THE LOGIC IN COMPUTER SCIENCE COLUMN [1]

by

Yuri GUREVICH [2]

From Invariants to Canonization

Abstract

We show that every polynomial-time full-invariant algorithm for graphs gives rise to a polynomial-time canonization algorithm for graphs.

1 Motivation

In theoretical computer science, the standard computation model is (still) that of Turing machines. Inputs and outputs of Turing machines are strings. Accordingly algorithms transform strings to strings. In real-life computing, often, inputs and outputs can be beneficially viewed as structures [3]. A good example is relational databases. In principle, structures can be encoded with strings. Let us examine this more carefully.

We start with the special case of ordered structures. An ordered graph $(G,<)$ can be encoded by listing the rows of its adjacency matrix one after another; the given order of vertices gives rise to the order of rows and the order of columns. Alternatively, $(G,<)$ can be encoded by listing, in the lexicographical order, all expressions $E(i,j)$ such that the ith and jth vertices of $(G,<)$ are adjacent. In either case, Code$(G,<)$ uniquely defines the isomorphism type of $(G,<)$, and the other way round: $\text{Code}(G_1,<_1) = \text{Code}(G_2,<_2) \iff (G_1,<_1) \cong (G_2,<_2)$.

[1]The Bull. of Euro. Assoc. for Theor. Computer Sci.,no. 63, October 1997.

[2]Partially supported by NSF and ONR. Address: EECS Dept., Univ. of Michigan, Ann Arbor, MI 48109-2122, USA. Email: gurevich@umich.edu.

[3]The term *structure* is used in the sense of mathematical logic. In this article, by default, structures are finite.

One is tempted to say that $(G, <)$ and $\mathrm{Code}(G, <)$ are easily computable from each other, even though this is not quite correct: $(G, <)$ is not a string and thus cannot serve as input or output in the conventional model[4]. Nevertheless $(G, <)$ and $\mathrm{Code}(G, <)$ are closely related, and $\mathrm{Code}(G, <)$ does represent $(G, <)$ faithfully. Both encodings, but especially the second, generalize easily from the class of graphs to an arbitrary class of structures[5]. In the rest of the article, we pretend that Turing machines can compute with ordered structures as inputs and/or outputs.

Now we turn our attention to structures that are not necessarily ordered. *A priori*, there is no problem; a structure A can be represented by any ordered version $(A, <)$ of it. This means of course that a structure with at least two elements is assigned multiple codes. So what? Is there any problem with that? Actually yes, there is a problem. In combinatorics, database theory, *etc.*, it is often important to not distinguish between isomorphic structures. For example, a database query is supposed to give you information about the database rather than its implementation. The question arises whether one can encode structures in polynomial time in a way that respects isomorphisms. We recall two well-known definitions.

Definition 1 A *full-invariant algorithm* for graphs is an algorithm I that assigns a binary string[6] to every ordered graph in such a way that

$$I(G_1, <_1) = I(G_2, <_2) \iff G_1 \cong G_2.$$

The graph invariant hypothesis asserts that there exists a polynomial time full-invariant algorithm for graphs. $\square$

An example of a full-invariant algorithm is an algorithm that, given an ordered version of a graph G, computes the lexicographically first string of the form $\mathrm{Code}(G, <)$. Unfortunately, it is not known (or believed) that there is a polynomial time version of that algorithm. If I is any full-invariant algorithm, then all information about a graph G is contained in the string $I(G) = I(G, <)$. But can one extract any information about G from $I(G)$ in polynomial time? Of course, you could recapture the graph if $I(G, <)$ has the form $\mathrm{Code}(G, <')$. Such an invariant would define a canonical ordering $<'$ of the vertices of G that does not depend on the given ordering $<$; the resulting ordered graph $(G, <')$ could be seen as the canonical (or normal) form of $(G, <)$.

[4]This remark gives away the bias of this author who proposed a computation model which works with arbitrary structures [Gu95]. Of course, the encoding problem for ordered structures should not be exaggerated; once understood, it is handled easily.

[5]For simplicity, throughout this article, a class of structures always consists of structures of the same vocabulary.

[6]Binary strings are strings in the alphabet $\{0, 1\}$.

Definition 2 *A graph canonization algorithm* is an algorithm C which transforms ordered graphs to ordered graphs and satisfies the following two requirements:

$$\begin{aligned} C(G,<) = (G',<') &\implies G' \cong G, \\ G_1 \cong G_2 &\implies C(G_1,<_1) \cong C(G_2,<_2). \end{aligned}$$

The graph canonization hypothesis asserts that there exists a graph canonization algorithm that works in polynomial time. □

The two definitions readily generalize from the class of graphs to other classes of structures. If I is a full-invariant algorithm for a class K, $A \in K$ and $(A,<)$ is an ordered version of A, define $I(A) = I(A,<)$.

Clearly the graph canonization hypothesis implies the graph invariant hypothesis. In the next section, we will prove the converse. The reader can skip the rest of this section.

The isomorphism relation on graphs gives rise to an equivalence relation on binary strings: $x \equiv y$ if either neither string codes an ordered graph or else there are ordered graphs $(G_1,<_1)$ and $(G_2,<_2)$ such that $x = \mathrm{Code}(G_1,<_1)$, $y = \mathrm{Code}(G_2,<_2)$ and $G_1 \cong G_2$. The two definitions can be reformulated in terms of the equivalence relation $\equiv$. Being unable to prove much about $\equiv$, the authors of [BG1], generalized the two definitions (and their relatives) to arbitrary equivalence relations E on binary strings and analyzed what is or is not true for all such relations E. Theorem 2 of [BG1] asserts, in particular, that the canonization problem for E is not necessarily Cook-reducible[7] to the invariant problem for E. Let's weaken the canonization hypothesis by replacing polynomial time with nondeterministic polynomial time. More exactly *the weak canonization hypothesis* for a string equivalence relation E is this: There exists a canonization algorithm $y = f(x)$ for E such that the length $|y|$ is bounded by a polynomial of $|x|$ and the graph of f is NP. According to Theorem 3 of [BG2], the statement "for all E, the invariant hypothesis implies the weak canonization hypothesis" implies improbable complexity phenomena. One such phenomenon is that, for all NP sets A, B (say of integers), there would exist disjoint $A' \subseteq A$ and $B' \subseteq B$ with $A' \cup B' = A \cup B$.

It turned out that the equivalence relation $\equiv$ is special.

2 The Theorem

Theorem 1 *The graph invariant hypothesis implies the graph canonization hypothesis.*

Proof Assume that I_0 is a polynomial time full-invariant algorithm for graphs. Define *an expanded graph* to be a graph with an additional binary relation on its vertices.

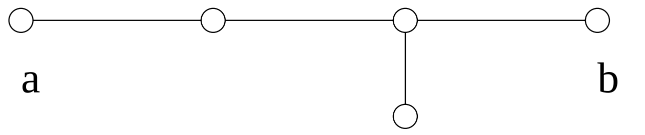
[7] For logicians: Cook reducibility is polynomial time Turing reducibility.

Lemma 1 *There is a polynomial time full-invariant algorithm I for expanded graphs.*

Proof of the lemma. We will exhibit a polynomial time transformation T of expanded graphs into ordinary graphs such that two expanded graphs (G_1, R_1) and (G_2, R_2) are isomorphic if only if the graphs $T(G_1, R_1)$ and $T(G_2, R_2)$ are isomorphic. The desired $I(G, R) = I_0(T(G, R))$.

Remark. More precisely (taking into account the limitations of the conventional computation model), the algorithm transforms an arbitrary ordered expanded graph $(G, R, <_0)$ into an ordered graph $(H, <)$ in such a way that the isomorphism type of H does not depend on the given order $<_0$, and the isomorphism type of H uniquely defines the isomorphism type of (G, R). The desired $I(G, R, <_0) = I_0(H, <)$. We will continue to use the more convenient language of the previous paragraph.

Construction The graph $H = T(G, R)$ is an extension of G. Decorate every element a of G with two adjacent elements a', a'' of degree one. For every pair $(a, b) \in R$, connect a to b by means of three auxiliary elements as follows.



That ends the construction of H. If G contains n vertices and R contains r pairs, then H contains $3n + 3r$ vertices.

To check that H uniquely defines (G, R), notice that the vertices of G are the only vertices of H with two neighbors of degree one. The lemma is proved. $\square$

We continue the proof of the theorem. Let I be a full-invariant algorithm for expanded graphs.

Notation. If v is a vertex of a graph G, let $[v]$ be the binary relation $\{(v, v)\}$. If $v_1, ..., v_k$ are different vertices of G and $k > 1$, let $[v_1, ..., v_k]$ be the partial order $\{(v_i, v_j) : 1 \le i < j \le k\}$ on the vertices of G.

Construction We construct the desired canonization algorithm C. Let $(G, <_0)$ be an ordered graph with n vertices. Here and in the rest of the proof, we ignore the case $n = 1$.

Step 1. Let s_1 be the lexicographically first string of the form $I(G, [v])$ and let v_1 be the first (with respect to $<_0$) vertex v with $I(G, [v]) = s_1$.

Step k, where $1 < k \le n$. We assume that vertices $v_1, \ldots, v_{k-1}$ have been constructed. Let s_k be the lexicographically first string of the form $I(G, [v_1, \ldots, v_{k-1}, v])$ and let v_k be the first (with respect to $<_0$) vertex v with $I(G, [v_1, \ldots, v_{k-1}, v]) = s_k$.

The desired $C(G, <_0) = (G, [v_1, \ldots, v_n])$. End of construction.

Obviously C works in polynomial time and $C(G,<) = (H,<') \Longrightarrow G \cong H$. In the remainder of the proof, we check that $G \cong H \Longrightarrow C(G,<_1) \cong C(H,<_2)$.

Let $C(G,<_1) = (G,[v_1,\dots,v_m])$ and $C(H,<_2) = (H,[w_1,\dots,w_n])$, and assume that $G \cong H$. Then G, H have the same number of vertices; hence $m = n$. By induction on k, we show that $(G,[v_1,\dots,v_k]) \cong (H,[w_1,\dots,w_k])$. When $k = n$, this gives the desired conclusion.

Let f be an isomorphism from G to H. Then $I(G,[v]) = I(H,[fv])$ for all v in G, so that

$$\min\{I(G,[v]) : v \in |G|\} = \min\{I(H,[w]) : w \in |H|\}$$

and therefore $(G,[v_1]) \cong (H,[w_1])$. Here $|X|$ is the set of vertices of the graph X and min is taken of course with respect to the lexicographic order.

Suppose that $k \le n$ and we have already proved that there exists an isomorphism

$$f : (G,[v_1,\dots,v_{k-1}]) \longrightarrow (H,[w_1,\dots,w_{k-1}]).$$

Then $I(G,[v_1,\dots,v_{k-1},v]) = I(H,[w_1,\dots,w_{k-1},fv])$ for all v in $|G|-\{v_1,\dots,v_{k-1}\}$, so that

$$\begin{aligned}&\min\Big\{I(G,[v_1,\dots,v_{k-1},v]) : v \in |G| - \{v_1,\dots,v_{k-1}\}\Big\} =\\ &\min\Big\{I(H,[w_1,\dots,w_{k-1},w]) : w \in |H| - \{w_1,\dots,w_{k-1}\}\Big\}\end{aligned}$$

and therefore $(G,[v_1,\dots,v_k]) \cong (H,[w_1,\dots,w_k])$. □

Corollary 1 *Let K be any class of structures. The graph invariant hypothesis implies the canonization hypothesis for K.*

Proof First check that Lemma 1 generalizes from expanded graphs to any class of structures and in particular to expanded K structures. Then repeat the remaining part of the proof of Theorem 1 replacing graphs with K structures. □

Acknowledgment Thanks to Andreas Blass for useful comments.

References

BG1 A. Blass and Y. Gurevich, "Equivalence relations, invariants, and normal forms", SIAM Journal on Computing 13 (1984), 682–689.

BG2 A. Blass and Y. Gurevich, "Equivalence relations, invariants, and normal forms, II", Springer Lecture Notes in Computer Science 171 (1984), 24–42.

Gu Yuri Gurevich, "Evolving Algebra 1993: Lipari Guide", in "Specification and Validation Methods", Ed. E. Boerger, Oxford University Press, 1995, 9–36. Later updates of the guide appear at http://www.eecs.umich.edu/gasm/