Foundations of Computing

Lecture 15

Arkady Yerukhimovich

March 7, 2024

Outline

Lecture 14 Review

- Decidable and Turing-recognizable languages
- Decidability of regular and context-free languages

Outline

Characterizing Computability of Languages

Definition: Decidable languages

A language L is *decidable* or *recursive* if some TM M decides it

- M halts on ALL inputs, accepts those in L and rejects those not in L
- Seems to match informal definition we wanted before

Definition: Turing-recognizable languages

A language L is *Turing-recognizable* or *recursively enumerable* if some TM M recognizes it

- M halts and accepts all strings in L
- M may not halt on strings not in L – does not necessarily have to reject

Observation

Every Decidable language is also Turing-recognizable, but the reverse direction is not true.

Decidable Languages

We showed the following languages are decidable:

- Languages about Finite Automata
  1. $A_{DFA} = \{\langle B, w\rangle \mid B$ is a DFA that accepts input string $w\}$
  2. $A_{NFA} = \{\langle B, w\rangle \mid B$ is a NFA that accepts input string $w\}$
  3. $A_{REX} = \{\langle R, w\rangle \mid R$ is a reg. exp. that generates the string $w\}$
  4. $E_{DFA} = \{\langle A\rangle \mid A$ is a DFA and $L(A) = \emptyset\}$
  5. $EQ_{DFA} = \{\langle A, B\rangle \mid A, B$ are DFAs and $L(A) = L(B)\}$
- Languages about CFGs
  1. $A_{CFG} = \{\langle G, w\rangle \mid G$ is a CFG that generates $w\}$
  2. $E_{CFG} = \{\langle G\rangle \mid G$ is a CFG and $L(G) = \emptyset\}$

A Language About Turing Machines

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M(w) = 1\}$$

A Language About Turing Machines

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M(w) = 1\}$$

- Observation: A_{TM} is Turing-recognizable
  On input $\langle M, w\rangle$:
  1. Simulate M on input w
  2. If M ever enters its accept state, halt and accept. If M ever enters its reject state, halt and reject

A Language About Turing Machines

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M(w) = 1\}$$

- Observation: A_{TM} is Turing-recognizable
  On input $\langle M, w\rangle$:
  1. Simulate M on input w
  2. If M ever enters its accept state, halt and accept. If M ever enters its reject state, halt and reject
- Is A_{TM} Decidable?
  - The problem: M may never halt
  - In this case, above algorithm will never output accept or reject
  - If could determine that M will never halt (i.e, it has entered an infinite loop), could reject.

A Language About Turing Machines

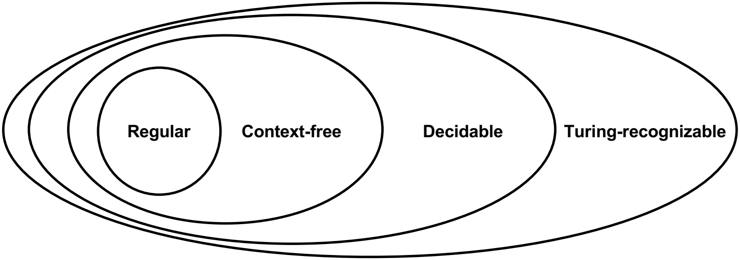
$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M(w) = 1\}$$

- Observation: A_{TM} is Turing-recognizable
  On input $\langle M, w\rangle$:
  1. Simulate M on input w
  2. If M ever enters its accept state, halt and accept. If M ever enters its reject state, halt and reject
- Is A_{TM} Decidable?
  - The problem: M may never halt
  - In this case, above algorithm will never output accept or reject
  - If could determine that M will never halt (i.e, it has entered an infinite loop), could reject.

An Undecidable Problem

- We will prove today that A_{TM} is undecidable

Relationships Among Language Classes



Outline

Set Cardinality

- Cardinality of a set S is the number of elements in that set ($|S|$)

Set Cardinality

- Cardinality of a set S is the number of elements in that set ($|S|$)
- A set S can be finite ($|S| < \infty$) or infinite ($|S| = \infty$)

Set Cardinality

- Cardinality of a set S is the number of elements in that set ($|S|$)
- A set S can be finite ($|S| < \infty$) or infinite ($|S| = \infty$)
- $|S_1| = |S_2|$ if there's a one-to-one and onto mapping from S_1 to S_2

Set Cardinality

- Cardinality of a set S is the number of elements in that set ($|S|$)
- A set S can be finite ($|S| < \infty$) or infinite ($|S| = \infty$)
- $|S_1| = |S_2|$ if there's a one-to-one and onto mapping from S_1 to S_2
- Example:
  $A = \{0, 1, 2, 3\}$
  $B = \{a, b, c, d\}$
  $f(0) = a, f(1) = b, f(2) = c, f(3) = d$

Countable and Uncountable (Infinite) Sets

Intuition: Countable sets are ones where we can arrange elements into a "first element", "second element", and so on.

Countable and Uncountable (Infinite) Sets

Intuition: Countable sets are ones where we can arrange elements into a "first element", "second element", and so on.

- An infinite set A is *countably infinite* if it has the same cardinality as the natural numbers: $\mathcal{N} = 1, 2, 3, \ldots$

Countable and Uncountable (Infinite) Sets

Intuition: Countable sets are ones where we can arrange elements into a "first element", "second element", and so on.

- An infinite set A is *countably infinite* if it has the same cardinality as the natural numbers: $\mathcal{N} = 1, 2, 3, \ldots$
- A set A is countable if it is finite or countably infinite

Countable and Uncountable (Infinite) Sets

Intuition: Countable sets are ones where we can arrange elements into a "first element", "second element", and so on.

- An infinite set A is *countably infinite* if it has the same cardinality as the natural numbers: $\mathcal{N} = 1, 2, 3, \ldots$
- A set A is countable if it is finite or countably infinite
- A set that is not countable is *uncountable*

Example 1: Evens

Evens

The set of even numbers is

Example 1: Evens

Evens

The set of even numbers iscountable

Example 2: Rationals

Rationals

The set of rational numbers is

Example 2: Rationals

Rationals

The set of rational numbers is countable

Example 2: Rationals

Rationals

The set of rational numbers is countable

	1	2	3	$\cdots$
1	1/1	1/2	1/3	
2	2/1	2/2	2/3	$\cdots$
3	3/1	3/2	3/3	
4	4/1	4/2	$\vdots$	

Example 3: Strings

Strings

The set of strings in $\{0, 1\}^*$ is

Example 3: Strings

Strings

The set of strings in $\{0, 1\}^*$ is countable

Diagonalization

Real Numbers

The set of real numbers ($\mathcal{R}$) is uncountable

Diagonalization

Real Numbers

The set of real numbers ($\mathcal{R}$) is uncountable

Proof: By diagonalization

Diagonalization

Real Numbers

The set of real numbers ($\mathcal{R}$) is uncountable

Proof: By diagonalization

- Assume that $\mathcal{R}$ is countable

Diagonalization

Real Numbers

The set of real numbers ($\mathcal{R}$) is uncountable

Proof: By diagonalization

- Assume that $\mathcal{R}$ is countable
- Then there is a one-to-one and onto mapping f from $\mathcal{N}$ to $\mathcal{R}$

Diagonalization

Real Numbers

The set of real numbers ($\mathcal{R}$) is uncountable

Proof: By diagonalization

- Assume that $\mathcal{R}$ is countable
- Then there is a one-to-one and onto mapping f from $\mathcal{N}$ to $\mathcal{R}$

n	f(n)
1	1.234...
2	3.141...
3	5.556...
$\vdots$	$\vdots$

Diagonalization

Real Numbers

The set of real numbers ($\mathcal{R}$) is uncountable

Proof: By diagonalization

- Assume that $\mathcal{R}$ is countable
- Then there is a one-to-one and onto mapping f from $\mathcal{N}$ to $\mathcal{R}$

n	f(n)
1	1.234...
2	3.141...
3	5.556...
$\vdots$	$\vdots$

- We construct a value $x \in \mathcal{R}$ s.t $x \neq f(n)$ for any n
  Idea: For all $i \in \mathcal{N}$, make $x_i \neq f(i)_i$

Diagonalization

Real Numbers

The set of real numbers ($\mathcal{R}$) is uncountable

Proof: By diagonalization

- Assume that $\mathcal{R}$ is countable
- Then there is a one-to-one and onto mapping f from $\mathcal{N}$ to $\mathcal{R}$

n	f(n)
1	1.234...
2	3.141...
3	5.556...
$\vdots$	$\vdots$

- We construct a value $x \in \mathcal{R}$ s.t $x \neq f(n)$ for any n
  Idea: For all $i \in \mathcal{N}$, make $x_i \neq f(i)_i$
- Contradiction – f is not mapping between $\mathcal{R}$ and $\mathcal{N}$

Outline

The Set of Turing Machines

Turing Machines

The set of all Turing Machines is

The Set of Turing Machines

Turing Machines

The set of all Turing Machines is countable

The Set of Turing Machines

Turing Machines

The set of all Turing Machines is countable

- We already showed that the set of strings $\{0, 1\}^*$ is countable

The Set of Turing Machines

Turing Machines

The set of all Turing Machines is countable

- We already showed that the set of strings $\{0,1\}^*$ is countable
- Can similarly show that for any finite alphabet Σ, Σ^* is countable

The Set of Turing Machines

Turing Machines

The set of all Turing Machines is countable

- We already showed that the set of strings $\{0,1\}^*$ is countable
- Can similarly show that for any finite alphabet Σ, Σ^* is countable
- But, a TM M can be written as a string $\langle M \rangle \in \Sigma^*$

The Set of Turing Machines

Turing Machines

The set of all Turing Machines is countable

- We already showed that the set of strings $\{0,1\}^*$ is countable
- Can similarly show that for any finite alphabet Σ, Σ^* is countable
- But, a TM M can be written as a string $\langle M \rangle \in \Sigma^*$
- Hence, by omitting all strings that are not encodings of valid TMs we get a mapping of TMs to $\mathcal{N}$

The Set of Languages

Languages over alphabet Σ

$\mathcal{L}$ – the set of all languages over alphabet Σ is

The Set of Languages

Languages over alphabet Σ

$\mathcal{L}$ – the set of all languages over alphabet Σ is uncountable

The Set of Languages

Languages over alphabet Σ

$\mathcal{L}$ – the set of all languages over alphabet Σ is uncountable

Proof:

1. Consider the set B of infinite binary sequences
    - An infinite binary sequence is an infinite length string of 0's and 1's

The Set of Languages

Languages over alphabet Σ

$\mathcal{L}$ – the set of all languages over alphabet Σ is uncountable

Proof:

1. Consider the set B of infinite binary sequences
   - An infinite binary sequence is an infinite length string of 0's and 1's
   - B is uncountable

The Set of Languages

Languages over alphabet Σ

$\mathcal{L}$ – the set of all languages over alphabet Σ is uncountable

Proof:

1. Consider the set B of infinite binary sequences
   - An infinite binary sequence is an infinite length string of 0's and 1's
   - B is uncountable
2. $|\mathcal{L}| = |B|$

The Set of Languages

Languages over alphabet Σ

$\mathcal{L}$ – the set of all languages over alphabet Σ is uncountable

Proof:

1. Consider the set B of infinite binary sequences
   - An infinite binary sequence is an infinite length string of 0's and 1's
   - B is uncountable
2. $|\mathcal{L}| = |B|$
   - Define the characteristic sequence χ_A of language $A \in \mathcal{L}$

$$
\begin{array}{rclccccccccl}
\Sigma^* & = & \{ & \epsilon & 0 & 1 & 00 & 01 & 11 & 000 & \cdots & \} \\
A & = & \{ & & & 1 & 00 & & & 000 & \cdots & \} \\
\chi_A & = & & 0 & 0 & 1 & 1 & 0 & 0 & 1 & \cdots &
\end{array}
$$

The Set of Languages

Languages over alphabet Σ

$\mathcal{L}$ – the set of all languages over alphabet Σ is uncountable

Proof:

1. Consider the set B of infinite binary sequences
   - An infinite binary sequence is an infinite length string of 0's and 1's
   - B is uncountable
2. $|\mathcal{L}| = |B|$
   - Define the characteristic sequence χ_A of language $A \in \mathcal{L}$

$$
\begin{array}{rclcccccccccl}
\Sigma^* & = & \{ & \epsilon & 0 & 1 & 00 & 01 & 11 & 000 & \cdots & \} \\
A & = & \{ & & & 1 & 00 & & & 000 & \cdots & \} \\
\chi_A & = & & 0 & 0 & 1 & 1 & 0 & 0 & 1 & \cdots &
\end{array}
$$

   - This is a one-to-one and onto mapping from $\mathcal{L}$ to B, so $|\mathcal{L}| = |B|$

The Set of Languages

Languages over alphabet Σ

$\mathcal{L}$ – the set of all languages over alphabet Σ is uncountable

Proof:

1. Consider the set B of infinite binary sequences
   - An infinite binary sequence is an infinite length string of 0's and 1's
   - B is uncountable
2. $|\mathcal{L}| = |B|$
   - Define the characteristic sequence χ_A of language $A \in \mathcal{L}$

$$
\begin{array}{rcl cccccccc l}
\Sigma^* & = & \{ & \epsilon & 0 & 1 & 00 & 01 & 11 & 000 & \cdots & \} \\
A & = & \{ & & & 1 & 00 & & & 000 & \cdots & \} \\
\chi_A & = & & 0 & 0 & 1 & 1 & 0 & 0 & 1 & \cdots &
\end{array}
$$

   - This is a one-to-one and onto mapping from $\mathcal{L}$ to B, so $|\mathcal{L}| = |B|$
3. Therefore, $\mathcal{L}$ is uncountable

Some Languages are not Turing-recognizable

We have proven:

1. The set of Turing Machines is countable
2. The set of languages is uncountable

Some Languages are not Turing-recognizable

We have proven:

1. The set of Turing Machines is countable
2. The set of languages is uncountable

Therefore:

Some Languages are not Turing-recognizable

We have proven:

1. The set of Turing Machines is countable
2. The set of languages is uncountable

Therefore:

- There is no one-to-one and onto mapping from languages to Turing Machines

Some Languages are not Turing-recognizable

We have proven:

1. The set of Turing Machines is countable
2. The set of languages is uncountable

Therefore:

- There is no one-to-one and onto mapping from languages to Turing Machines
- Thus, there exist languages that have no corresponding TM that recognizes them

Some Languages are not Turing-recognizable

We have proven:

1. The set of Turing Machines is countable
2. The set of languages is uncountable

Therefore:

- There is no one-to-one and onto mapping from languages to Turing Machines
- Thus, there exist languages that have no corresponding TM that recognizes them
- Note, that such languages are also undecidable

Where are we now?

Some Languages are not Turing-recognizable

We have proven:

1. The set of Turing Machines is countable
2. The set of languages is uncountable

Therefore:

- There is no one-to-one and onto mapping from languages to Turing Machines
- Thus, there exist languages that have no corresponding TM that recognizes them
- Note, that such languages are also undecidable

Where are we now?

- We have proven that some languages are not Turing-recognizable

Some Languages are not Turing-recognizable

We have proven:

1. The set of Turing Machines is countable
2. The set of languages is uncountable

Therefore:

- There is no one-to-one and onto mapping from languages to Turing Machines
- Thus, there exist languages that have no corresponding TM that recognizes them
- Note, that such languages are also undecidable

Where are we now?

- We have proven that some languages are not Turing-recognizable
- But, we have not given any examples of such a language

A_{TM} is Undecidable

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M(w) = 1\}$$

A_{TM} is Undecidable

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M(w) = 1\}$$

Proof: By contradiction

A_{TM} is Undecidable

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M(w) = 1\}$$

Proof: By contradiction

- Assume that A_{TM} is decided by a TM H

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

A_{TM} is Undecidable

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M(w) = 1\}$$

Proof: By contradiction

- Assume that A_{TM} is decided by a TM H

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

- Use H to build the following TM D:

A_{TM} is Undecidable

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M(w) = 1\}$$

Proof: By contradiction

- Assume that A_{TM} is decided by a TM H

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

- Use H to build the following TM 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

A_{TM} is Undecidable

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M(w) = 1\}$$

Proof: By contradiction

- Assume that A_{TM} is decided by a TM H

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

- Use H to build the following TM 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

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

A_{TM} is Undecidable

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M(w) = 1\}$$

Proof: By contradiction

- Assume that A_{TM} is decided by a TM H

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

- Use H to build the following TM 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

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

- Now consider what happens if we run D on $\langle D\rangle$

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

- Contradiction!

How Is This a Diagonalization?

How Is This a Diagonalization?

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	$\underline{\textit{accept}}$	*reject*	*accept*		*accept*	
M_2	*reject*	$\underline{\textit{reject}}$	*reject*	$\ldots$	*accept*	$\ldots$
M_3	*accept*	*accept*	$\underline{\textit{accept}}$		*reject*	
$\vdots$		$\vdots$		$\ddots$		
D	*reject*	*accept*	*reject*		?	

- We have defined D to do the opposite of what M_i does on input $\langle M_i \rangle$
- But what does D do on input $\langle D \rangle$??

A Turing-unrecognizable Language

$\overline{L_{TM}}$

The language

$$\overline{L_{TM}} = \{\langle M, w \rangle \mid M \text{ is a TM and } M(w) \neq 1\}$$

is not Turing-recognizable

Outline

Another Way to Prove Undecidability

Reductions Between Problems

Another Way to Prove Undecidability

Reductions Between Problems

There is a reduction from a problem A to a problem B if we can use a solution to problem B to solve problem A

$$A \leq B$$

Another Way to Prove Undecidability

Reductions Between Problems

There is a reduction from a problem A to a problem B if we can use a solution to problem B to solve problem A

$$A \leq B$$

Examples:

1. Finding area of a rectangle $\leq$ Finding its length and width

Another Way to Prove Undecidability

Reductions Between Problems

There is a reduction from a problem A to a problem B if we can use a solution to problem B to solve problem A

$$A \leq B$$

Examples:

1. Finding area of a rectangle $\leq$ Finding its length and width
2. Finding temperature outside $\leq$ Reading a thermometer

Another Way to Prove Undecidability

Reductions Between Problems

There is a reduction from a problem A to a problem B if we can use a solution to problem B to solve problem A

$$A \leq B$$

Examples:

1. Finding area of a rectangle $\leq$ Finding its length and width
2. Finding temperature outside $\leq$ Reading a thermometer

Observations:

Another Way to Prove Undecidability

Reductions Between Problems

There is a reduction from a problem A to a problem B if we can use a solution to problem B to solve problem A

$$A \leq B$$

Examples:

1. Finding area of a rectangle $\leq$ Finding its length and width
2. Finding temperature outside $\leq$ Reading a thermometer

Observations:

- Reductions not always symmetrical: $A \leq B$ does not mean $B \leq A$

Another Way to Prove Undecidability

Reductions Between Problems

There is a reduction from a problem A to a problem B if we can use a solution to problem B to solve problem A

$$A \leq B$$

Examples:

1. Finding area of a rectangle $\leq$ Finding its length and width
2. Finding temperature outside $\leq$ Reading a thermometer

Observations:

- Reductions not always symmetrical: $A \leq B$ does not mean $B \leq A$
- For now, no restriction on how the reduction works

Another Way to Prove Undecidability

Reductions Between Problems

There is a reduction from a problem A to a problem B if we can use a solution to problem B to solve problem A

$$A \leq B$$

Examples:

1. Finding area of a rectangle $\leq$ Finding its length and width
2. Finding temperature outside $\leq$ Reading a thermometer

Observations:

- Reductions not always symmetrical: $A \leq B$ does not mean $B \leq A$
- For now, no restriction on how the reduction works

Intuition

$A \leq B$ means that:

Another Way to Prove Undecidability

Reductions Between Problems

There is a reduction from a problem A to a problem B if we can use a solution to problem B to solve problem A

$$A \leq B$$

Examples:

1. Finding area of a rectangle $\leq$ Finding its length and width
2. Finding temperature outside $\leq$ Reading a thermometer

Observations:

- Reductions not always symmetrical: $A \leq B$ does not mean $B \leq A$
- For now, no restriction on how the reduction works

Intuition

$A \leq B$ means that:

- problem A is no harder than problem B.

Another Way to Prove Undecidability

Reductions Between Problems

There is a reduction from a problem A to a problem B if we can use a solution to problem B to solve problem A

$$A \leq B$$

Examples:

1. Finding area of a rectangle $\leq$ Finding its length and width
2. Finding temperature outside $\leq$ Reading a thermometer

Observations:

- Reductions not always symmetrical: $A \leq B$ does not mean $B \leq A$
- For now, no restriction on how the reduction works

Intuition

$A \leq B$ means that:

- problem A is no harder than problem B.
- Equivalently, problem B is no easier than problem A

Main Observation

Suppose that $A \leq B$, then:

Main Observation

Suppose that $A \leq B$, then:

- If A is undecidable
- B must also be undecidable

Main Observation

Suppose that $A \leq B$, then:

- If A is undecidable
- B must also be undecidable

Proof: (by contradiction)

Main Observation

Suppose that $A \leq B$, then:

- If A is undecidable
- B must also be undecidable

Proof: (by contradiction)

- Suppose that B is decidable

Reductions and Undecidability

Main Observation

Suppose that $A \leq B$, then:

- If A is undecidable
- B must also be undecidable

Proof: (by contradiction)

- Suppose that B is decidable
- Since $A \leq B$, there exists an algorithm (i.e., a reduction) that uses a solution to B to solve A

Main Observation

Suppose that $A \leq B$, then:

- If A is undecidable
- B must also be undecidable

Proof: (by contradiction)

- Suppose that B is decidable
- Since $A \leq B$, there exists an algorithm (i.e., a reduction) that uses a solution to B to solve A
- But, this means that A is decidable by running the machine for B as needed by the reduction

Undecidability of HALT

$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$$

Undecidability of HALT

$$HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$$

Theorem: $HALT$ is undecidable

Undecidability of HALT

$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$$

Theorem: $HALT$ is undecidable

Proof Sketch:

- We show that $A_{TM} \leq HALT$

Undecidability of HALT

$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$$

Theorem: $HALT$ is undecidable

Proof Sketch:

- We show that $A_{TM} \leq HALT$
- Since we know that A_{TM} is undecidable, this shows that $HALT$ is also undecidable

Undecidability of HALT

$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$$

Theorem: $HALT$ is undecidable

Proof Sketch:

- We show that $A_{TM} \leq HALT$
- Since we know that A_{TM} is undecidable, this shows that $HALT$ is also undecidable

Proof:

Construct algorithm S that decides A_{TM} given a TM R that decides $HALT$

Undecidability of HALT

$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$$

Theorem: $HALT$ is undecidable

Proof Sketch:

- We show that $A_{TM} \leq HALT$
- Since we know that A_{TM} is undecidable, this shows that $HALT$ is also undecidable

Proof:

Construct algorithm S that decides A_{TM} given a TM R that decides $HALT$

On input $\langle M, w\rangle$, S does the following:

Undecidability of HALT

$$HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$$

Theorem: $HALT$ is undecidable

Proof Sketch:

- We show that $A_{TM} \leq HALT$
- Since we know that A_{TM} is undecidable, this shows that $HALT$ is also undecidable

Proof:

Construct algorithm S that decides A_{TM} given a TM R that decides $HALT$

On input $\langle M, w \rangle$, S does the following:

- Run $R(\langle M, w \rangle)$

Undecidability of HALT

$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$$

Theorem: $HALT$ is undecidable

Proof Sketch:

- We show that $A_{TM} \leq HALT$
- Since we know that A_{TM} is undecidable, this shows that $HALT$ is also undecidable

Proof:

Construct algorithm S that decides A_{TM} given a TM R that decides $HALT$

On input $\langle M, w\rangle$, S does the following:

- Run $R(\langle M, w\rangle)$
- If R rejects – $M(w)$ doesn't halt – halt and reject

Undecidability of HALT

$$HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$$

Theorem: $HALT$ is undecidable

Proof Sketch:

- We show that $A_{TM} \leq HALT$
- Since we know that A_{TM} is undecidable, this shows that $HALT$ is also undecidable

Proof:

Construct algorithm S that decides A_{TM} given a TM R that decides $HALT$

On input $\langle M, w \rangle$, S does the following:

- Run $R(\langle M, w \rangle)$
- If R rejects – $M(w)$ doesn't halt – halt and reject
- if R accepts – $M(w)$ halts – Simulate $M(w)$ until it halts

Undecidability of HALT

$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$$

Theorem: $HALT$ is undecidable

Proof Sketch:

- We show that $A_{TM} \leq HALT$
- Since we know that A_{TM} is undecidable, this shows that $HALT$ is also undecidable

Proof:

Construct algorithm S that decides A_{TM} given a TM R that decides $HALT$

On input $\langle M, w\rangle$, S does the following:

- Run $R(\langle M, w\rangle)$
- If R rejects – $M(w)$ doesn't halt – halt and reject
- if R accepts – $M(w)$ halts – Simulate $M(w)$ until it halts
- Output whatever M output