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 $\mathcal{F}$} = Σ^***

  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 $\mathcal{F}$ is *consistent* or *sound* if no false statement has a valid proof in $\mathcal{F}$
(Proof in $\mathcal{F}$ implies Truth in $\mathcal{F}$)

A formal system $\mathcal{F}$ is *complete* if every true statement has a valid proof in $\mathcal{F}$
(Truth in $\mathcal{F}$ implies Proof in $\mathcal{F}$)

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) | 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 $\mathcal{F}$,

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

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

Theorem 3. (Church-Turing 1936) The problem of checking whether a given statement in $\mathcal{F}$ 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 $S' = \neg S_{G,\varepsilon}$**
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 $\mathcal{F}$ cannot be proved within any interesting consistent $\mathcal{F}$

Proof: Suppose we can prove "$\mathcal{F}$ is consistent" in $\mathcal{F}$

We constructed $\neg S_{G,\,\varepsilon}$ = "G does not accept ε"

which we showed is *true*, but *has no proof* in $\mathcal{F}$

G does not accept ε ⇔ There is no proof of $\neg S_{G,\,\varepsilon}$ 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 $\neg S_{G,\,\varepsilon}$ (here's the proof):

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

This contradicts the previous theorem.

Undecidability in Mathematics

PROVABLE$_{\mathcal{F}}$ = {S | there's a proof in $\mathcal{F}$ of S, or there's a proof in $\mathcal{F}$ of ¬S}

(Church-Turing 1936) For every interesting consistent $\mathcal{F}$, PROVABLE$_{\mathcal{F}}$ is undecidable

Proof: Suppose PROVABLE$_{\mathcal{F}}$ 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 $\mathcal{F}$

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 of Algorithms = Turing Machines

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

A Universal Theory of Computation

Is there 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**
= 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 π_1 and $\pi_2 : \Sigma^* \rightarrow \Sigma^*$ such that:

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

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 $K(x) \leq |\langle M,x\rangle| \leq 2|M| + |x| + 1 \leq |x| + c$

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 = "On <M,w>, let s = M(w). Print ss."

Suppose <M,w> is the shortest description of x.

Then <N,<M,w>> is a description of xx

Therefore

$$K(xx) \leq |\langle N,\langle M,w\rangle\rangle| \leq 2|N| + |\langle M,w\rangle| + 1$$
$$\leq 2|N| + K(x) + 1 \leq c + K(x)$$

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 = "On input <n,M,w>,
Let x = M(w). Print x for n times."

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

Then $K(x^n) \leq K(\langle N, \langle n, M, w\rangle\rangle) \leq 2|N| + d \log n + K(x)$

$\leq c \log n + K(x)$

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 = 0101010101010101010101010101010101

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

$$p : \Sigma^* \to \Sigma^*$$

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 a integer c so that for all x ϵ {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 $\langle M, d_p(x) \rangle$ is a description of x, and

$$K(x) \leq |\langle M, d_p(x) \rangle|$$

$$\leq 2|M| + K_p(x) + 1 \leq c + K_p(x)$$

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)

$\leq$ (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 ≥ 1,

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

"Most strings are highly incompressible"

Proof: (Number of binary strings of length n) = 2^n

but (Number of descriptions of length < n-c)

≤ (Number of binary strings of length < n-c)

= $2^{n-c} - 1$

Hence the probability that a *random* x satisfies

$$K(x) < n\text{-}c$$

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. **12358132134558914423337761098 7**
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. **0100011011000001010011100101110111110000**

2. **12358132134558914423337761098 7**

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

Determining Compressibility

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

COMPRESS = { (x,c) | K(x) ≤ 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

COMPRESS = {(x,c) | K(x) ≤ c}

Theorem: COMPRESS is undecidable!

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

M = *"On input x ϵ {0,1}*, interpret x as a number N.*
For all y ϵ {0,1} in lexicographical order,*
If (y,N) ∉ COMPRESS *then print y 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!