THE LOGIC IN COMPUTER SCIENCE COLUMN[1]

by

Yuri GUREVICH[2]

The Value, if any, of Decidability

Quisani: Hello. It's that time of the year again when we have a discussion, isn't it?

Author: Come on in. It is a busy time but I am happy to see you.

Q: I am afraid my mood is rather critical today.

A: That is all right. Speak your mind.

Q: It is about computability and decidability. It seems to me that logicians' computability has little to do with real computability. Some so-called computable functions are not computable in any practical sense. It is ridiculous to call them computable.

A: It is well known that some computable functions are not feasibly computable.

Q: But in what sense *are* they computable? You, logicians, have hijacked the notion of computability. Computability should mean practical computability.

A: Practical computability is too hard to define. Take a function f, say, from binary strings to binary strings which is not computable by any Turing machine, and define

$$g(x) = \begin{cases} 1 & \text{if } |x| < 2^{10000}, \\ f(x) & \text{otherwise.} \end{cases}$$

Is g practically computable?

Q: It is obvious that g is not computable. It seems reasonable to argue that g is practically computable because it is so easily computable on practical instances.

[1]Bulletin of European Assoc. for Theor. Computer Sci., Feb. 1995, 129–135.

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

But if g is practically computable then computability does not cover practical computability. Doesn't this contradict the Church-Turing thesis?

A: No, Church and Turing spoke about functions computable by algorithms rather than practically computable functions [Ch, T]. According to, say, Turing's thesis, a function computable by any algorithm is computable by a Turing machine (Turing computable).

Q: Do you believe Turing's thesis?

A: One may have reasonable doubts about the thesis, but on the whole it seems convincing to me.

Q: Because different formalizations of computability turned out to be equivalent?

A: No. Once I heard Dana Scott say that all those different formalizations are not that different after all, and I agree with him (though some of the proposed formalizations may seem weaker than others; for example, it is not obvious that Church's λ-calculus is powerful enough to program all recursive functions). But the thesis has survived more than half a century of experimentation. Also, I find Turing's original speculative proof of his thesis very convincing.

Q: What proof? How can one prove such a thesis? Sorry, I have not read Turing's original paper. In general, I lack a taste for the original sources. I like mathematical ideas, but often an idea clears up with progress and can be found in a purer form in later papers.

A: Turing analyzes how a computer executes an algorithm. You will greatly enjoy this particular original source.

Q: What computer? There were no computers in his time.

A: A human computer. Turing argues that, without altering the input-output function, certain assumptions can be made about the algorithm, and he winds up with a Turing machine computing the same function. It is a beautiful piece of speculative philosophy.

Q: What assumptions? Please explain.

A: No, I do not want to spoil your reading, only to whet your appetite.

Q: Does Church give any speculative proof of his thesis?

A: Yes, he does. It isn't as elaborate as Turing's but it is convincing too.

Q: What are the reasonable doubts about Turing's thesis that you alluded to?

A: A closer analysis of Turing's argument reveals various implicit assumptions about the physical world [Ga]. One cannot rule out the possibility that quantum phenomena or some other natural phenomena can be used to construct an algorithm that cannot be simulated by a Turing machine.

Q: If you allow algorithms to exploit nature, then it is easy to refute Turing's thesis. I am sure that no Turing machine can compute whether it will snow in Ann Arbor on a given day, yet there is an obvious, though slow, algorithm for computing that Boolean-valued function: keep watching to see if it snows.

A: I do not buy that this watching procedure is an algorithm.

Q: What is an algorithm?

A: It is not easy to define what is and what is not an algorithm. In a sense, Church-Turing's thesis has been too successful. We tend to define algorithms in a way consistent with the thesis.

Q: I also do not think that the watching procedure is an algorithm, but what is wrong with it? Is it that the result comes out of thin air?

A: Your watching procedure as a genuine algorithm with an oracle, but the oracle cannot be used repeatedly. If physics leads us to an oracle that can be used repeatedly and that behaves consistently, then the oracle can be seen as an algorithm, can't it?

Q: Let me come back to my criticism of the computability concept. I did not argue with Turing's thesis that every function computable by an algorithm is Turing computable. It is the converse that bothers me. Should an arbitrary Turing machine be seen as an algorithm? Some conditions should be imposed on the notion of algorithm.

A: The need for such restrictions is well recognized in computer science.

Q: I question the value of unconstrained computability and related notions like decidability or recursive enumerability. It seems all these notions belong to logic, not to computer science.

A: They are indeed notions of mathematical logic. For example, recursive enumerability is Σ^0_1 definability in formal logical systems of a certain kind, e.g. Peano Arithmetic. I think though that these notions belong to computer science as well. In particular, Turing computability is programmability in usual programming languages if one abstracts from resource limitations; it provides an upper bound for what programs compute.

Q: Maybe computable functions should be called programmable?

A: It is too late to change terminology.

Q: I am not surprised that you defend computability, decidability, *etc.*. You spent years proving decidability/undecidability results; didn't you?

A: Yes, I did.

Q: Your PhD thesis was about the decidability of something. What was it?

A: The first-order theory of ordered Abelian groups (OAGs). Given a first-order sentence about OAGs, my algorithm decides whether the sentence is true in all OAGs.

Q: Is the algorithm practical?

A: No, not at all.

Q: I suspected that. The argument that computability is programmability makes sense to me. It provides a justification for undecidability results; no program solves an undecidable problem. But what about decidability?

A: The decidability of a problem implies that it is not undecidable and it may be a first step toward a more practical algorithm. Sometimes there is an element of sport. Solving an old open question may be honorable; do not underestimate the value of vanity in scientific progress.

Q: The sport argument applies to everything, and I wonder how important that first step is. A decidable problem may be, I am sure, as hard as an undecidable one in practice. Has your algorithm for ordered Abelian groups been eventually refined to a practical algorithm?

A: No. In a sense, there is no practical algorithm for the purpose; that follows from a result of Albert Meyer [M].

Q: Did people think at the time that all or most natural decidable problems have good decision algorithms?

A: I suspect that some did. *A priori*, the thought is not silly. Here is one analogy. There exists an infinite hierarchy of undecidable decision problems that are easier than the halting problem for Turing machines [R], but no natural decision problem of that kind is known as far as I know.

Q: What does easier mean here?

A: Problem $\mathcal{P}_1$ is *easier* than $\mathcal{P}_2$ if $\mathcal{P}_1$ reduces to $\mathcal{P}_2$ but not the other way round. Here $\mathcal{P}_1$ *reduces* to $\mathcal{P}_2$ if some algorithm R takes positive, respectively negative, instances of $\mathcal{P}_1$ to positive, respectively negative, instances of $\mathcal{P}_2$, so that any decision algorithm $A(y)$ for $\mathcal{P}_2$ yields a decision algorithm $A(R(x))$ for $\mathcal{P}_1$.

Q: As you search for pro-decidability arguments, let me ask you if you regret time spent on proving the decidability of various problems?

A: No. It was exciting. Things should be put into a historical perspective. Do you know that Church and Turing came up with their theses to solve a particular problem?

Q: I think I do. It was the decision problem for first-order logic, wasn't it? Given a first-order sentence, tell if it is true in all structures where it is defined. You may have forgotten that we spoke at length about that [Gu1].

A: Right, Church and Turing used their theses to prove that problem undecidable. A wave of additional undecidability results followed. Algebraic theories provided an ample playing ground [ELTT]. Alfred Tarski originated a program of classifying first-order versions (that is, the formalizations in the classical first-order logic) of popular algebraic theories as decidable or undecidable. Numerous theories have been proved undecidable: group theory, the theory of rings, the theory of fields, the theory of the rational field, *etc.*. Some theories turned out to be decidable, but that was rare. Alfred Tarski devised decision procedures for the theories of the field of real numbers and the field of complex numbers. He and Wanda Szmielew proved the decidability of the theory of Abelian groups and suspected the decidability of the theory of ordered Abelian groups. At this point I joined the game.

Q: It seems counterintuitive that the rational field is undecidable but the real field is decidable.

A: Indeed there were some surprises.

Q: Was there a clear goal in the enterprise?

A: You sound like a granting agency. Mathematics may be compared to archaeology; you find a curious spot and dig and see what is there. I guess, logicians wanted to know where the frontier between decidable and undecidable is.

Q: Have they succeeded?

A: To a large extent they did. It isn't easy today to surprise experts with a new natural decision problem whose decidability status is unknown.

Q: I return to my question about the value of mere decidability.

A: In interesting cases, in order to prove the decidability of a problem, one has to make a nontrivial advance in the mathematics of the problem. That side effect is probably the main justification of decidability results.

Q: Did Tarski's program influence algebra?

A: It has changed a part of algebra and refocused it on formal theories. Here is an instructive example. Tarski's proof of the decidability of the first-order theory of the field $\mathbf{C}$ of complex numbers showed that all algebraically closed fields of characteristic zero are elementarily equivalent, that is, indistinguishable by first-order sentences. This made precise the informal Lefschetz's principle (restricted to fields of characteristic zero) that whatever is true for $\mathbf{C}$ is true also for all algebraically closed fields of characteristic zero.

Q: I know nothing about Lefschetz's principle, but something sounds suspicious to me. Was the informal principle restricted to first-order properties?

A: A good point. The Lefschetz principle covers more than first-order properties. Some of the "more" can also be explained logically [BE].

Q: I have an impression that you, logicians, concentrate too much on the first-order logic.

A: Logicians study and apply various logics, but you may be right. First-order logic is *the* classical logic, well studied and beautiful; it is natural for a logician to use it. In many applications, first-order logic isn't the right logic, however,

Q: Was it the right logic for ordered Abelian groups?

A: Not really. Most of interesting known theorems about OAGs were not first-order.

Q: It is possible *a priori* that no formal logic is just right for a given application.

A: Yet some logics may be more appropriate than others. The case of OAGs may be instructive. Later I found a richer formal theory of OAGs; the known theorems could be formulated in the richer theory or slight extensions of it.

Q: But was the richer theory decidable?

A: Yes. Moreover, the richer theory allowed for a much simpler decidability proof.

Q: First-order logic is used widely in computer science. Do you think that in some cases it should be replaced by different logics?

A: Yes, I do.

Q: One day I would like to hear more about that. But let me come back to computability. What exactly is feasible computability? Is it precisely polynomial time computability?

A: I do not think so [Gu2], but some experts, notably Steve Cook, do [Co]. A short discussion will not do justice to this important subject. If you are interested, we can devote another conversation to it.

Acknowledgment

Jon Barwise, Jim Huggins and Chuck Wallace reacted promptly to my last-moment request to read and comment. The article has been discussed more leisurely with Andreas Blass some of whose comments became a part of the article. I am thankful to all of them.

References

BE Jon Barwise and Paul C. Eklof, "Lefschetz's Principle", Journal of Algebra 13 (1969), 554–570.

Ch Alonzo Church, "An Unsolvable Problem of Elementary Number Theory", American J. Math. 58 (1936), 345-363. Reprinted in "The Undecidable", Ed. Martin Davis, Raven Press, New York, 1965, 88–107.

Co Stephen A. Cook, "Computational Complexity of Higher Type Functions", Proc. 1990 Intl. Cong. of Mathematicians, Kyoto, Japan, Springer–Verlag, 1991, 55–69.

ELTT Yuri L. Ershov, Igor A. Lavrov, Asan D. Taimanov and Michael A. Taitslin, "Elementary Theories", Russian Math. Surveys 20 (1965), 35–100 (English translation).

Ga Robin Gandy, "Church's Thesis and Principles for Mechanisms", in: "The Kleene Symposium" (Ed. J. Barwise et al.), North-Holland, 1980, 123–148.

Gu1 Yuri Gurevich, "Zero-One Laws", Bulletin of the European Assoc. for Theor. Computer Science, No. 46 (Feb. 1992), 90–106.

Gu2 Yuri Gurevich, "Feasible Functions", London Math. Soc. Newsletter, No. 206, June 1993, 6–7.

M Albert R. Meyer, "The Inherent Computational Complexity of Theories of Ordered Sets", in Proc. 1974 Intl. Cong. of Mathematicians, Vancouver, B.C., Canada (1974), 477–482.

R Hartley Rogers, "Theory of Recursive Functions and Effective Computability", McGraw-Hill, New York, 1967.

T Alan M. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem", Proc. London Math. Society 2, no. 42 (1936), 230-236, and no. 43 (1936), 544-546. Reprinted in "The Undecidable", Ed. Martin Davis, Raven Press, New York, 1965, 115–154.