Logical operations and Kolmogorov complexity

Alexander Shen* Nikolai Vereshchagin†

Abstract

Conditional Kolmogorov complexity $K(x|y)$ can be understood as the complexity of the problem "$Y \to X$", where X is the problem "construct x" and Y is the problem "construct y". Other logical operations ($\wedge, \vee, \leftrightarrow$) can be interpreted in a similar way, extending Kolmogorov interpretation of intuitionistic logic and Kleene realizability. This leads to interesting problems in algorithmic information theory. Some of these questions are discussed.

Keywords: intuitionistic logic, realizability, Kolmogorov complexity.

Kolmogorov complexity $K(x)$ of a binary string x is defined as minimal length of a program that generates this string. This definition can be extended to sets of strings. Let A be a (finite or infinite) set of strings. We define the complexity $K(A)$ as the length of a shortest program that generates some string $x \in A$. Informally, we consider A as a problem "Generate any element of A"; $K(A)$ is complexity of this problem. Evidently, $K(A) = \min\{K(x) \mid x \in A\}$, so this generalization gives nothing really new.

However, it can be combined with the definition of logical operations on sets of strings that goes back to Kolmogorov's paper [3] and Kleene's notion of realizability [2, p. 501]. Let A and B be two sets of strings. We define sets $A \wedge B$, $A \vee B$ and $A \to B$ as follows:

- $A \wedge B = \{\langle a, b\rangle \mid a \in A, b \in B\}$
- $A \vee B = \{\langle 0, a\rangle \mid a \in A\} \cup \{\langle 1, b\rangle \mid b \in B\}$
- $A \to B = \{p \mid [p](x) \in B$ for all $x \in A\}$

Here $\langle\cdot,\cdot\rangle$ is computable encoding of pair of strings; $[p](x)$ stands for the output of p (considered as a program) when applied to input x. Note that a program p in $A \to B$ may give arbitrary results (or no results) when applied to $x \notin A$.

Example 1. Let x and y be two strings. Consider the set $x \to y$ (to simplify notation we identify a string s and the singleton $\{s\}$). This set contains all programs that map x to y. It is easy to see that $K(x \to y) = K(y|x) + O(1)$ where $K(y|x)$ denotes conditional complexity of the string y when x is known.

Note also that $K(x \wedge y)$ is the complexity of the pair (x, y) and $K(x \vee y) = \min(K(x), K(y))$.

Example 2. Now consider the set $x \leftrightarrow y$ defined as $(x \to y) \wedge (y \to x)$. By definition, $K(x \leftrightarrow y)$ is the complexity of pair of programs transforming x to y and vice versa. It turns out (as proved in [1]) that $K(x \leftrightarrow y) = \max(K(x|y), K(y|x)) + O(\log(K(x|y) + K(y|x)))$.

Example 3. Let x and y be two strings. Consider the set $(x \to y) \to y$. Its elements are programs that map every program in $(x \to y)$ (i.e., transforming x to y) to y. Let us prove that $K((x \to y) \to y) = \min(K(x), K(y)) + O(\log(|x| + |y|))$. (Here $|s|$ stands for the length of s.)

It is easy to see that $K((x \to y) \to y) \leq K(y) + O(1)$. Indeed, any program that prints y can be considered as a program that maps every element of $(x \to y)$ to y.

It is also easy to see that $K((x \to y) \to y) \leq K(x) + O(1)$. Indeed, for a given string x consider the program p_x that maps any program s to $[s](x)$. If s belongs to $x \to y$, then $[p_x](s) = [s](x) = y$. Therefore, $p_x \in ((x \to y) \to y)$. On the other hand, $K(p_x) \leq K(x) + O(1)$.

Therefore, $K((x \to y) \to y) \leq \min(K(x), K(y)) + O(1)$. It remains to prove that $\min(K(x), K(y)) \leq K((x \to y) \to y) + O(\log n)$ if x and y are strings of length at most n.

*Institute of Problems of Information Transmission (Moscow); Independent University of Moscow; e-mail: shen@landau.ac.ru, shen@mccme.ru.

†Dept. of Mathematical Logic and Theory of Algorithms, Moscow State University, Vorobjevy Gory, Moscow 119899, Russia; e-mail: ver@mech.math.msu.su, ver@mccme.ru.

Let s be a program in $(x \to y) \to y$. Let S be the set of all strings of length at most n. For any function $\tau\colon S \to S$ we fix some program l_τ that computes this function; therefore, we have $|S|^{|S|}$ different programs l_τ. A pair $(u, v) \in S \times S$ is called *s-coherent* if $[s](l_\tau) = v$ for any function τ such that $\tau(u) = v$. (Note that we do not require that $[s](p) = v$ for any program $p \in (u \to v)$.)

By definition the pair (x, y) is s-coherent. Other coherent pairs may exist. (For example, if $s(p) = y$ for any p, then any pair (x', y) is s-coherent. If $s(p) = p(x)$ for any p, then any pair (x, y') will be s-coherent.) However, either all coherent pairs have x as the first component or all coherent pairs have y as the second component. To prove that, it is enough to prove the following statement: if (x', y') and (x'', y'') are s-coherent pairs then either $x' = x''$ or $y' = y''$. If it is not the case and $x' \neq x''$, $y \neq y''$, consider a function τ such that $\tau(x') = y'$ and $\tau(x'') = y''$. Then we have $[s](l_\tau) = y'$ and $[s](l_\tau) = y''$ at the same time, which is impossible, since $y' \neq y$.

If all s-coherent pairs have x as the first component, then $K(x) \leq K(s) + O(\log n)$, because we can find x when s and n are given (searching for a s-coherent pair and taking its first component). Similarly, if all s-coherent pairs have y as the second component, then $K(y) \leq K(s) + O(\log n)$. Therefore, $\min(K(x), K(y)) \leq K(s) + O(\log n)$. (End of the proof.)

Similar argument can be used to prove that $\min(K(x), K(z)) \leq K((x \to y) \to z) + O(\log n)$ for any strings x, y, z having length at most n. Indeed, let t be a program in $(x \to y) \to z$. A triple (x, y, z) is called t-coherent if $t(l_\tau) = z$ for any τ such that $\tau(x) = y$. If two triples (x', y', z') and (x'', y'', z'') are t-coherent then either $x' = x''$ or $z' = z''$. Therefore, either x or z can be reconstructed from t.

In particular, for $x = z$ we get $K((x \to y) \to x) = K(x) + O(\log n)$.

Example 4. For any three strings x, y, z having length at most n we have $K((x \vee y) \to z) = \max(K(z|x), K(z|y)) + O(\log n)$. This was recently proved by Andrei Muchnik with a very nice argument. One direction is easy: $K(z|x) = K(x \to z) \leq K((x \vee y) \to z)$; for the same reason $K(z|y) \leq K((x \vee y) \to z)$. To prove the reverse inequality, Muchnik assumes that $K(z|x) \leq k$ and $K(z|y) \leq k$ and proves that one can find a "fingerprint" f of z having length k such that z can be reconstructed from f and x and also from f and y. The proof uses expander graphs and will be published elsewhere.

Question. Let $A(p, q, \ldots)$ be a propositional formula with connectivities $\wedge, \vee, \to$. For any strings $x, y, \ldots$ consider the set $A(x, y \ldots)$ defined in a natural way. The question is whether $K(A(x, y, \ldots))$ is determined by complexities $K(x), K(y) \ldots$ and conditional complexities of their combinations up to $O(\log n)$-term if $x, y, \ldots$ are strings of length at most n. Examples 1–4 support this conjecture.

Remark 1. The goal of Kolmogorov and Kleene was to provide an interpretation of the intuitionistic logic. Following this idea, we can prove the following statement: if $A(p, q, \ldots)$ is provable in the intuitionistic propositional calculus (IPC), then $K(A(x, y, \ldots)) = O(1)$ for any strings $x, y, \ldots$. Indeed, there exists a string s that belongs to $A(x, y, \ldots)$ for all $x, y, \ldots$ (induction by the length of the proof in IPC). See also [4] where a slightly different approach using Scott domains is used.

We do not know whether the reverse implication is true (i.e., whether $K(A(x, y, \ldots)) = O(1)$ implies that A is provable in IPC) for any propositional formula A containing $\wedge$, $\vee$ and $\to$. It is easy to see, however, that if $K(A(x, y, \ldots)) = O(1)$ then A is provable in classical propositional calculus.

Remark 2. It is easy to see that

$$K(B) \leq K(A) + K(A \to B) + O(\log K(A \to B)))$$

Indeed, one can combine (self-delimiting encoding of) a program from $A \to B$ and (encoding of) any element of A to get an encoding of some element of B.

We can combine this remark with the previous one: if $A(p, q, \ldots) \to B(p, q, \ldots)$ is provable in IPC, then $K(B(x, y, \ldots)) \leq K(A(x, y, \ldots)) + O(1)$ for all strings $x, y, \ldots$ For example, the formula $(x \vee y) \to ((x \to y) \to y))$ is provable in IPC, therefore $K((x \to y) \to y)) \leq K(x \vee y) + O(1) = \min(K(x), K(y)) + O(1)$ (as we mentioned in example 3 above).

Example 5. The Pierce law $((x \to y) \to x) \to x$ (being provable in classical logic) is not provable in IPC. However, Pierce law has low complexity: $K(((x \to y) \to x) \to x) = O(\log n)$ for any strings x, y of length at most n. Indeed, let s belong to $(x \to y) \to x$. This time call a pair $(u, v) \in S \times S$ *s-coherent* if $[s](l_\tau) = u$ for any τ such that $\tau(u) = v$. If (u, v) is s-coherent then $u = x$, since otherwise there exists τ with $\tau(x) = y$, $\tau(u) = v$ and we have $x = [s](l_\tau) = u$. Given s find an s-coherent pair and output its first component. This instruction describes a program from $((x \to y) \to x) \to x$ of complexity $\log n$ (note that we need to know n).

Recalling Remark 2, we see again that $K(x) \leq K((x \to y) \to x) + O(\log n)$ (cf. Example 3, last sentence).

However, as the next example shows, the inequality for complexities may be valid even in the case when the corresponding implication has large complexity.

Example 6. $K(((x \to y) \to y) \to (x \vee y)) = \min(K(x|y), K(y|x)) + O(\log(|x| + |y|))$.

(Recall that $K((x \to y) \to y) = K(x \vee y) + O(\log n)$, as we have seen in Example 3, so one may expect that $K(((x \to y) \to y) \to (x \vee y)) = O(\log n)$. But this is not the case.)

The inequality $K(((x \to y) \to y) \to (x \vee y)) \leq K(y|x) + O(1)$ is straightforward since given a program p with $[p](x) = y$ and a program s in $(x \to y) \to y$ we can find y (because $s[p] = y$).

Let us prove that $K(((x \to y) \to y) \to (x \vee y)) \leq K(x|y) + O(\log n)$, where $n = \max\{|x|, |y|\}$. It suffices to prove that given the triple $\langle n$, a program p with $[p](y) = x$, a program s in the set $(x \to y) \to y\rangle$ we are able to find either x or y. Recall the notion of s-coherent pair (see Example 3). Given n, p and s find an s-coherent pair (u, v). Then continue to enumerate other s-coherent pairs and run in parallel p on input v. We stop if we either find another s-coherent pair (u', v'), or we find out that $[p](v) = u$. In the former case we know either x, or y: if $v' \neq v$ then $x = u$ and if $u' \neq u$ then $y = v$ (recall that either the first component of all s-coherent pairs is equal to x, or the second component of all s-coherent pairs is equal to y). In the latter case (when $[p](v) = u$) we know that $x = u$. Indeed, if $x \neq u$ then $y = v$ hence $u = [p](v) = [p](y) = x$. Note that computation terminates (if there are no other s-coherent pairs except (u, v) then $(u, v) = (x, y)$ hence $[p](v) = u$).

Let us prove that $K(((x \to y) \to y) \to (x \vee y)) \geq \min\{K(y|x), K(x|y)\} - O(1)$. Assume that a program p is in the set $((x \to y) \to y) \to (x \vee y)$. We wish to prove that either given p we can find a program r_1 with $[r_1](x) = y$, or given p we can find a program r_2 with $[r_2](y) = x$.

We know that $p[s] = \langle 0, x\rangle$ or $p[s] = \langle 1, y\rangle$ for any s in $(x \to y) \to y$. Using a standard trick from recursion theory, we choose a pair A, B of enumerable inseparable subsets of $\mathbb{N}$. For any $i \in \mathbb{N}$ and any strings u, v consider the following program $q_i(u, v)$ in $(u \to v) \to v$: given a program s run it on input u and enumerate in parallel A and B; if it turns out that $[s](u) = v$ then output v and stop, if it turns out that $i \in A$ then output v and stop, if it turns out that $i \in B$ and $[s](u)$ is defined then output $[s](u)$ and stop (note that it does not matter which option to choose if several of them happen simultaneously; note also that output is undefined if $[s](u)$ is undefined and $i \notin B$).

As for any i the program $q_i(x, y)$ is in $(x \to y) \to y$, the program p applied to $q_i(x, y)$ outputs either $\langle 0, x\rangle$ or $\langle 1, y\rangle$. And either there is $i \in A$ such that $[p](q_i(x, y)) = \langle 0, x\rangle$, or there is $i \in B$ such that $[p](q_i(x, y)) = \langle 1, y\rangle$ (otherwise the decidable set $\{i \mid [p](q_i(x, y)) = \langle 1, y\rangle\}$ separates A and B). In the former case given p and y we are able to find x: find $i \in A$ and u such that $[p](q_i(u, y)) = \langle 0, u\rangle$ and output u; note that $q_i(u, y)$ is in $(x \to y) \to y$ hence $u = x$. In the latter case given p and x we are able to find y: find $i \in B$ and v such that $[p](q_i(x, v)) = \langle 1, v\rangle$ and output v; note that $q_i(x, v)$ is in $(x \to y) \to y$ hence $v = y$. (End of proof.)

Acknowledgements. This work was partially done while visiting Ecole Normale Supérieure de Lyon. Authors are also grateful to participants of Kolmogorov seminar (Moscow), especially to Andrei Muchnik who presented his proof (Example 4) at the seminar and kindly allowed us to mention his unpublished result. We are grateful to referees for their remarks.

References

[1] C.H. Bennett, P. Gacs, M. Li, P. Vitanyi and W. Zurek. Thermodynamics of Computation and Information Distance, STOC-1993 Proceedings, 21–30.

[2] S.C. Kleene. *Introduction to metamathematics.* New York, Toronto, 1952.

[3] A. Kolmogoroff. Zur Deutung der Intuitionistischen Logik. *Mathematische Zeitschrift*, Bd. 35 (1932), H. 1, S. 57–65.

[4] A. Shen. Algorithmic variants of entropy. *Soviet Math. Dokl.*, **29** (1984), No. 3, pp. 569–573.