CS 154

Foundations of Math and Kolmogorov Complexity
Computability and 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 $F$
- $\{\text{Mathematical statements in } F\} = \Sigma^*$
  - String $w$ represents the statement “M accepts $w$”
- $\{\text{True statements in } F\} = L(M)$
- A proof that “M accepts $w$” can be defined to be an accepting computation history for M on $w$
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:* $\{(S, P) \mid P \text{ is a proof of } S \text{ in } \mathcal{F}\}$

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 $S_{M,w}$*
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 $F$)
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 in $F$ 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.
Unprovable Truths in Mathematics

(Gödel) Every consistent interesting $\mathcal{F}$ is incomplete: there are true statements that cannot be proved.

Let $S_{M, w}$ in $\mathcal{F}$ be true if and only if $M$ accepts $w$

Proof: Define Turing machine $G(x)$:

1. Obtain own description $G$ [Recursion Theorem!]
2. Construct statement $S' = \neg S_{G, x}$
3. Search for a proof of $S'$ in $\mathcal{F}$ over all finite length strings. Accept if a proof is found.

Claim: $S'$ is true in $\mathcal{F}$, but has no proof in $\mathcal{F}$

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

Proof: Suppose we can prove “$F$ is consistent” in $F$

We constructed $\neg S_{G, x} = \text{“G does not accept x”}$

which we showed is true, but has no proof in $F$

$G$ does not accept $x \iff$ There is no proof of $\neg S_{G, x}$ in $F$

But if there’s a proof in $F$ of “$F$ is consistent” then

there is a proof in $F$ of $\neg S_{G, x}$ (here’s the proof):

“If $S_{G, x}$ is true, then there is a proof in $F$ of $\neg S_{G, x}$.

$F$ is consistent, therefore $\neg S_{G, x}$ is true.

But $S_{G, x}$ and $\neg S_{G, x}$ cannot both be true.

Therefore, $\neg S_{G, x}$ is true”

This contradicts the previous theorem.
Undecidability in Mathematics

PROVABLE$_F = \{S \mid \text{there's a proof in } F \text{ of } S, \text{ or there's a proof in } F \text{ of } \neg S\}

(Church-Turing 1936) For every interesting consistent $F$, PROVABLE$_F$ is undecidable

Proof: Suppose PROVABLE$_F$ is decidable with TM P.

Then we can decide $A_{TM}$ 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 $F$

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
A Universal Theory of *Information*?

Can we quantify how much *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 $x$ is the length of the *shortest description* of $x$

How should we “describe” strings?

Use Turing machines with inputs!

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

**Def:** A *description of $x$* is a string $<M,w>$ such that

$M$ on input $w$ halts with only $x$ on its tape.

**Def:** 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 
\(<,> : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*\) and computable functions 
\(\pi_1 \) and \(\pi_2 : \Sigma^* \rightarrow \Sigma^*\) such that:

\[ z = <M,w> \iff \pi_1(z) = M \text{ and } \pi_2(z) = w \]

Define: \( <M,w> := 0^{|M|}1 \ M \ w \)

(Example: \(<10110,101> = 00000110110101\) )

Note that \( |<M,w>| = 2|M| + |w| + 1 \)
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.
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.
Observe that $<M,x>$ is a description of $x$.

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

Theorem: 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 constants $c$ and $d$
Repetitive Strings have Low K-Complexity

Theorem: 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 $w = (01)^n$, we have $K(w) \leq K(01) + c \log n$

So for all $n$, $K((01)^n) \leq d + c \log n$ 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 \( K_p(x) := |d_p(x)| \).
Theorem: For every interpreter \( p \), there is a fixed \( c \) 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 \( M = \text{"On } w, \text{ simulate } p(w) \text{ and write its output to tape"} \)

Then \( <M,d_p(x)> \) is a description of \( x \), so

\[
K(x) \leq |<M,d_p(x)>| \\
\leq 2|M| + K_p(x) + 1 \leq c + K_p(x)
\]
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: (Number of binary strings of length \( n \)) = \( 2^n \)
but (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 < $n-c$) 
\[ \leq (\text{Number of binary strings of length } < n-c) \]
\[ = 2^{n-c} - 1 \]

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. 010001101100000101001110010111011000
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?
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!

**Idea:** 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?

COMPRESSION = \{(x,c) \mid K(x) \leq c\}

Theorem: COMPRESSION is undecidable!

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

M = “On input x ∈ \{0,1\}*, let N = 2^{|x|}.

For all y ∈ \{0,1\}* in lexicographical order,

If (y,N) ∉ COMPRESSION then print y and halt.”

M(x) prints the shortest string y’ with K(y’) > 2^{|x|}.

<M,x> is a description of y’, and |<M,x>| ≤ d + |x|

So 2^{|x|} < K(y’) ≤ d + |x|. CONTRADICTION for large x!
Yet Another Proof that $A_{TM}$ is Undecidable!

$COMPRESS = \{ (x,c) \mid K(x) \leq c \}$

Theorem: $A_{TM}$ is undecidable.

Proof: Reduction from $COMPRESS$ to $A_{TM}$.
Given a pair $(x,c)$, our reduction constructs a TM:

$M_{x,c} = \text{On input } w,$

For all pairs $<M',w'>$ with $|<M',w'>| \leq c$, simulate each $M'$ on $w'$ in parallel.

If some $M'$ halts and prints $x$, then accept.

$K(x) \leq c \iff M_{x,c}$ accepts $\varepsilon$
More on Interesting Formal Systems

A formal system $F$ is *interesting* if it is finite and:

1. Any mathematical statement about computation can also be effectively described within $F$.
   
   For all strings $x$ and integers $c$, there is a $S_{x,c}$ in $F$ that is equivalent to “$K(x) \geq c$”

2. Proofs are convincing: it should be possible to check that a proof of a theorem is correct
   
   This set is decidable: $\{ (S,P) \mid P$ is a proof of $S$ in $F \}$
The Unprovable Truth About K-Complexity

Theorem: For every interesting consistent $\mathcal{F}$, there is a $t$ s.t. for all $x$, “$K(x) > t$” is unprovable in $\mathcal{F}$

Proof: Define an $M$ that treats its input as an integer:

$M(k) := \text{Search over all strings } x \text{ and proofs } P \text{ for a proof } P \text{ in } \mathcal{F} \text{ that } K(x) > k. \text{ Output } x \text{ if found}$

Suppose $M(k)$ halts. It must print some output $x'$

Then $K(x') = K(<M,k>) \leq c + |k| \leq c + \log k$ for some $c$

Because $\mathcal{F}$ is consistent, $K(x') > k$ is true

But $k < c + \log k$ only holds for small enough $k$

If we choose $t$ to be greater than these $k$...

then $M(t)$ cannot halt, so “$K(x) > t$” has no proof!
Random Unprovable Truths

Theorem: For every interesting consistent $\mathcal{F}$, there is a $t$ s.t. for all $x$, “$K(x) > t$” is unprovable in $\mathcal{F}$

For a randomly chosen $x$ of length $t+100$, “$K(x) > t$” is true with probability at least $1 - 1/2^{100}$.

We can randomly generate true statements in $\mathcal{F}$ which have no proof in $\mathcal{F}$, with high probability!

For every interesting formal system $\mathcal{F}$ there is always some finite integer (say, $t=10000$) so that you’ll never be able to prove in $\mathcal{F}$ that a random 20000-bit string requires a 10000-bit program!