Reducibility and Information

Dr. Chuck Rocca
roccac@wcsu.edu
http://sites.wcsu.edu/roccac

WESTERN CONNECTICUT STATE UNIVERSITY
MACRICOSTAS SCHOOL OF ARTS & SCIENCES

Table of Contents

Table of Contents

Oracles

Definition

An **oracle** for a language B is an external device that is capable of always deciding membership in B. An **oracle Turing machine** is a modification of a Turing machine M denoted M^B that has the added ability to query an oracle for B.

Oracles

Definition

An **oracle** for a language B is an external device that is capable of always deciding membership in B. An **oracle Turing machine** is a modification of a Turing machine M denoted M^B that has the added ability to query an oracle for B.

$M^B =$ *"On input w:*

1. *Query the oracle for B to determine if $w \in B$.*
2. *If the oracle answers YES,* **accept**; *if NO,* **reject**."

Oracles

Definition

An **oracle** for a language B is an external device that is capable of always deciding membership in B. An **oracle Turing machine** is a modification of a Turing machine M denoted M^B that has the added ability to query an oracle for B.

$\overline{M}^B =$ "*On input* w*:*

1. *Query the oracle for* B *to determine if* $w \in B$.
2. *If the oracle answers NO,* **accept**; *if YES,* **reject**."

E_{TM} and A_{TM}

$T^{A_{TM}}$ = "*On input* $\langle M \rangle$, *where* M *is a TM:*

1. *Construct the TM* N.
   N = "*On any input:*
   1. *Run* M *in parallel on all strings in* Σ^*.
   2. *If* M *accepts any of these strings,* **accept**."
2. *Query the oracle for* A_{TM} *to determine if* $\langle N, 0 \rangle \in A_{TM}$.
3. *If the oracle answers NO,* **accept**; *if YES,* **reject**."

E_{TM} and A_{TM}

$T^{A_{TM}}$ = "*On input* $\langle M \rangle$, *where* M *is a TM:*

1. *Construct the TM* N.
   N = "*On any input:*
   1. *Run* M *in parallel on all strings in* Σ^*.
   2. *If* M *accepts any of these strings,* **accept**."
2. *Query the oracle for* A_{TM} *to determine if* $\langle N, 0 \rangle \in A_{TM}$.
3. *If the oracle answers NO,* **accept**; *if YES,* **reject**."

E_{TM} and A_{TM}

$T^{A_{TM}}$ = "*On input* $\langle M \rangle$, *where* M *is a TM:*

1. *Construct the TM* N.
   N = "*On any input:*
   1. *Run* M *in parallel on all strings in* Σ^*.
   2. *If* M *accepts any of these strings,* **accept**."
2. *Query the oracle for* A_{TM} *to determine if* $\langle N, 0 \rangle \in A_{TM}$.
3. *If the oracle answers NO,* **accept**; *if YES,* **reject**."

E_{TM} and A_{TM}

$T^{A_{TM}}$ = "*On input* $\langle M \rangle$, *where* M *is a TM:*

1. *Construct the TM* N.
   N = "*On any input:*
   1. *Run* M *in parallel on all strings in* Σ^*.
   2. *If* M *accepts any of these strings,* **accept**."
2. *Query the oracle for* A_{TM} *to determine if* $\langle N, 0 \rangle \in A_{TM}$.
3. *If the oracle answers NO,* **accept**; *if YES,* **reject**."

Can an oracle for A_{TM} actually exist?

Turing Reducible

Definition

A language A is **Turing decidable** relative to B if there exists an oracle machine M^B which decides A. If this is the case we say that A is Turing reducible to B and write $A \leq_T B$.

Turing Reducible

Definition

A language A is **Turing decidable** relative to B if there exists an oracle machine M^B which decides A. If this is the case we say that A is Turing reducible to B and write $A \leq_T B$.

- For all machines A: $A \leq_T A$

Turing Reducible

Definition

A language A is **Turing decidable** relative to B if there exists an oracle machine M^B which decides A. If this is the case we say that A is Turing reducible to B and write $A \leq_T B$.

- For all machines A: $A \leq_T A$
- For all machines A and their compliments $\overline{A}$: $\overline{A} \leq_T A$

Turing Reducible

Definition

A language A is **Turing decidable** relative to B if there exists an oracle machine M^B which decides A. If this is the case we say that A is Turing reducible to B and write $A \leq_T B$.

- For all machines A: $A \leq_T A$
- For all machines A and their compliments $\overline{A}$: $\overline{A} \leq_T A$
- $E_{TM} \leq_T A_{TM}$

Turing Reduction vs. Mapping Reduction

Recall, A is mapping reducible to B, $A \leq_m B$, if there is a computable function f such that

$$w \in A \Leftrightarrow f(w) \in B.$$

That is, if we can change questions about A into questions about B in some computable way. It is "clear" that $A \leq_m B \Rightarrow A \leq_T B$.

Turing Reduction vs. Mapping Reduction

Recall, A is mapping reducible to B, $A \leq_m B$, if there is a computable function f such that

$$w \in A \Leftrightarrow f(w) \in B.$$

That is, if we can change questions about A into questions about B in some computable way. It is "clear" that $A \leq_m B \Rightarrow A \leq_T B$.

$M^B =$ *"On input w:*

1. *Query the oracle for B to determine if $f(w) \in B$.*
2. *If the oracle answers YES,* **accept**; *if NO,* **reject**."

Turing Reduction vs. Mapping Reduction

Theorem

There exists A and B such that $A \leq_T B$ and $A \not\leq_m B$.

Turing Reduction vs. Mapping Reduction

Theorem

There exists A and B such that $A \leq_T B$ and $A \not\leq_m B$.

- $A = \overline{A_{TM}}$

Turing Reduction vs. Mapping Reduction

Theorem

There exists A and B such that $A \leq_T B$ and $A \not\leq_m B$.

- $A = \overline{A_{TM}}$
- $B = A_{TM}$

Turing Reduction vs. Mapping Reduction

Theorem

There exists A and B such that $A \leq_T B$ and $A \not\leq_m B$.

- $A = \overline{A_{TM}}$
- $B = A_{TM}$
- $\overline{A_{TM}} \leq_T A_{TM}$

Turing Reduction vs. Mapping Reduction

Theorem

There exists A and B such that $A \leq_T B$ and $A \not\leq_m B$.

- $A = \overline{A_{TM}}$
- $B = A_{TM}$
- $\overline{A_{TM}} \leq_T A_{TM}$
- If $\overline{A_{TM}} \leq_m A_{TM}$, then $\overline{A_{TM}}$ would be recognizable, a contradiction.

Problem 6.3 Transitivity

Theorem

If $A \leq_T B$ and $B \leq_T C$, then $A \leq_T C$

Problem 6.3 Transitivity

Theorem

If $A \leq_T B$ and $B \leq_T C$, then $A \leq_T C$

- Suppose M_1^B decides if $w \in A$ using an oracle for B.

Problem 6.3 Transitivity

Theorem

If $A \leq_T B$ and $B \leq_T C$, then $A \leq_T C$

- Suppose M_1^B decides if $w \in A$ using an oracle for B.
- Suppose M_2^C decides if $w \in B$ using an oracle for C.

Problem 6.3 Transitivity

Theorem

If $A \leq_T B$ and $B \leq_T C$, then $A \leq_T C$

- Suppose M_1^B decides if $w \in A$ using an oracle for B.
- Suppose M_2^C decides if $w \in B$ using an oracle for C.
- Define M_3^C by replacing the oracle for B in M_1^B with M_2^C.

Problem 6.3 Transitivity

Theorem

If $A \leq_T B$ and $B \leq_T C$, then $A \leq_T C$

- Suppose M_1^B decides if $w \in A$ using an oracle for B.
- Suppose M_2^C decides if $w \in B$ using an oracle for C.
- Define M_3^C by replacing the oracle for B in M_1^B with M_2^C.

$M_1^B =$ "*On input w: Query the oracle for B to determine if $w \in B$ and so also in A.*"

$M_2^C =$ "*On input w: Query the oracle for C to determine if $w \in C$ and so also in B.*"

Problem 6.3 Transitivity

Theorem

If $A \leq_T B$ and $B \leq_T C$, then $A \leq_T C$

- Suppose M_1^B decides if $w \in A$ using an oracle for B.
- Suppose M_2^C decides if $w \in B$ using an oracle for C.
- Define M_3^C by replacing the oracle for B in M_1^B with M_2^C.

$M_2^C =$ "On input w: Query the oracle for C to determine if $w \in C$ and so also in B."

$M_3^C =$ "On input w: Query M_2^C to determine if $w \in B$ and so also in A."

Decidability

Theorem

If $A \leq_T B$ and B is decidable, then A is decidable.

Decidability

Theorem

If $A \leq_T B$ and B is decidable, then A is decidable.

Given an oracle TM M^B which decides A, replace the oracle for B with the decider for B.

Table of Contents

Comments of Information

- $A = 10101010101010101010101010$
- $B = 11010101010100101100001100 10$

Comments of Information

- $A = 10101010101010101010101010$
- $B = 11010101010100101100001100 10$
- A is "clearly" compressible where as B is not
  - $M =$"On input w and $n \in \mathbb{N}$: repeat w n times."
  - $A = M(\text{"10"}, 14)$

Comments of Information

- $A = 10101010101010101010101010$
- $B = 11010101010100101100001100 10$
- A is "clearly" compressible where as B is not
  - $M =$"On input w and $n \in \mathbb{N}$: repeat w n times."
  - $A = M(\text{“10”}, 14)$
- $\langle M, \text{“01”}, 14 \rangle$

$$|\langle M, \text{“10”}, 14 \rangle| = \underbrace{1011\,0110}_{|\langle M \rangle|}\ \underbrace{11101010001 \cdots 11101010}_{\langle M \rangle}\ \underbrace{10}_{\text{“10”}}\ \underbrace{1110}_{14}$$

Information and Minimal Descriptions

Definition

The **minimal description** of a binary string x, written $d(x)$, is the shortest string $\langle M, w \rangle$ where the TM M prints x on input w. The **descriptive complexity** of x, $K(x)$, is $K(x) = |d(x)|$.

Upper Bound for x and xx

Theorem

For any binary string x

$$\exists c_0 \forall x : K(x) \leq |x| + c_0$$

and

$$\exists c_1 \forall x : K(xx) \leq K(x) + c_1$$

Upper Bound for x and xx

Theorem

For any binary string x

$$\exists c_0 \forall x : K(x) \leq |x| + c_0$$

and

$$\exists c_1 \forall x : K(xx) \leq K(x) + c_1$$

- c_0 is the length of a Turing machine that outputs its input x (the identity function).

Upper Bound for x and xx

Theorem

For any binary string x

$$\exists c_0 \forall x : K(x) \leq |x| + c_0$$

and

$$\exists c_1 \forall x : K(xx) \leq K(x) + c_1$$

- c_0 is the length of a Turing machine that outputs its input x (the identity function).
- c_1 is the length of a machine that runs a Turing machine N on input w and prints the result twice; we run this on $d(x)$, the minimal description of x.

Bounding xy

Suppose $x = 1010$ and $y = 1101$;
we could represent $\langle xy \rangle$ by

- 11001100_01_1101

Bounding xy

Suppose $x = 1010$ and $y = 1101$; we could represent $\langle xy \rangle$ by

- $11001100\llcorner 01\llcorner 1101$

Then $\exists c \forall x, y$ so that $K(xy) = |d(xy)|$ is bounded by

- $2K(x) + K(y) + c$

Bounding xy

Suppose $x = 1010$ and $y = 1101$; we could represent $\langle xy \rangle$ by

- 11001100␣01␣1101
- 110000␣01␣1010␣1101

Then $\exists c \forall x, y$ so that $K(xy) = |d(xy)|$ is bounded by

- $2K(x) + K(y) + c$

Bounding xy

Suppose $x = 1010$ and $y = 1101$; we could represent $\langle xy \rangle$ by

- 11001100␣01␣1101
- 110000␣01␣1010␣1101

Then $\exists c \forall x, y$ so that $K(xy) = |d(xy)|$ is bounded by

- $2K(x) + K(y) + c$
- $2\log_2(K(x)) + K(x) + K(y) + c$

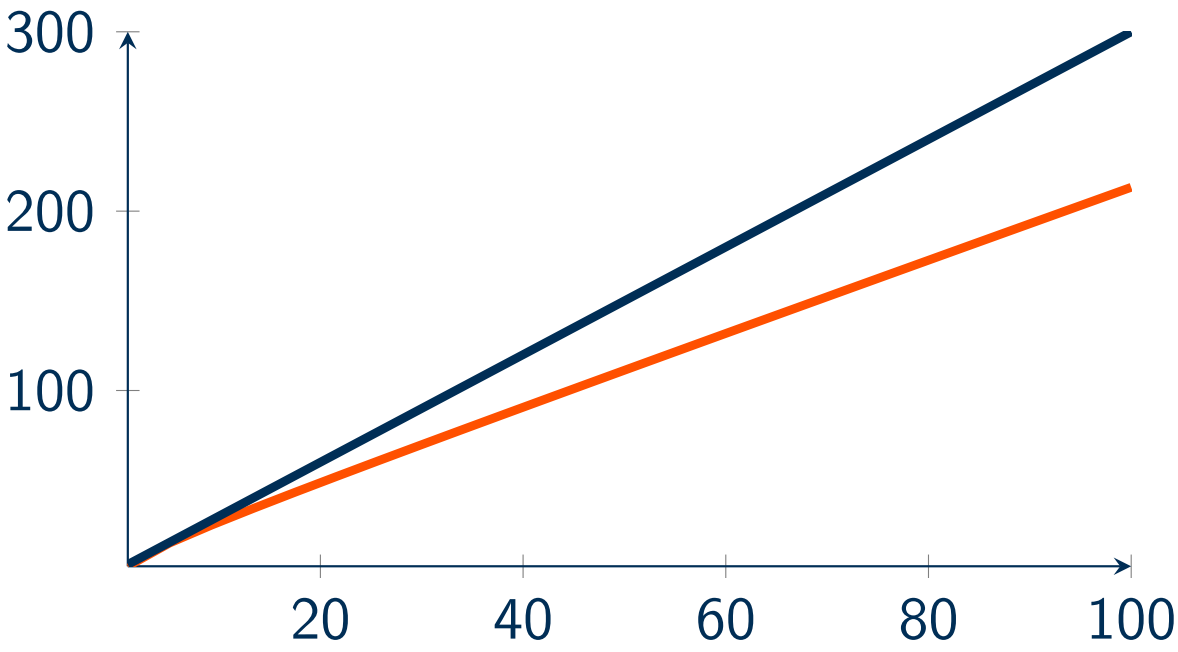
Bounding xy

Suppose $x = 1010$ and $y = 1101$; we could represent $\langle xy \rangle$ by

- 11001100_01_1101
- $110000\_01\_1010\_1101$

Then $\exists c \forall x, y$ so that $K(xy) = |d(xy)|$ is bounded by

- $2K(x) + K(y) + c$
- $2\log_2(K(x)) + K(x) + K(y) + c$



Descriptive Languages

Theorem

For any descriptive language p (programming language or language language), a fixed constant exists that depends only on p such that

$$\forall x : K(x) \leq K_p(x) + c,$$

where $K_p(x) = |d_p(x)|$ *is the descriptive complexity of a program in language p which prints x.*

Descriptive Languages

Theorem

For any descriptive language p (programming language or language language), a fixed constant exists that depends only on p such that

$$\forall x : K(x) \leq K_p(x) + c,$$

where $K_p(x) = |d_p(x)|$ is the descriptive complexity of a program in language p which prints x.

- Let $K_p(x) = |d_p(x)|$ as above.

Descriptive Languages

Theorem

For any descriptive language p (programming language or language language), a fixed constant exists that depends only on p such that

$$\forall x : K(x) \leq K_p(x) + c,$$

where $K_p(x) = |d_p(x)|$ *is the descriptive complexity of a program in language p which prints x.*

- Let $K_p(x) = |d_p(x)|$ as above.
- Let M be an interpreter for p with $|\langle M \rangle| = c$.

Descriptive Languages

Theorem

For any descriptive language p (programming language or language language), a fixed constant exists that depends only on p such that

$$\forall x : K(x) \leq K_p(x) + c,$$

where $K_p(x) = |d_p(x)|$ *is the descriptive complexity of a program in language p which prints x.*

- Let $K_p(x) = |d_p(x)|$ as above.
- Let M be an interpreter for p with $|\langle M \rangle| = c$.
- Then, $\langle M \rangle\, d_p(x)$ prints x and

$$K(x) \leq K_p(x) + |\langle M \rangle| = K_p(x) + c$$

Table of Contents

c-Compressible

Definition

Given a string x and constant c we say that x is c-**compressible** if

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

If x is not c-compressible, we say that x is **incompressible by** c. If x is incompressible by 1, we say that x is **incompressible**.

c-Compressible

Definition

Given a string x and constant c we say that x is c-**compressible** if

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

If x is not c-compressible, we say that x is **incompressible by** c. If x is incompressible by 1, we say that x is **incompressible**.

- $A = 10101010101010101010101010$

c-Compressible

Definition

Given a string x and constant c we say that x is c-**compressible** if

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

If x is not c-compressible, we say that x is **incompressible by** c. If x is incompressible by 1, we say that x is **incompressible**.

- $A = 10101010101010101010101010$
- $B = 11010101010100101100001100 10$

c-Compressible

Definition

Given a string x and constant c we say that x is c-**compressible** if

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

If x is not c-compressible, we say that x is **incompressible by** c. If x is incompressible by 1, we say that x is **incompressible**.

- $A = 10101010101010101010101010$
- $B = 11010101010100101100001100 10$
- $xy = 11001100\_01\_1101$

c-Compressible

Definition

Given a string x and constant c we say that x is c-**compressible** if

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

If x is not c-compressible, we say that x is **incompressible by** c. If x is incompressible by 1, we say that x is **incompressible**.

- $A = 10101010101010101010101010$
- $B = 11010101010100101100001100 10$
- $xy = 11001100\_01\_1101$
- $xy = 110000\_01\_1010\_1101$

Incompressible by c

Theorem

There exists at least $2^n - (2^{n-c+1} - 1)$ strings of length n which are incompressible by c.

Incompressible by c

Theorem

There exists at least $2^n - (2^{n-c+1} - 1)$ strings of length n which are incompressible by c.

- 2^n strings of length n

Incompressible by c

Theorem

There exists at least $2^n - (2^{n-c+1} - 1)$ strings of length n which are incompressible by c.

- 2^n strings of length n
- $\sum_{i=0}^{n-c} 2^i = 1 + 2 + 4 + \cdots + 2^{n-c} = 2^{n-c+1} - 1$

Incompressible by c

Theorem

There exists at least $2^n - (2^{n-c+1} - 1)$ strings of length n which are incompressible by c.

- 2^n strings of length n
- $\sum_{i=0}^{n-c} 2^i = 1 + 2 + 4 + \cdots + 2^{n-c} = 2^{n-c+1} - 1$
- # of "c-shorter" descriptions $<$ # of length n descriptions

Incompressible by c

Theorem

There exists at least $2^n - (2^{n-c+1} - 1)$ strings of length n which are incompressible by c.

- 2^n strings of length n
- $\sum_{i=0}^{n-c} 2^i = 1 + 2 + 4 + \cdots + 2^{n-c} = 2^{n-c+1} - 1$
- # of "c-shorter" descriptions $<$ # of length n descriptions
- $\therefore$ there exists strings incompressible by c

Incompressible by c

Theorem

There exists at least $2^n - (2^{n-c+1} - 1)$ strings of length n which are incompressible by c.

- 2^n strings of length n
- $\sum_{i=0}^{n-c} 2^i = 1 + 2 + 4 + \cdots + 2^{n-c} = 2^{n-c+1} - 1$
- # of "c-shorter" descriptions $<$ # of length n descriptions
- $\therefore$ there exists strings incompressible by c
- Specifically, $2^n - (2^{n-c+1} - 1)$

Incompressible Strings

Corollary

There exists incompressible strings.

Incompressible Strings

Corollary

There exists incompressible strings.

- let $c = 1$ and apply the theorem

Incompressible Strings

Corollary

There exists incompressible strings.

- let $c = 1$ and apply the theorem
- $\therefore$ there exists incompressible strings

Computable Properties and Compressibility

Definition

A **property** of a string x is a binary function f, i.e. x either has the property or it does not. Given $F_n = \{x | f(x) = FALSE \text{ and } |x| \leq n\}$, a property holds for **almost all** strings if

$$\frac{|F_n|}{|All\ Strings\ with\ |x| \leq n|} \longrightarrow 0$$

as n goes to infinity.

Computable Properties and Compressibility

Theorem

Given a computable property f that holds for almost all strings, and $b > 0$, then the property f is FALSE for only finitely many strings that are incompressible by b.

Computable Properties and Compressibility (proof part 1)

- $F_n = \{x | f(x) = FALSE \text{ and } |x| \leq n\}$

Computable Properties and Compressibility (proof part 1)

- $F_n = \{x | f(x) = FALSE \text{ and } |x| \leq n\}$
- $F_\infty = \{x | f(x) = FALSE\}$, F_∞ is enumerable

Computable Properties and Compressibility (proof part 1)

- $F_n = \{x | f(x) = FALSE \text{ and } |x| \leq n\}$
- $F_\infty = \{x | f(x) = FALSE\}$, F_∞ is enumerable
- Let M be a machine that finds the i^{th} string s in F_∞

Computable Properties and Compressibility (proof part 1)

- $F_n = \{x | f(x) = FALSE \text{ and } |x| \leq n\}$
- $F_\infty = \{x | f(x) = FALSE\}$, F_∞ is enumerable
- Let M be a machine that finds the i^{th} string s in F_∞
- For each $x \in F_\infty$, let i_x be its position in F_∞

Computable Properties and Compressibility (proof part 1)

- $F_n = \{x | f(x) = FALSE \text{ and } |x| \leq n\}$
- $F_\infty = \{x | f(x) = FALSE\}$, F_∞ is enumerable
- Let M be a machine that finds the i^{th} string s in F_∞
- For each $x \in F_\infty$, let i_x be its position in F_∞
- Thus $\langle M, i_x \rangle$ returns x and $|d(x)| = K(x) \leq |i_x| + c$

Computable Properties and Compressibility (proof part 1)

- $F_n = \{x | f(x) = FALSE \text{ and } |x| \leq n\}$
- $F_\infty = \{x | f(x) = FALSE\}$, F_∞ is enumerable
- Let M be a machine that finds the i^{th} string s in F_∞
- For each $x \in F_\infty$, let i_x be its position in F_∞
- Thus $\langle M, i_x \rangle$ returns x and $|d(x)| = K(x) \leq |i_x| + c$
- Choose n so that

$$\frac{|F_n|}{|All\ Strings\ with\ |x| \leq n|} \leq \frac{1}{2^{b+c+1}}$$

Computable Properties and Compressibility (proof part 2)

- Choose n so that

$$\frac{|F_n|}{|All\ Strings\ with\ |x| \leq n|} \leq \frac{1}{2^{b+c+1}}$$

Computable Properties and Compressibility (proof part 2)

- Choose n so that

$$\frac{|F_n|}{|All\ Strings\ with\ |x| \leq n|} \leq \frac{1}{2^{b+c+1}}$$

- There are $2^{n+1} - 1$ strings of length n, so

$$i_x \leq \frac{2^{n+1} - 1}{2^{b+c+1}} < \frac{2^{n+1}}{2^{b+c+1}} = 2^{n-b-c}$$

Computable Properties and Compressibility (proof part 2)

- Choose n so that

$$\frac{|F_n|}{|All\ Strings\ with\ |x| \leq n|} \leq \frac{1}{2^{b+c+1}}$$

- There are $2^{n+1} - 1$ strings of length n, so

$$i_x \leq \frac{2^{n+1} - 1}{2^{b+c+1}} < \frac{2^{n+1}}{2^{b+c+1}} = 2^{n-b-c}$$

- Then for all x such that $|x| \geq n$ and $f(x) = FALSE$

$$|d(x)| = K(x) \leq |i_x| + c \leq (n - b - c) + c = n - b$$

Computable Properties and Compressibility (proof part 2)

- Choose n so that

$$\frac{|F_n|}{|All\ Strings\ with\ |x| \leq n|} \leq \frac{1}{2^{b+c+1}}$$

- There are $2^{n+1} - 1$ strings of length n, so

$$i_x \leq \frac{2^{n+1} - 1}{2^{b+c+1}} < \frac{2^{n+1}}{2^{b+c+1}} = 2^{n-b-c}$$

- Then for all x such that $|x| \geq n$ and $f(x) = FALSE$

$$|d(x)| = K(x) \leq |i_x| + c \leq (n - b - c) + c = n - b$$

- $\therefore$ If x is incompressible by b, then its length is less than n, so there are only finitely many such x.

Nearly Incompressible

Theorem

There exists $b > 0$, for all strings x, such that the minimal description $d(x)$ of x is incompressible by b.

Nearly Incompressible

Theorem

There exists $b > 0$, for all strings x, such that the minimal description $d(x)$ of x is incompressible by b.

- Define a machine M:
  $M =$ "On input $\langle R, y \rangle$:

Nearly Incompressible

Theorem

There exists $b > 0$, for all strings x, such that the minimal description $d(x)$ of x is incompressible by b.

- Define a machine M:
  $M =$ "On input $\langle R, y \rangle$:
  1. Run R on y and **reject** if the output is not of the form $\langle S, z \rangle$.

Nearly Incompressible

Theorem

There exists $b > 0$, for all strings x, such that the minimal description $d(x)$ of x is incompressible by b.

- Define a machine M:
  $M =$ "On input $\langle R, y \rangle$:
  1. Run R on y and **reject** if the output is not of the form $\langle S, z \rangle$.
  2. Run S on z and halt with its output on the tape."

Nearly Incompressible

Theorem

There exists $b > 0$, for all strings x, such that the minimal description $d(x)$ of x is incompressible by b.

- Define a machine M:
  $M =$"On input $\langle R, y \rangle$:
  1. Run R on y and **reject** if the output is not of the form $\langle S, z \rangle$.
  2. Run S on z and halt with its output on the tape."
- Let $b = |\langle M \rangle| + 1$ so that $|\langle M \rangle| = b - 1$

Nearly Incompressible

Theorem

There exists $b > 0$, for all strings x, such that the minimal description $d(x)$ of x is incompressible by b.

- Define a machine M:
  $M =$"On input $\langle R, y \rangle$:
  1. Run R on y and **reject** if the output is not of the form $\langle S, z \rangle$.
  2. Run S on z and halt with its output on the tape."
- Let $b = |\langle M \rangle| + 1$ so that $|\langle M \rangle| = b - 1$
- If $d(x)$ is b-compressible:

$$\begin{aligned} |\langle M, d(d(x)) \rangle| &\leq |\langle M \rangle| + |d(d(x))| \\ &\leq (b-1) + |d(x)| - b = |d(x)| - 1 \end{aligned}$$

Nearly Incompressible

Theorem

There exists $b > 0$, for all strings x, such that the minimal description $d(x)$ of x is incompressible by b.

- Define a machine M:
  $M =$"On input $\langle R, y \rangle$:
  1. Run R on y and **reject** if the output is not of the form $\langle S, z \rangle$.
  2. Run S on z and halt with its output on the tape."
- Let $b = |\langle M \rangle| + 1$ so that $|\langle M \rangle| = b - 1$
- If $d(x)$ is b-compressible:

$$\begin{aligned} |\langle M, d(d(x)) \rangle| &\leq |\langle M \rangle| + |d(d(x))| \\ &\leq (b-1) + |d(x)| - b = |d(x)| - 1 \end{aligned}$$

- But $d(x)$ is minimal, so this is a contradiction.

Table of Contents

Next Class

- Have a Good Summer

Reducibility and Information

Dr. Chuck Rocca
roccac@wcsu.edu
http://sites.wcsu.edu/roccac

WESTERN CONNECTICUT STATE UNIVERSITY
MACRICOSTAS SCHOOL OF ARTS & SCIENCES