Chapter 7
Logic and Proof in Computer Science:
Categories and Limits of Proof Techniques

John W. Coffey
University of West Florida, USA

ABSTRACT

Computer software pervades our lives today. Nevertheless, software is one of the few products for which producers generally provide no express or implied warranties, a truly striking fact since peoples' lives depend in such fundamental ways on these products. This article addresses why such an unintuitive (and undesirable) situation might exist. It will catalog a range of computer science proof techniques and their historical antecedents, the purposes they serve, and several foundational concerns that elude proof techniques of any kind. Along the way, the concept of intractability and its role in computing will be explored as it pertains to algorithmic complexity and to proofs of the meanings of computer programs.

INTRODUCTION

Software is everywhere in peoples' lives today. Computer programs run the gamut from 50-year-old legacy COBOL programs that compute peoples' salaries to programs in the Java and Swift programming languages that bring their smart phones to life. Computer Science is characterized as one of the "hard" sciences – a colloquial term indicating that the field is characterized by applicability of the scientific method, mathematical rigor, and susceptibility to the formulation of testable hypotheses and quantifiable results. While all these attributions are true, and an impressive slate of proof techniques is available, there are still some rather surprising deficiencies within in the field that negatively impact the ability to address some very important problems. Just as one might be disappointed to discover that empirical psychology is based upon 95% probabilities that some phenomenon might occur, people interested in computer science might be disappointed to discover all the "loose ends" that exist in the field.

Computer algorithms and programs are based upon logic, and therefore, would seem to be amenable to proof techniques. Indeed they are, and many different techniques are used. Direct proof techniques

DOI: 10.4018/978-1-5225-2443-4.ch007

including the somewhat misleadingly named math induction technique (which is actually a deductive method), proof by contradiction, constructive proofs, and existence proofs are routinely employed. These techniques will be discussed here. However, it turns out that many critically important, foundational problems in the field present substantial difficulties or are not (at least, so far) amenable to proof techniques at all.

In fact, one of the more interesting aspects of the application of proof techniques to computer science is the disconcerting way in which very important, foundational problems crop up that are either intractable in any other than small quantities (and typically provable as such) or simply unprovable at all. An example of the former is the very large category of problems termed NP-Complete. While an in-depth discussion of NP-Completeness is beyond the scope of this paper, its implications for the field will be briefly discussed. A second critically important example of the former is the application of techniques to prove characteristics of a running computer program, its dynamic semantics. An example of the latter is the seemingly simple problem of proving whether or not a running program will actually halt. It is in these latter examples that the reason behind the "no expressed or implied warranties" issue becomes more comprehensible.

This work is ambitious because it seeks to summarize a broad swath of foundational concerns regarding logic and proof applied to computer science. For this reason, it will only be possible to provide basic motivational examples of the techniques described. The remainder of this paper will point the reader to literature and other resources that allow more in-depth exploration of the techniques described here. After the literature, a survey of proof techniques is presented, along with some examples, and the range of problems they can address is discussed. Following that is a section on intractable and unprovable problems. The paper concludes with a summary and discussion.

BACKGROUND: THE EVOLUTION OF PROOF TECHNIQUES USED IN COMPUTER SCIENCE

Many resources pertaining to proof techniques in computer science are available. Numerous scholarly articles describe seminal (but now well-understood and accepted) work in defining foundational knowledge in the field. Other scholarly articles have been published pertaining to aspects of open or unresolved issues while trying to get traction on aspects of the problems which might lead to a final resolution. Additionally, many educational resources are available online and in hardcopy that describe the importance of proof techniques in the field and their applications. This literature review will contain a summary of some of these major results and resources.

Proof techniques in computer science run the gamut from highly mathematical and formal, to quite informal but still persuasive. Lehman, Leighton and Meyer (2016) state that proofs entail various methods of establishing truth, and that these methods take different forms in different fields. They cite different forms that search for the truth might entail including legal truth (that determined by a judge or jury) authoritative truth (established by an expert), probability truth (established by statistics) and philosophical truth, generally in the form of deductive reasoning, established by "careful exposition and persuasion." They go on to provide a good overview of techniques employed in computer science.

The ancient Greeks laid the groundwork for logic and proof techniques that are very useful in computer science today. Euclid created Euclidian Geometry by posing axioms (self-evidently true statements) and using them to prove theorems. His work gave rise to deductive methods including modus ponens, modus