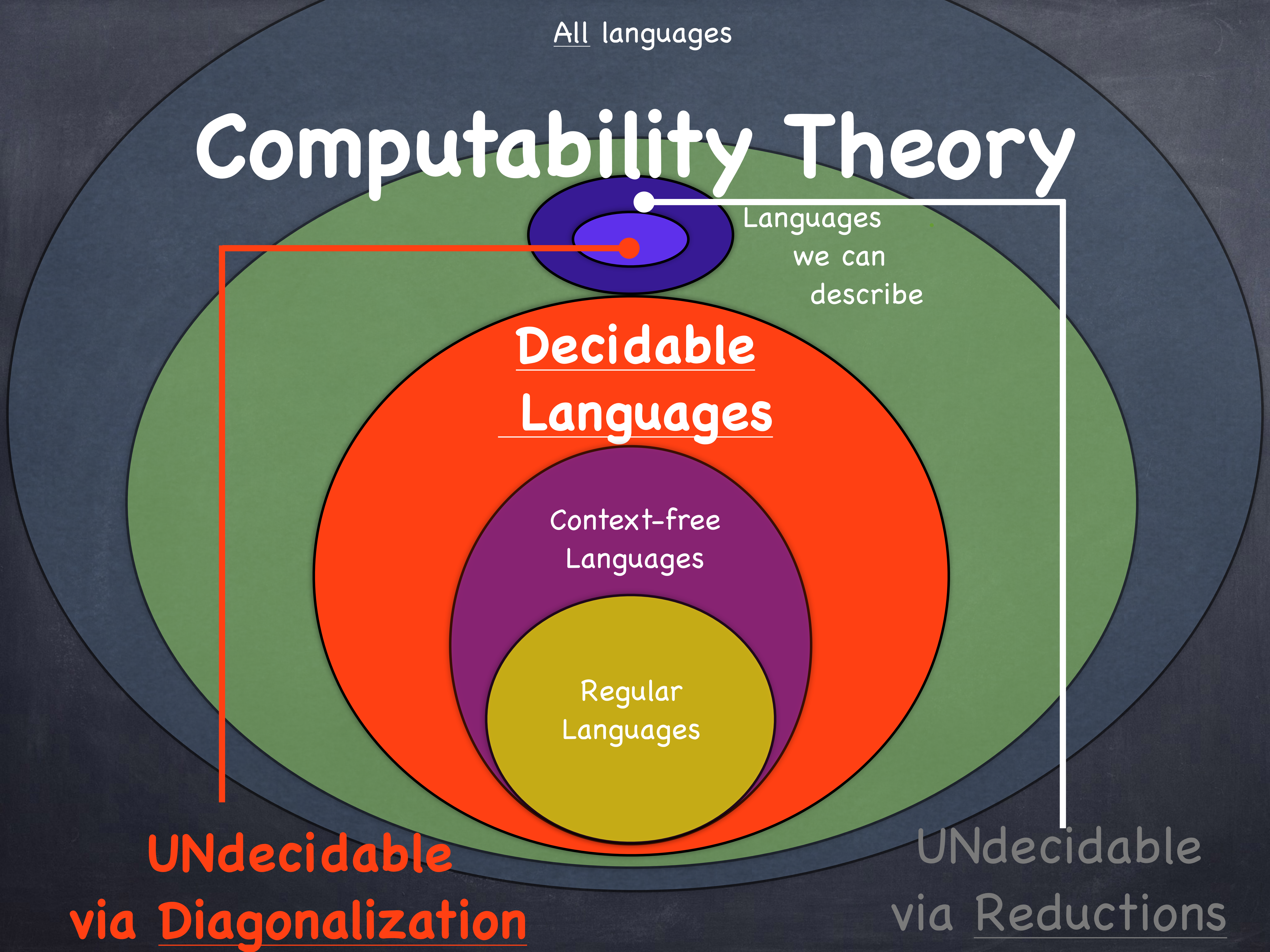
COMP-330
Theory of Computation

Fall 2019 -- Prof. Claude Crépeau

Lec. 18-19 : Turing (UN)Decidability



All languages
Computability Theory
Languages we can describe
Decidable Languages
Context-free Languages
Regular Languages
UNdecidable via Diagonalization
UNdecidable via Reductions

Turing Decidability

Alan Turing

Format & Notations

- Represent objects as strings
- $\langle O_1, O_2, \ldots, O_k \rangle$ is the string representing objects $O_1, O_2, \ldots, O_k$
- Many encodings are possible.
- Implicitly, at beginning of an algorithm, check that input is in the correct format, otherwise reject.

Format & Notations

EXAMPLE 3.23

Let A be the language consisting of all strings representing undirected graphs that are connected. Recall that a graph is ***connected*** if every node can be reached from every other node by traveling along the edges of the graph. We write

$$A = \{\langle G \rangle |\ G \text{ is a connected undirected graph}\}.$$

The following is a high-level description of a TM M that decides A.

Format & Notations

$M =$ "On input $\langle G \rangle$, the encoding of a graph G:

1. Select the first node of G and mark it.

2. Repeat the following stage until no new nodes are marked:

3. For each node in G, mark it if it is attached by an edge to a node that is already marked.

4. Scan all the nodes of G to determine whether they all are marked. If they are, *accept*; otherwise, *reject*."

Decidable Languages about DFA

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

THEOREM 4.1

A_{DFA} is a decidable language.

Decidable Languages about DFA

PROOF IDEA We simply need to present a TM M that decides A_{DFA}.

$M =$ "On input $\langle B, w \rangle$, where B is a DFA and w is a string:

1. Simulate B on input w.
2. If the simulation ends in an accept state, *accept*. If it ends in a nonaccepting state, *reject*."

Decidable Languages about DFA

We can prove a similar theorem for nondeterministic finite automata. Let

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

THEOREM 4.2

A_{NFA} is a decidable language.

Decidable Languages about DFA

$N =$ "On input $\langle B, w\rangle$ where B is an NFA, and w is a string:

1. Convert NFA B to an equivalent DFA C, using the procedure for this conversion given in Theorem 1.39.
2. Run TM M from Theorem 4.1 on input $\langle C, w\rangle$.
3. If M accepts, *accept*; otherwise, *reject*."

Decidable Languages about DFA

Similarly, we can determine whether a regular expression generates a given string. Let $A_{\mathsf{REX}} = \{\langle R, w\rangle \mid R \text{ is a regular expression that generates string } w\}$.

THEOREM 4.3

A_{REX} is a decidable language.

Decidable Languages about DFA

PROOF The following TM P decides A_{REX}.

$P =$ "On input $\langle R, w \rangle$ where R is a regular expression and w is a string:

1. Convert regular expression R to an equivalent NFA A by using the procedure for this conversion given in Theorem 1.54.
2. Run TM N on input $\langle A, w \rangle$.
3. If N accepts, *accept*; if N rejects, *reject*."

Decidable Languages about DFA

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

THEOREM 4.4 ....

E_{DFA} is a decidable language.

Decidable Languages

PROOF A DFA accepts some string iff reaching an accept state from the start state by traveling along the arrows of the DFA is possible. To test this condition we can design a TM T that uses a marking algorithm similar to that used in Example 3.23.

T = "On input $\langle A \rangle$ where A is a DFA:

1. Mark the start state of A.
2. Repeat until no new states get marked:
3. Mark any state that has a transition coming into it from any state that is already marked.
4. If no accept state is marked, *accept*; otherwise, *reject*."

Decidable Languages about DFA

The next theorem states that determining whether two DFAs recognize the same language is decidable. Let

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

THEOREM 4.5

EQ_{DFA} is a decidable language.

PROOF To prove this theorem we use Theorem 4.4. We construct a new DFA C from A and B, where C accepts only those strings that are accepted by either A or B but not by both. Thus, if A and B recognize the same language, C will accept nothing. The language of C is

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

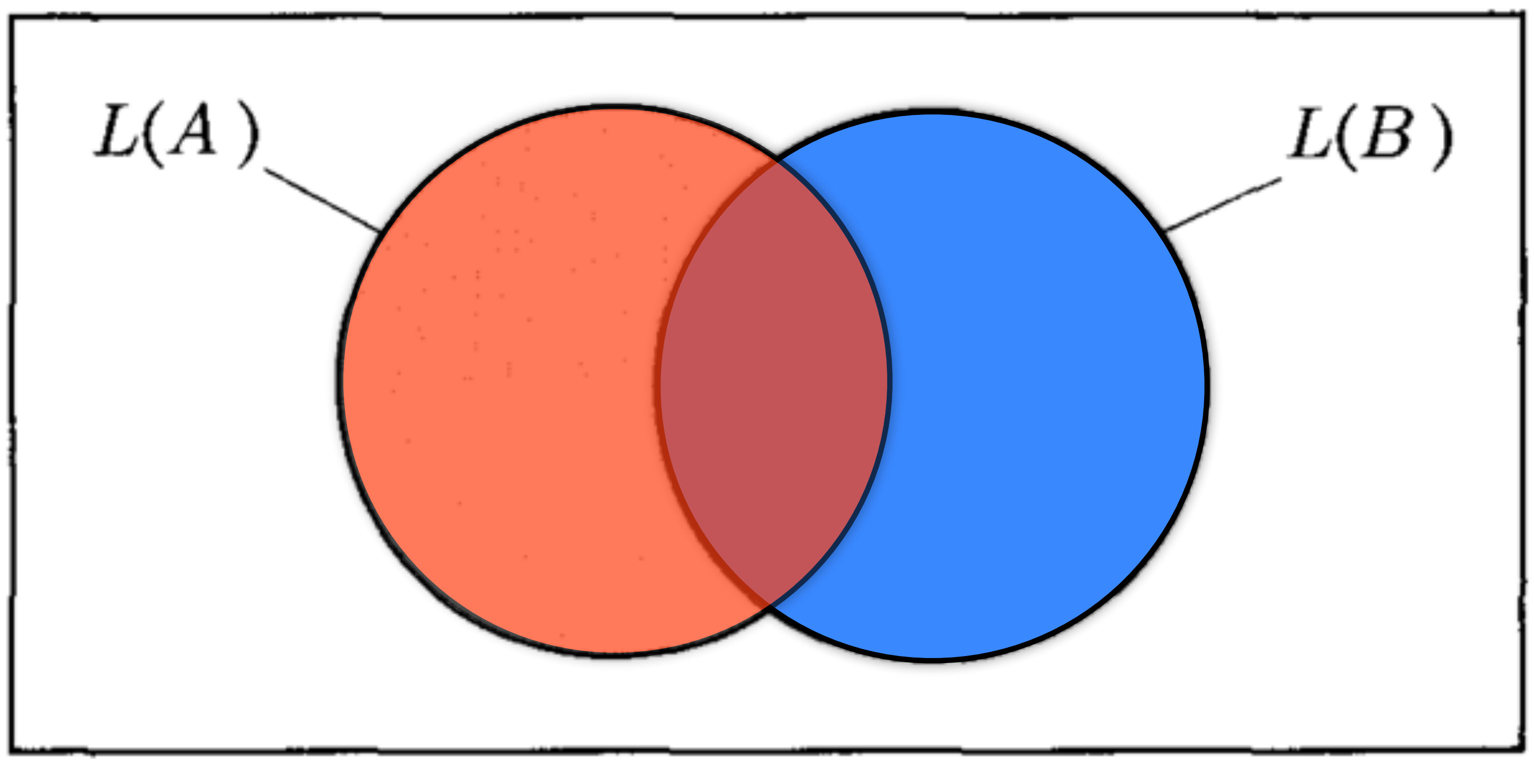


FIGURE 4.6
The symmetric difference of $L(A)$ and $L(B)$

Decidable Languages about DFA

Once we have constructed C we can use Theorem 4.4 to test whether $L(C)$ is empty. If it is empty, $L(A)$ and $L(B)$ must be equal.

$F =$ "On input $\langle A, B\rangle$, where A and B are DFAs:

1. Construct DFA C as described.
2. Run TM T from Theorem 4.4 on input $\langle C\rangle$.
3. If T accepts, *accept*. If T rejects, *reject*."

Decidable Languages about CFG

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

THEOREM 4.7 ....

A_{CFG} is a decidable language.

Decidable Languages about CFG

PROOF The TM S for A_{CFG} follows.

$S =$ "On input $\langle G, w\rangle$, where G is a CFG and w is a string:

1. Convert G to an equivalent grammar in Chomsky normal form.
2. List all derivations with $2n - 1$ steps, where n is the length of w, except if $n = 0$, then instead list all derivations with 1 step.
3. If any of these derivations generate w, *accept*; if not, *reject*."

Decidable Languages about CFG

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

THEOREM 4.8 ....

E_{CFG} is a decidable language.

Decidable Languages about CFG

PROOF

$R =$ "On input $\langle G \rangle$, where G is a CFG:

1. Mark all terminal symbols in G.
2. Repeat until no new variables get marked:
3. Mark any variable A where G has a rule $A \rightarrow U_1 U_2 \cdots U_k$ and each symbol $U_1, \ldots, U_k$ has already been marked.
4. If the start variable is not marked, *accept*; otherwise, *reject*."

Decidable Languages about CFG

THEOREM 4.9

Every context-free language is decidable.

Decidable Languages about CFG

PROOF Let G be a CFG for A and design a TM M_G that decides A. We build a copy of G into M_G. It works as follows.

$M_G =$ "On input w:

1. Run TM S on input $\langle G, w \rangle$
2. If this machine accepts, *accept*; if it rejects, *reject*."

Decidable Languages

Decidable	Undecidable
A_{DFA}	EQ_{CFG}
A_{NFA}	A_{TM}
A_{REX}	$HALT_{TM}$
E_{DFA}	E_{TM}
EQ_{DFA}	$REGULAR_{TM}$
A_{CFG}	EQ_{TM}
E_{CFG}	PCP

Undecidable Languages about CFG

Next we consider the problem of determining whether two context-free grammars generate the same language. Let

$$EQ_{\mathsf{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}.$$

Undecidable Languages about TM

A_{DFA} and A_{CFG} were decidable, A_{TM} is not. Let

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

THEOREM 4.11

A_{TM} is undecidable.

Undecidable Languages about TM

A_{DFA} and A_{CFG} were decidable, A_{TM} is not. Let

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

THEOREM 4.11 ....

A_{TM} is undecidable.

A_{TM} is Turing-recognizable.

$U =$ "On input $\langle M, w\rangle$, where M is a TM and w is a string:

1. Simulate M on input w.
2. If M ever enters its accept state, *accept*; if M ever enters its reject state, *reject*."

Undecidable Language about TM

THE ACCEPTANCE PROBLEM IS UNDECIDABLE

Now we are ready to prove Theorem 4.11, the undecidability of the language

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

Undecidable Language about TM

Assumption: H exists

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

Undecidable Language about TM

H exists ⇒ D exists

$D =$ "On input $\langle M \rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M \rangle \rangle$.

2. Output the opposite of what H outputs; that is, if H accepts, *reject* and if H rejects, *accept*."

Undecidable Language about TM

Properties of D

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

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

Undecidable Language about TM

Properties of D

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

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

CONTRADICTION

Undecidable Language about TM

- 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 $\langle D \rangle$.

Undecidable Language about TM

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

- 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 $\langle D\rangle$.

Undecidable Language about TM

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

- 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 $\langle D\rangle$.

$D =$ "On input $\langle M\rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M\rangle\rangle$.

2. Output the opposite of what H outputs; that is, if H accepts, *reject* and if H rejects, *accept*."

Undecidable Language about TM

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

- 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 $\langle D \rangle$.

CONTRADICTION

$D =$ "On input $\langle M \rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M \rangle \rangle$.
2. Output the opposite of what H outputs; that is, if H accepts, *reject* and if H rejects, *accept*."

Undecidable Language about TM

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

- 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 $\langle D\rangle$.

CONTRADICTION

CONTRADICTION

$D =$ "On input $\langle M\rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. Output the opposite of what H outputs; that is, if H accepts, *reject* and if H rejects, *accept*."

Undecidable Language about TM

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

CONTRADICTION
CONTRADICTION
CONTRADICTION

- 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 $\langle D\rangle$.

$D =$ "On input $\langle M\rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. Output the opposite of what H outputs; that is, if H accepts, *reject* and if H rejects, *accept*."

Undecidable Language about TM

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*		*accept*		
M_2	*accept*	*accept*	*accept*	*accept*	
M_3					$\cdots$
M_4	*accept*	*accept*			
$\vdots$		$\vdots$			

FIGURE 4.19
Entry i, j is *accept* if M_i accepts $\langle M_j \rangle$

Undecidable Language about TM

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*	
M_4	*accept*	*accept*	*reject*	*reject*	
$\vdots$		$\vdots$			

FIGURE 4.20
Entry i, j is the value of H on input $\langle M_i, \langle M_j\rangle\rangle$

Undecidable Language about TM

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*		*accept*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*		*reject*	
M_4	*accept*	*accept*	*reject*	*reject*		*accept*	
$\vdots$		$\vdots$			$\ddots$		
D	*reject*	*reject*	*accept*	*accept*		?	
$\vdots$		$\vdots$					$\ddots$

FIGURE 4.21
If D is in the **table**, a contradiction occurs at "?"

Diagonalization

Decidable	Undecidable
A_{DFA}	EQ_{CFG}
A_{NFA}	**A_{TM}**
A_{REX}	$HALT_{TM}$
E_{DFA}	E_{TM}
EQ_{DFA}	$REGULAR_{TM}$
A_{CFG}	EQ_{TM}
E_{CFG}	PCP

Unrecognizable Language about TM

THEOREM 4.22

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

Unrecognizable Language about TM

THEOREM 4.22 ....

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

Let M_1 and M_2 be TMs respectively recognizing L and its complement $\bar{L}$.

Unrecognizable Language about TM

THEOREM 4.22

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

Let M_1 and M_2 be TMs respectively recognizing L and its complement $\bar{L}$.

M = "On input w:
1. Run both M_1 and M_2 on input w in parallel.
2. If M_1 accepts, *accept*; if M_2 accepts, *reject*."

Unrecognizable Language about TM

COROLLARY 4.23

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

PROOF We know that A_{TM} is Turing-recognizable. If $\overline{A_{TM}}$ also were Turing-recognizable, A_{TM} would be decidable. Theorem 4.11 tells us that A_{TM} is not decidable, so $\overline{A_{TM}}$ must not be Turing-recognizable.

COMP-330
Theory of Computation

Fall 2019 -- Prof. Claude Crépeau

Lec. 18-19 : Turing (UN)Decidability