CS 154

Lecture 12:
Foundations of Math and Kolmogorov Complexity
Computability and the Foundations of Mathematics
The Foundations of Mathematics

A *formal system* describes a formal language for
- writing (finite) mathematical statements,
- has a definition of what statements are “true”
- has a definition of a proof of a statement

Example: Every TM M defines some formal system $\mathcal{F}$

- \{Mathematical statements in $\mathcal{F}$\} = $\Sigma^*$
  String $w$ represents the statement “M accepts $w$”
- \{True statements in $\mathcal{F}$\} = $L(M)$
- A proof that “M accepts $w$” can be defined to be an accepting computation history for M on $w$
Consistency and Completeness

A formal system $F$ is **consistent** or **sound** if no false statement has a valid proof in $F$
(Proof in $F$ implies Truth in $F$)

A formal system $F$ is **complete** if every true statement has a valid proof in $F$
(Truth in $F$ implies Proof in $F$)
Interesting Formal Systems

Define a formal system $\mathcal{F}$ to be *interesting* if:

1. Any mathematical statement about computation can be (computably) described as a statement of $\mathcal{F}$. Given $(M, w)$, there is a (computable) $S_{M,w}$ in $\mathcal{F}$ such that $S_{M,w}$ is true in $\mathcal{F}$ if and only if $M$ accepts $w$.

2. Proofs are “convincing” – a TM can check that a proof of a theorem is correct. This set is decidable: $\{(S, P) \mid P$ a proof of $S$ in $\mathcal{F}\}$

3. If $S$ is in $\mathcal{F}$ and there is a proof of $S$ describable as a computation, then there’s a proof of $S$ in $\mathcal{F}$. If $M$ accepts $w$, then there is a proof $P$ in $\mathcal{F}$ of $S_{M,w}$
Limitations on Mathematics

For every consistent and interesting $F$,

Theorem 1. (Gödel 1931) $F$ is incomplete: There are mathematical statements in $F$ that are true but cannot be proved in $F$.

Theorem 2. (Gödel 1931) The consistency of $F$ cannot be proved in $F$.

Theorem 3. (Church-Turing 1936) The problem of checking whether a given statement in $F$ has a proof is undecidable.
Proof: Define Turing machine G(x):
1. Obtain own description G [Recursion Theorem]
2. Construct statement \( S' = \neg S_G, \varepsilon \)
3. Search for a proof of \( S' \) in \( F \) over all finite length strings. Accept if a proof is found.

Claim: \( S' \) is true in \( F \), but has no proof in \( F \)
\( S' \) basically says “There is no proof of \( S' \) in \( F \)”
(Gödel 1931) The consistency of $\mathcal{F}$ cannot be proved within any interesting consistent $\mathcal{F}$

Proof: Suppose we can prove “$\mathcal{F}$ is consistent” in $\mathcal{F}$
We constructed $\neg S_{G,\epsilon} = "G$ does not accept $\epsilon"$ which we showed is true, but has no proof in $\mathcal{F}$
$G$ does not accept $\epsilon$ $\iff$ There is no proof of $\neg S_{G,\epsilon}$ in $\mathcal{F}$

But if there’s a proof in $\mathcal{F}$ of “$\mathcal{F}$ is consistent” then there is a proof in $\mathcal{F}$ of $\neg S_{G,\epsilon}$ (here’s the proof):

“If $S_{G,\epsilon}$ is true, then there is a proof in $\mathcal{F}$ of $\neg S_{G,\epsilon}$. $\mathcal{F}$ is consistent, therefore $\neg S_{G,\epsilon}$ is true. But $S_{G,\epsilon}$ and $\neg S_{G,\epsilon}$ cannot both be true. Therefore, $\neg S_{G,\epsilon}$ is true”

This contradicts the previous theorem.
Proof: Suppose \( \text{PROVABLE}_\mathcal{F} \) is decidable with TM \( P \).

Then we can decide \( A_{\text{TM}} \) using the following procedure:

On input \((M, w)\), run the TM \( P \) on input \( S_{M,w} \)

If \( P \) accepts, examine all possible proofs in \( \mathcal{F} \)

If a proof of \( S_{M,w} \) is found then accept
If a proof of \( \neg S_{M,w} \) is found then reject

If \( P \) rejects, then reject.

Why does this work?
Kolmogorov Complexity: A Universal Theory of Data Compression
The Church-Turing Thesis:

Everyone’s
Intuitive Notion  =  Turing Machines of Algorithms

This is not a theorem –
* it is a falsifiable scientific hypothesis.  

A Universal Theory of Computation
Is there a Universal Theory of \textit{Information}?

Can we quantify how much \textit{information} is contained in a string?

A = 01010101010101010101010101010101

B = 110010011101110101101001011001011

Idea: The more we can “compress” a string, the less “information” it contains....
Information as Description

Thesis: The amount of information in a string = Shortest way of describing that string

How should we “describe” strings?

Use Turing machines with inputs!

Let $x \in \{0,1\}^*$

**Definition:** The shortest description of $x$, denoted as $d(x)$, is the lexicographically shortest string $<M,w>$ such that $M(w)$ halts with only $x$ on its tape.
A Specific Pairing Function

Theorem. There is a 1-1 computable function $\langle , \rangle : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ and computable functions $\pi_1$ and $\pi_2 : \Sigma^* \rightarrow \Sigma^*$ such that:

$$z = \langle M, w \rangle \iff \pi_1(z) = M \text{ and } \pi_2(z) = w$$

For $x_i \in \Sigma$, let $Z(x_1 \ x_2 \ldots \ x_k) = 0 \ x_1 \ 0 \ x_2 \ldots \ 0 \ x_k \ 1$

Then we can define:

$$\langle M, w \rangle := Z(M) \ w$$

(Example: $\langle 10110,101 \rangle = 01000101001101$)

Note that $|\langle M, w \rangle| = 2|M| + |w| + 1$
Kolmogorov Complexity (1960’s)

**Definition:** The shortest description of $x$, denoted as $d(x)$, is the lexicographically shortest string $<M, w>$ such that $M(w)$ halts with only $x$ on its tape.

**Definition:** The Kolmogorov complexity of $x$, denoted as $K(x)$, is $|d(x)|$.

**EXAMPLES??**
Let’s first determine some properties of $K$. Examples will fall out of this.
Kolmogorov Complexity

Theorem: There is a fixed $c$ so that for all $x$ in $\{0,1\}^*$

\[ K(x) \leq |x| + c \]

“The amount of information in $x$ isn’t much more than $|x|$”

Proof: Define a TM $M$ = “On input $w$, halt.”

On any string $x$, $M(x)$ halts with $x$ on its tape.

Let $c = 2|M| + 1$

Then $K(x) \leq |<M,x>| \leq 2|M| + |x| + 1 \leq |x| + c$
Repetitive Strings have Low Information

Theorem: There is a fixed \( c \) so that for all \( x \in \{0,1\}^* \)
\[
K(xx) \leq K(x) + c
\]

“The information in xx isn’t much more than that in x”

Proof: Let \( N = “\text{On } <M,w>, \text{ let } s = M(w). \text{ Print } ss.” \)

Suppose \( <M,w> \) is the shortest description of \( x \).
Then \( <N,<M,w>> \) is a description of \( xx \)

Therefore
\[
K(xx) \leq |<N,<M,w>>| \leq 2|N| + |<M,w>| + 1
\leq 2|N| + K(x) + 1 \leq c + K(x)
\]
Repetitive Strings have Low Information

Corollary: There is a fixed $c$ so that for all $n \geq 2$, and all $x \in \{0,1\}^*$, $K(x^n) \leq K(x) + c \log n$

“The information in $x^n$ isn’t much more than that in $x$”

Proof: Define the TM

$N = \text{“On input } <n,M,w>, \text{ Let } x = M(w). \text{ Print } x \text{ for } n \text{ times.”}$

Let $<M,w>$ be the shortest description of $x$.

Then $K(x^n) \leq K(<N,<n,M,w>>) \leq 2|N| + d \log n + K(x) \\
\leq c \log n + K(x)$

for some constant $c$ and $d$
Repetitive Strings have Low Information

Corollary: There is a fixed $c$ so that for all $n \geq 2$, and all $x \in \{0,1\}^*$, $K(x^n) \leq K(x) + c \log n$

“The information in $x^n$ isn’t much more than that in $x$”

Recall:

$A = 01010101010101010101010101010101$

For $w = (01)^n$, $K(w) \leq K(01) + c \log_2 n$

So for all $n$, $K((01)^n) \leq d + c \log_2 n$ for a fixed $c, d$
Does The Computational Model Matter?

Turing machines are one “programming language.” If we use other programming languages, could we get significantly shorter descriptions?

An interpreter is a “semi-computable” function

\[ p : \Sigma^* \rightarrow \Sigma^* \]

Takes programs as input, and (may) print their outputs

Definition: Let \( x \in \{0,1\}^* \). The shortest description of \( x \) under \( p \), called \( d_p(x) \), is the lexicographically shortest string \( w \) for which \( p(w) = x \).

Definition: The \( K_p \) complexity of \( x \) is \( K_p(x) := |d_p(x)| \).
Does The Computational Model Matter?

Theorem: For every interpreter $p$, there is an integer $c$ so that for all $x \in \{0,1\}^*$, $K(x) \leq K_p(x) + c$

Moral: Using another programming language would only change $K(x)$ by some additive constant

Proof: Define $M =$ “On $w$, simulate $p(w)$ and write its output to tape”

Then $<M,d_p(x)>$ is a description of $x$, and

$$K(x) \leq |<M,d_p(x)>|$$

$$\leq 2|M| + K_p(x) + 1 \leq c + K_p(x)$$
There Exist Incompressible Strings

Theorem: For all $n$, there is an $x \in \{0,1\}^n$ such that $K(x) \geq n$

“There are incompressible strings of every length”

Proof: $(\text{Number of binary strings of length } n) = 2^n$
but $(\text{Number of descriptions of length } < n) \leq (\text{Number of binary strings of length } < n)$
$= 1 + 2 + 4 + \cdots + 2^{n-1} = 2^n - 1$

Therefore there is at least one $n$-bit string $x$ that does not have a description of length $< n$
Random Strings Are Incompressible!

Theorem: For all $n$ and $c \geq 1$,

$$\Pr_{x \in \{0,1\}^n}[K(x) \geq n-c] \geq 1 - \frac{1}{2^c}$$

“Most strings are highly incompressible”

Proof: (Number of binary strings of length $n$) = $2^n$
but (Number of descriptions of length < $n-c$)
\leq (Number of binary strings of length < $n-c$)
= $2^{n-c} - 1$

Hence the probability that a random $x$ satisfies

$K(x) < n-c$

is at most $(2^{n-c} - 1)/2^n < 1/2^c$. 
Kolmogorov Complexity: Try it!

Give short algorithms for generating the strings:

1. 01000110110000010100111001011101110000
2. 123581321345589144233377610987
3. 126241207205040403203628803628800
Kolmogorov Complexity: Try it!

Give short algorithms for generating the strings:

1. 01000110110000010100111001011101110000
2. 123581321345589144233377610987
3. 126241207205040403203628803628800
Kolmogorov Complexity: Try it!

Give short algorithms for generating the strings:

1. 0100011010111000010100111001011101110000

2. 123581321345589144233377610987

3. 126241207205040403203628803628800
Kolmogorov Complexity: Try it!

Give short algorithms for generating the strings:

1. 01000110110000010100111001011101110000
2. 123581321345589144233377610987
3. 126241207205040403203628803628800

This seems hard to determine in general. Why? We’ll give a formal answer in just one moment...
KOLMOGOROV DIRECTIONS

HOW DO I GET TO YOUR PLACE FROM LEXINGTON?

Hmm...

OK, STARTING FROM YOUR DRIVEWAY, TAKE EVERY LEFT THAT DOESN'T PUT YOU ON A PRIME-NUMBERED HIGHWAY OR STREET NAMED FOR A PRESIDENT.

WHEN PEOPLE ASK FOR STEP-BY-STEP DIRECTIONS, I WORRY THAT THERE WILL BE TOO MANY STEPS TO REMEMBER, SO I TRY TO PUT THEM IN MINIMAL FORM.
Determining Compressibility

Can an algorithm perform optimal compression? Can algorithms tell us if a given string is compressible?

\[ \text{COMPRESS} = \{ (x,c) \mid K(x) \leq c \} \]

**Theorem:** COMPRESS is undecidable!

**Intuition:** If decidable, we could design an algorithm that prints the shortest incompressible string of length \( n \)

*But such a string could then be succinctly described, by providing the algorithm code and \( n \) in binary!*

**Berry Paradox:** “The smallest integer that cannot be defined in less than thirteen words.”
Determining Compressibility

\[
\text{COMPRESS} = \{(x, c) \mid K(x) \leq c\}
\]

Theorem: COMPRESS is undecidable!

Proof: Suppose it’s decidable. Consider the TM:

\[M = \text{“On input } x \in \{0,1\}^*, \text{ interpret } x \text{ as a number } N. \]
\[\text{For all } y \in \{0,1\}^* \text{ in lexicographical order,}
\]
\[\text{If } (y,N) \notin \text{COMPRESS then print } y \text{ and halt.”}
\]

M(x) prints the shortest string y’ with K(y’) > N.

But <M,x> describes y’, and |<M,x>| ≤ d + log N

So N < K(y’) ≤ d + 2 log N. CONTRADICTION!