Computing Theory

COMP 147 (4 units)

Chapter 4: Decidability

Section 4.2: Undecidable languages

Last Time: Decidable Problems

Acceptance Tests:

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

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

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

$A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}.$

Emptiness Tests:

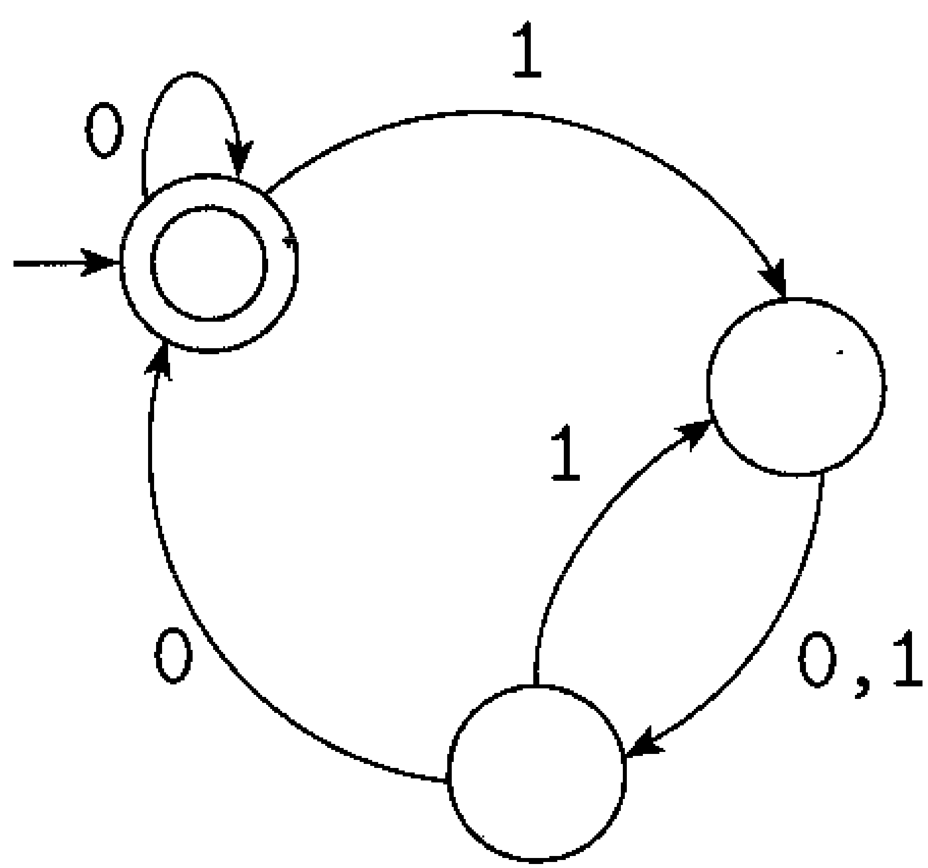
$E_{\text{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$

$E_{\text{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}.$

Equivalence Tests:

$EQ_{\text{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}.$

Answer all parts for the following DFA M and give reasons for your answers.



a. Is $\langle M, 0100\rangle \in A_{DFA}$? ✓

b. Is $\langle M, 011\rangle \in A_{DFA}$? ✗

c. Is $\langle M\rangle \in A_{DFA}$? ✗ Input invalid

d. Is $\langle M, 0100\rangle \in A_{REX}$? ✗ Input invalid

e. Is $\langle M\rangle \in E_{DFA}$? ✗ it accepts strings

f. Is $\langle M, M\rangle \in EQ_{DFA}$? ✓

Let $A\varepsilon_{CFG}$ = { <G> | G is a CFG that generates ε }.
Show that $A\varepsilon_{CFG}$ is decidable.

We can construct the following Turing Machine M

M: on input <G> where G is a CFG
1) Run TM S for A_{CFG} on input <G, ε> where S decides A_{CFG}
2) If S accepts, *accept* if S rejects *reject*.

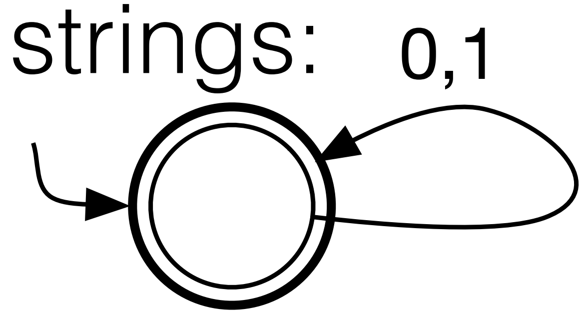
Show that the following language is decidable

$$ALL_{DFA} = \{< A > | A\ is\ a\ DFA\ and\ L(A) = \Sigma^*\}$$

- Solution1: Use TM that decides for EQ_{DFA}

M: on input <A> where A is a DFA

1. Construct DFA B that accepts all strings:



2. Run TM S for EQ_{DFA} on input <A, B> where S decides EQ_{DFA}
3. If S accepts, *accept* if S rejects *reject*.

Show that the following language is decidable

$$ALL_{DFA} = \{< A > | A \ is \ a \ DFA \ and \ L(A) = \Sigma^*\}$$

- Solution2: Use TM that decides for E_{DFA}

M: on input <A> where A is a DFA

1. Run TM S for E_{DFA} on input <A> where S decides E_{DFA}
2. If S accepts, *reject* if S rejects *accepts*.

Is this a correct solution? What does S rejecting tell you

Show that the following language is decidable

$$ALL_{DFA} = \{< A > | A \ is \ a \ DFA \ and \ L(A) = \Sigma^*\}$$

- Correct Solution2: Use TM that decides for E_{DFA}

M: on input <A> where A is a DFA

1. Construct DFA B that accepts the complement of L(A)
2. Run TM S for E_{DFA} on input <B> where S decides E_{DFA}
3. If S accepts, *accepts* if S rejects *reject*.

Countable and Uncountable Sets

How to count elements in a set?

Let A be a set:

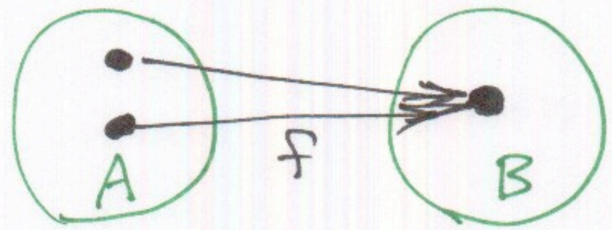
- If A is finite ==> counting is trivial
- If A is infinite ==> how do we count?
- And, how do we compare two infinite sets by their size?

DEFINITIONS

Assume $f : A \to B$

"ONE-TO-ONE"

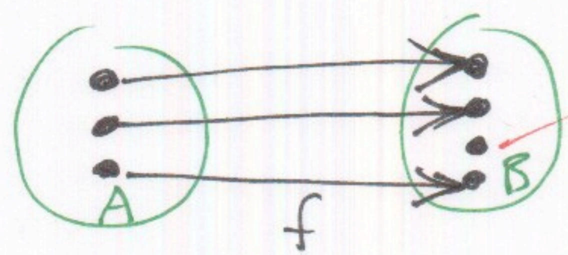
If $a \neq b$ then $f(a) \neq f(b)$



NO!

"ONTO"

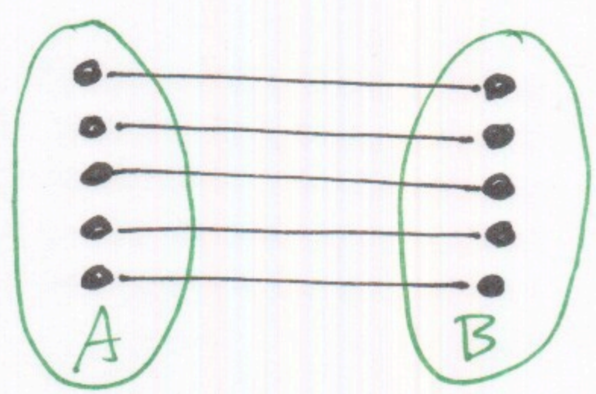
EVERY ELEMENT IN B IS "HIT."



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 $\mathbb{N}$

Cantor's definition of set "size" for infinite sets (1873 A.D.)

Let N = {1,2,3,…} (all natural numbers)

Let E = {2,4,6,…} (all even numbers)

Q) Which is bigger?

- A) Both sets are of the same size
  - "***Countably infinite***"
  - Proof: Show by one-to-one, onto set correspondence from N ==> E

n	*f(n)*
1	2
2	4
3	6
.	.
.	.

i.e, for every element in N, there is a unique element in E, and vice versa.

Example #2

- Let Q be the set of all rational numbers
- Q = { m/n | for all m,n ∈ N }
- Claim: Q is also countably infinite; => |Q|=|N|

1/1	1/2	1/3	1/4	1/5	….
2/1	2/2	2/3	2/4	2/5	….
3/1	3/2	3/3	3/4	3/5	….
4/1	4/2	4/3	4/4	4/5	….
5/1	5/2		….		

Really, really big sets!
(even bigger than countably infinite sets)

Uncountable sets

Example:

- Let R be the set of all real numbers
- Claim: R is uncountable

The set of irrational numbers is UNCOUNTABLY INFINITE!

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

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

Really, really big sets!
(even bigger than countably infinite sets)

Uncountable sets

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!

Paradox

- The barber in this town is the "one who shaves all those, and those only, who do not shave themselves."

- The question is, does the barber shave himself?

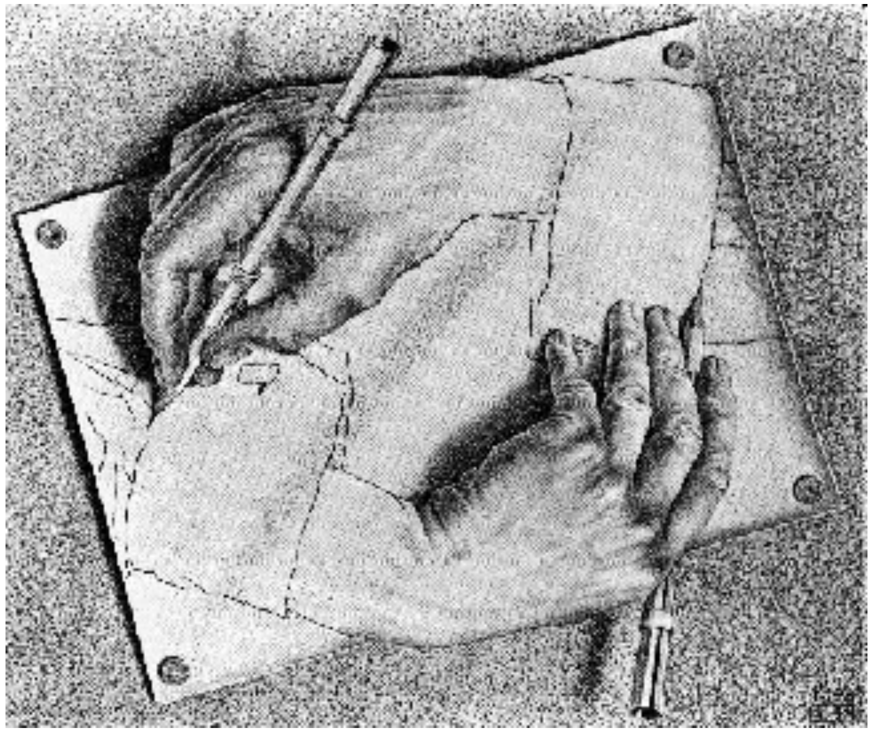
- Answering this question results in a contradiction.

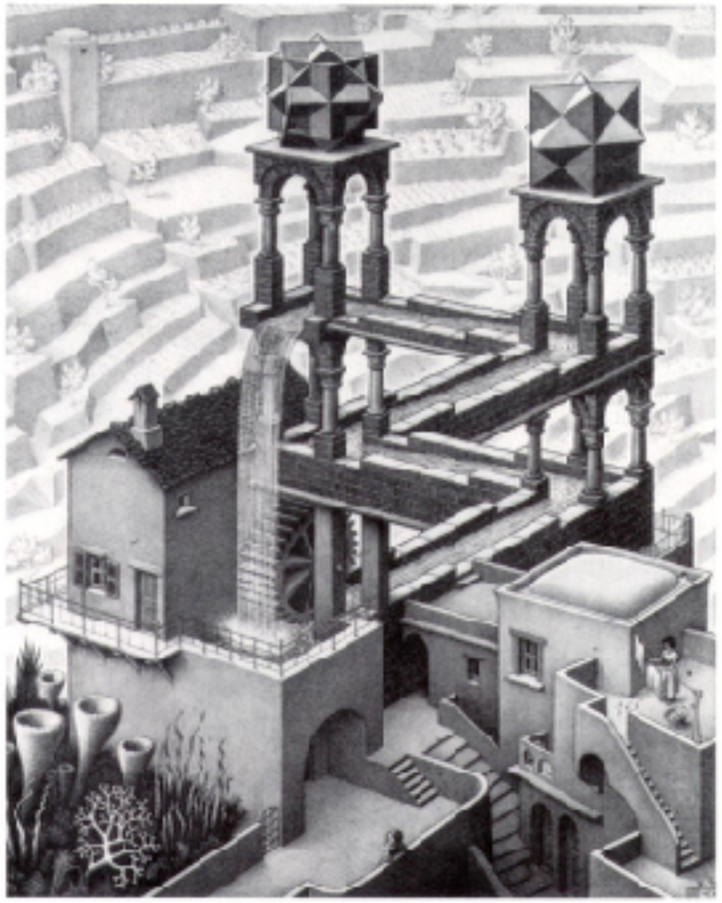
    - The barber cannot shave himself as he only shaves those who do not shave themselves. As such, if he shaves himself he ceases to be a barber.

    - Conversely, if the barber does not shave himself, then he fits into the group of people who would be shaved by the barber, and thus, as the barber, he must shave himself.

Of Paradoxes & Strange Loops

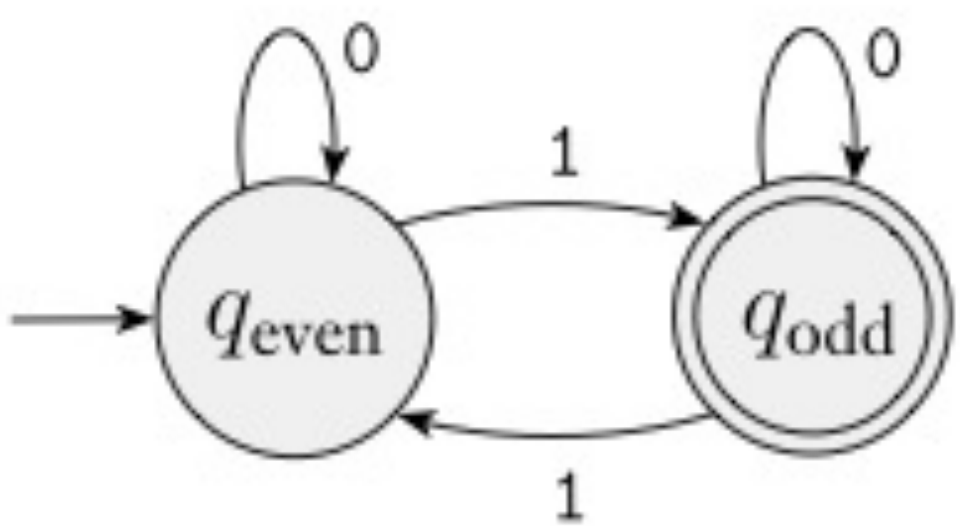
E.g., Barber's paradox, Achilles & the Tortoise (Zeno's paradox)
MC Escher's paintings





A "fun" book for further reading:
"Godel, Escher, Bach: An Eternal Golden Braid"
by Douglas Hofstadter (Pulitzer winner, 1980)

FSM encoded as binary strings



FSM encode as binary: 0010100100100 1100

does this machine accept itself?

L= {<A> | A is FSM that does not accept itself}

Barber paradox again

- is there a FSM that accepts exactly all the binary strings that represent finite state machines that do not accept themselves?
- Contradiction just like the barber paradox
- What does this machine do when it takes itself as input? Contradiction!!

Binary-Strings from TM's

- We shall restrict ourselves to TM's with input alphabet {0, 1}.
- Assign positive integers to the three classes of elements involved in moves:
  - States: q_1(start state), q_2 (final state), q_3, …
  - Symbols X_1 (0), X_2 (1), X_3 (blank), X_4, …
  - Directions D_1 (L) and D_2 (R).

Binary Strings from TM's – (2)

- Suppose $\delta(q_i, X_j) = (q_k, X_l, D_m)$.
- Represent this rule by string $0^i10^j10^k10^l10^m$.
- Key point: since integers i, j, … are all > 0, there cannot be two consecutive 1's in these strings.

Binary Strings from TM's – (2)

- Represent a TM by concatenating the codes for each of its moves, separated by 11 as punctuation.
  - That is: $Code_1 11 Code_2 11 Code_3 11 \ldots$

An Undecidable Problem

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

- A_{TM} Turing-recognizable, but not Turing-decidable

- Build a Turing Machine U_{TM} that can simulate M on w

TM U_{TM}

- M accepts w, then *accept*
- M rejects w, the *reject*
- M loops on w, then our machine will not halt

so not a decider

The Universal Turing Machine

- Input:
  - M the description of some TM
  - w an input string for M
- Action:
  - Simulate M
  - Behave like M would (accept reject or loop)

U_{TM} recognizes (but does not decide) A_{TM}

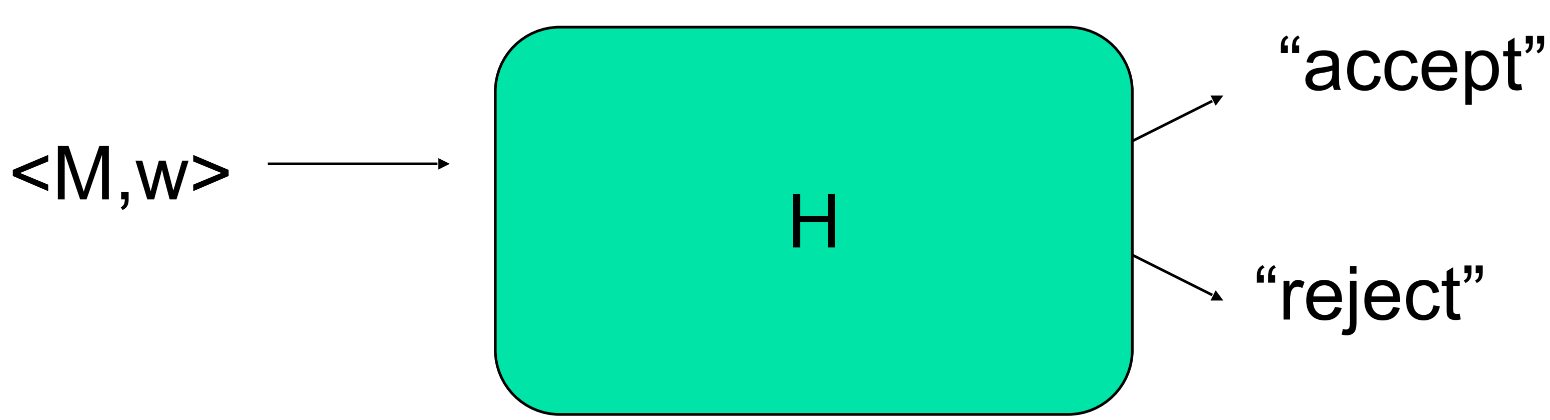
A Claim

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

- A_{TM} is undecidable

- Claim: No TM decider H that is always guaranteed to halt, can exist!

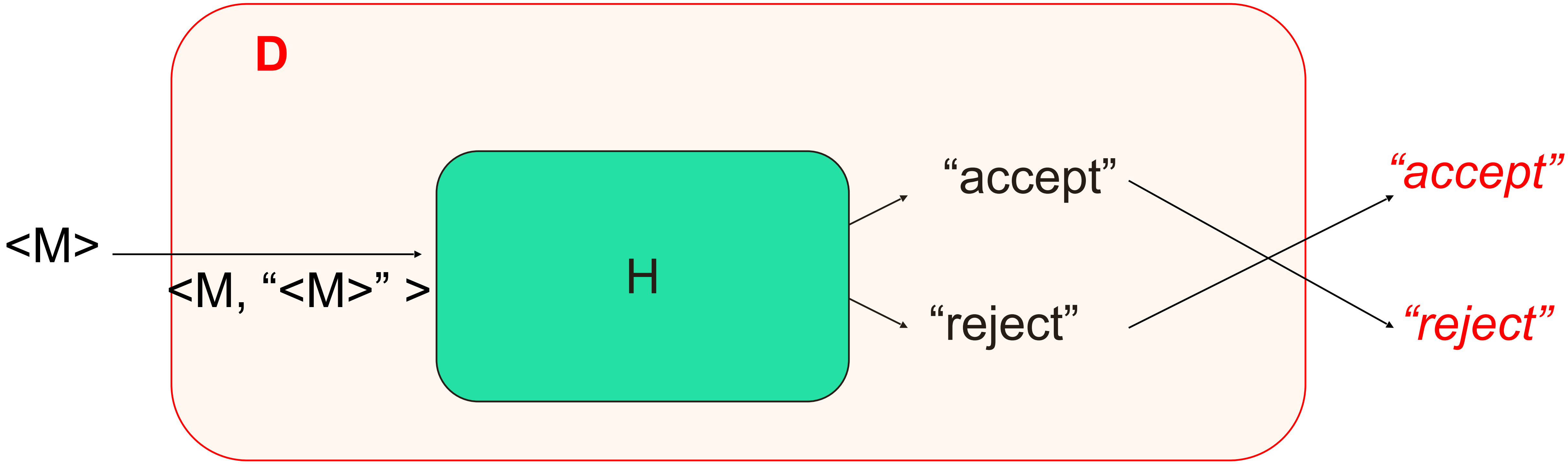
- Proof: (Alan Turing, 1936)
  - By contradiction, let us assume H exists



Therefore, if H exists ➔ D also should exist.
<u>But can such a D exist?</u> (if not, then H also cannot exist)

HP Proof (step 1)

- Let us construct a new TM **D** using H as a subroutine:
  - On input <M>:
    1. Run H on input <M, <M> >; //(i.e., run M on M itself)
    2. Output the ***opposite*** of what H outputs;



HP Proof (step 2)

- The notion of inputing “<M>” to M itself
  - A program can be input to itself (e.g., a compiler is a program that takes any program as input)

$$D(\langle M\rangle) = \begin{cases} \textit{accept}, & \text{if M does } \textit{not} \text{ accept } \langle M\rangle \\ \textit{reject}, & \text{if M accepts } \langle M\rangle \end{cases}$$

Now, what happens if D is input to itself?

$$D(\langle D\rangle) = \begin{cases} \textit{accept}, & \text{if D does not accept } \langle D\rangle \\ \textit{reject}, & \text{if D accepts } \langle D\rangle \end{cases}$$

A contradiction!!! ==> Neither D nor H can exist

Therefore, some languages cannot have TMs…

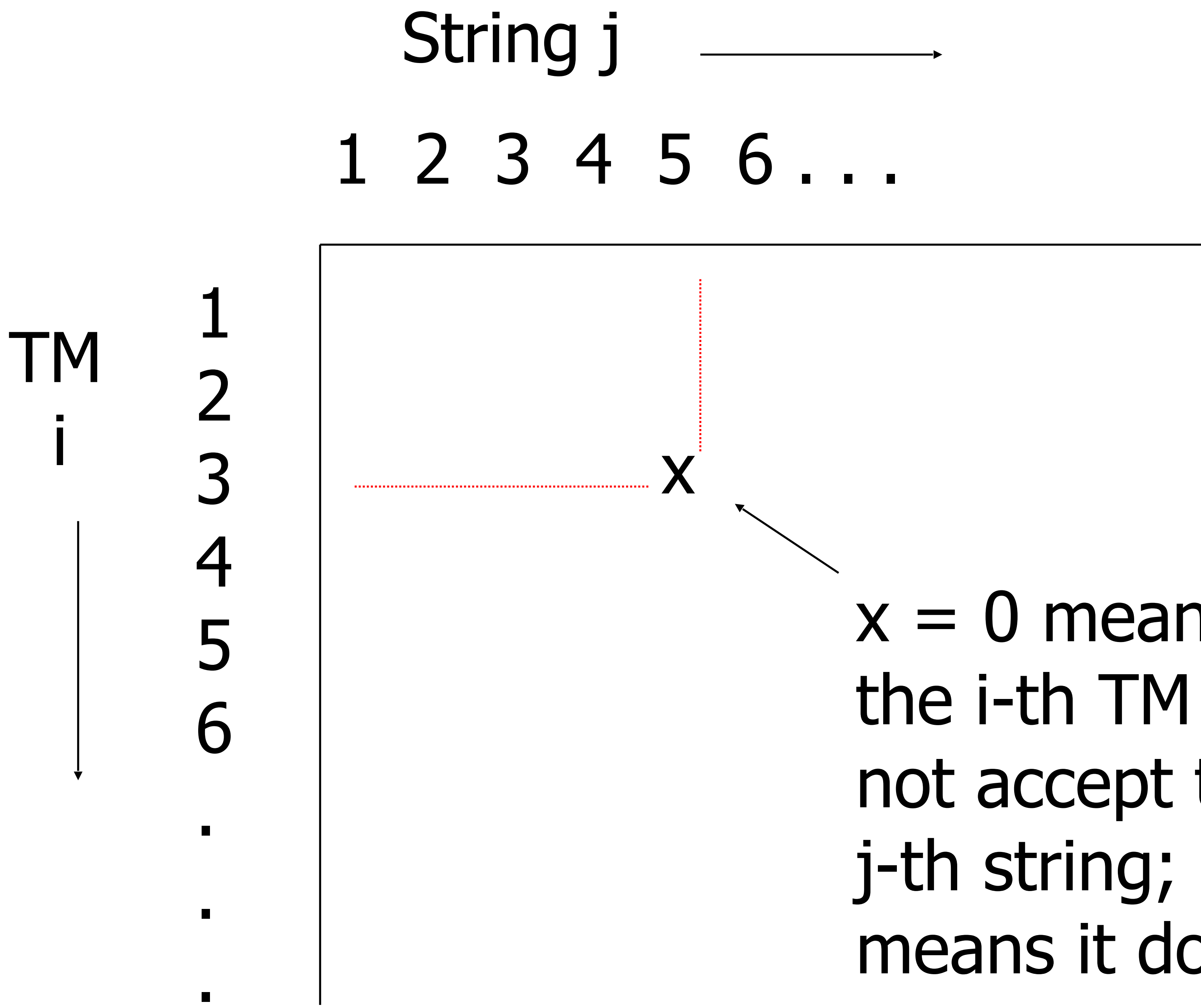
- The set of all TMs is countable
- The set of all Languages is uncountable
- ==> There should be some languages without TMs (by Pigeon Hole Principle)

The Diagonalization Language

Example of a language that is
<u>not Turing Recognizable</u>

(i.e, no TMs exist)

Table of Acceptance



x = 0 means the i-th TM does not accept the j-th string; 1 means it does.

Diagonalization Again

- Whenever we have a table like the one on the previous slide, we can diagonalize it.
  - That is, construct a sequence D by complementing each bit along the major diagonal.
- Formally, $D = a_1a_2\ldots$, where $a_i = 0$ if the (i, i) table entry is 1, and vice-versa.

The Diagonalization Language

- $L_d = \{ w_i \mid w_i \notin L(M_i) \}$
  - The language of all strings whose corresponding machine does ***not*** accept itself (i.e., its own code)

(input word w)

j →

(TMs)

i ↓

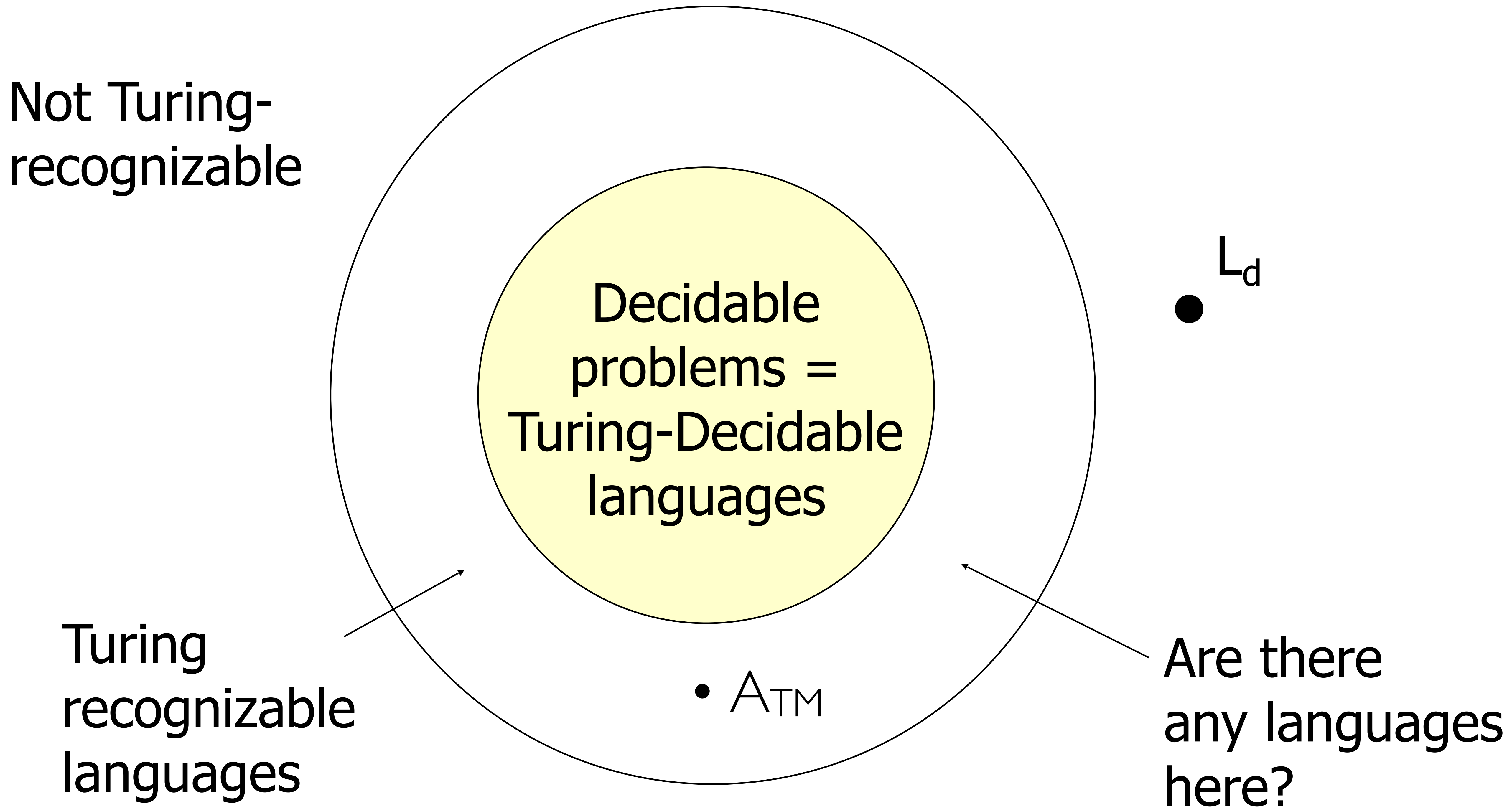
	1	2	3	4	…
1	0	1	0	1	…
2	1	1	0	0	…
3	0	1	0	1	…
4	1	0	0	1	…
⋮					⋱

diagonal

- <u>Table:</u> $T[i,j] = 1$, if M_i accepts w_j
  $= 0$, otherwise.

- Make a new language called
  $L_d = \{w_i \mid T[i,i] = 0\}$

Bullseye Picture



Important (but Obvious) Theorem

THEOREM 4.22

A language is decidable iff it is Turing-recognizable and co-Turing-recognizable.

In other words, a language is decidable exactly when both it and its complement are Turing-recognizable.

A Language that is not Turing-Recognizable

$A_{\mathsf{TM}} = \{\langle M, w\rangle |\ M \text{ is a TM and } M \text{ accepts } w\}$

A_{TM} is Turing-Recognizable, but not decidable.

A language is decidable iff it is Turing-recognizable and co-Turing-recognizable.

COROLLARY 4.23 ....

$\overline{A_{\mathsf{TM}}}$ is not Turing-recognizable.