Foundations of Computing

Lecture 16

Arkady Yerukhimovich

March 19, 2024

Outline

Lecture 15 Review

- Countable and Uncountable Sets
  - Diagonalization
- Proving A_{TM} is Undecidable

Outline

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

$$A \leq B$$

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

$$A \leq B$$

Intuition

$A \leq B$ means that:

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

Reductions and Undecidability

Main Observation

Suppose that $A \leq B$, then:

- If A is undecidable
- B must also be undecidable

Reductions and Undecidability

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 reduction using the decider machine for B.

Undecidability of $HALT_{TM}$

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

Undecidability of $HALT_{TM}$

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

Theorem: $HALT$ is undecidable

Undecidability of $HALT_{TM}$

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

Theorem: $HALT$ is undecidable

Proof Sketch:

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

Undecidability of $HALT_{TM}$

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

Theorem: $HALT$ is undecidable

Proof Sketch:

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

Undecidability of $HALT_{TM}$

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

Theorem: $HALT$ is undecidable

Proof Sketch:

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

Proof:

Construct reduction R that decides A_{TM} given a TM D that decides $HALT$

Undecidability of $HALT_{TM}$

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

Theorem: $HALT$ is undecidable

Proof Sketch:

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

Proof:

Construct reduction R that decides A_{TM} given a TM D that decides $HALT$

On input $\langle M, w\rangle$, R does the following: R is trying to decide if

$M(w) = 1$

Undecidability of $HALT_{TM}$

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

Theorem: $HALT$ is undecidable

Proof Sketch:

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

Proof:

Construct reduction R that decides A_{TM} given a TM D that decides $HALT$

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

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

Undecidability of $HALT_{TM}$

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

Theorem: $HALT$ is undecidable

Proof Sketch:

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

Proof:

Construct reduction R that decides A_{TM} given a TM D that decides $HALT$

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

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

Undecidability of $HALT_{TM}$

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

Theorem: $HALT$ is undecidable

Proof Sketch:

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

Proof:

Construct reduction R that decides A_{TM} given a TM D that decides $HALT$
On input $\langle M, w \rangle$, R does the following:

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

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 $A_{TM} \leq REGULAR_{TM}$

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 $A_{TM} \le REGULAR_{TM}$
- Specifically, reduction builds another TM M' s.t.

$\langle M, w \rangle$ Does $M(w) = 1$

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 $A_{TM} \leq REGULAR_{TM}$
- Specifically, reduction builds another TM M' s.t.
  - If M accepts w, M' 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 $A_{TM} \leq REGULAR_{TM}$
- Specifically, reduction builds another TM M' s.t.
    - If M accepts w, M' recognizes Σ^* – regular language
    - If M does not accept w, M' recognizes $\{0^n1^n\}$ – 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 $A_{TM} \leq REGULAR_{TM}$
- Specifically, reduction builds another TM M' s.t.
  - If M accepts w, M' recognizes Σ^* – regular language
  - If M does not accept w, M' recognizes $\{0^n 1^n\}$ – not regular
- If we can decide whether M' recognizes a regular language or not, can use that to decide whether M accepts w or not

1. On input $\langle M, w \rangle$ build M'
2. Run $D(M')$
3. if $D(M') = 1 \Rightarrow M(w) = 1$, if $D(M') = 0 \Rightarrow M$ does not accept w

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:
Reduction R that decides A_{TM} given a TM D 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:
Reduction R that decides A_{TM} given a TM D 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:
Reduction R that decides A_{TM} given a TM D that decides $REGULAR_{TM}$
On input $\langle M, w \rangle$:

1. Construct TM $M'_{\langle M,w \rangle}$ s.t. $M'_{\langle M,w \rangle}(x)$ is as follows:

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:

Reduction R that decides A_{TM} given a TM D that decides $REGULAR_{TM}$

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

1. Construct TM $M'_{\langle M,w \rangle}$ s.t. $M'_{\langle M,w \rangle}(x)$ is as follows:
    - 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:

Reduction R that decides A_{TM} given a TM D that decides $REGULAR_{TM}$

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

1. Construct TM $M'_{\langle M,w \rangle}$ s.t. $M'_{\langle M,w \rangle}(x)$ is as follows:
    - If $x = 0^n 1^n$, accept
    - 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:
Reduction R that decides A_{TM} given a TM D that decides $REGULAR_{TM}$
On input $\langle M, w \rangle$:

1. Construct TM $M'_{\langle M,w \rangle}$ s.t. $M'_{\langle M,w \rangle}(x)$ is as follows:
   - If $x = 0^n 1^n$, accept
   - If x does not have this form, run $M(w)$ and accept if it accepts
2. Run D on input $\langle M' \rangle$

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

Theorem: $REGULAR_{TM}$ is undecidable
Proof:
Reduction R that decides A_{TM} given a TM D that decides $REGULAR_{TM}$
On input $\langle M, w \rangle$:

1. Construct TM $M'_{\langle M,w \rangle}$ s.t. $M'_{\langle M,w \rangle}(x)$ is as follows:
   - If $x = 0^n 1^n$, accept
   - If x does not have this form, run $M(w)$ and accept if it accepts
2. Run D on input $\langle M' \rangle$
3. Output what D outputs

$M(w) = 1$: M' accepts all strings $L(M') = \Sigma^* -$ Reg

M does not accept w : M' accept $0^n 1^n$ $L(M')$ - Non Reg.

Other Undecidable Languages – Exercise

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

Outline

Summary

Algorithms

Algorithms are critical for understanding decidability of problems

Summary

Algorithms

Algorithms are critical for understanding decidability of problems

1. To show that a problem is decidable – give an algorithm that always terminates and outputs the answer

Summary

Algorithms

Algorithms are critical for understanding decidability of problems

1. To show that a problem is decidable – give an algorithm that always terminates and outputs the answer
2. To show that a problem is undecidable – give an algorithm (a reduction) that shows that this problem can be used to solve one of the undecidable problems

What About Turing-Unrecognizable Problems?

Question

Can reductions help us determine if a language is Turing-unrecognizable?

What About Turing-Unrecognizable Problems?

Question

Can reductions help us determine if a language is Turing-unrecognizable?

Recall: $\overline{A_{TM}}$ is Turing-unrecognizable

What About Turing-Unrecognizable Problems?

Question

Can reductions help us determine if a language is Turing-unrecognizable?

Recall: $\overline{A_{TM}}$ is Turing-unrecognizable

Problem: $\overline{A_{TM}} \leq A_{TM}$

What About Turing-Unrecognizable Problems?

Question

Can reductions help us determine if a language is Turing-unrecognizable?

Recall: $\overline{A_{TM}}$ is Turing-unrecognizable

Problem: $\overline{A_{TM}} \leq A_{TM}$
but A_{TM} is Turing-recognizable

What About Turing-Unrecognizable Problems?

Question

Can reductions help us determine if a language is Turing-unrecognizable?

Recall: $\overline{A_{TM}}$ is Turing-unrecognizable

Problem: $\overline{A_{TM}} \leq A_{TM}$
but A_{TM} is Turing-recognizable

Solution

We need to restrict what our reductions can do.

Outline

Mapping Reductions

Definition

Language A is mapping reducible to language B ($A \leq_m B$) if there is a computable function $f : \Sigma^* \to \Sigma^*$, where for every w,

$$w \in A \iff f(w) \in B$$

Mapping Reductions

Definition

Language A is mapping reducible to language B ($A \leq_m B$) if there is a computable function $f : \Sigma^* \to \Sigma^*$, where for every w,

$$w \in A \iff f(w) \in B$$

- Function f is computable if it can be computed by a TM / algorithm
  - There is a TM M that starts with w on its tape, writes $f(w)$ on its tape

Mapping Reductions

Definition

Language A is mapping reducible to language B ($A \leq_m B$) if there is a computable function $f : \Sigma^* \to \Sigma^*$, where for every w,

$$w \in A \iff f(w) \in B$$

- Function f is computable if it can be computed by a TM / algorithm
  - There is a TM M that starts with w on its tape, writes $f(w)$ on its tape
- Such reductions are also called:
  - many-one reductions
  - Karp reductions (when only considering poly-time reductions)

Mapping Reductions

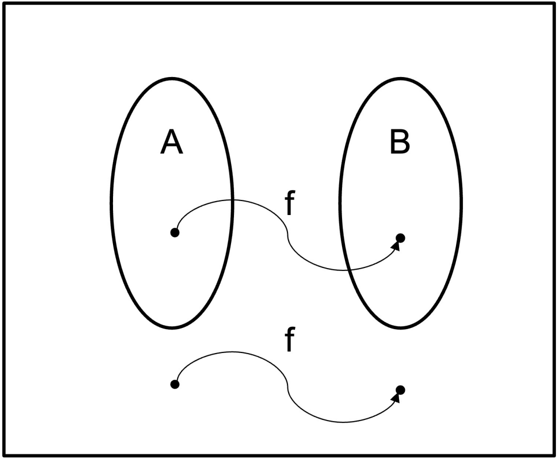
Definition

Language A is mapping reducible to language B ($A \leq_m B$) if there is a computable function $f : \Sigma^* \to \Sigma^*$, where for every w,

$$w \in A \iff f(w) \in B$$

- Function f is computable if it can be computed by a TM / algorithm
  - There is a TM M that starts with w on its tape, writes $f(w)$ on its tape
- Such reductions are also called:
  - many-one reductions
  - Karp reductions (when only considering poly-time reductions)
- Works by mapping input $\in A$ to input $\in B$ and vice-versa

Mapping Reductions



Mapping Reduction Properties

Mapping reductions are very useful:

Mapping Reduction Properties

Mapping reductions are very useful:

1. If $A \leq_m B$
   - If B is decidable then A is decidable

Mapping Reduction Properties

Mapping reductions are very useful:

1. If $A \leq_m B$
   - If B is decidable then A is decidable
   - If A is undecidable then B is undecidable

Mapping Reduction Properties

Mapping reductions are very useful:

1. If $A \leq_m B$
   - If B is decidable then A is decidable
   - If A is undecidable then B is undecidable
2. If $A \leq_m B$
   - If B is Turing-recognizable then

Mapping Reduction Properties

Mapping reductions are very useful:

1. If $A \leq_m B$
   - If B is decidable then A is decidable
   - If A is undecidable then B is undecidable
2. If $A \leq_m B$
   - If B is Turing-recognizable then A is Turing-recognizable

Mapping Reduction Properties

Mapping reductions are very useful:

1. If $A \leq_m B$
   - If B is decidable then A is decidable
   - If A is undecidable then B is undecidable
2. If $A \leq_m B$
   - If B is Turing-recognizable then A is Turing-recognizable
   - If A is not Turing-recognizable than B is not Turing-recognizable

Turing Reductions

Definition

Language A is Turing reducible to language B ($A \leq_T B$) if can use a decider for B to decide A.

Turing Reductions

Definition

Language A is Turing reducible to language B ($A \leq_T B$) if can use a decider for B to decide A.

- The reduction may make multiple calls to decider for B and may not directly use the result.

Turing Reductions

Definition

Language A is Turing reducible to language B ($A \leq_T B$) if can use a decider for B to decide A.

- The reduction may make multiple calls to decider for B and may not directly use the result.
- For example, in the proof that $L_{TM} \leq L_{E_{TM}}$, we flipped the result of R deciding $L_{E_{TM}}$

Turing Reduction Properties

Turing reductions are more general than mapping reductions:

Turing Reduction Properties

Turing reductions are more general than mapping reductions:

1. If $A \leq_m B$, then $A \leq_T B$

Turing Reduction Properties

Turing reductions are more general than mapping reductions:

1. If $A \leq_m B$, then $A \leq_T B$
2. If $A \leq_T B$, then it is not necessarily the case that $A \leq_m B$

Turing Reduction Properties

Turing reductions are more general than mapping reductions:

1. If $A \leq_m B$, then $A \leq_T B$
2. If $A \leq_T B$, then it is not necessarily the case that $A \leq_m B$
   - In particular, $L_{TM} \leq_T \overline{L_{TM}}$, but $L_{TM} \not\leq_m \overline{L_{TM}}$

Turing Reduction Properties

Turing reductions are more general than mapping reductions:

1. If $A \leq_m B$, then $A \leq_T B$
2. If $A \leq_T B$, then it is not necessarily the case that $A \leq_m B$
   - In particular, $L_{TM} \leq_T \overline{L_{TM}}$, but $L_{TM} \not\leq_m \overline{L_{TM}}$

But, they have weaker implications than mapping reductions:

Turing Reduction Properties

Turing reductions are more general than mapping reductions:

1. If $A \leq_m B$, then $A \leq_T B$
2. If $A \leq_T B$, then it is not necessarily the case that $A \leq_m B$
   - In particular, $L_{TM} \leq_T \overline{L_{TM}}$, but $L_{TM} \not\leq_m \overline{L_{TM}}$

But, they have weaker implications than mapping reductions:

3. If $A \leq_T B$
   - If B is decidable then A is decidable

Turing Reduction Properties

Turing reductions are more general than mapping reductions:

1. If $A \leq_m B$, then $A \leq_T B$
2. If $A \leq_T B$, then it is not necessarily the case that $A \leq_m B$
   - In particular, $L_{TM} \leq_T \overline{L_{TM}}$, but $L_{TM} \not\leq_m \overline{L_{TM}}$

But, they have weaker implications than mapping reductions:

3. If $A \leq_T B$
   - If B is decidable then A is decidable
   - If A is not decidable, then B is not decidable

Turing Reduction Properties

Turing reductions are more general than mapping reductions:

1. If $A \leq_m B$, then $A \leq_T B$
2. If $A \leq_T B$, then it is not necessarily the case that $A \leq_m B$
   - In particular, $L_{TM} \leq_T \overline{L_{TM}}$, but $L_{TM} \not\leq_m \overline{L_{TM}}$

But, they have weaker implications than mapping reductions:

3. If $A \leq_T B$
   - If B is decidable then A is decidable
   - If A is not decidable, then B is not decidable
4. If $A \leq_T B$
   - If B is Turing-recognizable,

Turing Reduction Properties

Turing reductions are more general than mapping reductions:

1. If $A \leq_m B$, then $A \leq_T B$
2. If $A \leq_T B$, then it is not necessarily the case that $A \leq_m B$
   - In particular, $L_{TM} \leq_T \overline{L_{TM}}$, but $L_{TM} \not\leq_m \overline{L_{TM}}$

But, they have weaker implications than mapping reductions:

3. If $A \leq_T B$
   - If B is decidable then A is decidable
   - If A is not decidable, then B is not decidable
4. If $A \leq_T B$
   - If B is Turing-recognizable,A is not necessarily Turing-recognizable

Turing Reduction Properties

Turing reductions are more general than mapping reductions:

1. If $A \leq_m B$, then $A \leq_T B$
2. If $A \leq_T B$, then it is not necessarily the case that $A \leq_m B$
    - In particular, $L_{TM} \leq_T \overline{L_{TM}}$, but $L_{TM} \not\leq_m \overline{L_{TM}}$

But, they have weaker implications than mapping reductions:

3. If $A \leq_T B$
    - If B is decidable then A is decidable
    - If A is not decidable, then B is not decidable
4. If $A \leq_T B$
    - If B is Turing-recognizable,A is not necessarily Turing-recognizable
    - If A is not Turing-recognizable, cannot say if B is Turing-recognizable