Unifying Verification Paradigms  
(Extended Abstract)*

Natarajan Shankar

Computer Science Laboratory  
SRI International  
Menlo Park CA 94025 USA  
shankar@csl.sri.com  
URL: http://www.csl.sri.com/~shankar/  
Phone: +1 (415) 859-5272 Fax: +1 (415) 859-2844

Abstract. The field of formal methods is blessed with an overabundance of formalisms (functional, relational, automata-theoretic, modal, and temporal), techniques (resolution, rewriting, induction, and model checking), and application areas (hardware, reactive, fault-tolerant, real-time, and hybrid systems). No single verification approach has proven convincingly superior to the others. I argue that it is both necessary and desirable to develop a unified framework within which different approaches can coexist. The paper outlines some preliminary efforts in this direction in the context of SRI's PVS system. These efforts include the embedding of special-purpose formalisms (e.g., the Duration Calculus) into the general-purpose PVS logic, the integration of theorem proving with various forms of model checking, and the application of theorem proving and model checking to the analysis of tabular specifications.

1 Introduction

There has been a dramatic proliferation of verification formalisms in recent times. Volume B of the Handbook of Theoretical Computer Science [40] takes up the bulk of nearly 1300 pages to survey a small fraction of these formalisms (e.g., Hoare logics, modal and temporal logics, process algebras). Clearly, many of

* Supported by the Air Force Office of Scientific Research under contract F49620-95-C0044 and by the National Science Foundation under contract CCR-9509931 and CCR-930044. Some of the applications described were undertaken for NASA Langley Research Center under contracts NASI-18969 and NASI-20334 and for ARPA through NASA Ames Research Center under contract NASA-NAG-2-891. The views presented in this paper bear the influence of several of my colleagues including John Rushby, Sam Owre, Jens Skakkebøk, Judy Crow, David Cyrluk, Mandayam Srinivas, Patrick Lincoln, and Klaus Havelund. Whether they or the funding agencies would concur with these views is another matter. The word paradigm is perhaps being misused in the title, but the author is not the first to perpetrate this abuse.
these formalisms are useful in obtaining a metatheoretic characterization of completeness, expressiveness, and decidability, but from a practical viewpoint, one has to wonder whether this cornucopia of formalisms really offers any significant advantage over conventional mathematics (e.g., predicate calculus, set theory, algebra, analysis, combinatorics). This multiplicity of formalisms raises several obvious questions and challenges:

- Who needs so many formalisms?
- Does any one formalism address a practical problem in its totality?
- How can we make it easier to implement a novel idea or formalism?
- How can we make different formalisms interact in a coherent manner?

In response to these questions, I believe that:

- Though ordinary mathematics is adequate for describing computational phenomena, formalism provides a convenient layer of abstraction along with notational and deductive tools. For example, temporal logic [28] can succinctly capture the fairness assumptions and the safety and liveness properties of finite-state programs. Furthermore, the propositional fragment of temporal logic is decidable. This fragment of temporal logic can also be used for model checking and for synthesizing control programs [6]. Similar arguments can be made for the numerous variants of Hoare logics, process algebras, algebraic datatype specification languages, dataflow languages, and others.

- In my experience, no individual special-purpose formalism fully addresses the verification problems of practical significance. Though formalism does make it convenient to use mathematics, it does not avoid the need for mathematics. For many practical problems, the main burden of verification is still in the sheer amount of conventional mathematics required to carry out a successful verification. For example, Dutertre [8] found it necessary to build a library of basic facts of mathematical analysis as a prelude to the formalization of hybrid systems. Other work in the verification of real-time and fault-tolerant systems [7] also exhibits a heavy dependence on basic mathematics that are not alleviated by the use of special-purpose formalisms.

- Logical frameworks like Isabelle [33] do make it easier to implement proof checkers for new formalisms. However, as we have already seen, individual formalisms are not adequate for practical verification. For verification purposes, we can obtain a more synergistic interaction between different formalisms by embedding these as theories and notational front-ends within a general-purpose framework for conventional mathematics. In this paper, we describe a few such attempts at integrating special-purpose formalisms and deductive procedures into the framework of PVS.

- Even if we can embed special-purpose formalisms within a general-purpose verification framework, very little has been achieved unless these formalisms can interact smoothly. For example, a unified approach should allow the use of induction, rewriting, and propositional logic for the verification of hardware components, and the application of temporal logic or process algebra for verifying the asynchronous interaction between hardware components.