Foundations of Computing

Lecture 17

Arkady Yerukhimovich

March 23, 2023

Outline

Lecture 16 Review

- Proving undecidability
- Reductions
- Examples of undecidable languages

Outline

Reductions

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

$$A \leq B$$

Reductions

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

$$A \leq B$$

Using Reductions

Suppose that $A \leq B$, then:

- If A is undecidable
- B must also be undecidable

Reductions

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

$$A \leq B$$

Using Reductions

Suppose that $A \leq B$, then:

- If A is undecidable
- B must also be undecidable

We have actually seen two different types of reductions

Reduction 1

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

Reduction 1

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

Goal: Prove that $L_{TM} \leq EMPTY - STRING_{TM}$

Reduction 1

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

Goal: Prove that $L_{TM} \leq EMPTY - STRING_{TM}$

The Reduction: S

Reduction 1

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

Goal: Prove that $L_{TM} \leq EMPTY - STRING_{TM}$

The Reduction: S

On input $\langle M, w \rangle$ (to L_{TM}) do:

Reduction 1

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

Goal: Prove that $L_{TM} \leq EMPTY - STRING_{TM}$

The Reduction: S

On input $\langle M, w \rangle$ (to L_{TM}) do:

- Construct a TM M_2 that does the following
  - On input ϵ, M_2 runs $M(w)$ and returns what it returns
  - On any other input M_2 halts and returns 0

Reduction 1

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

Goal: Prove that $L_{TM} \leq EMPTY - STRING_{TM}$

The Reduction: S

On input $\langle M, w \rangle$ (to L_{TM}) do:

- Construct a TM M_2 that does the following
  - On input ϵ, M_2 runs $M(w)$ and returns what it returns
  - On any other input M_2 halts and returns 0
- Note that M_2 accepts ϵ if and only if $M(w) = 1$

Reduction 1

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

Goal: Prove that $L_{TM} \leq EMPTY - STRING_{TM}$

The Reduction: S

On input $\langle M, w \rangle$ (to L_{TM}) do:

- Construct a TM M_2 that does the following
  - On input ϵ, M_2 runs $M(w)$ and returns what it returns
  - On any other input M_2 halts and returns 0
- Note that M_2 accepts ϵ if and only if $M(w) = 1$
- Assume you have a decider R for $EMPTY - STRING_{TM}$

Reduction 1

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

Goal: Prove that $L_{TM} \leq EMPTY - STRING_{TM}$

The Reduction: S

On input $\langle M, w \rangle$ (to L_{TM}) do:

- Construct a TM M_2 that does the following
  - On input ϵ, M_2 runs $M(w)$ and returns what it returns
  - On any other input M_2 halts and returns 0
- Note that M_2 accepts ϵ if and only if $M(w) = 1$
- Assume you have a decider R for $EMPTY - STRING_{TM}$
- Run $R(\langle M_2 \rangle)$ and output what it outputs

Reduction 1

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

Goal: Prove that $L_{TM} \leq EMPTY - STRING_{TM}$

The Reduction: S

On input $\langle M, w \rangle$ (to L_{TM}) do:

- Construct a TM M_2 that does the following
  - On input ϵ, M_2 runs $M(w)$ and returns what it returns
  - On any other input M_2 halts and returns 0
- Note that M_2 accepts ϵ if and only if $M(w) = 1$
- Assume you have a decider R for $EMPTY - STRING_{TM}$
- Run $R(\langle M_2 \rangle)$ and output what it outputs

$\langle M, w \rangle \rightarrow \langle M_2 \rangle$ s.t. $\langle M, w \rangle \in L_{TM} \iff M_2 \in EQ_{TM}$

- If R decides $EMPTY - STRING_{TM}$, S decides L_{TM}, contradiction

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

Reduction 1

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

Goal: Prove that $L_{TM} \leq EMPTY - STRING_{TM}$

The Reduction: S

On input $\langle M, w \rangle$ (to L_{TM}) do:

- Construct a TM M_2 that does the following
  - On input ϵ, M_2 runs $M(w)$ and returns what it returns
  - On any other input M_2 halts and returns 0
- Note that M_2 accepts ϵ if and only if $M(w) = 1$
- Assume you have a decider R for $EMPTY - STRING_{TM}$
- Run $R(\langle M_2 \rangle)$ and output what it outputs

- If R decides $EMPTY - STRING_{TM}$, S decides L_{TM}, contradiction
- S maps input of L_{TM} to an input of $EMPTY - STRING_{TM}$

$$\langle M, w \rangle \in L_{TM} \iff \langle M_2 \rangle \in EMPTY - STRING_{TM}$$

Reduction 2

$$L_{E_{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

Reduction 2

$$L_{E_{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

Goal: Prove that $L_{TM} \leq L_{E_{TM}}$

Reduction 2

$$L_{E_{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

Goal: Prove that $L_{TM} \leq L_{E_{TM}}$

The Reduction: S

Reduction 2

$$L_{E_{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

Goal: Prove that $L_{TM} \leq L_{E_{TM}}$

The Reduction: S

On input $\langle M, w \rangle$ (to L_{TM}) do:

Reduction 2

$$L_{E_{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

Goal: Prove that $L_{TM} \leq L_{E_{TM}}$

The Reduction: S

On input $\langle M, w \rangle$ (to L_{TM}) do:

- Construct a TM M_2 that does the following
  - On input x, if $x \neq w$, reject
  - If $x = w$, run $M(w)$ and accept if M does

Reduction 2

$$L_{E_{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

Goal: Prove that $L_{TM} \leq L_{E_{TM}}$

The Reduction: S

On input $\langle M, w \rangle$ (to L_{TM}) do:

- Construct a TM M_2 that does the following
  - On input x, if $x \neq w$, reject
  - If $x = w$, run $M(w)$ and accept if M does
- Note that M_2 either accepts only w (if $M(w) = 1$) or $\emptyset$ (if $M(w) \neq 1$)

Reduction 2

$$L_{E_{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

Goal: Prove that $L_{TM} \leq L_{E_{TM}}$

The Reduction: S

On input $\langle M, w \rangle$ (to L_{TM}) do:

- Construct a TM M_2 that does the following
  - On input x, if $x \neq w$, reject
  - If $x = w$, run $M(w)$ and accept if M does
- Note that M_2 either accepts only w (if $M(w) = 1$) or $\emptyset$ (if $M(w) \neq 1$)
- Assume you have a decider R for $L_{E_{TM}}$

Reduction 2

$$L_{E_{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

Goal: Prove that $L_{TM} \leq L_{E_{TM}}$

The Reduction: S

On input $\langle M, w \rangle$ (to L_{TM}) do:

- Construct a TM M_2 that does the following
  - On input x, if $x \neq w$, reject
  - If $x = w$, run $M(w)$ and accept if M does
- Note that M_2 either accepts only w (if $M(w) = 1$) or $\emptyset$ (if $M(w) \neq 1$)
- Assume you have a decider R for $L_{E_{TM}}$
- Run $R(\langle M_2 \rangle)$, and output opposite of what it outputs

Reduction 2

$$L_{E_{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

Goal: Prove that $L_{TM} \leq L_{E_{TM}}$

The Reduction: S

On input $\langle M, w \rangle$ (to L_{TM}) do:

- Construct a TM M_2 that does the following
  - On input x, if $x \neq w$, reject
  - If $x = w$, run $M(w)$ and accept if M does
- Note that M_2 either accepts only w (if $M(w) = 1$) or $\emptyset$ (if $M(w) \neq 1$)
- Assume you have a decider R for $L_{E_{TM}}$
- Run $R(\langle M_2 \rangle)$, and output opposite of what it outputs

- If R decides $L_{E_{TM}}$, S decides L_{TM}, contradiction

Reduction 2

$$L_{E_{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

Goal: Prove that $L_{TM} \leq L_{E_{TM}}$

The Reduction: S

On input $\langle M, w \rangle$ (to L_{TM}) do:

- Construct a TM M_2 that does the following
  - On input x, if $x \neq w$, reject
  - If $x = w$, run $M(w)$ and accept if M does
- Note that M_2 either accepts only w (if $M(w) = 1$) or $\emptyset$ (if $M(w) \neq 1$)
- Assume you have a decider R for $L_{E_{TM}}$
- Run $R(\langle M_2 \rangle)$, and output opposite of what it outputs

- If R decides $L_{E_{TM}}$, S decides L_{TM}, contradiction
- But, S does not map input in L_{TM} into an input in $L_{E_{TM}}$

$$\langle M, w \rangle \in L_{TM} \iff \langle M_2 \rangle \notin L_{E_{TM}}$$

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

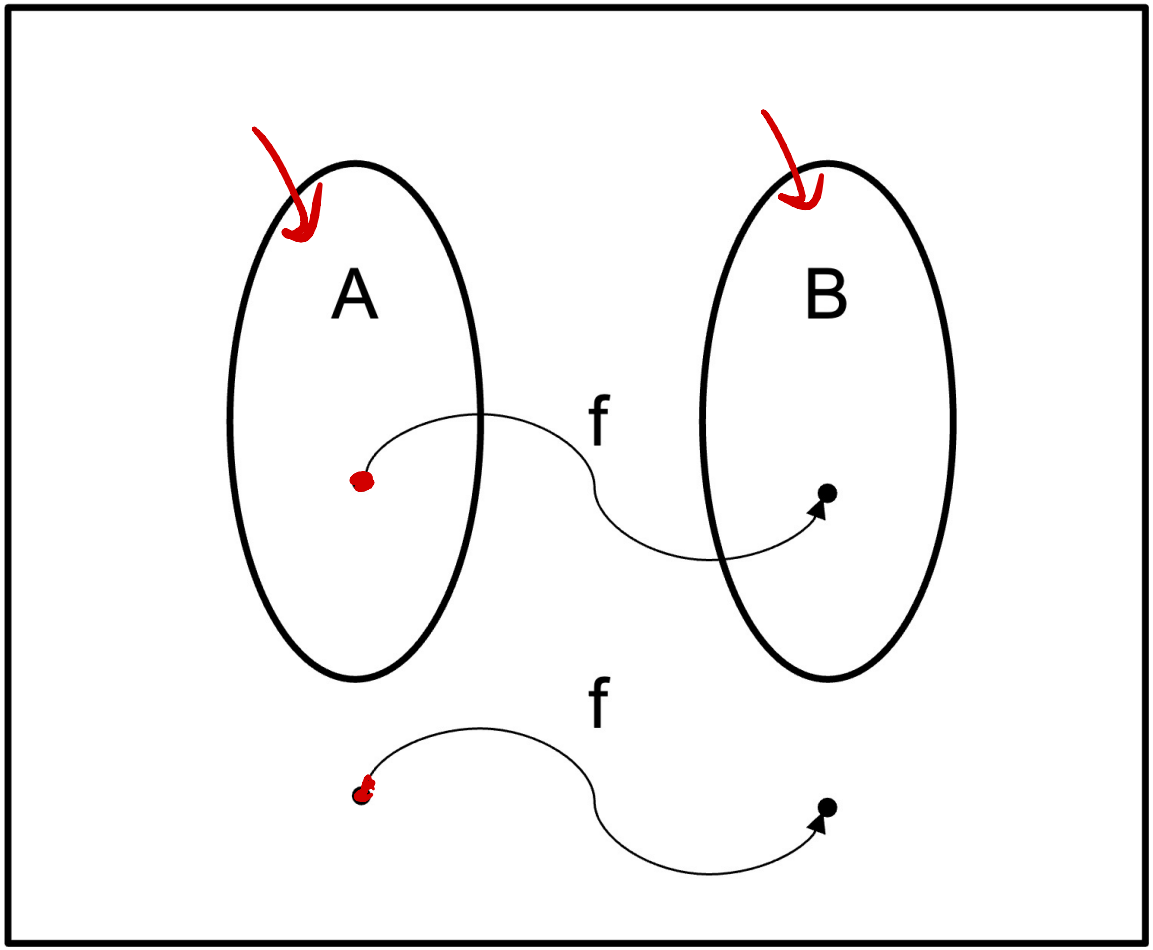
Definition

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

Outline

Information in a String

$$A = 01010101010101010101010101$$

$$B = 11010010001110001011111 1$$

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 TH 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 TH 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 TH M on input w halts with x on its tape
2. The Kolmogorov complexity of x is

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

- $K(x)$ 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 TH M on input w halts with x on its tape
2. The Kolmogorov complexity of x is

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

- $K(x)$ 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. $\forall x, y, K(xy) \leq 2 \cdot K(x) + K(y) + c$

$x = 0\,0\,1$

$0\,0\,0\,0\,1\,1\,0\,1$ w y

$x \cdot 2$ w

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

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

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

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 imcompressible

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 imcompressible

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 imcompressible

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

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 imcompressible

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

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

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 imcompressible

1. Incompressible strings of every length exist
   Proof:
   - There are 2^n strings of length n
   - The number of strings 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 imcompressible

1. Incompressible strings of every length exist
   Proof:
   - There are 2^n strings of length n
   - The number of strings 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 imcompressible

1. Incompressible strings of every length exist
   Proof:
   - There are 2^n strings of length n
   - The number of strings 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 cannot decide whether a string is incompressible