# Foundations of Computing

## Lecture 17

Arkady Yerukhimovich

March 21, 2024

# Outline


1. Lecture 16 Review
2. Where Are We Now?
3. Reduction Types
4. A Computational Definition of Information – Kolmogorov Complexity

# Lecture 16 Review

- Proofs by reduction
- Undecidable languages
  - $HALT_{TM}$
  - $REGULAR_{TM}$

# Exercise

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

# Outline


1. Lecture 16 Review
2. Where Are We Now?
3. Reduction Types
4. A Computational Definition of Information – Kolmogorov Complexity

# 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 an 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

Takeaway: General reductions do not work for Turing-unrecognizable languages

**Solution**

We need to restrict what our reductions can do.

# Outline


1. Lecture 16 Review
2. Where Are We Now?
3. Reduction Types
4. A Computational Definition of Information – Kolmogorov Complexity

# 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

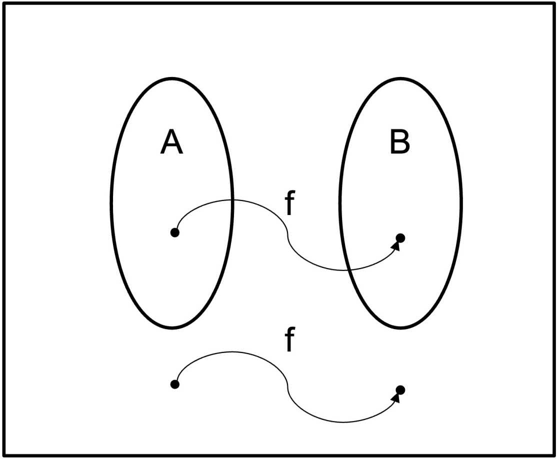
## Definition

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:

If $A \leq_m B$

- If $B$ is decidable then $A$ is decidable

# Mapping Reduction Properties

Mapping reductions are very useful:

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:

If $A \leq_m B$

- If $B$ is decidable then $A$ is decidable
- If $A$ is undecidable then $B$ is undecidable

- If $B$ is Turing-recognizable then $A$ is Turing-recognizable

# Mapping Reduction Properties

Mapping reductions are very useful:

If $A \leq_m B$

- If $B$ is decidable then $A$ is decidable
- If $A$ is undecidable then $B$ is undecidable

- If $B$ is Turing-recognizable then $A$ is Turing-recognizable
- If $A$ is not Turing-recognizable than $B$ is not Turing-recognizable

**Observation:**

Mapping reductions work for both decidability and Turing-recognizability.

# 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
    - 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
   - 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

   - If $B$ is Turing-recognizable, $A$ is not necessarily Turing-recognizable
   - If $A$ is not Turing-recognizable, cannot say if $B$ is Turing-recognizable

# Outline


1. Lecture 16 Review
2. Where Are We Now?
3. Reduction Types
4. A Computational Definition of Information – Kolmogorov Complexity

# Information in a String

$$A = 010101010101010101010101$$

$$B = 110100100011100010111111$$

**Question**

Which of these strings contains more information?

# Kolmogorov Complexity

## Definition

Consider $x \in \{0,1\}^*$.

# Kolmogorov Complexity

## Definition

Consider $x \in \{0,1\}^*$.

1. The minimal description of $x$ ($d(x)$) is the shortest string $\langle M, w \rangle$ such that TM $M$ on input $w$ halts with $x$ on its tape

# Kolmogorov Complexity

## Definition

Consider $x \in \{0,1\}^*$.

1. The minimal description of $x$ ($d(x)$) is the shortest string $\langle M, w\rangle$ such that TM $M$ on input $w$ halts with $x$ on its tape
2. The Kolmogorov complexity of $x$ is

$$K(x) = |d(x)|$$

# Kolmogorov Complexity

## Definition

Consider $x \in \{0,1\}^*$.

1. The minimal description of $x$ ($d(x)$) is the shortest string $\langle M, w \rangle$ such that TM $M$ on input $w$ halts with $x$ on its tape
2. The Kolmogorov complexity of $x$ is

$$K(x) = |d(x)|$$

- Intuitively: $K(x)$ is the length of the shortest program that outputs $x$
- $K(x)$ is the minimal description of $x$

# Kolmogorov Complexity

## Definition

Consider $x \in \{0,1\}^*$.

1. The minimal description of $x$ ($d(x)$) is the shortest string $\langle M, w \rangle$ such that TM $M$ on input $w$ halts with $x$ on its tape
2. The Kolmogorov complexity of $x$ is

$$K(x) = |d(x)|$$

- Intuitively: $K(x)$ is the length of the shortest program that outputs $x$
- $K(x)$ is the minimal description of $x$
- This captures the “amount of information” in $x$

# Properties of Kolmogorov Complexity

1. $\forall x, K(x) \leq |x| + c$ for some constant $c$

# Properties of Kolmogorov Complexity

1. $\forall x, K(x) \leq |x| + c$ for some constant $c$
   - Can always describe a TM $M$ that given $x$ just leaves it on it's tape

# Properties of Kolmogorov Complexity

1. $\forall x, K(x) \leq |x| + c$ for some constant $c$
   - Can always describe a TM $M$ that given $x$ just leaves it on it's tape
   - Size of description of $M$ is independent of $|x|$

# Properties of Kolmogorov Complexity

1. $\forall x, K(x) \leq |x| + c$ for some constant $c$
   - Can always describe a TM $M$ that given $x$ just leaves it on it's tape
   - Size of description of $M$ is independent of $|x|$
   - Can describe $x$ as $\langle M \rangle || x$

# Properties of Kolmogorov Complexity

1. $\forall x, K(x) \leq |x| + c$ for some constant $c$
   - Can always describe a TM $M$ that given $x$ just leaves it on it's tape
   - Size of description of $M$ is independent of $|x|$
   - Can describe $x$ as $\langle M \rangle || x$
     - Need to have some way to indicate where description of $M$ ends and description of $x$ begins (no special characters to do this)

# Properties of Kolmogorov Complexity

1. $\forall x, K(x) \leq |x| + c$ for some constant $c$
   - Can always describe a TM $M$ that given $x$ just leaves it on it's tape
   - Size of description of $M$ is independent of $|x|$
   - Can describe $x$ as $\langle M \rangle || x$
     - Need to have some way to indicate where description of $M$ ends and description of $x$ begins (no special characters to do this)
2. $\forall x, K(xx) \leq K(x) + c$ for some constant $c$

# Properties of Kolmogorov Complexity

1. $\forall x, K(x) \leq |x| + c$ for some constant $c$
   - Can always describe a TM $M$ that given $x$ just leaves it on it's tape
   - Size of description of $M$ is independent of $|x|$
   - Can describe $x$ as $\langle M \rangle || x$
     - Need to have some way to indicate where description of $M$ ends and description of $x$ begins (no special characters to do this)
2. $\forall x, K(xx) \leq K(x) + c$ for some constant $c$
   - Use $K(x)$ bits to describe $x$, then use $c$ bits to describe TM that repeats its input

# Properties of Kolmogorov Complexity

1. $\forall x, K(x) \leq |x| + c$ for some constant $c$
   - Can always describe a TM $M$ that given $x$ just leaves it on it's tape
   - Size of description of $M$ is independent of $|x|$
   - Can describe $x$ as $\langle M \rangle || x$
     - Need to have some way to indicate where description of $M$ ends and description of $x$ begins (no special characters to do this)
2. $\forall x, K(xx) \leq K(x) + c$ for some constant $c$
   - Use $K(x)$ bits to describe $x$, then use $c$ bits to describe TM that repeats its input
3. $\forall x, y, K(xy) \leq$

# Properties of Kolmogorov Complexity

1. $\forall x, K(x) \leq |x| + c$ for some constant $c$
   - Can always describe a TM $M$ that given $x$ just leaves it on it's tape
   - Size of description of $M$ is independent of $|x|$
   - Can describe $x$ as $\langle M \rangle || x$
     - Need to have some way to indicate where description of $M$ ends and description of $x$ begins (no special characters to do this)
2. $\forall x, K(xx) \leq K(x) + c$ for some constant $c$
   - Use $K(x)$ bits to describe $x$, then use $c$ bits to describe TM that repeats its input
3. $\forall x, y, K(xy) \leq 2K(x) + K(y) + c$

# Compressibility of Strings

## Definition

For string $x$, $x$ is $c$-compressible if

$$K(x) \leq |x| - c$$

# Compressibility of Strings

## Definition

For string $x$, $x$ is $c$-compressible if

$$K(x) \le |x| - c$$

If $K(x) \ge |x|$, then $x$ is incompressible

# Compressibility of Strings

**Definition**

For string $x$, $x$ is $c$-compressible if

$$K(x) \leq |x| - c$$

If $K(x) \geq |x|$, then $x$ is incompressible

1. Incompressible strings of every length exist

# Compressibility of Strings

## Definition

For string $x$, $x$ is $c$-compressible if

$$K(x) \leq |x| - c$$

If $K(x) \geq |x|$, then $x$ is incompressible

1. Incompressible strings of every length exist
   Proof:

# Compressibility of Strings

## Definition

For string $x$, $x$ is $c$-compressible if

$$K(x) \leq |x| - c$$

If $K(x) \geq |x|$, then $x$ is incompressible

1. Incompressible strings of every length exist
   Proof:
   - There are $2^n$ (binary) strings of length $n$

# Compressibility of Strings

## Definition

For string $x$, $x$ is $c$-compressible if

$$K(x) \le |x| - c$$

If $K(x) \ge |x|$, then $x$ is incompressible

1. Incompressible strings of every length exist
   Proof:
   - There are $2^n$ (binary) strings of length $n$
   - The number of programs of length less than $n$ is

$$\sum_{0 \le i \le n-1} 2^i = 1 + 2 + 4 + \cdots + 2^{n-1} = 2^n - 1$$

# Compressibility of Strings

## Definition

For string $x$, $x$ is $c$-compressible if

$$K(x) \leq |x| - c$$

If $K(x) \geq |x|$, then $x$ is incompressible

1. Incompressible strings of every length exist
   Proof:
   - There are $2^n$ (binary) strings of length $n$
   - The number of programs of length less than $n$ is

     $$\sum_{0 \leq i \leq n-1} 2^i = 1 + 2 + 4 + \cdots + 2^{n-1} = 2^n - 1$$

   - So, there exists at least one string that is incompressible

# Compressibility of Strings

## Definition

For string $x$, $x$ is $c$-compressible if

$$K(x) \leq |x| - c$$

If $K(x) \geq |x|$, then $x$ is incompressible

1. Incompressible strings of every length exist
   Proof:
   - There are $2^n$ (binary) strings of length $n$
   - The number of programs of length less than $n$ is

   $$\sum_{0 \leq i \leq n-1} 2^i = 1 + 2 + 4 + \cdots + 2^{n-1} = 2^n - 1$$

   - So, there exists at least one string that is incompressible
2. In fact, incompressible strings look like random strings

# Compressibility of Strings

## Definition

For string $x$, $x$ is $c$-compressible if

$$K(x) \leq |x| - c$$

If $K(x) \geq |x|$, then $x$ is incompressible

1. Incompressible strings of every length exist
   Proof:
   - There are $2^n$ (binary) strings of length $n$
   - The number of programs of length less than $n$ is

   $$\sum_{0 \leq i \leq n-1} 2^i = 1 + 2 + 4 + \cdots + 2^{n-1} = 2^n - 1$$

   - So, there exists at least one string that is incompressible
2. In fact, incompressible strings look like random strings
3. But, $K(x)$ is not computable, moreover it is undecidable whether a string is incompressible