Foundations of Computing

Lecture 16

Arkady Yerukhimovich

March 21, 2023

Outline

1. Lecture 15 Review
2. An Undecidable Language
3. Reducibility
4. Where Are We Now?

Lecture 15 Review

- Labguages about Machines
- Countable and Uncountable Sets
  - Diagonalization
- Proving L_{TM} is Undecidable

Outline

L_{TM} is Turing-recognizable

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

L_{TM} is Turing-recognizable

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

Proof: By construction of machine $M_{L_{TM}}$

$M_{L_{TM}}$: On input $\langle M, w\rangle$,

1. Run M on input w

L_{TM} is Turing-recognizable

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

Proof: By construction of machine $M_{L_{TM}}$

$M_{L_{TM}}$: On input $\langle M, w\rangle$,

1. Run M on input w
2. If M halts, halt and output what M outputs

L_{TM} is Turing-recognizable

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

Proof: By construction of machine $M_{L_{TM}}$

$M_{L_{TM}}$: On input $\langle M, w\rangle$,

1. Run M on input w
2. If M halts, halt and output what M outputs

Correctness:

- For any input $\langle M, w\rangle \in L_{TM}$, M is a TM, and $M(w)$ halts and outputs 1.

L_{TM} is Turing-recognizable

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

Proof: By construction of machine $M_{L_{TM}}$

$M_{L_{TM}}$: On input $\langle M, w\rangle$,

1. Run M on input w
2. If M halts, halt and output what M outputs

Correctness:

- For any input $\langle M, w\rangle \in L_{TM}$, M is a TM, and $M(w)$ halts and outputs 1.
- Hence, $M_{L_{TM}}$, also halts and outputs 1

L_{TM} is Turing-recognizable

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

Proof: By construction of machine $M_{L_{TM}}$

$M_{L_{TM}}$: On input $\langle M, w\rangle$,

1. Run M on input w
2. If M halts, halt and output what M outputs

Correctness:

- For any input $\langle M, w\rangle \in L_{TM}$, M is a TM, and $M(w)$ halts and outputs 1.
- Hence, $M_{L_{TM}}$, also halts and outputs 1
- Thus, $M_{L_{TM}}$ accepts all inputs in L_{TM}

L_{TM} is Turing-recognizable

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

Proof: By construction of machine $M_{L_{TM}}$

$M_{L_{TM}}$: On input $\langle M, w\rangle$,

1. Run M on input w
2. If M halts, halt and output what M outputs

Correctness:

- For any input $\langle M, w\rangle \in L_{TM}$, M is a TM, and $M(w)$ halts and outputs 1.
- Hence, $M_{L_{TM}}$, also halts and outputs 1
- Thus, $M_{L(TM}$ accepts all inputs in L_{TM}
- Note that $M_{L_{TM}}$ may not halt on all inputs – doesn't decide L_{TM}

L_{TM} is Undecidable

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

L_{TM} is Undecidable

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

Proof: By contradiction

L_{TM} is Undecidable

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

Proof: By contradiction

- Assume that L_{TM} is decided by TM H

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

L_{TM} is Undecidable

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

Proof: By contradiction

- Assume that L_{TM} is decided by 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 a TM D that checks whether a TM M accepts its own description, and then does the opposite:

L_{TM} is Undecidable

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

Proof: By contradiction

- Assume that L_{TM} is decided by TM H

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

- Use H to build a TM D that checks whether a TM M accepts its own description, and then does the opposite:

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

L_{TM} is Undecidable

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

Proof: By contradiction

- Assume that L_{TM} is decided by TM H

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

- Use H to build a TM D that checks whether a TM M accepts its own description, and then does the opposite:
  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

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

L_{TM} is Undecidable

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

Proof: By contradiction

- Assume that L_{TM} is decided by 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 a TM D that checks whether a TM M accepts its own description, and then does the opposite:

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 $\langle D\rangle$

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

How Is This a Diagonalization?

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

1. Lecture 15 Review
2. An Undecidable Language
3. Reducibility
4. Where Are We Now?

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:

Reductions and Undecidability

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

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
- 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 $L_{TM} \leq HALT$

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

Proof:

Construct algorithm S that decides L_{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 $L_{TM} \leq HALT$
- Since we know that L_{TM} is undecidable, this shows that $HALT$ is also undecidable

Proof:

Construct algorithm S that decides L_{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 $L_{TM} \leq HALT$
- Since we know that L_{TM} is undecidable, this shows that $HALT$ is also undecidable

Proof:

Construct algorithm S that decides L_{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 $L_{TM} \leq HALT$
- Since we know that L_{TM} is undecidable, this shows that $HALT$ is also undecidable

Proof:

Construct algorithm S that decides L_{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 $L_{TM} \leq HALT$
- Since we know that L_{TM} is undecidable, this shows that $HALT$ is also undecidable

Proof:

Construct algorithm S that decides L_{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 $L_{TM} \leq HALT$
- Since we know that L_{TM} is undecidable, this shows that $HALT$ is also undecidable

Proof:

Construct algorithm S that decides L_{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

Other Undecidable Languages

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

Other Undecidable Languages

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

Theorem: $REGULAR_{TM}$ is undecidable

Other Undecidable Languages

$$REGULAR_{TM} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

Theorem: $REGULAR_{TM}$ is undecidable
Proof Sketch:

- We show that $L_{TM} \leq REGULAR_{TM}$

Other Undecidable Languages

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

Theorem: $REGULAR_{TM}$ is undecidable
Proof Sketch:

- We show that $L_{TM} \leq REGULAR_{TM}$
- Specifically, reduction builds a TM M_2 s.t.

Other Undecidable Languages

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

Theorem: $REGULAR_{TM}$ is undecidable
Proof Sketch:

- We show that $L_{TM} \leq REGULAR_{TM}$
- Specifically, reduction builds a TM M_2 s.t.
  - If M accepts w, M_2 recognizes Σ^* – regular language

Other Undecidable Languages

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

Theorem: $REGULAR_{TM}$ is undecidable
Proof Sketch:

- We show that $L_{TM} \leq REGULAR_{TM}$
- Specifically, reduction builds a TM M_2 s.t.
  - If M accepts w, M_2 recognizes Σ^* – regular language
  - If M does not accept w, M_2 recognizes $\{0^n 1^n\}$ – not regular

Other Undecidable Languages

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

Theorem: $REGULAR_{TM}$ is undecidable
Proof Sketch:

- We show that $L_{TM} \leq REGULAR_{TM}$
- Specifically, reduction builds a TM M_2 s.t.
  - If M accepts w, M_2 recognizes Σ^* – regular language
  - If M does not accept w, M_2 recognizes $\{0^n 1^n\}$ – not regular
- If we can decide whether M_2 recognizes a regular language or not, can use that to decide whether M accepts w or not

Other Undecidable Languages

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

Theorem: $REGULAR_{TM}$ is undecidable
Proof:

Other Undecidable Languages

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

Theorem: $REGULAR_{TM}$ is undecidable
Proof:
Construct algorithm S that decides L_{TM} given a TM R that decides $REGULAR_{TM}$

Other Undecidable Languages

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

Theorem: $REGULAR_{TM}$ is undecidable
Proof:
Construct algorithm S that decides L_{TM} given a TM R that decides $REGULAR_{TM}$
On input $\langle M, w \rangle$:

Other Undecidable Languages

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

Theorem: $REGULAR_{TM}$ is undecidable
Proof:
Construct algorithm S that decides L_{TM} given a TM R that decides $REGULAR_{TM}$
On input $\langle M, w \rangle$:

1. Construct TM M_2 s.t. on input x

Other Undecidable Languages

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

Theorem: $REGULAR_{TM}$ is undecidable

Proof:

Construct algorithm S that decides L_{TM} given a TM R that decides $REGULAR_{TM}$

On input $\langle M, w \rangle$:

1. Construct TM M_2 s.t. on input x
    1. If $x = 0^n 1^n$, accept

Other Undecidable Languages

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

Theorem: $REGULAR_{TM}$ is undecidable

Proof:

Construct algorithm S that decides L_{TM} given a TM R that decides $REGULAR_{TM}$

On input $\langle M, w \rangle$:

1. Construct TM M_2 s.t. on input x
    1. If $x = 0^n 1^n$, accept
    2. If x does not have this form, run $M(w)$ and accept if it accepts

Other Undecidable Languages

$$REGULAR_{TM} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

Theorem: $REGULAR_{TM}$ is undecidable
Proof:
Construct algorithm S that decides L_{TM} given a TM R that decides $REGULAR_{TM}$
On input $\langle M, w\rangle$:

1. Construct TM M_2 s.t. on input x
   1. If $x = 0^n1^n$, accept
   2. If x does not have this form, run $M(w)$ and accept if it accepts
2. Run R on input $\langle M_2\rangle$

Other Undecidable Languages

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

Theorem: $REGULAR_{TM}$ is undecidable

Proof:

Construct algorithm S that decides L_{TM} given a TM R that decides $REGULAR_{TM}$

On input $\langle M, w \rangle$:

1. Construct TM M_2 s.t. on input x
   1. If $x = 0^n 1^n$, accept
   2. If x does not have this form, run $M(w)$ and accept if it accepts
2. Run R on input $\langle M_2 \rangle$
3. Output what R outputs

Other Undecidable Languages

$$EMPTY-STRING_{TM} = \{\langle M\rangle \mid M \text{ is a TM and } M(\epsilon)=1\}$$

Assume R that decides

$L_{TM} \leq EMPTY\text{-}STRING_{TM}$

L_{TM} = $\langle M, w\rangle$ accept if $M(w)=1$

S, Given $\langle M, w\rangle$

$M_w(\epsilon)$: write w onto its tape

Run $M(w)$ accepts if it accepts

Run $R(\langle M_w\rangle)$