DFA C TO ACCEPT THE SYMMETRIC DIFFERENCE.

---

USE THE TM FROM PREVIOUS

THEOREM [  $E_{DFA} \equiv$  empty language ]

TO TEST.

---

PROOF

CONSTRUCT A TM

INPUT:  $\langle A, B \rangle$  (2 DFA'S)

CONSTRUCT A DFA C TO ACCEPT

$$L(C) = (\overline{L(A)} \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$$

USE THE PREVIOUS TM TO TEST

WHETHER THE LANGUAGE THAT

C ACCEPTS IS EMPTY.  
ACCEPT IF SO; REJECT OTHERWISE.

12

## CONTEXT-FREE LANGUAGES

GIVEN A CFG, WILL IT GENERATE  
A GIVEN STRING,  $w$ ?

DECIDABLE!

GIVEN A CFG, IS THE LANGUAGE  
IT GENERATES EMPTY?

DECIDABLE!

GIVEN TWO ~~CFGs~~ CFGs, DO THEY  
ACCEPT THE SAME LANGUAGE?

NOT DECIDABLE!

IS A CFG AMBIGUOUS?

DO TWO CFGs HAVE ANY  
STRINGS THEY GENERATE  
IN COMMON?

IS THE COMPLEMENT OF A CFG  
ALSO A CFG?

NOT DECIDABLE!

## THEOREM

THE LANGUAGE

$$A_{CFG} = \left\{ \langle G, w \rangle \mid G \text{ is a CFG THAT GENERATES STRING } w. \right\}$$

IS DECIDABLE.

IN OTHER WORDS:

"GIVEN A CFG AND A STRING, WE CAN WRITE A PROGRAM THAT WILL ALWAYS HALT THAT WILL TELL US YES/NO WHETHER THE GRAMMAR GENERATES THE STRING."

ALL GRAMMARS ARE "PARSABLE".

SOME KINDS OF GRAMMARS

(eg. LL(k) OR LR(k) GRAMMARS)

CAN BE PARSED VERY EFFICIENTLY.

$O(N)$

BUT IN GENERAL, THE PARSER MAY

~~TAKE~~ TAKE  $O(N^3)$  TIME.

(WHERE  $N$  IS THE LENGTH OF THE STRING.)

PROOF

IDEA #1:

ENUMERATE ALL LEFTMOST DERIVATIONS.

FOR EACH, TEST TO SEE IF IT  
GENERATES  $w$ .

PROBLEM: WHAT IF  $w$  IS NOT  
IN THE LANGUAGE?

⇒ MAY NOT HALT!

**PROOF**

INPUTS:  $G = \text{A CFG.}$   
 $w = \text{A STRING.}$

STEP 1: CONVERT  $G$  INTO CHOMSKY NORMAL FORM.

**DERIVATIONS USING CNF GRAMMARS**

At each step, the length grows by exactly 1.

$S \rightarrow SS$   
 $S \rightarrow a$

$N-1$  steps

$S \Rightarrow SS \Rightarrow SSS \Rightarrow SSSS \Rightarrow SSSSS$

... Plus 1 additional step for each terminal symbol.

$\Rightarrow aSSSS \Rightarrow aaSSS \Rightarrow aaaSS \Rightarrow aaaaS \Rightarrow aaaaa$

$\therefore$  EVERY DERIVATION HAS EXACTLY  $2N-1$  steps.



Step 2:

Let  $N$  be the length of  $w$ .

List all derivations of  
length  $2N-1$ .

(There are only finitely many.)

CHECK EACH DERIVATION TO  
SEE IF IT GENERATES  $w$ .

IF ANY DERIVATION GENERATES  $w$ ,  
THEN ACCEPT.  
ELSE REJECT.

## THEOREM

THE LANGUAGE

$$E_{CFG} = \left\{ \langle G \rangle \mid \begin{array}{l} G \text{ is a CFG and} \\ L(G) = \emptyset \end{array} \right\}$$

IS DECIDABLE.

TO PROVE  $E_{CFG}$  IS DECIDABLE...

TO PROVE SOME PROBLEM  
IS DECIDABLE...

GIVE AN ALGORITHM (i.e., A T.M.)

VERIFY THE ALGORITHM IS CORRECT.

\* ALWAYS HALTS.

\* GIVES THE RIGHT ANSWER.

CONSIDER THIS GRAMMAR.

WHICH NONTERMINALS CAN  
GENERATE STRINGS OF TERMINALS?

S → ABCDE

A → BCA

A → abc

B → CA

B → AB

B → BBBd

C → CB

C → dd

E → EE

E → BE

E → EC

Underline  
in RED

Can generate a  
string:

A

C

B

## ALGORITHM

INPUT: A CFG " $G$ ".

A "MARKING" ALGORITHM:

MARK ALL TERMINAL SYMBOLS

REPEAT

LOOK FOR A RULE:

$A \rightarrow XXX$

WHERE ALL SYMBOLS ON RIGHT  
SIDE HAVE BEEN MARKED.

$B \rightarrow \underline{CA}$

MARK THE NONTERMINAL ON  
THE LEFT SIDE.

UNTIL NOTHING MORE CAN BE MARKED.

IF THE START SYMBOL IS MARKED

THEN ACCEPT

ELSE REJECT

DOES IT TERMINATE?  
IS IT CORRECT?

## THEOREM

THE LANGUAGE

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a Turing Machine} \\ \text{and } M \text{ accepts } w \}$$

IS TURING RECOGNIZABLE.

GIVEN THE DESCRIPTION OF A TURING MACHINE AND SOME INPUT, CAN WE DETERMINE WHETHER THE MACHINE ACCEPTS IT?

SURE!

JUST SIMULATE/RUN THE TM ON THE INPUT.

M ACCEPTS w

OUR ALGORITHM WILL HALT & ACCEPT

M REJECTS w

OUR ALGORITHM WILL HALT & REJECT.

M LOOPS ON w

OUR ALGORITHM WILL NOT HALT!

SO IT IS NOT A DECIDER!

## THE "UNIVERSAL" TURING MACHINE

INPUT:

$M$  = The description of some T.M.

$w$  = An input string for  $M$ .

ACTION:

SIMULATE  $M$ .

BEHAVE JUST LIKE  $M$  WOULD.

(MAY ACCEPT, REJECT, OR LOOP)

THE UNIVERSAL TURING MACHINE

IS A RECOGNIZER

(BUT NOT A DECIDER)

FOR

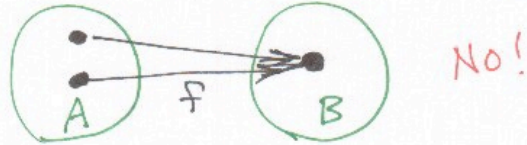
$$A_{TM} = \left\{ \langle M, w \rangle \mid \begin{array}{l} M \text{ is a TM and} \\ M \text{ accepts } w \end{array} \right\}$$

**DEFINITIONS**

Assume  $f: A \rightarrow B$

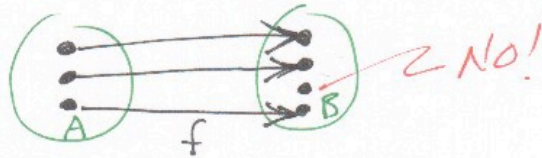
"ONE-TO-ONE"

If  $a \neq b$  then  $f(a) \neq f(b)$



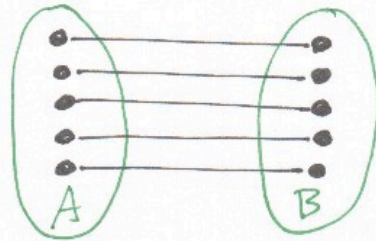
"ONTO"

EVERY ELEMENT IN  $B$  IS "HIT."



"CORRESPONDENCE"

ONE-TO-ONE AND ONTO



## INFINITY: COUNTABLE AND UNCOUNTABLE

GEORG CANTOR: "Two sets have the same size iff there exists a CORRESPONDENCE between them."

---

A set is "COUNTABLE" iff  
It has a finite size, or  
There is a CORRESPONDENCE  
with  $\mathbb{N}$

---

$$\mathbb{N} = \{1, 2, 3, \dots\}$$

---

$$\text{ODD NUMBERS} = \{1, 3, 5, \dots\}$$

COUNTABLY INFINITE

ALSO A SUBSET OF  $\mathbb{N}$

$$\text{Rational Numbers} = \left\{ \frac{m}{n} \mid m \text{ and } n \in \mathbb{N} \right\}$$

COUNTABLY INFINITE

Irrational Numbers / REAL NUMBERS

UNCOUNTABLY INFINITE



THE SET OF RATIONAL NUMBERS IS COUNTABLY INFINITE.

PROOF: FIND A WAY TO LIST THEM.  
MUST INCLUDE ALL OF THEM.

$$\text{RATIONALS:} = \left\{ \frac{m}{n} \mid m \text{ and } n \in \mathbb{N} \right\}$$

	$\frac{1}{7}$	$\frac{4}{3}$	$\frac{22}{29}$	$\frac{1}{2}$	$\frac{17}{3}$	$\frac{4}{2}$	...
CORRESPONDENCE	1	2	3	4	5	6	...

	1	2	3	4	5	6	7
1	$\frac{1}{1}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{1}{6}$	$\frac{1}{7}$
2	$\frac{2}{1}$	$\frac{2}{2}$	$\frac{2}{3}$	$\frac{2}{4}$	$\frac{2}{5}$	$\frac{2}{6}$	$\frac{2}{7}$
3	$\frac{3}{1}$	$\frac{3}{2}$	$\frac{3}{3}$	$\frac{3}{4}$	$\frac{3}{5}$	$\frac{3}{6}$	$\frac{3}{7}$
4	$\frac{4}{1}$	$\frac{4}{2}$	$\frac{4}{3}$	$\frac{4}{4}$	$\frac{4}{5}$	$\frac{4}{6}$	
5	$\frac{5}{1}$	$\frac{5}{2}$	$\frac{5}{3}$	$\frac{5}{4}$			
6	$\frac{6}{1}$	$\frac{6}{2}$	$\frac{6}{3}$				
7	$\frac{7}{1}$	$\frac{7}{2}$					

...

WE CAN ENUMERATE THE  
SET OF ALL TURING MACHINES.

APPROACH #1

- ⊗ EVERY TURING MACHINE CAN BE ENCODED INTO A STRING (OF FINITE LENGTH)
- ⊗ EVERY STRING IS EITHER A VALID TM REPRESENTATION OR NOT.
- ⊗ GENERATE ALL STRINGS, ONE AFTER THE OTHER.
- ⊗ CHECK TO SEE IF IT IS A VALID TM.

APPROACH #2

THE ALPHABETS ARE FINITE.  
THERE ARE A FINITE # OF KINDS  
OF TRANSITIONS

$a \rightarrow b, R$

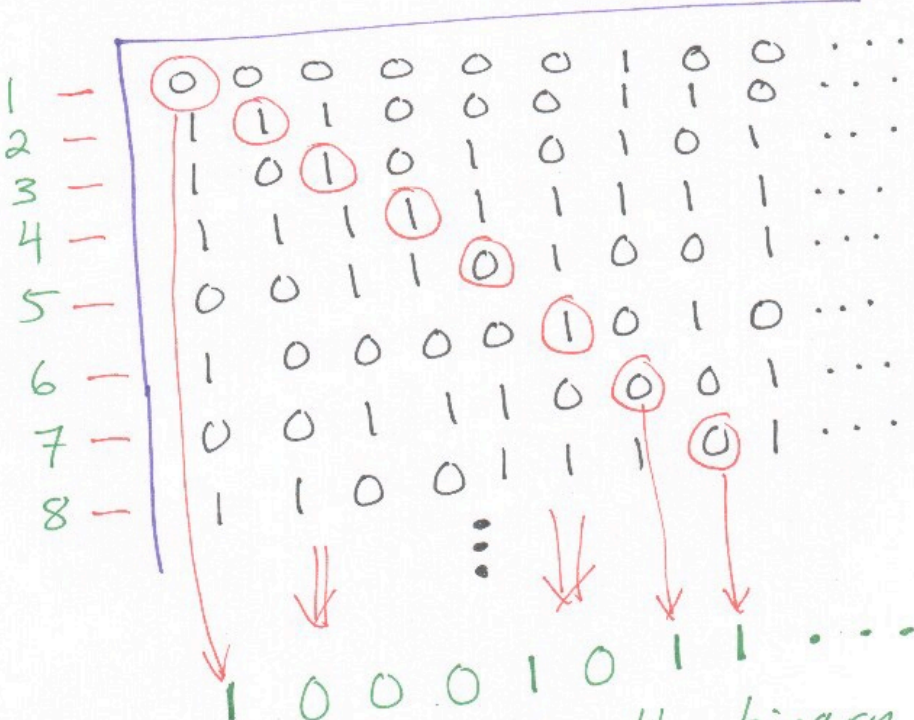
FOR  $i = 1, 2, 3, \dots$   
THERE ~~IS~~ <sup>IS</sup> A FINITE NUMBER  
OF DIRECTED GRAPHS ~~WITH~~  
WITH  $i$  NODES.  
THERE IS A FINITE NUMBER  
OF WAYS TO LABEL THE EDGES.  
GENERATE THEM ALL.  
END

**THEOREM**

THE SET OF ALL INFINITE LENGTH STRINGS OVER  $\{0, 1\}$  IS UNCOUNTABLY INFINITE.

PROOF (By DIAGONALIZATION METHOD)

Assume the set of ~~the~~ infinite binary strings can be enumerated. [i.e., correspondence with  $\mathbb{N}$ ]



is an infinite length binary string which is not equal to any string on this list!  $\square$

The set of all finite length strings over  $\Sigma = \{a, b\}$  is countable.

$\in$  a b ab ba aa bb aaa aab ...

A LANGUAGE contains some of those strings and not others.

$\in$  a b ab ba aa bb aaa aab ...

A LANGUAGE can be fully specified by giving an infinite length binary string.

1 0 1 1 0 0 1 1 0 ...

There are uncountably many infinite length binary strings.

**THEREFORE**

The number of languages is  
**UNCOUNTABLY INFINITE!**

### THEOREM

THE SET OF ALL TURING MACHINES  
IS COUNTABLY INFINITE.

### COROLLARY

THE SET OF ALL TURING-RECOGNIZABLE  
LANGUAGES IS COUNTABLY INFINITE.

### THEOREM

THE SET OF ALL LANGUAGES  
IS UNCOUNTABLY INFINITE.

### COROLLARY

SOME LANGUAGES ARE NOT  
TURING RECOGNIZABLE!

*A whole lot, too!*

## THE HALTING PROBLEM

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a Turing Machine and } M \text{ accepts } w. \}$$

THIS SET IS UNDECIDABLE.

### PROOF

Assume  $A_{TM}$  is decidable.

Let  $H$  be the algorithm/TM that decides  $A_{TM}$ .

$$H(\langle M, w \rangle) = \begin{cases} \text{ACCEPT, if } M \text{ accepts } w \\ \text{REJECT, if } M \text{ does not} \\ \text{accept } w \end{cases}$$

Doesn't really exist; We are headed toward a contradiction.

Using  $H$ , construct a new machine,  $D$ . "Devil Machine"

Try to run  $D$ .

Find a contradiction.

$\therefore H =$  A decider for Halting Problem can not exist.

## PROOF - MORE DETAIL

### HALTING DECIDER

$$H(\langle M, w \rangle) = \begin{cases} \text{ACCEPT, if } M \text{ accepts } w. \\ \text{REJECT, if } M \text{ does not} \\ \text{accept } w. \end{cases}$$

### DEVIL MACHINE

INPUT TO D:  $\langle M \rangle$  = The description of a TM,  $M$ .

ACTION:

RUN  $H$  AS A SUBROUTINE.

ASK WHETHER MACHINE  $M$  WOULD ACCEPT IF GIVEN A DESCRIPTION OF ITSELF AS INPUT.

OUTPUT: DO THE OPPOSITE.

$H$  ACCEPTS  $\Rightarrow$  REJECT

$H$  REJECTS  $\Rightarrow$  ACCEPT.

## RECAP OF D'S BEHAVIOR

Run  $H(\langle M, \langle M \rangle \rangle)$

GIVEN A MACHINE  $M$ , ASK WHETHER THAT MACHINE WOULD ACCEPT WHEN GIVEN A DESCRIPTION OF ITSELF AS INPUT.

$$D(\langle M \rangle) = \begin{cases} \text{ACCEPT, if } M \text{ does not accept } \langle M \rangle. \\ \text{REJECT, if } M \text{ accepts } \langle M \rangle. \end{cases}$$

Now Run  $D$  ON ITSELF!

$$D(\langle D \rangle) = \begin{cases} \text{ACCEPT, if } D \text{ does not accept } \langle D \rangle. \\ \text{REJECT, if } D \text{ accepts } \langle D \rangle. \end{cases}$$

**PARADOX!**

The next sentence is true.  
The previous sentence is false.



H ACCEPTS  $\langle M, w \rangle$  EXACTLY WHEN M ACCEPTS  $w$ .

D REJECTS  $\langle M \rangle$  EXACTLY WHEN M ACCEPTS  $\langle M \rangle$ .

D REJECTS  $\langle D \rangle$  EXACTLY WHEN D ACCEPTS  $\langle D \rangle$ .

"IF A LANGUAGE  $L$  IS DECIDABLE,  
THEN  $L$  IS TURING RECOGNIZABLE  
AND ITS COMPLEMENT  $\bar{L}$  IS  
TURING RECOGNIZABLE."

---

⊗ EVERY DECIDABLE LANGUAGE IS  
TURING RECOGNIZABLE.

⊗ WANT TO RECOGNIZE  $\bar{L}$ ?  
i.e., IS  $x$  IN  $\bar{L}$ ?

JUST RUN THE DECIDER FOR  $L$   
AND GIVE THE OPPOSITE ANSWER.

"IF A LANGUAGE  $L$  IS TURING  
RECOGNIZABLE AND ITS COMPLEMENT  
 $\bar{L}$  IS ALSO TURING RECOGNIZABLE,  
THEN  $L$  IS DECIDABLE."

---

WANT TO DECIDE  $L$ ?

i.e., Is  $x$  in  $L$ ?

Let  $M_1$  be the recognizer for  $L$ .

Let  $M_2$  be the recognizer for  $\bar{L}$ .

Run  $M_1$  and  $M_2$  in parallel.

$x$  is in either  $L$  or  $\bar{L}$ .

Either  $M_1$  or  $M_2$  (or both)  
will eventually halt and *one of*  
*them will accept.*

IF  $M_1$  ACCEPTS, THEN ACCEPT.

IF  $M_2$  ACCEPTS, THEN REJECT.

Either way, you'll eventually  
halt with a decision.

### THEOREM

A LANGUAGE  $L$  IS DECIDABLE  
IFF  $L$  AND  $\overline{L}$  ARE  
TURING RECOGNIZABLE.

### DEFINITION

A LANGUAGE IS "CO-TURING RECOGNIZABLE"  
IF ITS COMPLEMENT IS  
TURING RECOGNIZABLE.

### RESTATING THE THEOREM

A LANGUAGE IS DECIDABLE  
IFF IT IS TURING RECOGNIZABLE  
AND CO-TURING RECOGNIZABLE.

RECALL:

$A_{TM}$  IS TURING RECOGNIZABLE.

$A_{TM}$  IS NOT DECIDABLE.

THEREFORE:

$\overline{A_{TM}}$  IS NOT TURING RECOGNIZABLE.

**PROOF**

Assume  $\overline{A_{TM}}$  is Turing Recognizable.

If a Language and its complement are both Turing Recognizable, then the language is decidable.

But we know  $A_{TM}$  is not decidable.