Genericity and Depth

Ang Li

University of Wisconsin–Madison

UW–Madison Logic Seminar
December 12, 2023

Outline

Kolmogorov Complexity

Definition

The *prefix-free Kolmogorov complexity* $K_M(x)$ of a binary string x w.r.t. a prefix-free machine M is

$$K_M(x) := \min\{|\sigma| : M(\sigma) = x\},$$

where $|\sigma|$ is the length of σ.

Definition

A prefix-free machine R is an *optimal* prefix-free machine if there is a constant d_M for each prefix-free machine M such that

$$\forall x K_R(x) \leq K_M(x) + d_M.$$

The constant d_M is called the coding constant of M (w.r.t. R).

Time-bounded Kolmogorov Complexity

We fix one such machine $\mathbb{U}$. For any natural number s,
$K_s(x) = \min\{|\sigma| : \mathbb{U}(\sigma) = x \text{ in at most } s \text{ many steps}\}$.

Definition

For a computable function $t\colon \mathbb{N} \to \mathbb{N}$ and a prefix-free machine M, the *prefix-free Kolmogorov complexity with time bound t* relative to M is

$$K^t_M(x) := \min\{|\sigma| : M(\sigma) \downarrow = x \text{ in at most } t(|x|) \text{ many steps}\},$$

and we write K^t for $K^t_{\mathbb{U}}$.

Logical Depth

Definition (Bennett 1988)

For $X \in 2^{\mathbb{N}}$, we say that X is deep if for every computable time bound t and $c \in \mathbb{N}$,

$$(\forall^{\infty} n)[K^t(X \upharpoonright n) - K(X \upharpoonright n) \geq c].$$

A set that is not *deep* is called *shallow*.

Example

1. The halting set $\emptyset'$ is deep.
2. A computable or a ML-random set is shallow.

Genericity

Computationally “easy enough” sets cannot be deep.

Theorem (Downey, McInerney, & Ng 2017)

There exists a superlow c.e. deep set.

Definition

For a set S of finite binary strings, a set A *meets* S if $(\exists n) A \upharpoonright n \in S$, and A *avoids* S if $(\exists n)[A \upharpoonright n \preccurlyeq \sigma \rightarrow \sigma \notin S]$.

A set A is *n-generic* if it meets or avoids every Σ^0_n set of strings.
A set A is *weakly n-generic* if it meets all dense Σ^0_n sets of strings.
Every weakly $(n+1)$-generic set is n-generic.

Even though generic sets are not necassarily “simple” as they are comeager in the Cantor space, they are computationally weak.

Upper Bound

Definition

A function $f : \mathbb{N} \to \mathbb{N}$ is a *Solovay function* if $K(n) \leq^+ f(n)$ for all n and $K(n) =^+ f(n)$ infinitly often.

Proposition (Bienvenu, Delle Rose, & Merkle)

Every weakly 2-generic set is shallow.

Proof.

We want to build a sequence $\{V_n\}_{n\in\mathbb{N}}$ of dense Δ_2^0 sets of finite binary strings where V_n only contains strings of length at least n and there is an $\tau \in V_n$ extending any σ such that $K^t(\tau) \leq^+ K(|\tau|)$.

For every weakly 2-generic A, $\exists t, c K^t(A \upharpoonright n) \leq K(A \upharpoonright n) + c$ infinitely often since A meets each V_n.

Every weakly 2-generic set is shallow

Proof.

Define a computable Solovay function h as the following:

$$h(\langle n, s\rangle) = \begin{cases} K_s(n) & \text{if } K_s(n) \neq K_{s-1}(n) \\ +\infty & \text{otherwise.} \end{cases}$$

Using $\emptyset'$, we can compute the number of steps s_n such that $K_{s_n}(n) = K(n) \neq K_{s_n-1}(n)$.

Let σ_n be the nth string in $\{0,1\}^*$. Let $\tau_n = \sigma_n 0^{\langle n, s_n\rangle - |\sigma_n|}$. We enumerate τ_n into all V_i for any $i \leq |\tau_n|$.

There is a machine M that outputs $\sigma_n 0^{\langle n,s\rangle - |\sigma_n|}$ given input δ where $n = \mathbb{U}(\delta)$ and $\mathbb{U}$'s computation takes exactly s steps. Thus,

$$K^t(\tau_n) \leq^+ K_M(\tau_n) \leq h(\langle n, s_n\rangle) = K(n) \leq^+ K(\langle n, s_n\rangle) = K(|\tau_n|).$$

$\square$

A Little Bit of Technicality

Remark

For any computable t and prefix-free machine M, there exists a computable function g_M such that there is some constant c and $K^{g_M(t)}(x) \leq K^t_M(x) + c$.

Given an effective list of Turing machines, we can acquire an effective list $\{M_e\}_{e\in\mathbb{N}}$ of self-delimiting machines that simulate prefix-free machines in exponential time. We choose $\mathbb{U}$ such that $\mathbb{U}(0^{e-1}1\sigma) = M_e(\sigma)$.

If we restrict ourselves to self-delimiting machines, g_M can be replaced by $ct\log t$ [LV+08, Example 7.1.1].

The question of whether g_M can be efficient is still open.

Lower Bound

Theorem

There exists a deep 1-generic set.

Proof.

We shall use a $\emptyset''$-construction to build a 1-generic set A.

Requirements:

D_i : If φ_i is an order function,
$(\forall c)(\forall^\infty m)[K^{\varphi_i}(A \upharpoonright m+1) > K(A \upharpoonright m+1) + c]$,

G_e : A meets or avoids S_e, where $\{S_e\}_{e\in\mathbb{N}}$ is an effective listing of c.e. set of strings.

Primitive G_e Strategy

Associate a restraint l_e to each G_e requirement. At stage s, wait until $A_{s-1} \upharpoonright n$ has an extension $(A_{s-1} \upharpoonright n)^\frown\sigma$ in $S_e[s]$ for $n > l_e$. Let $A_s = (A_{s-1} \upharpoonright n)^\frown\sigma$.

Primitive D_i Strategy

Let $\{\varphi_i\}_{i\in\mathbb{N}}$ be a listing of all partial computable functions.
We partition $\mathbb{N}$ into consecutive intervals $I_0, I_1, \ldots$
Then, we assign a φ to each I.

$$\begin{array}{ccccccc}
\{0\} & [1,2] & [3,6] & [7,14] & [15,30] & [31,62] & \ldots \\
I_0 & I_1 & I_2 & I_3 & I_4 & I_5 & \ldots \\
\varphi_0 & & \varphi_0 & & \varphi_0 & & \ldots \\
 & \varphi_1 & & & & \varphi_1 & \ldots \\
 & & & \varphi_2 & & & \ldots
\end{array}$$

Primitive D_i Strategy

1. Set $x = 0$.
2. Wait for $\varphi_{i,s}(x)\downarrow$.
3. We "move in" the next φ_i interval I_j: if $\varphi_{i,s}(\max I_{j+2^{i+1}} + 1)\downarrow$, we run $\mathbb{U}$ on all inputs of length $< |I_j|$ for $g_M(g_N(\varphi_i(\max I_{j+2^{i+1}} + 1)))$ many steps each. Then, we choose the leftmost string τ of length $|I_j|$ which was not among the outputs, and alter A_{s-1} so that $A_s \upharpoonright \max I_j + 1 = (A_{s-1} \upharpoonright \min I_j)^\frown\tau$.

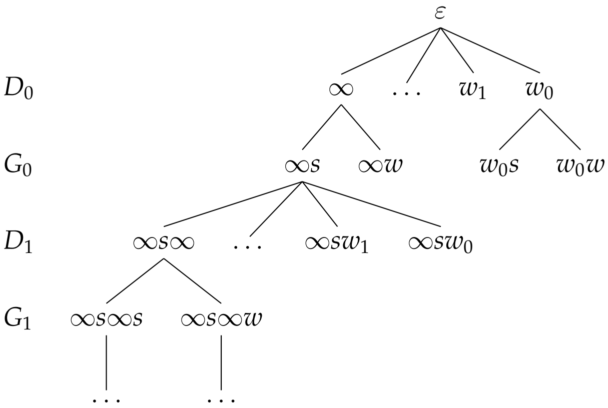
   Increase x by 1 and go back to step 2.

Tree of Strategies

We order the requirements as follows: $D_0 < G_0 < D_1 < G_1 < \cdots$

Let $\Lambda = \{\infty < \cdots < w_n < \cdots < w_2 < w_1 < w_0 < s < w\}$ be the set of outcomes.

Tree of strategies:



Primitive Strategy

Problems:

- Extensions made by G_e-strategy might destroy moved-in intervals.
- New intervals being moved in might destroy the extension of G_e-strategy.

For the second problem, we can delay the extensions so that relevant intervals are all moved in. For the first problem, we need another strategy.

A Second Strategy

For the first problem, we shall let the G_e-strategy take over. When we get the chance to extend the initial segment of A_{s-1}, we also compress the initial segments of A to make sure that $K^{\varphi_i}(A \upharpoonright m+1) - K(A \upharpoonright m+1) \geq e + c_\alpha$ for $i \leq e$.

Definition

A c.e. set $W \subseteq \mathbb{N} \times 2^{<\omega}$ is a *bounded request set* if $\sum_{\langle r,y \rangle \in W} 2^{-r} \leq 1$.

Theorem (Machine Existence)

For each bounded request set W, one can effectively obtain a prefix-free machine M such that

$$\forall r, y[\langle r, y \rangle \in W \leftrightarrow \exists w(|w| = r \wedge M(w) = y)].$$

A Second Strategy

The weight of the strings at time φ_i is

$$w_{\alpha,i} = \sum_{\theta \in N_{\alpha,i}} 2^{-K^{\varphi_i}(\theta)},$$

where

$$N_{\alpha,i} := \{\theta : \max I_{j_0} < |\theta| \leq \max I_{j_1} + 1\}.$$

We need to enumerate the request $(K^{\varphi_i}(\theta) - e - c_\alpha, \theta)$ for every $\theta \in N_{\alpha,i}$.

We assign weight to each $N_{\alpha,i}$ in advance to make sure the weight of the request set ≤ 1.

If there are infinitely many n through which an initial of A requires attention, there must exist a pair of large enough n and stage so that $w_{\alpha,i}$ is small enough for all $i \leq e$, and the relevant overlapping intervals have been moved in since $\sum_\theta 2^{-K^{\varphi_i}(\theta)} \leq 1$ for any i.

Strategy

Define $l_\varepsilon = 0$ where ε is the empty string.

D_i-strategy:

1. If node α is visited the first time, we initialize α by setting $l_\alpha = 1 + \max\{\max\{l_\sigma : \sigma \text{ has been initialized}\}, \max\{l'_\sigma : \sigma \text{ has outcome stop}\}\}$.
2. Set $x = 0$.
3. Wait for $\varphi_{i,s}(x) \downarrow$.
4. For the least interval I_j assigned to φ_i such that I_j has not been moved in or was marked fresh, $\varphi_{i,s}(\max I_{j+2^{i+1}} + 1) \downarrow$, and $\min I_j > l_\alpha$, we move in I_j.

   For any moved-in interval I with $\min I > \max I_j$, we mark it fresh.

   Increase x by 1 and go back to step 3.

Strategy

G_e-strategy:

1. If node α is visited the first time, we initialize α by setting l_α beyond any interval $I_{j+2^{i+1}}$ such that I_j has been moved in and assigned to some φ_i with D_i having a finite current outcome and $i \leq e$, and larger than any l'_σ. Assign a number c_α.
2. α requires attention through $n \geq l_\alpha$ at stage s if $A_{s-1} \upharpoonright n$ has an extension in $S_e[s]$ and any interval I assigned to some φ_i such that $D_i < G_e$, the current outcome of the D_i-strategy is infinite, and I overlaps the concatenating segment of the extension, has been moved in.
3. α looks for an n that makes sure we can compress the strings cheaply enough. If such an n is found, we act by extending $A_{s-1} \upharpoonright n$ and define $l'_\alpha = n + |\sigma|$. Otherwise, let $A_s = A_{s-1}$.

Construction

Let a strategy α of length t be eligible to act at a substage t (α is visited) of stage $s \geq t$ if and only if α has the correct guess about the current outcomes of all $\beta \prec \alpha$.

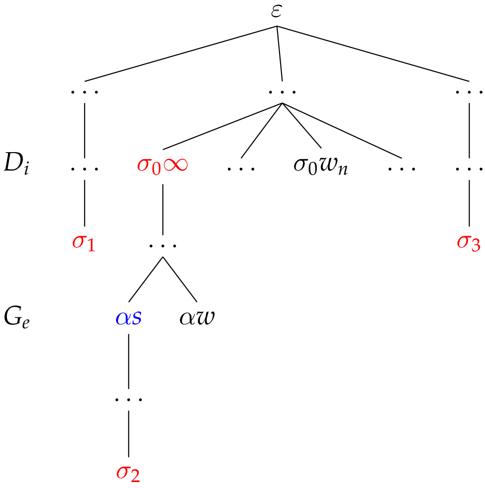
We define the current true path f_s at stage s to be the longest strategy eligible to act at stage s.

Let $f = \liminf_s f_s$.

Verification

Lemma

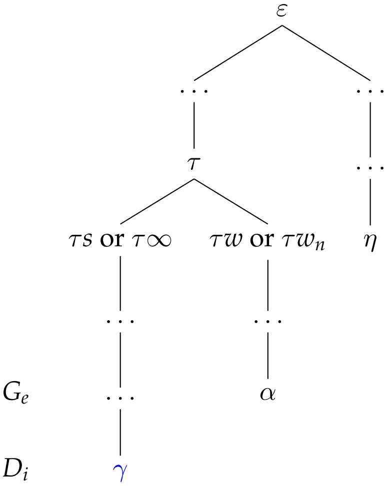
The eventual G_e node α on the true path is not injured.



Verification

Lemma

A is deep.



References I

- Charles H Bennett, *Logical depth and physical complexity*, Citeseer, 1988.
- R.G. Downey and D.R. Hirschfeldt, *Algorithmic randomness and complexity*, Theory and Applications of Computability, Springer New York, 2010.
- Rod Downey, Michael McInerney, and Keng Meng Ng, *Lowness and logical depth*, Theoretical Computer Science **702** (2017), 23–33.
- Rupert Hölzl, Thorsten Kräling, and Wolfgang Merkle, *Time-bounded Kolmogorov complexity and Solovay functions*, Theory of Computing Systems **52** (2013), no. 1, 80–94.
- David W Juedes and Jack H Lutz, *Modeling time-bounded prefix Kolmogorov complexity*, Theory of Computing Systems **33** (2000), 111–123.
- Steffen Lempp, *Priority argument in computability theory, model theory, and complexity theory.*

References II

- Ming Li, Paul Vitányi, et al., *An introduction to Kolmogorov complexity and its applications*, vol. 3, Springer, 2008.
- André Nies, *Computability and randomness*, vol. 51, OUP Oxford, 2012.

Thank You!