DECIDABILITY - OVERVIEW

- EVERY QUESTION ABOUT REGULAR LANGUAGES IS DECIDABLE.
- SOME QUESTIONS ABOUT CFLs ARE DECIDABLE, BUT SOME ARE NOT.
- THE "HALTING PROBLEM" IS NOT DECIDABLE.
- SOME LANGUAGES ARE NOT TURING RECOGNIZABLE.
- MANY QUESTIONS ABOUT TURING MACHINES ARE NOT DECIDABLE, AND SOME ARE NOT EVEN TURING RECOGNIZABLE.

THE "HALTING PROBLEM"

GIVEN A PROGRAM ...
WILL IT HALT?

GIVEN A TURING MACHINE, WILL IT HALT WHEN RUN ON SOME PARTICULAR GIVEN INPUT STRING?

GIVEN SOME PROGRAM WRITTEN IN [JAVA/C/any other language], WILL IT EVER GET INTO AN INFINITE LOOP? OR WILL IT ALWAYS TERMINATE?

ANSWER

IN GENERAL, WE CAN'T ALWAYS KNOW!

THE BEST WE CAN DO IS RUN THE PROGRAM AND SEE WHETHER IT HALTS... BUT...

FOR MANY PROGRAMS, WE CAN PROVE "IT WILL ALWAYS HALT." [OR "IT MAY SOMETIMES LOOP."]

BUT FOR PROGRAMS IN GENERAL, THE QUESTION IS **UNDECIDABLE**.

PROBLEM:

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

IS THIS PROBLEM DECIDABLE?

<u>LANGUAGES</u> 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...

$$A_{DFA} = \left\{ \langle B, w \rangle \;\middle|\; \begin{array}{l} B \text{ is a DFA that} \\ \text{accepts string } w \end{array} \right\}$$

...is decidable.

Possible Confusion

String

w

Language

DFA ≡ regular language

String

$\langle B, w \rangle$

Language

A_{DFA}

Not regular, not CFG, but it is decidable

PROOF THAT

$$A_{DFA} = \left\{ \langle B, w \rangle \;\middle|\; \begin{array}{l} B \text{ is a DFA that} \\ \text{accepts string } w \end{array} \right\}$$

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 RERRESENTATION.

THE TM THEN SIMULATES B
ON w.

IF B REACHES ~~[illegible]~~ 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...

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

... IS DECIDABLE.

PROOF

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

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 A_{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...

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

...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_{REX}.

AND

THAT ALGORITHM / TM WILL ALWAYS HALT.

HALTING?

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

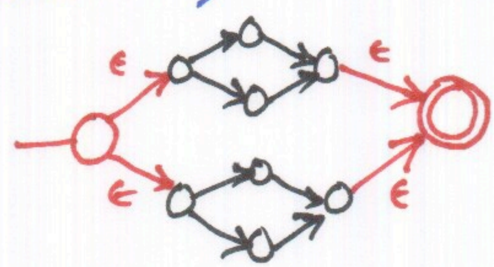
MANY PROGRAMS CAN BE PROVEN TO ALWAYS HALT.
BUT THIS DOES NOT MEAN THE HALTING PROBLEM IS DECIDABLE!

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

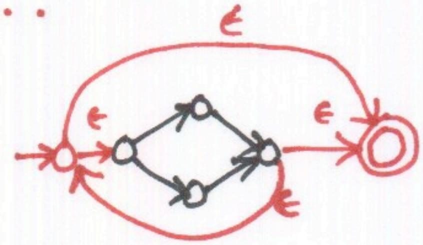
$\boxed{\text{ALGORITHM / TM}}$

STEP 1: CONVERT R INTO A NFA, B'.

(RECALL THE ALGORITHM)

$X \mid Y \Rightarrow$ 

$X \circ Y \Rightarrow \ldots$

$X^* \Rightarrow$ 

STEP 2: CONSTRUCT THE STRING

$$\langle B', w \rangle$$

STEP 3: USE THE TM FROM THE LAST THEOREM TO DECIDE WHETHER THIS IS IN

$$A_{NFA} = \left\{ \langle B, w \rangle \;\middle|\; \begin{array}{l} B \text{ is a NFA} \\ \text{that accepts } w \end{array} \right\}$$

$\boxed{\text{ACCEPTANCE TESTING}}$

IS STRING w IN THE LANGUAGE?

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

$\boxed{\text{EMPTINESS TESTING}}$

IS THE LANGUAGE EMPTY?

$L = \emptyset$?

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

$\boxed{\text{EQUALITY TESTING}}$

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.

$\Rightarrow$ 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 MARED.

THEOREM

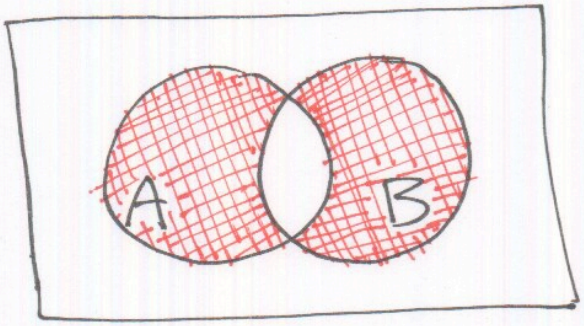
THE LANGUAGE

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

IS DECIDABLE.

PROOF

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



"ANYTHING IN A OR B BUT NOT BOTH."

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

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} ≡ empty language] TO TEST.

PROOF

CONSTRUCT A TM

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

CONSTRUCT A DFA C TO ACCEPT

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

USE THE PREVIOUS TM TO TEST WHETHER THE LANGUAGE THAT C ACCEPTS IS EMPTY.

ACCEPT IF SO; REJECT OTHERWISE.

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, 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

$$A_{CFG} = \left\{ \langle G, w \rangle \;\middle|\; \begin{array}{l} G \text{ is a CFG THAT} \\ \text{GENERATES STRING } w. \end{array} \right\}$$

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 (e.g. LL(k) OR LR(k) GRAMMARS) CAN BE PARSED VERY EFFICIENTLY. $O(N)$

BUT IN GENERAL, THE PARSER MAY TAKE $O(N^3)$ TIME.

(Where N IS THE LENGTH OF THE STRING.)

PROOF

IDEA #1:

ENUMERATE <u>ALL</u> LEFTMOST DERIVATIONS.

FOR EACH, TEST TO SEE IF IT GENERATES w.

PROBLEM: WHAT IF w IS NOT IN THE LANGUAGE?

$\Rightarrow$ MAY NOT HALT!

PROOF

INPUTS: G = A CFG.
w = A STRING.

STEP 1: CONVERT G INTO CHOMSKY NORMAL FORM.

DERIVATIONS USING CNF GRAMMARS

At each step, the length grows by exactly 1.

$$S \to SS$$
$$S \to a$$

N-1 steps

$$S \underset{1}{\Rightarrow} SS \underset{2}{\Rightarrow} SSS \underset{3}{\Rightarrow} SSSS \underset{4}{\Rightarrow} \underbrace{SSSSS}_{5}$$

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

$$\underset{1}{\Rightarrow} aSSSS \underset{2}{\Rightarrow} aaSSS \underset{3}{\Rightarrow} aaaSS \underset{4}{\Rightarrow} aaaaS$$

N steps

$$\underset{5}{\Rightarrow} aaaaa$$

$\therefore$ 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

$$E_{CFG} = \left\{ \langle G \rangle \;\middle|\; \begin{array}{c} G \text{ is a CFG and} \\ L(G) = \emptyset \end{array} \right\}$$

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 A STRING OF TERMINALS?

$S \to ABCD$

$A \to BCA$

$A \to xyz$

$B \to CA$

$B \to AB$

$B \to BBBw$

$C \to CB$

$C \to ww$

$D \to DD$

$D \to BD$

$D \to DC$

ALGORITHM

INPUT: A CFG "G".

A "MARKING" ALGORITHM:

MARK ALL TERMINAL SYMBOLS

REPEAT

LOOK FOR A RULE:

$$A \rightarrow XXXX$$

WHERE ALL SYMBOLS ON RIGHT SIDE HAVE BEEN MARED.

$$B \rightarrow \underline{C}\,\underline{A}$$

MARKS THE NONTERMINAL ON THE LEFT SIDE.

UNTIL NOTHING MORE CAN BE MARKED.

IF THE START SYMBOL IS NOT MARKED

THEN ACCEPT $(L(G) = \emptyset)$

ELSE REJECT $(L(G) \neq \emptyset)$

DOES IT TERMINATE?
IS IT CORRECT?

THEOREM

THE LANGUAGE

$$A_{TM} = \left\{ \langle M, w \rangle \;\middle|\; \begin{array}{l} M \text{ is a Turing Machine} \\ \text{and } M \text{ accepts } w \end{array} \right\}$$

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

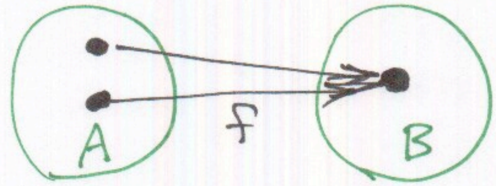
$$A_{TM} = \left\{ \langle M, w \rangle \;\middle|\; \begin{array}{c} M \text{ is a TM and} \\ M \text{ accepts } w \end{array} \right\}$$

DEFINITIONS

Assume $f: A \to B$

"ONE-TO-ONE"

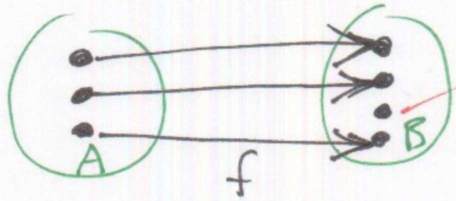
If $a \neq b$ then $f(a) \neq f(b)$



NO!

"ONTO"

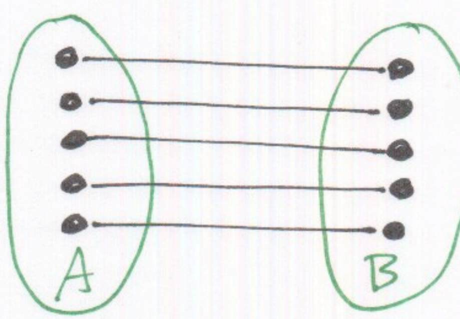
EVERY ELEMENT IN B IS "HIT."



NO!

"CORRESPONDENCE"

ONE-TO-ONE <u>AND</u> 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 $\mathbb{N}$

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

ODD NUMBERS $= \{1, 3, 5, \cdots\}$

COUNTABLY INFINITE

ALSO A <u>SUBSET</u> OF $\mathbb{N}$

Rational Numbers $= \{M/N \mid M \text{ and } N \in \mathbb{N}\}$

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 <u>ALL</u> OF THEM.

RATIONALS: $= \left\{ M/N \;\middle|\; M \text{ and } N \in \mathbb{N} \right\}$

	$1/7$	$4/3$	$22/29$	$1/2$	$17/3$	$4/2$	...
CORRESPONDENCE	|	|	|	|	|	|	
	1	2	3	4	5	6	...

	1	2	3	4	5	6	7
1	$1/1$	$1/2$	$1/3$	$1/4$	$1/5$	$1/6$	$1/7$
2	$2/1$	$2/2$	$2/3$	$2/4$	$2/5$	$2/6$	$2/7$
3	$3/1$	$3/2$	$3/3$	$3/4$	$3/5$	$3/6$	$3/7$
4	$4/1$	$4/2$	$4/3$	$4/4$	$4/5$	$4/6$	
5	$5/1$	$5/2$	$5/3$	$5/4$			
6	$6/1$	$6/2$	$6/3$				
7	$7/1$	$7/2$					

...

The set of irrational numbers is UNCOUNTABLY INFINITE!

$$\pi = 3.14159265 \cdots$$
$$\sqrt{2} = 1.4142135 \cdots$$
$$e = 2.718281828 \cdots$$
$$= 5.67932043 \cdots$$

$$1/3 = .333{,}333{,}33\bar{3} < .333{,}333{,}5{,}71{,}29 \cdots$$
$$.333{,}334{,}00\bar{0}$$

PROOF: Assume it is COUNTABLY INFINITE.

1	.3	1	4	1	5	9	...
2	.1	4	1	4	2	1	...
3	.2	7	1	8	2	8	...
4	.5	6	7	9	3	2	...
5	.7	4	2	5	3	1	...
6	.3	9	2	4	5	0	...

.4 5 2 8 4 1 ...

This Number is NOT in the table!

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 RERESENTATION 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

$$\xrightarrow{a \to b, R}$$

<u>FOR</u> $i = 1, 2, 3, \ldots$

THERE IS A FINITE NUMBER OF DIRECTED GRAPHS WITH i NODES.

THERE IS A FINITE NUMBER OF WAYS TO LABEL THE EDGES.

GENERATE THEM ALL.

<u>END</u>

THEOREM

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

PROOF (By DIAGONALIZATION METHOD)

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

1 —	0	0	0	0	0	0	1	0	0	...
2 —	1	1	1	0	0	0	1	1	0	...
3 —	1	0	1	0	1	0	1	0	1	...
4 —	1	1	1	1	1	1	1	1	1	...
5 —	0	0	1	1	0	1	0	0	1	...
6 —	1	0	0	0	0	1	0	1	0	...
7 —	0	0	1	1	1	0	0	0	1	...
8 —	1	1	0	0	1	1	1	0	1	...

⋮

1 0 0 0 1 0 1 1 ...

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

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

ϵ a b ab ba aa bb aaa aab ...

A LANGUAGE contains some of those strings and not others.

ϵ 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

$$A_{TM} = \left\{ \langle M, w \rangle \;\middle|\; \begin{array}{l} M \text{ is a Turing Machine} \\ \text{and } M \text{ accepts } w. \end{array} \right\}$$

THIS SET IS UNDECIDABLE.

PROOF

Assume A_{TM} is decidable.

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

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

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

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

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 $\overline{L}$ IS TURING RECOGNIZABLE."

- EVERY DECIDABLE LANGUAGE IS TURING RECOGNIZABLE.
- WANT TO RECOGNIZE $\overline{L}$?
  i.e., IS x IN $\overline{L}$?
  JUST RUN THE DECIDER FOR L AND GIVE THE OPPOSITE ANSWER.

"IF A LANGUAGE L IS TURING RECOGNIZABLE AND ITS COMPLEMENT $\overline{L}$ 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 $\overline{L}$.

Run M_1 and M_2 in parallel.

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

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.