NASA-CR-205329

Software Development Technologies for Reactive, Real-Time, and Hybrid Systems

1996 Progress Report
NASA grant NAG2-892

P.I. : Professor Zohar Manna
Computer Science Department
Stanford University
Stanford, CA. 94301-9045

Objective

The research is directed towards the design and implementation of a comprehensive deductive environment for the development of high-assurance systems, especially reactive (concurrent, real-time, and hybrid) systems. Reactive systems maintain an ongoing interaction with their environment, and are among the most difficult to design and verify. The project aims to provide engineers with a wide variety of tools within a single, general, formal framework in which the tools will be most effective. The entire development process is considered, including the construction, transformation, validation, verification, debugging, and maintenance of computer systems. The goal is to automate the process as much as possible and reduce the errors that pervade hardware and software development.

Approach

The on-going research proceeds simultaneously on two fronts: theoretical investigations and incremental implementation. Several parts of the proposed toolkit can be realized only after developing the necessary theoretical basis.

The formal framework is based on the generic computational model of Transition Systems, specialized to Fair Transition Systems, Timed Transition Systems, and Phase Transition Systems for the classes of concurrent, real-time, and hybrid systems, respectively. These models cover both hardware and software systems, and underlie all the proposed languages and methods. The most complex of these models, phase transition systems, combines discrete transition systems with continuous processes whose parameters are controlled by differential equations.

On the implementation front, the basic verification environment provides a unique combination of tools, and is being used as the basis for prototype components that test and validate new methods and approaches. A software or hardware designer can communicate with the deductive environment through two classes of languages: system components can be described in a graphical language that extends Statecharts; system specifications and interfaces are described in Temporal Logic, extended to deal with real-time and continuous change.

Progress for 1996

Our research in 1996 focused on the following areas:

- **Combining Deductive and Algorithmic Verification:** The two main approaches to verifying temporal properties of reactive systems are *deductive verification* on the one hand, and *model checking* on the other. (We summarize these in [SUM96b].) In the deductive approach, the validity of a given temporal property over a given program is reduced to the general validity of a set of first-order formulas, through the use of a given set of *verification rules*. In *model checking*, the state space of the program is systematically explored in search for a computation that violates the property being verified, until such a counterexample is found or it is shown that it cannot exist.

  Model checking procedures are usually automatic, while deductive verification often relies on user interaction to identify suitable lemmas and auxiliary assertions. However, model checking is usually applicable only to systems with a finite, fixed number of states, while the deductive approach can verify infinite-state systems and parameterized finite-state systems of arbitrary size.

  In [SUM96a], we present an extension of classical tableau-based model checking procedures to the case of infinite-state systems, using deductive methods in an incremental construction of the behavior graph. Logical formulas are used to represent infinite sets of states in an abstraction of this graph, which is repeatedly refined in the search for a counterexample computation, ruling out large portions of the graph before they are expanded to the state-level. This can lead to large savings, even in the case of finite-state systems. Only local conditions need to be checked at each step, and previously proven properties can be used to further constrain the search. This framework is currently being extended to incorporate well-founded orders and progress measures as well.

In [dAM96] we present a methodology for the verification of temporal properties of systems based on the gradual construction and algorithmic checking of *fairness diagrams*. These are abstractions of the system and its progress properties, and have a simple graphical representation. A proof of a temporal property consists of a chain of diagram transformations, starting from a diagram representing the original system and ending with a diagram that either corresponds directly to the specification, or that can be shown to satisfy it by purely algorithmic methods. Each diagram transformation captures a natural step of the gradual process of system analysis and proof discovery. In [dAKM97] we extend this approach to hybrid systems (see below).

- **Hierarchical Verification Diagrams:** Graphical formalisms can facilitate the task of guiding and understanding a deductive proof. *Verification Diagrams*, introduced by Manna and Pnueli, provide a graphical representation of the direct proof of temporal properties. We previously extended this framework to arbitrary temporal formulas, using *generalized verification diagrams*. In [BMS96], we present a formal framework that allows the combination of multiple diagrams into one proof, thus facilitating incremental proof construction of complex systems and properties. In addition, we extend the applicability of verification diagrams to properties specified by existentially quantified temporal formulas, which are strictly more expressive than quantifier-free temporal formulas.

- **Graphical Formalisms:** In [BdAM$^+$96] we summarize the common features of verification diagrams, fairness diagrams, and deductive model checking. They all can describe and verify infinite-state systems using a finite representation which can be incrementally constructed, where verification conditions are local and where global properties can be checked algorithmically. Diagrams (or sequences of diagrams) are formal proof objects, which succinctly represent a set of verification conditions that replaces a combination of textual verification rules.

The graphical nature of diagrams makes them easier to construct and understand than text-based proofs and specifications.

- **Real-Time Systems:** The current release of our STeP systems (see below) supports the verification of real-time systems, modeled as *clocked transition systems*. We are collaborating with Dr. Jonathan Ostroff of York University, Canada, who is using STeP in the compositional verification of real-time systems. Dr. Ostroff visited our group and is providing feedback on the application of the system to large-scale, compositional real-time verification [ON96, Ost97].

- **Hybrid Systems:** In [dAKM97], we present a methodology for the verification of temporal properties of hybrid systems. The methodology is based on the deductive transformation of *hybrid diagrams*, which represent the system and its properties, and which can be algorithmically checked against the specification. This check either gives a positive answer to the verification problem, or provides guidance for the further transformation of the diagrams. The resulting methodology is complete for quantifier-free linear-time temporal logic.

  We have also continued our collaboration with Prof. Tom Henzinger (see Cornell/Berkeley subcontract below).

- **Probabilistic System Verification:** In [dA97], we present a methodology for the verification of performance and reliability properties of discrete real-time systems. The methodology relies on a temporal logic that can express bounds on the probability of events and on the average time between them. The semantics of the logics is defined with respect to timed systems that exhibit both probabilistic and nondeterministic behavior. We developed model checking algorithms for the algorithmic verification of the specifications.

- **Deductive Support (STeP):** We have continued work on our implementation testbed, the Stanford Temporal Prover (STeP) [BBC+96], which combines deductive methods with algorithmic techniques to verify linear-time temporal logic specifications of reactive and real-time systems. STeP uses verification rules, verification diagrams, automatically generated invariants [BBM97], model checking, and a collection of decision procedures, to verify finite- and infinite-state systems.

  An initial version of STeP (version 1.1) has been released [BBC+95], and is being used for educational and research purposes in over 30 sites around the world. At Stanford, STeP and its deductive components are being used in courses on formal verification and introductory logic for computer science. Information on obtaining the system is available by sending e-mail to `step-request@CS.Stanford.EDU`. See also

`http://theory.stanford.edu/~zm/step.html`.

Version 1.2, ported to the latest relase of SML of New Jersey (109.x), will be available soon, including a Linux version of the system.

Improvements done in 1996 include: stronger decision procedures; rules for modular verification; visual representation of proof searches; support for the verification of real-time systems; and improved interfaces for the non-clausal resolution-based theorem proving component.

Experimental features of STeP developed in 1996 include: a preliminary implementation of deductive model checking [SUM96a] and automata for monadic second order logic, which can automatically verify a class of parameterized programs. More powerful deductive support, integrating decision procedures and first-order reasoning, has been developed and implemented, in collaboration with Dr. Mark Stickel at SRI International [BSU96] (see below).

A Java implementation of Manna and Waldinger's Deductive Tableau proof system, based on non-clausal resolution, was developed this year. This tool was used in teaching CS157 (Introduction to Logic and Automated Reasoning) at Stanford. It will be available for educational use, and will serve as the basis for further work in deductive synthesis.

- **Industrial collaboration.:** We have continued our contact with researchers from SUN and Intel. STeP has been used to verify circuits proposed by Intel and SUN. We have initiated further industrial collaborations (e.g. Yago Systems) that will explore how our deductive technology can be applied to complex hardware designs, which will drive new research and development.

SRI subcontract

- **Deductive Support:** In collaboration with Dr. Mark Stickel at SRI, we developed a procedure for proving the validity of first-order formulas in the presence of decision procedures for an interpreted subset of the language [BSU96]. The procedure is designed to be practical: formulas can have large complex boolean structure, and include structure sharing in the form of `let`- expressions. We accommodate different kinds of decision procedures, from those that can suggest refuting instantiations to those that cannot. The procedure has been implemented as part of STeP. Although the procedure is incomplete, it is able to eliminate the need for user interaction in the proofs of many verification conditions.

- **Program Synthesis:** In collaboration with Dr. Richard Waldinger, recent efforts have been devoted to the synthesis of programs which have side effects as part of their intended behavior. As an application domain for the synthesis of imperative programs, we have been constructing pictures, graphical displays, and animated cartoons from declarative specifications. This has many potential applications, including the automated development of visual displays to illustrate the results of computations.

  The declarative specifications can describe the entities (geometric figures, photographs, pieces of text, and even film clips) and relationships between them (left and right, bigger and smaller). Explicit states allow the description of changes in the configuration.

Cornell/Berkeley subcontract

On January 1, 1996, Prof. Henzinger moved from Cornell to the University of California at Berkeley. There we continued to develop the theory and improve the practice of model checking for real-time and hybrid systems.

- **Theory of real-time model checking:** We developed an on-the-fly algorithm, which is the first method for real-time model checking that is optimal with respect to both time and space requirements [HKV96]. We also discovered a reduction of real-time model checking to untimed model checking, which dramatically increases the applicability of existing tools [KH97].

  In a recent development, we propose and study a new formal model for real-time and hybrid system which is more robust than previous approaches [GHJ97]. In particular, if a robust timed automaton accepts a real-time sequence of inputs, then it must also accept all sequences where the input times are slightly perturbed.

- **Theory of hybrid model-checking:** We classified hybrid automata as to whether they induce finite bisimilarity, similarity, or language-equivalence quotients [HK96]. This classification gives rise to a structural explanation of previous decidability results and to improved verification algorithms [Hen96].

- **Practice of hybrid model checking:** After switching from formula-based to polyhedra-based algorithms, we have been steadily improving HyTech, our symbolic model checker for hybrid automata. The current release, HyTech 1.04, is available at

  `http://www.eecs.berkeley.edu/~tah/.`

  The improvements are driven by case studies that involve parameter-synthesis tasks and nonlinear behavior [HW96].

Future Research (1997)

Our research plans for the following year include:

- **Integration of Techniques:** Further integration of deductive and algorithmic techniques. We will examine more closely how automata-theoretic techniques can be used in the general case of infinite-state systems.

- **Graphical Formalisms:** Although our graphical formalisms are interactive in nature, we want to identify the areas where automation can be most useful, to minimize the necessary interaction.

  We also plan to develop a uniform graphical environment, implemented in Java, which can be used to apply the various graphical formalisms we have developed, using a common set of tools.

- **Hybrid Systems:** Develop a practical graphical environment for the verification of hybrid systems. This includes exploring further the combination of the deductive and algorithmic approaches in the context of hybrid systems, and using the graphical toolkit mentioned above.

- **Modular Verification:** Develop practical techniques that enable the modular verification of concurrent and real-time systems. This includes, for example, finding efficient methods for determining the validity of complex temporal verification conditions.

- **Verification of Object-Oriented Programs:** Apply the temporal logic methodology to the verification of object-oriented systems; in particular, identify compositional and modular verification techniques that can handle and exploit the abstraction, encapsulation and specialization features of object-oriented languages.
- **Hardware-Specific Applications:** Adapt and specialize the techniques used in STeP to the special case of hardware systems, finding methods for generating hardware invariants, and developing specialized deductive support for finite-state systems.
- **Program Synthesis:** Identify new practical applications of deductive program synthesis, and develop a synthesis environment that can use the deductive tools already embodied in our STeP system.
- **Model checking for Hybrid and Real-time Systems:** Continue to extend the practical applicability of our model checking methods for hybrid and real-time systems, following the insights provided by our recent theoretical results.

Publications (1996)

The following references are publications supported by this grant in 1996. Selected entries, marked with ⋆, are included with this report.

[BBC+96] N. S. Bjørner, A. Browne, E. Chang, M. Colón, A. Kapur, Z. Manna, H.B. Sipma, and T.E. Uribe. STeP: Deductive-algorithmic verification of reactive and real-time systems. In *Proc. 8th Intl. Conference on Computer Aided Verification*, volume 1102 of *LNCS*, pages 415–418. Springer-Verlag, July 1996. ⋆

[BBM97] N.S. Bjørner, A. Browne, and Z. Manna. Automatic generation of invariants and intermediate assertions. *Theor. Comp. Sci.*, 1997. To appear. ⋆

[BdAM+96] A. Browne, L. de Alfaro, Z. Manna, H.B. Sipma, and T.E. Uribe. Diagram-based formalisms for the verification of reactive systems. In *CADE-14 Workshop on Visual Reasoning*, 1996. ⋆

[BMS96] A. Browne, Z. Manna, and H.B. Sipma. Hierarchical verification using verification diagrams. In *Second Asian Computing Science Conf.*, volume 1179 of *LNCS*, pages 276–286. Springer-Verlag, December 1996. ⋆

[BSU96] N.S. Bjørner, M.E. Stickel, and T.E. Uribe. A practical combination of first-order reasoning and decision procedures. Computer Science Department, Stanford University. Submitted, December 1996.

[dA97] L. de Alfaro. Temporal logics for the specification of performance and reliability. In *14th Symp. on Theoretical Aspects of Computer Science*, February 1997. ★

[dAKM97] L. de Alfaro, A. Kapur, and Z. Manna. Hybrid diagrams: A deductive-algorithmic approach to hybrid system verification. In *14th Symp. on Theoretical Aspects of Computer Science*, February 1997. ★

[dAM96] L. de Alfaro and Z. Manna. Temporal verification by diagram transformations. In *Proc. 8th Intl. Conference on Computer Aided Verification*, volume 1102 of *LNCS*, pages 287–299, July 1996. ★

[GHJ97] V. Gupta, T.A. Henzinger, and R. Jagadeesan. Robust timed automata. In *Proceedings of the First International Workshop on Hybrid and Real-time Systems (HART)*, LNCS. Springer-Verlag, 1997. To appear.

[Hen96] T.A. Henzinger. The theory of hybrid automata. In *Proc. 11th IEEE Symp. Logic in Comp. Sci.*, pages 278–292. IEEE Computer Society Press, 1996.

[HK96] T.A. Henzinger and P.W. Kopke. State equivalences for rectangular hybrid automata. In *Seventh International Conference on Concurrency Theory (CONCUR)*, volume 1119 of *LNCS*, pages 530–545. Springer-Verlag, 1996.

[HKV96] T.A. Henzinger, O. Kupferman, and M.Y. Vardi. A space-efficient on-the-fly algorithm for real-time model checking. In *Seventh International Conference on Concurrency Theory (CONCUR)*, LNCS, pages 514–529. Springer-Verlag, 1996.

[HW96] T.A. Henzinger and H. Wong-Toi. Using HYTECH to synthesize control parameters for a steam boiler. In *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, volume 1165 of *LNCS*. Springer-Verlag, 1996.

[KH97] O. Kupferman and T.A. Henzinger. From quantity to quality. In *Proceedings of the First International Workshop on Hybrid and Real-time Systems (HART)*, LNCS. Springer-Verlag, 1997. To appear.

[SUM96a] H.B. Sipma, T.E. Uribe, and Z. Manna. Deductive model checking. In *Proc. 8th Intl. Conference on Computer Aided Verification*, volume 1102 of *LNCS*, pages 208–219. Springer-Verlag, July 1996. ⋆

[SUM96b] H.B. Sipma, T.E. Uribe, and Z. Manna. Model checking and deduction for infinite-state systems. Computer Science Department, Stanford University. Submitted, December 1996.

Other References:

[BBC+95] N.S. Bjørner, A. Browne, E. Chang, M. Colón, A. Kapur, Z. Manna, H.B. Sipma, and T.E. Uribe. STeP: The Stanford Temporal Prover, User's Manual. Technical Report STAN-CS-TR-95-1562, Computer Science Department, Stanford University, November 1995. ⋆

[ON96] J.S. Ostroff and H.K. Ng. Verifying real-time systems using untimed tools. In *Proc. Third AMAST Workshop on Real-Time Systems*, pages 132–146, March 1996.

[Ost97] J.S. Ostroff. A visual toolset for the design of real-time discrete event systems. *IEEE Trans. on Control Systems Technology*, May 1997. To appear.