DOI:10.1145/1953122.1953149

Technical Perspective
The Quest for a Logic for Polynomial-Time Computation

By Phokion G. Kolaitis

THE INTERACTION BETWEEN computation and logic goes back to the early beginnings of computer science with the development of computability theory in the 1930s by A. Turing, K. Gödel, A. Church, S. Kleene, and the other great logicians of that era. In more recent decades, the interaction between computation and logic has spanned the entire spectrum of computer science, from programming languages to artificial intelligence and from computational complexity to database systems. Consider, for example, the P vs. NP problem, which is justly regarded to be the central open problem in theoretical computer science.[1] Formally, the P vs. NP problem asks whether or not the class NP of all algorithmic problems solvable by a non-deterministic Turing machine in polynomial time coincides with the class P of all algorithmic problems solvable by a deterministic Turing machine in polynomial time. Informally, the P vs. NP problem asks whether every algorithmic problem whose solutions can be efficiently verified has the property that its solutions can also be efficiently computed. Logic has played a key role in the investigation of this problem. Indeed, in 1971, S. Cook established the existence of "hardest" problems in NP by showing that *Boolean Satisfiability*, a fundamental problem in propositional logic, is NP-complete. This implies that P = NP if and only if there is an efficient algorithm for telling whether or not a given Boolean formula has a satisfying truth assignment.

Soon after Cook's seminal discovery, R. Fagin found an even tighter connection between logic and NP by showing that NP coincides with the class of all algorithmic problems that can be specified using a logical formalism known as *existential second-order logic* or, in short, *ESO*. To illustrate this result, consider *3-Colorability*, another prominent NP-complete problem, which asks whether or not a given graph G (that is, a set of nodes V and a set of edges E) can be colored by assigning one of three colors, say blue, red, and yellow, to each of its nodes, so that no two nodes joined by an edge have the same color. This problem can be specified by the *ESO*-formula shown at the bottom of the page.

Intuitively, this formula asserts that the set V of nodes can be partitioned into three subsets (color classes) B, R, and Y such that if two nodes x and y are joined by an edge $E(x, y)$, then x and y cannot be in the same color class. Fagin's result reinforces the unity between computation and logic by providing a logic-based, machine-free characterization of NP that makes no mention of computational resources (time) or bounds (polynomial) on such resources.

Is there a logic for P? In other words, is there a logical formalism L on the class of all graphs such P coincides with the class of all algorithmic problems on graphs that can be specified in L? This problem was raised by Y. Gurevich in 1984, who actually conjectured that there is *no* logic for P. A related (and essentially equivalent) problem had been raised in 1982 by A. Chandra and D. Harel, who asked: is there is an algorithm that enumerates all polynomial-time computable properties of graphs? Having a logic L for P will make it possible to recast the P vs. NP problem as a problem of pure logic, namely, as the problem of whether or not *ESO* and L have the same expressive power on the collection of all graphs. Furthermore, a logic L for P will serve as a high-level specification language for expressing precisely those algorithmic problems about graphs whose solutions can be efficiently computed.

The quest for a logic for P on the class of all graphs has been going on for the past 30 years. While the problem remains unresolved to date, the quest has given rise to a fascinating journey that has brought together several different strands of research. The following paper by Martin Grohe contains an account of one of the most sophisticated and beautiful results obtained to date in the quest for a logic for P on the class of all graphs. In a nutshell, Grohe shows there is indeed a logic for P for many large classes of graphs of algorithmic interest and mathematical significance. More precisely, his result asserts that there is a logic for P for *every* class of graphs consisting of all graphs that exclude some fixed graph as a minor, that is, as a graph obtained by edge contractions on a subgraph. This result vastly generalizes earlier work showing that a logic for P exists for large classes of graphs, including the class of all planar graphs.

The logic used in Grohe's result is fixed-point logic with counting, a natural extension of first-order logic with a recursion and a cardinality-counting mechanism that has found many applications in other areas of computer science and, in particular in databases. From a technical viewpoint, Grohe's proof is a real tour de force that requires the development of deep graph-theoretic machinery that uncovers new insights into the structure of graphs with excluded minors. Furthermore, the interplay between logic and graph theory yields another bonus, namely, for every class of graphs with excluded minors, the graph isomorphism problem can be solved in polynomial time via a simple combinatorial algorithm. Whether a polynomial-time algorithm exists for the graph isomorphism problem on the class of all graphs is yet another major open problem in computer science.

References

1. Fortnow, L. The status of the P versus NP problem. *Commun. ACM 52*, 9 (Sept. 2009), 78–86.

Phokion G. Kolaitis (kolaitis@cs.ucsc.edu) is a professor of computer science at the University of California Santa Cruz and a research staff member at IBM Research—Almaden.

© 2011 ACM 0001-0782/11/06 $10.00

$$\exists B \exists R \exists Y \forall x ((B(x) \vee R(x) \vee Y(x)) \wedge (\forall y)(E(x, y) \rightarrow \neg(B(x) \wedge B(y)) \wedge \neg(R(x) \wedge R(y)) \wedge \neg(Y(x) \wedge Y(y))))$$