3-2014

Deep Mathematical Results Are The Ones That Connect Seemingly Unrelated Areas: Towards A Formal Proof Of Gian-Carlo Rota's Thesis

Olga Kosheleva
The University of Texas at El Paso, olgak@utep.edu

Vladik Kreinovich
The University of Texas at El Paso, vladik@utep.edu

Comments:
Technical Report: UTEP-CS-14-29
Published in *Applied Mathematical Sciences*, 2014, Vol. 8, No. 48, pp. 2391-2396.

DEEP MATHEMATICAL RESULTS ARE THE ONES THAT CONNECT SEEMINGLY UNRELATED AREAS: TOWARDS A FORMAL PROOF OF GIAN-CARLO ROTA'S THESIS

Olga Kosheleva[1] **and Vladik Kreinovich**[2]

University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
olgak@utep.edu, vladik@utep.edu

Abstract

When is a mathematical result deep? At first glance, the answer to this question is subjective: what is deep for one mathematician may not sound that deep for another. A renowned mathematician Gian-Carlo Rota expressed an opinion that the notion of deepness is more objective that we may think: namely, that a mathematical statement is deep if and only if it connects two seemingly unrelated areas of mathematics. In this paper, we formalize this thesis, and show that in this formalization, Gian Carlo Rota's thesis becomes a provable mathematical result.

Mathematics Subject Classification: 00A30, 03A05, 68Q30

Keywords: deep mathematical results, seemingly unrelated areas of mathematics, Kolmogorov complexity, Gian-Carlo Rota's thesis

1 Formulation of the Problem: How to Formalize (and Prove) Gian-Carlo Rota's Thesis

When is a mathematical result deep? How can we distinguish deep mathematical results from the ones which may be technically complicated – but not really deep?

Is this distinction subjective? At first glance, this is a very subjective distinction, depending on the mathematicians' opinions and tastes.

Gian-Carlo Rota's idea: the distinction is reasonably objective. Many mathematicians believe that deep results are the ones that connect two seemingly unrelated areas of mathematics. This view was explicitly stated by the famous mathematician Gian-Carlo Rota in his essay [2].

Gian Carlo Rota's thesis has empirical support. This thesis is well supported empirically. There are many examples of results which most mathematicians view as deep, and in most cases, these results indeed connect two unrelated areas of mathematics.

The most widely cited example is Euler's identity $e^{\mathrm{i}\cdot\pi} = -1$, which relates three seemingly unrelated quantities:

- the basis e of natural logarithms;
- the imaginary unit i – the square root of -1; and
- the number π, the ratio of the circle's circumference to its diameter.

There are many other examples of this type.

How can we describe it formally? Since Gian-Carlo Rota's thesis seems to be supported by many examples, a natural idea is: can we describe it – and prove it – in precise terms?

What we do in this paper. In this paper, we formalize Gian-Carlo Rota's thesis as a precise statement, and we prove that the resulting precise statement is indeed true.

2 Let us Formalize Gian-Carlo Rota's Thesis

When is a result deep: towards formalization. We would like to distinguish proofs which are really complex from proofs which are long but not complex and not deep at all – e.g., that consist of many similar routine parts.

From a formal viewpoint, a proof is nothing else but a sequence of symbols – a sequence which satisfies some easy-to-check conditions (that each step indeed follows the rules of the corresponding proof system). For general sequences of symbols, a similar distinction problem is well known since the 1960s papers of A. N. Kolmogorov, P Martin-Löf, R. Solomonoff, and G. Chaitin, who tried to distinguish between complex-to-reproduce sequences – such as truly random

sequences obtained by flipping coins – and seemingly similar sequences like 0101...01, which also have half zeros and half ones, but follow a simple rule. Their solution to this problem naturally follows from the problem itself:

- we fix a (universal) programming language, and then
- we define the *Kolmogorov complexity* $K(s)$ of a sequence s as the shortest possible bit length of a program (in the fixed language) which generates the sequence s; see, e.g., [1].

A repeating sequence 0101...01 can be described by a short for-loop program, so its Kolmogorov complexity is small. Similarly, any sequence which can be generated by a simple formula or a simple algorithm has a small Kolmogorov complexity. On the other hand, an intuitive understanding of what is a truly random sequence of 0s and 1s is that it cannot be generated by any simple algorithm. So, the shortest way to generate this sequence $s = 0110\ldots$ is to simply print this sequence bit by bit, i.e., to use a program `println(0110...)`. Thus, for a truly random sequence s, its Kolmogorov complexity $K(s)$ (the length of this shortest program) is large: namely, it is approximately equal to the length $\mathrm{len}(s)$ of the sequence s: $K(s) \approx \mathrm{len}(s)$.

We will therefore call a sequence if symbols *complex* if its Kolmogorov complexity is large $K(s) \gg 0$, i.e., formally, larger than a certain threshold C: $K(s) \geq C$.

Comment. We can always use a print statement to generate any sequence s. In other words, for each string s, there is a program of length $\mathrm{len}(s) + c_0$ which generates s – where c_0 is the overhead needed to invoke the print statement. Since the Kolmogorov complexity $K(s)$ is the shortest length of all programs computing s, we can conclude that $K(s) \leq \mathrm{len}(s) + c_0$.

When are results dependent and when they are independent: towards formalization. The notion of dependence was also formalized as part of Kolmogorov complexity-related research. Intuitively, a sequence s depends of the sequence s' if the use of s' can help us compute s. Formally we can define *conditional Kolmogorov complexity* $K(s \,|\, s')$ as the shortest length of a program that computes s by possibly using s' as an input. In these terms, dependence means that if we allow to use s' as an input, then the complexity of s drastically decreases, i.e., that $K(s \,|\, s') \ll K(s)$. Formally, we can describe this as $K(s \,|\, s') \leq K(s) - C$ for some threshold value C.

Correspondingly, independence can be described as $K(s \,|\, s') \geq K(s) - c_0$ for some threshold value c_0.

Now, we are ready for formalize and prove our result.

Comment. In the following text, we will use the following two relations between the conditional Kolmogorov complexity and the original (unconditional) Kolmogorov complexity.

First, in the definition of conditional Kolmogorov complexity $K(s \,|\, s')$, we do not require that the correspondingly program actually do something with the string s'. We also allow programs which simply ignore the string s' and simple compute s "from scratch". In particular, we are allowing the shortest of such "ignoring" programs – whose length is $K(s)$. Since $K(s \,|\, s')$ is the shortest length of all such programs, ignoring s' and not ignoring s', we thus conclude that

$$K(s \,|\, s') \le K(s). \tag{1}$$

The second relation comes from the fact that any program which computes s without using s' can also be viewed as a program which This relation comes from the fact that, by definition, $K(s')$ is the shortest length of a program that computes s'. Thus, for each strong s', there is a program of length $K(s')$ which computes this string s'. Similarly, there exists a program of length $K(s \,|\, s')$ which computes s based on s'. We can therefore combine these two programs into a single program of length $K(s \,|\, s') + K(s') + c_0$ which computes s – where c_0 is the 'overhead" needed to make sure that we first compute s' and then s.

By definition, $K(s)$ is the smallest possible length of a program for computing s. Thus, the fact that we have a program of length $K(s \,|\, s') + K(s') + c_0$ which computes the string s means that

$$K(s) \le K(s \,|\, s') + K(s') + c_0. \tag{2}$$

3 Main Result: Formulation and Proof

Proposition 3.1 *There exists number $c_0 > 0$ and $c > 0$ such that, for every natural number L:*

- *if $K(x) \ge L$, then there exist y and z for which $K(y \,|\, z) \ge K(y) - c$ but $K(y \,|\, z, x) \le K(y \,|\, x) - L + c$;*
- *if there exist y and z for which $K(y \,|\, z) \ge K(y) - c_0$ but $K(y \,|\, x, z) \le K(y \,|\, x) - L$, then $K(x) \ge L - c$.*

Comment. The first implications says that if x is complex (= "truly deep"), then there exist two statements which are independent from each other ($K(y \,|\, z) \ge K(y) - c_0$) but which become strongly dependent in the presence of x: $K(y \,|\, x, z) \le K(y \,|\, x) - L + c_0$. The second implication states that, vice versa, if the presence of x makes two previously independent statements strongly dependent, this means that x is complex.

In other words, this proposition says that a statement x is complex if and it only if it provides a connection between two previously unconnected statements y and z – this is exactly what Gian-Carlo Rota stated in his informal thesis.

Proof. Let us first prove the first implication. Let x be a string for which $K(x) \geq L$. Since $K(x) \leq \text{len}(x) + c_0$, we thus conclude that $\ell \stackrel{\text{def}}{=} \text{len}(x) \geq K(x) - c_0 \geq L - c_0$, and $\ell \geq L - c_0$.

Let us prove that there exists a string y of length ℓ for which $K(y \,|\, x) \geq \ell$. Indeed, let us assume that, vice versa, for every string y of length ℓ, we have $K(y \,|\, x) \leq \ell - 1$. This would means that every such string can be computed by a program of length $\leq \ell - 1$. How many such programs are possible? There are two possible 1-bit sequences, so there are no more than 2 different programs of length 1. Similarly, there are $\leq 2^2$ possible programs of length 2, $\leq 2^3$ possible programs of length 3, etc. Overall, there are $\leq 2 + 2^2 + \ldots + 2^{\ell-1} = 2^\ell - 2$ possible programs of length $\leq \ell - 1$. Since different programs compute different strings, all these programs compute only $\leq 2^\ell - 2$ strings of length ℓ. Overall, there are $2^\ell > 2^\ell - 2$ strings of length ℓ, so there is indeed at least one string y of length ℓ for which $K(y \,|\, x) \geq \ell$.

Let us now take $z = y \oplus x$, where $\oplus$ means bitwise addition modulo 2. Then, due to the known property of addition modulo 2, we get $y = z \oplus x$. In other words, if we know z and x, then we can use a very short program to compute y and therefore, $K(y \,|\, z, x) \leq c_1$ for some constant c_1. On the other hand, by our selection of y, we have $K(y \,|\, x) \geq \ell \geq L - c_0$. Thus, $0 \leq K(y \,|\, x) - L + c_0$, hence $c_1 \leq K(y \,|\, x) - L + c_0 + c_1$ and $K(y \,|\, z, x) \leq K(y \,|\, x) - L + c$ for $c \stackrel{\text{def}}{=} c_0 + c_1$ (and for any larger value c).

To conclude the proof of the first implication, we need to prove that $K(y \,|\, z) \geq K(y) - c$. In other words, we want to prove that the length $\text{len}(p)$ of any program p which computes y based on z is larger than or equal to $K(y) - c$, for some $c > 0$. Indeed, based on each such program p, we can design a new program which computes y based on x:

- first, we compute $z = y \oplus x$ (which takes length $\leq c_1$);
- then, we apply the program p to this z and compute y.

Thus, we get a new program q of length $\text{len}(q) = \text{len}(p) + c_1 + c_2$ (where c_2 is an overhead) that computes y based on x. We selected y in such a way that $K(y \,|\, x) \geq \ell$, i.e., that the shortest length of a program computing y from x is greater than or equal to ℓ. Thus, we have $\text{len}(q) = \text{len}(p) + c_1 + c_2 \geq \ell$, and hence $\text{len}(p) \geq \ell - c_1 - c_2$. On the other hand, since $K(y) \leq \ell + c_0$, we have $\ell \geq K(y) - c_0$. Therefore, $\text{len}(p) \geq K(y) - c$ for $c = c_0 +_1 + c_2$. Since this is true for each program p which computes y from z, we conclude that $K(y \,|\, z) \geq K(y) - c$. The first implication is proven.

Let us now prove the second implication. Let us assume that $K(y\,|\,z) \geq K(y) - c_0$ but $K(y\,|\,x,z) \leq K(y\,|\,x) - L$. The formula (2) implies that $K(y\,|\,z) \leq K(y\,|\,z,x) + K(x) + c_0$. Thus, we conclude that

$$K(y) - c_0 \leq K(y\,|\,z,x) + K(x) + c_0.$$

We also have $K(y\,|\,z,x) \leq K(y\,|\,x) - L$. Due to the inequality (1), we have $K(y\,|\,x) \leq K(y)$ and thus, $K(y\,|\,z,x) \leq K(y) - L$. So, $K(y) - c_0 \leq K(y) - L + K(x) + c_0$. Moving all the terms except for $K(x)$ into the left-hand side, we get the desired inequality $K(x) \geq L - c$, with $c = 2c_0$ (any value $c \geq 2c_0$ will also work). The proposition is proven.

Acknowledgments. This work was supported in part by the National Science Foundation grants HRD-0734825 and HRD-1242122 (Cyber-ShARE Center of Excellence) and DUE-0926721.

References

[1] M. Li and P. M. B. Vitányi, *An Introduction to Kolmogorov Complexity*, Springer-Verlag, Berlin, Hiedelberg, New York, 2008.

[2] G.-C. Rota, "Indiscrete thoughts", In: G. C. Rota, *The Phenomenology of Mathematical Beauty*, Birkhäuser, Boston, Massachusetts, 1997, pp. 121–133.

Received: March 29, 2014