Against Three Formalist Computer Program Verification Methodologies (Extended Abstract)

Steven Meyer
Pragmatic C Software Corp., Boston, MA, USA
smeyer@tdl.com

Abstract

Three formalist computer program verification research methodologies are criticized using the philosophy of science developed by Lakatos, Feyerabend and Kuhn. The research programmes are 1) Dijkstra and Hoare verification by formal proofs, 2) De Millo, Lipton and Perlis verification by social processes within the computer science community, and 3) Fezter's philosophical disproof of the very idea of program verification using a semiotic-deductivist philosophical theory. This paper uses philosophy of science to study philosophy of computing instead of the too often unsuccessful method of using methods from strong AI and algorithm study to attempt to solve general philosophical problems. It is argued that further scientific progress in CS needs abandonment of current formalist phenomenological baggage. The paper concludes with an example from the physics of laminar flows that illustrates Feyerabend's problem specific and anti-formalist characterization of computational science.

Against Three Formalist Computer Program Verification Methodologies
Extended Abstract

1. Introduction

In the past, philosophy of computer science (CS) has too often meant applying methods from the strongest forms of AI and formalist algorithm study to solve philosophical problems. But after half a century of using this approach, there has been little scientific progress in CS and no contribution to classical philosophy. Most predictions from strong AI have not come true and little (or no) progress has been made on the P=NP problem that is the central CS theoretical problem. This paper uses philosophy of science to study three theories on the nature of the verification of computer programs. The original idea to study program verification as a road toward understanding the foundational and philosophical problems of CS was motivated by the observation that algorithm study and the P=NP problem are related to the philosophy of mathematical truth. This paper argues that further progress is CS requires giving up the formalist phenomenological baggage embodied by the three theories of program verification criticized here.[1]

Since Karl Popper and his students developed falsificationism and the improved Lakatos-Feyerabend-Kuhn (LFK) methodology of scientific research programmes (MSRP), science has been demarcated from pseudo-science and metaphysics because scientific theories make objective testable claims. An intellectually honest scientist must be willing to abandon a theory if it is falsified. The LFK theory improves naive falsification to allow both refutation and proof. Lakatos has shown that mathematical theories are also testable in this same sense.[2]

It is unimportant whether CS is similar to mathematics or to science because either theory must be testable if it has any claim to intellectual respectability. CS is not the only science that involves both empirical questions and mathematics. The development of physics has been as much a story of testing the correctness of computation outside the world of mathematical axiomatic conventionalism as CS (Smolin[2006], 278-282). Physics uses experiments to criticize mathematics that produces physically impossible infinities while CS uses experiments on computability to test theories of computation.

The author presented a general argument against formalist CS at ECAP 2005 using philosophical arguments from scientific natural philosophy (Meyer[2005]); This paper continues development of an anti-formalist research programme by criticizing three

---

1. Phenomenology as used here follows Pickering's definition as discourse pre-determined by assumptions about the nature of mathematics. According to Pickering[1984], 26-27, one needs a theory of science before cloud chambers (here program verifications) can be interpreted. Following LFK philosophy of science, progress requires avoidance of such assumptions.

2. See Popper[1959] for the original definition of falsification and see Lakatos[1978], 8-101 for its further development as MSRP. Lakatos[1978b], 24-42 develops the idea of quasi-empirical methodology of mathematical research programmes. See Meyer[2002] for further description of the contributions to concrete example based philosophy of science by Lakatos, Feyerabend and Kuhn. Meyer[2008] discusses the concepts of quasi-empiricism and phenomenology as used in this paper.

current computer program verification methodologies. The Dijkstra (Dijkstra[1976]) and Hoare (Hoare[1986]) research programme advocates verification by formal mathematical proof. The Demillo, Lipton and Perlis research programme (DeMillo[1979]) advocates verifying program correctness by the psychology of social processes (irrational group consensus). The Fetzer impossibility of computer program verification research programme (Fetzer[1988, 2001]) attempts to show program verification is impossible using a philosophical theory. It uses the distinction between computer programs and algorithms and assumes the philosophical theory that science works by using inductive discovery and deductive verification (in Fetzer's words from the very nature of causal systems).[3] Two of the theories criticized below advocate program verification and one opposes it. The methodological answer follows the LFK development of disproof and Feyerabend's defense of rationality (Feyerabend[1976]). Namely, there is no generally applicable method for programming, but using the 'cunning of reason', a method can be rationally chosen for any specific problem. Also see the discussion of Feyerabend's method in Smolin[2006] (292-299). The discussion below of Heisenberg's analysis of laminar flow illustrates the problem specific method.

2. Three Verification Methodologies

Due to lack of space in an extended abstract, the three arguments are only sketched here. The talk will provide the detailed arguments.

2.1 Dijkstra's book that created program verification is incorrect

Professor Dijkstra wrote the book *A Discipline of Programming* (Dijkstra[1976]) to define and explain mathematically certain computer programme verification by proof using syntactic structures that Dijkstra claimed simplify the proofs. Dijkstra attempted to apply formal refinement to probably the easiest of all non trivial CS problems. The problem is called the Dutch Nation Flag Problem (*Ibid.*, 111-116). It sorts tertiary values (3 colored balls) into 3 separate regions under some constraints. Dijkstra's formally verified algorithm is both incorrect (although it can be made correct at the cost of efficiency if a non standard English parsing of Dijkstra's prose is used) (Meyer[1983], 6, 11).

There can be no stronger refutation than an experimental dis-confirmation in the book that defines the research programme. The key to efficient algorithms and simple programs that are easy to debug (debugging is a kind of scientific experimentation) is to use a scientific duality. The problem can be viewed as as a two pass binary sort of only two colors in each pass, or as a linear scan that processes all 3 colors at once and distributes three balls to the correct buckets. Since such duality is not possible with axiomatic formal mathematics, Dijkstra's explication failed.

2.2 De Millo et. al. verification as irrational group consensus

De Millo, Lipton and Perlis (DeMillo[1970]) criticize formal program verification because it does not match their perception of how social interactions between

---

3. Fetzer bases his beliefs on the concept of semiotic systems that he attributes to Charles S. Pierce and Newell and Simon (Fetzer[2001], 43-44, 48-52).

mathematicians occur when theorems are proven. Their research programme is immediately pseudo science because there is no objective way to test it. They offer no criteria that would allow the correctness of a verification to be determined when two different communities disagree. Presumably they are assuming only one group exists in which decisions are based on academic hierarchy.

The computer program verification using social processes research programme is so disconnected from reality and so based on creating a closed elitist society that it is nothing more than metaphysics. See Yandell[2002] for a detailed discussion of how mathematicians really interact. The description is the result of Yandell's extensive interviews with prize winning mathematicians. Mackenzie[2004] documents efforts by the CS establishment to prevent the publication of Fetzer's criticizing paper. Meyer[1983], Chapter 3 discusses Dijkstra and De Millo et. al. prevention of publishing any scientific criticism of program verification.

2.3 Fetzer's criticism requires philosophically questionable assumptions

Fetzer (Fetzer[1988, 2001]) argues that program verification is impossible, but instead of using scientific testing of verification. he defends formalist CS by using untestable philosophical assumptions. All Fetzer shows is that verification is not possible in his philosophical world. Fetzer ignores incompleteness results from logic and ignores disagreements among mathematicians over acceptable methods of formal deduction and induction (for example Goedel[1986], Finsler[1996] and Bregner[1995]). Formal philosophy applied to concrete experimental problems (does a program work, it it efficient, does it have bugs?) is immediately refuted because its domain of application comes from our physical world. Fetzer's assumption actually defend formalist programme verification by continuing to carry the baggage of pre-determined semiotic-deductivist foundations of mathematical logic (Fetzer[2001], 280-284).

Fetzer also assumes that science (and therefore CS) follows the inductive-deductive model of science ([Fetzer[1988], 1051-1052) that has been disproven by Lakatos (Lakatos[1978b], 128-192). Finally, Fetzer (p. 1058) makes the mathematical assumption that there is a difference between an algorithm and the program that expresses it. Fetzer's criticism is therefore incorrect because it ignores problem specific knowledge.

3. Heisenberg Problem Specific Example

The physical testing of a hydro-dynamic calculation and the correctness of the related computer simulation from the first half of the 20th century remain open although the physics has been well understood since the 1920s. Heisenberg starts the discussion from his AHQP (Sources for the History of Quantum Physics) (Kuhn-AHQP[1962]) with 'I learned more from Bohr than anybody else the new type of theoretical physics which is almost more experimental than mathematics. That is you have to cover the experimental situation by means of concepts which fit.'

In 1922 Heisenberg wrote a paper on the instability of laminar flow (small oscillations around laminar flows). A year later mathematician Fritz Noether applied a general mathematical theory to show the flow was stable. The proof looked good to everyone including Heisenberg, but Heisenberg believed from his physical intuition that he was

correct.

In 1950, a pupil of Von Neumann performed a digital computer simulation that produced results close to Heisenberg's. This example shows that formalism can not replace science. Also, Heisenberg's result may still turn out to be wrong.

4. Conclusion

This criticism of the current theories of computer program verification shows the bleakness of engineering style formalism and the need to establish literature and science area computer science departments.[4] Unless CS is separated from information technology and business, the most important computational questions of our era will never even be asked. It is time to return to Thomas Kuhn's 1960 historical analysis: 'historically, science and technology have been relatively independent enterprises, going back as far as classical Greece and Imperial Rome!' (Mirowski[2005]).

5. References

Bregner[1995] Breger, H. (Gillies, D. Ed.). *Revolutions in Mathematics.* Oxford, 1995, 249-264.

DeMillo[1979] De Millo, R. A., Lipton, R. J., and Perlis, A. J. Social processes and proofs of theorems and programs. *CACM.* 22, 5(1979), 271-280.

Dijkstra[1976] Dijkstra, E. W. *A Discipline of Programming.* Prentice Hall, 1976.

Fetzer[1988] Fetzer, J. H. Program verification: the very idea. *CACM.* 31, 9(1988), 1048-1063.

Fetzer[2001] Fetzer, J. H. *Computers and Cognition: Why Minds are not Machines.* Kluwer Academic, 2001.

Feyerabend[1975] Feyerabend, P. *Against Method* (London 1975).

Finsler[1996] Finsler, P. (Booth, D. and Ziegler, R. eds.) *Finsler set theory: Platonism and Circularity.* Birkhauser, 1996.

Goedel[1986] Goedel, K. (Feferman, S. ed.) *Collected Works of Kurt Goedel.* vol. I, Oxford, 1986.

Hoare[1986] Hoare, C. A. R. Mathematics of programming. *Byte*. (August 1986), 115-149.

Kuhn-AHQP[1962] Kuhn, T. (Interviewer). Transcript of the AHQP (Sources for the History of Quantum Physics) interview of Werner Heisenberg by Thomas Kuhn, Film A 603.2 (2), 11-32.

Lakatos[1978] Lakatos, I. *Philosophical papers. Vol. 1. The Methodology of Scientific Research Programmes*, Ed. J. Worrall and G. Currie (Cambridge 1978).

Lakatos[1978b] Lakatos, I. *Philosophical papers. Vol. 2. Mathematics, Science*

4. Smolin describes a similar lack of scientific progress in theoretical physics that Smolin attributes to academic organization (Smolin[2001], chap. 19).

and epistemology, Ed. J. Worrall and G. Currie (Cambridge 1978).

MacKenzie[2004] MacKenzie, D. *Mechanizing Proof - Computing, Risk, and Trust.* MIT Press, 2001, 197-281.

Meyer[1983] Meyer, S. *Pragmatic Versus Structured Computer Programming.*, Unpublished (URL www.tdl.com/~smeyer/docs/StructProgBook-IntendedThesis.pdf), 1983.

Meyer[2004] Meyer, S. "Proposal to teach Lakatos-Feyerabend-Kuhn Philosophy of Science", unpublished, 2004, URL: www.tdl.com/~smeyer/docs/lfk-essay.dec22.pdf.

Meyer[2005] Meyer, S. "Toward Anti-Formalist Computer Science", ECAP'05 Abstracts, Computing and Philosophy, p. 27 (Extended abstract and presentation slides available at URL www.tdl.com/~smeyer/docs/antiformalist-cs.foils.pdf).

Meyer[2008] Meyer, S. "Modern Mathematics and Physics are Quasi-empirical in the Same Sense", URL: www.tdl.com/~smeyer/docs/quasi-empirical-math.pdf, 2008.

Mirowski[2005] Mirowski, P. Hoedown in the OK corral: more reflections on the 'social' in current philosophy of science. *Stud. Hist. Phil. Sci. 36 (2005)*, 791-799.

Pickering[1984] Pickering, A. *Constructing Quarks.* University of Chicago Press, 1984.

Smolin[2006] Smolin, L. *The Trouble with Physics - The Rise of String Theory, the Fall of a Science, and What Comes Next.* Houghton Mifflin, 2006.

Popper[1959] Popper, K. *Logic of Scientific Discovery* (New York 1959), section 12.

Yandell[2002] Yandell, B. *The Honors Class - Hilbert's Problems and Their Solvers.* A K Peters, 2002.