Towards Empirical and Scientific Theories of Computation

(Extended Abstract)

Steven Meyer
Pragmatic C Software Corp., Minneapolis, MN, USA
smeyer@tdl.com

Abstract

The current situation in empirical testing of computational theories is considered. First, the development of computational thinking as logic and set theory in the late 19th and early 20th centuries is discussed. Next, current computer science research programmes are described. The paper continues with a discussion of the alternative empirical research programme in the study of mathematics and computation. Finally, two possible antinomies in current computer science are described and the out of favor Finslerian empirical set theory method from the 1920s is discussed to show alternative ways of thinking about and testing computational theories.

1. Late 19th and Early 20th Century Computational Thinking

Computational and algorithmic study goes back at least to the 19th century. The study began with attempts to put the foundation of every day mathematics especially the use of infinity and real numbers on a sounder footing. This development occurred during a period in which mathematical formalism and structuralism were becoming the primary mathematical methodology. Mathematical objects are generated and existence is shown by recursive generation from axioms(Chandler[1982] and Kunen[2009]). This formalist research programme was seen as formalizing all thinking until Russell's paradox was discovered in the beginning of the 20th century. The problem was solved in mathematics by adoption of the Zermelo-Frankel set theory axioms (ZFC) in the beginning of the 20th century (Kunen[2009], 4-8). This "Standard Model" in which sets are everything allowed mathematicians to continue to work on problems without worrying too much about methodological foundations.

2. Computational Thinking After the Beginning of Computer Science

When computer science developed in the second half of the twentieth century, a number of factors combined to eliminate empiricism. 20th century logic and set theory replaced extensional (by listing and enumeration) descriptions by intensional (propositions stating what is desired) formulas and language. Because 19th century mathematics lacked computers, intensional descriptions of numbers and sets made sense because large numbers were not accessible (to calculation?), and it was felt proofs were needed to insure nothing different happened. Unfortunately, intensional descriptions led to antinomies (this Finsler[1996] term is used here instead of the possibly more common "paradox"). Meaning of statements such as *The set of all x such x is irrational* still pose

complications for mathematical foundations. Computational thinking has dealt with such antinomies by turning a blind eye.

Now 50 years after the first computers were constructed, large numbers are accessible, but extensional descriptions are still out of favor. Large but unbounded problems called potential infinity in the 19th century (Goldrei[1998], 66-73) has become 'infinity' by unconscious problem solving. This situation can be seen in the dominant research programmes in the following computer science areas.

2.1 Computer Program Development

In the area of algorithm implementation and computer programming, empiricism was barred by formalism as only computer programme proofs could show program correctness. Edsger Dijkstra's reply to criticism that proofs of computer programme correctness was different from mathematical proofs in a paper by Demillo et. al. (Demillo[1982], also cf. my empirical criticism in Meyer[1983])). The paper's title was "A Political Pamphlet from the Middle Ages". Dijkstra wrote "Besides political, the paper is pre-scientific in the sense that even the shallowest analogy is accepted as justification." (Dijkstra[2006]).

2.2 Artificial Intelligence

In the area that studied the potential and the limits of computation, the artificial intelligence research programme became dominant and actively attempted to prevent competing research programme's from even receiving funding (see Lighthill[1972]). The research programme claimed that it was only a matter of time before computers surpassed humans. Since AI requires performance of logical reasoning better than human mathematicians, no empirical testing of Church's Thesis occurred (Kunen, 198-202).

2.3 Concrete Algorithm Efficiency

In the area of concrete algorithm efficiency analysis, algorithm discovery by efficiency proof analysis became the dominant methodology (see Lakatos[1976] for a discussion of the mathematical method based on analysis of proofs). This method used mathematical methods so no empirical testing of foundation questions could occur, and algorithms created by human ingenuity not amenable to combinatoric proof were by definition impossible.

2.4 NP Completeness as Turing Machine Algorithm Efficiency

The dominant research programme has tied foundation problems in effective computability to the efficiency of non deterministic Turing Machines. The antinomies that bothered 19th and early 20th Century mathematical have been eliminated by convention. Mathematical logic and ZFC became the "standard model" because of the elimination of efficiency models other than P=NP and because of the acceptance of Tarski's satisfiability definition of truth (for criticism see Finsler[1976], 162, "the distinction between true and false [must] be made in an objective [non axiomatic] way"). The idea behind NP completeness is that Church's thesis is useless if computations take more steps than the number of seconds the universe has existed. Since the P=NP problem is axiomatic in nature, no consideration of the alternative Kalmogorov complexity or possible antinomies in interpretation of problems proven to be NP complete is possible.

3. Background of Empirical Testing in Computational Thinking

There are two traditions which are in contrast to current formalist theories. Physicists have believed that one discovers mathematical truths from studying nature. For example J. Fourier has written "The deep study of nature is the most fruitful source of mathematical discoveries" (Moritz[1914], item 612). Lakatos' mathematics as quasi-empiricism documented the empiricist tradition without non foundational mathematics (Lakatos[1976] was widely accepted by set theorist and Logicians in the 1960s but was then rejected starting in the 1970s).

Many mathematicians believe that mathematical concept are not merely conventionalist axioms but either exist or do not exist in the sense of images on Plato's cave wall. This view is called Platonism and implies that scientific testing and experiments can be used to determine the existence of mathematical theses such as the Church Turing Thesis which states that there is no possible computation outside recursively computable functions (Kunen[2009], 186-191). However, current formalist and ZFC set theorists have changed to meaning of Platonism to introspection (mathematical insight or deduction rather than induction from (possibly quasi-empirical) observation (Finsler[1996], p.6 introduction).

4. Two Antinomies

In spite of almost no study, there are a few antinomies in computer science that I have run into in the area of computer program language compiler implementation.

1. **Cooper's unexplained efficient flow graph algorithm** There is an interesting graph theory area algorithm for computing (pre)dominators in computer language compiler flow graphs (Cooper[2001]) that appears to be faster than an algorithm proven to be optimal using algorithm efficiency proof analysis based combinatorics (Tarjan[1974]). If this algorithm is really outside mathematical efficiency provability, it may lead to the discovery of many new more efficient intuitive algorithms.

2. **Computer instruction span dependent jump instruction selection** The problem of selecting optimal assembler jump instructions was proven to be NP complete (Symanzki[1978] and the later [Leverett[1980]). Older computers had two types of jump instructions. One type was a fast and small instruction used if a jump was to a near memory address, and another larger and slower instruction for the general case. Programs are smaller if more local jump instructions are used, but if longer span jump instructions are required, if one wrong instruction is selected, it may cause many short span jump instructions to be eliminated. The history of this is interesting. The first paper "proves" NP completeness and suggested simplified algorithms that run in polynomial time. The second version, substitutes algorithm change with problem change. My claim here is that the general case problem and NP completeness proof show an antinomie because span instruction selection is not NP complete for programs that have meaning in a rather weak consistency of the meaning of computer program execution sense.

5. The Finslerian Scientific Revolution Applied to Church's Thesis

The 20th century Swiss mathematician Paul Finsler developed a detailed Platonistic and empirical alternative to ZFC. Finsler's "revolution" was not accepted (Gillies[1992], essay by Breger H. "A restoration that failed: Paul Finsler's theory of sets", 249-264). However, Finsler provides an alternative research program that may be usable in testing Church's theses and other modern computer science research programmes. Finsler's research programme is especially interesting because he published detailed answers to his critics including detailed re-proofs of his opponents disproofs of his solutions to various antinomies. The remainder of this talk analyzes Finsler's conception of truth and mathematics outside formalist finite Church/Turing computability.

The aim of this talk is to encourage future computation thought that will allow various competing computational research programs to flourish, compete, and be used to test each other.

6. References

Chandler[1982] Chandler, B. and Magnus, W. *The History of Combinatorial Group Theory: A case Study in the History of Ideas*, Springer, New York, 1982.

Cook[1971] Cook, S. "The Complexity of Theorem-proving procedures," *Proceedings of the third Annual ACM symposium on Theory of Computing.*, May, 1971, 151-158.

Cooper[2001] Cooper, K, et. al. "A Simple, Fast Dominance Algorithm", *Softw. Pract. Expr.* 4(2001), 1-10.

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[2006] Dijkstra, E. Dijkstra Archive (URL: http://userweb.cs.utexas.edu/users/EWD), University of Texas, 2008.

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

Gillies[1982] Gillies, D. A. *Frege, Dedekind, and Peano on the Foundations of Arithmetic*. Van Gorcum, Assen, Netherlands, 1982.

Gillies[1992] Gillies, D. (ed.) *Revolutions in Mathematics.*. Oxford, 1992.

Goldrei[1998] Goldrei, D. *Classic Set Theory For Guided Independent Study.* CRC Press, Florida, 1996.

Kunen[2009] Kunen, K. *The Foundations of Mathematics.*, Studies in Logic Vol. 19, College Publications, London, 2009.

Lakatos[1976] Lakatos, I. *Proofs and Refutations.*, Cambridge, 1976.

Leverett[1980] Leverett, B. and Szymanski, T. "Chaining Span Dependent Jump Instructions", *ACM Trans Lang. and Sys. (TOPLAS)*, 3, 2(1980), 274-289.

Lighthill[1972] Lighthill, J, "Lighthill Report - Artificial Intelligence a General Survey" British Science Research Council (original report with

responses available at www.tdl.com/~smeyer), 1972.

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

Moritz[1914] Moritz, R. *On Mathematics and Mathematicians*, Dover, New York, 1914.

Szymanski[1978] Szymanski, T. "Assembling Code for machines with span-dependent instructions", *CACM*. 21, 4(1978), 300-308.

Tarjan[1974] Tarjan, R. "Finding Dominators in Directed Graphs", *SIAM J. Comput., 3(1):62-89, March, 1974.*