Description Complexity

Reijo Jaakkola

Mathematics Research Centre, Tampere University, Finland

November 27, 2024

Overview

What is randomness?

- Is the binary string

$$0101010101010101$$

less random than

$$1011101111011101$$

?

Kolmogorov complexity

- The Kolmogorov complexity $K(x)$ of a binary string x is the length of the shortest program P that produces x.
- Intuitively, the larger $K(x)$ is the more random x is.
- **Example:** the string

$$010101010101010101$$

is produced by the program "print 01 nine times".

Kolmogorov complexity

- $K(x) = O(|x|)$.
- For most $x \in \{0,1\}^n$ we have that $K(x) \geq |x|$ by a counting argument.

Description complexity

- The description complexity $D(x)$ of a binary string x is the length of the shortest program P that only accepts x.
- $D(x) \approx K(x)$.
- **Example:** the string

$$010101010101010101$$

is described by "starts with 0, 0 and 1 alternate, has length 18".

Conditional description complexity

- $D(x : |x|)$ is the length of the shortest program which, when given a string of length $|x|$, accepts it only if it is x.
- **Example:** in this setting the string

$$010101010101010101$$

is described by "starts with 0, 0 and 1 alternate".

Link to logic

- We move now from binary strings to general finite structures (e.g., strings, graphs, groups).
- Given a logic $\mathcal{L}$ and a structure $\mathfrak{A}$, we define $D_{\mathcal{L}}(\mathfrak{A})$ as

$$\min \left\{ \mathrm{len}(\varphi) \;\middle|\; \begin{array}{l} \varphi \in \mathcal{L} \text{ and } \varphi \text{ defines } \mathfrak{A} \text{ uniquely up to isomorphism} \\ \text{among structures of the same size} \end{array} \right\}$$

Example: finite graphs

- A graph is a pair $G = (V, E)$, where V is a set and

$$E \subseteq \{\{v, u\} \mid v, u \in V, v \neq u\}.$$

- Let $\mathcal{L}$ be the first-order logic FO.
- The clique of size n is described by the sentence

$$\forall x \forall y (x = y \vee E(x, y)).$$

This sentence has size

$$\underbrace{2}_{\text{quantifiers}} + \underbrace{1}_{\text{disjunction}} + \underbrace{2}_{\text{atomic formulas}}$$

so $D_{\mathrm{FO}}(G) \leq 5$.

- Every finite graph can be defined in FO up to isomorphism.

Description complexity is hard to understand

- The main challenge in studying $D_{\mathcal{L}}$ is that even for simple structures it is very difficult to calculate it.
- **Open problem:** we know that for most graphs G of size n we have that

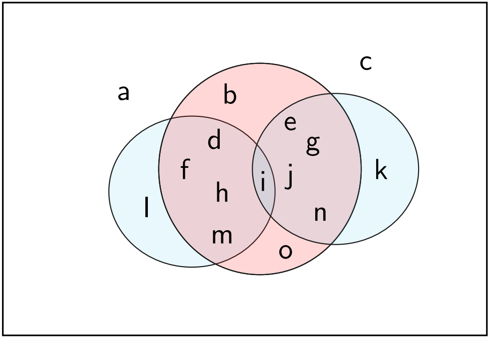
$$D_{\mathrm{FO}}(G) = \Omega\left(\frac{n^2}{\log(n)}\right)$$

but we do not know whether

$$D_{\mathrm{FO}}(G) = \Theta\left(\frac{n^2}{\log(n)}\right)$$

holds for most graphs.

- The simplest possible structures are the unary structures.
- Unary structure is a tuple $(A, P_1, \ldots, P_k)$, where A is a set and $P_1, \ldots, P_k \subseteq A$.



- **Terminology:** a type is $I \subseteq \{1, \ldots, k\}$. A type I is realized in $(A, P_1, \ldots, P_k)$ if there is $a \in A$ such that $a \in P_\ell$ iff $\ell \in I$.

Propositional logic with counting (PLC)

- In PLC we can say how many times a Boolean combination of unary relations occurs in a unary structure $A = (P_1, \ldots, P_k)$.
- **Examples:**

$$\exists^{=5} x \ P_1(x) \quad \exists^{\geq 2} x(P_1(x) \vee \neg P_2(x))$$

$$\exists^{=3} x \ P_1(x) \wedge \exists^{=5} x P_2(x)$$

- The size of e.g. $\exists^{=5} x \ P_1(x)$ is

$$\underbrace{1}_{\text{existential quantifier}} + \underbrace{5}_{\text{index}} + \underbrace{1}_{\text{atomic formula}}$$

- Every (finite) unary structure can be defined uniquely up to isomorphism in PLC.

Some results on PLC

Theorem (J., Kuusisto, Vilander, 2023)

- *For every unary structure* $\mathfrak{M}$ *of size* n *we have that*

$$D_{\mathrm{PLC}}(\mathfrak{M}) = \min(n, 2(n-t)) + \mathcal{O}(1),$$

where t *is the size of the largest type in* $\mathfrak{M}$.

- *For most unary structures* $\mathfrak{M}$ *of size* n *we have that*

$$D_{\mathrm{PLC}}(\mathfrak{M}) = n + k2^{k+1} - 1,$$

where k *is the number of unary structures. Hence*

$$\mathop{\mathbb{E}}_{\mathfrak{M}}(D_{\mathrm{PLC}}(\mathfrak{M})) \sim n,$$

as $n \to \infty$.

Monadic first-order logic (MFO)

- FO over unary vocabularies.
- **Example:** $\forall x(\neg P(x) \vee \exists y(\neg x = y \ \wedge \ Q(y)))$. This formula has size

$$\underbrace{2}_{\text{quantifiers}} + \underbrace{4}_{\text{Boolean connectives}} + \underbrace{3}_{\text{atomic formulas}}$$

- Every (finite) unary structure can be defined uniquely up to isomorphism in MFO.

Some results on MFO

Theorem (J., Kuusisto, Vilander, 2024)

1. *Let $\mathfrak{M}$ be a unary structure of size n, t_1 the size of the largest type in $\mathfrak{M}$ and t_2 the size of the second largest type in $\mathfrak{M}$. Now*

$$3t_2 - 3 \leq D_{\mathrm{MFO}}(\mathfrak{M}) \leq \min(3t_1, 6t_2) + O(1).$$

2. *For a random unary structure $\mathfrak{M}$ of size n we have that*

$$\mathop{\mathbb{E}}_{\mathfrak{M}}(D_{\mathrm{MFO}}(\mathfrak{M})) \sim \frac{3n}{2^k}$$

as $n \to \infty$.

Lossy compression in PLC

- For each $d \geq 0$ we define PLC^d as the set of those formulas of PLC which can count only up to d.
- Given a structure $\mathfrak{M}$ of size n we define

$$[\mathfrak{M}]_d := \{\mathfrak{N} \mid \mathfrak{N} \text{ has size } n, \mathfrak{M} \text{ and } \mathfrak{N} \text{ are } \mathrm{PLC}^d \text{ equivalent}\}.$$

$D_{\mathrm{PLC}^d}(\mathfrak{M})$ is the description complexity of $[\mathfrak{M}_d]$.

Lossy compression in PLC

Theorem (J., Kuusisto, Vilander, 2023)

Let $\mathfrak{M}$ be a unary structure of size n. Let $I_1, \ldots, I_r$ denote the types realized in $\mathfrak{M}$. Define

$$t_s := \min\{\textit{size of } I_S, d\}.$$

Now either

$$D_{\mathrm{PLC}^d}(\mathfrak{M}) = \sum_{s=1}^{r} t_s + O(1)$$

or

$$D_{\mathrm{PLC}^d}(\mathfrak{M}) = n - \max\{t_s \mid 1 \leq s \leq r\} + O(1).$$

Conclusions

- Description complexity is a way of measuring the randomness of a deterministic object.
- A lot of recent progress on understanding the description complexity of unary structures.
- For more complex structures such as graphs and binary strings much remains to be done.

Thanks!