Lecture 14: 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 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 halts on $w$”
- A *proof* that “M halts on $w$” can be defined to be the *computation history* of M on $w$: the sequence of configurations $C_0 C_1 \cdots C_t$ that M goes through while computing on $w$

*Could sometimes prove “M doesn’t halt on $w$”...*
Interesting Systems of Mathematics

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) $S_{M,w}$ of $\mathcal{F}$ 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 candidate proof of a theorem is correct. This set is decidable: $\{(S, P) \mid P$ is 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 $M$ halts on $w$, then there’s either a proof $P$ of $S_{M,w}$ or a proof $P$ of $\neg S_{M,w}$.
Consistency and Completeness

A formal system $\mathcal{F}$ is **inconsistent** if there is a statement $S$ in $\mathcal{F}$ such that both $S$ and $\neg S$ are provable in $\mathcal{F}$.

$\mathcal{F}$ is **consistent** if it is NOT inconsistent.

A formal system $\mathcal{F}$ is **incomplete** if there is a statement $S$ in $\mathcal{F}$ such that neither $S$ nor $\neg S$ are provable in $\mathcal{F}$.

$\mathcal{F}$ is **complete** if it is NOT incomplete.
For every consistent and interesting $F$,

**Theorem 1.** (Gödel 1931) $F$ must be *incomplete*!
“There are mathematical statements that are *true* but cannot be proved.”

**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 statements that cannot be proved or disproved.

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

Proof: Define TM $G(w)$:
1. Obtain own description $G$ [Recursion Theorem!]
2. For all strings $P$ in lexicographical order,
   If ($P$ is a proof of $S_{G, w}$ in $\mathcal{F}$) then reject
   If ($P$ is a proof of $\neg S_{G, w}$ in $\mathcal{F}$) then accept

Note: If $\mathcal{F}$ is complete then $G$ cannot run forever!

1. If ($G$ accepts $w$) then have proof $P$ of “$G$ doesn’t accept $w$”
2. If ($G$ rejects $w$) then found proof $P$ of “$G$ accepts $w$”

In either case, $\mathcal{F}$ is inconsistent! Proof of $S_{G, w}$ and $\neg S_{G, w}$
(Gödel 1931) The consistency of $F$ cannot be proved within any interesting consistent $F$

Proof: Assume we can prove “$F$ is consistent” in $F$
We constructed $\neg S_{G, w} = \text{“G does not accept w”}$
which is true, but has no proof in $F$

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

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

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

But since $F$ is consistent, this cannot be true.
Therefore, $\neg S_{G, w}$ is true”

This contradicts the previous theorem!
Proof: Suppose $\text{PROVABLE}_F$ is decidable with TM $P$. Then we can decide $A_{TM}$ with the following procedure:

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

If $P$ accepts, examine all proofs in lex. order

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?

Undecidability in Mathematics

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

(Church-Turing 1936) For every interesting consistent $F$, $\text{PROVABLE}_F$ is undecidable
Kolmogorov Complexity: A Universal Theory of Data Compression
The Church-Turing Thesis:

Everyone’s Intuitive Notion of Algorithms = Turing Machines

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.
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:

$$<M,w> := 0^{|M|}1Mw$$

(Example: $<10110,101> = 0000011011010101$)

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 $M = \text{“On input } w, \text{ halt.”}$
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 $K(x) \leq |<M,x>| \leq 2|M| + |x| + 1 \leq |x| + c$
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

$N =$ “On input $<n,<M,w>>$, Let $x = M(w)$. Print $x$ for $n$ 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:

$A = 0101010101010101010101010101010101010101010101010101010101010101$

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 prints 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 TM $M =$ “On $w$, simulate $p(w)$ 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 Incomprressible 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 $< n$)  

$\leq$ (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 $c \geq 1$,

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

*“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
Computing 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.”
Computing Compressibility?

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

**Theorem:** COMPRESS is undecidable!

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

\[ M = \text{"On input } x \in \{0,1\}^*, \text{ let } N = 2^{|x|}. \]

\[ \text{For all } y \in \{0,1\}^* \text{ in lexicographical order, if } (y,N) \notin \text{COMPRESS then print } y \text{ and halt."} \]

\[ M(x) \text{ prints the shortest string } y' \text{ with } K(y') > 2^{|x|}. \]

\[ <M,x> \text{ is a description of } y', \text{ and } |<M,x>| \leq d + |x| \]

So \[ 2^{|x|} < K(y') \leq d + |x| . \text{ CONTRADICTION for large } x! \]
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 $\varepsilon$
Theorem: \( L = \{xx \mid x \in \{0, 1\}^*\} \) is not regular.

Proof: Suppose \( L \) is recognized by a DFA \( D \).
Let \( n \geq 0 \) and choose an \( x \in \{0, 1\}^* \) such that \( K(x) \geq n \).
Let \( q_x \) be the state of \( D \) reached after reading in \( x \).

Define a TM \( M(D, q, n) \):

- Find a path \( P \) in \( D \) of length \( n \) that starts from state \( q \) and ends in a final state.
- Print the \( n \)-bit string along path \( P \), and halt.

Claim: The pair \( <M,(D, q_x, n)> \) is a description of \( x \)!

So \( n \leq K(x) \leq |<M,(D, q_x, n)>| \leq O(\log n) \)

CONTRADICTION!
A formal system $F$ is *interesting* if:

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

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

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

**Proof:** Define a Turing machine \( M \) as follows:

\[ M(y) := \text{Search over all strings } x' \text{ and proofs } P \text{ for a proof } P \text{ in } F \text{ of } K(x') > b(y). \text{ Output } x' \text{ if found} \]

If \( M(y) \) halts, it prints some \( x' \). Then for some \( c \),

\[ K(x') = K(<M,y>) \leq c + |y| \leq c + \log(b(y)) \]

Therefore \( K(x') \leq c + \log(b(y)) \) has a proof in \( F \).

But \( K(x') > b(y) \) also has a proof \( P \) in \( F \)!

For \( t \gg b(y) \), have proof of “\( K(x') > t \)” and its negation! Therefore \( M(y) \) does not halt!
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!