To appear in 8th *International Conference on Computer Aided Verification*
Lecture Notes in Computer Science,
Springer-Verlag, July 1996.

STeP: Deductive-Algorithmic Verification of Reactive and Real-time Systems*

Nikolaj Bjørner, Anca Browne, Eddie Chang, Michael Colón,
Arjun Kapur, Zohar Manna, Henny B. Sipma, and Tomás E. Uribe

Computer Science Department, Stanford University
Stanford, CA 94305

Abstract. The Stanford Temporal Prover, STeP, 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, model checking, and a collection of decision procedures to verify finite- and infinite-state systems.

System Description: The Stanford Temporal Prover, STeP, supports the computer-aided formal verification of reactive, real-time (and, in particular, concurrent) systems based on temporal specifications. Reactive systems maintain an ongoing interaction with their environment; their specifications are typically expressed as constraints on their behavior over time. STeP is not restricted to finite-state systems, but combines algorithmic and deductive methods to allow the verification of a broad class of systems, including parameterized (N-component) circuit designs, parameterized (N-process) programs, and programs with infinite data domains.

The deductive methods of STeP verify temporal properties of systems by means of verification rules and verification diagrams. *Verification rules* are used to reduce temporal properties of systems to first-order verification conditions [8]. *Verification diagrams* [7, 3] provide a visual language for guiding, organizing, and displaying proofs. Verification diagrams allow the user to construct proofs hierarchically, starting from a high-level, intuitive proof sketch and proceeding incrementally, as necessary, through layers of greater detail.

Deductive verification almost always relies on finding, for a given program and specification, suitably strong auxiliary invariants and intermediate assertions. STeP implements a variety of techniques for automatic *invariant generation*. These methods include *local*, *linear* and *polyhedral* invariant generation, which perform an approximate, abstract propagation through the system [2]. Verification conditions can then be established using the automatically generated auxiliary invariants as background properties.

* This research was supported in part by the National Science Foundation under grant CCR-92-23226, the Advanced Research Projects Agency under NASA grant NAG2-892, the United States Air Force Office of Scientific Research under grant F49620-93-1-0139, the Department of the Army under grant DAAH04-95-1-0317, and a gift from Intel Corporation.

STeP also provides an integrated suite of simplification and decision procedures for automatically checking the validity of a large class of first-order and temporal formulas. This degree of automated deduction is intended to efficiently handle most verification conditions that arise in deductive verification. An interactive Gentzen-style theorem prover and a resolution-based prover are available to establish the verification conditions that are not proved automatically.

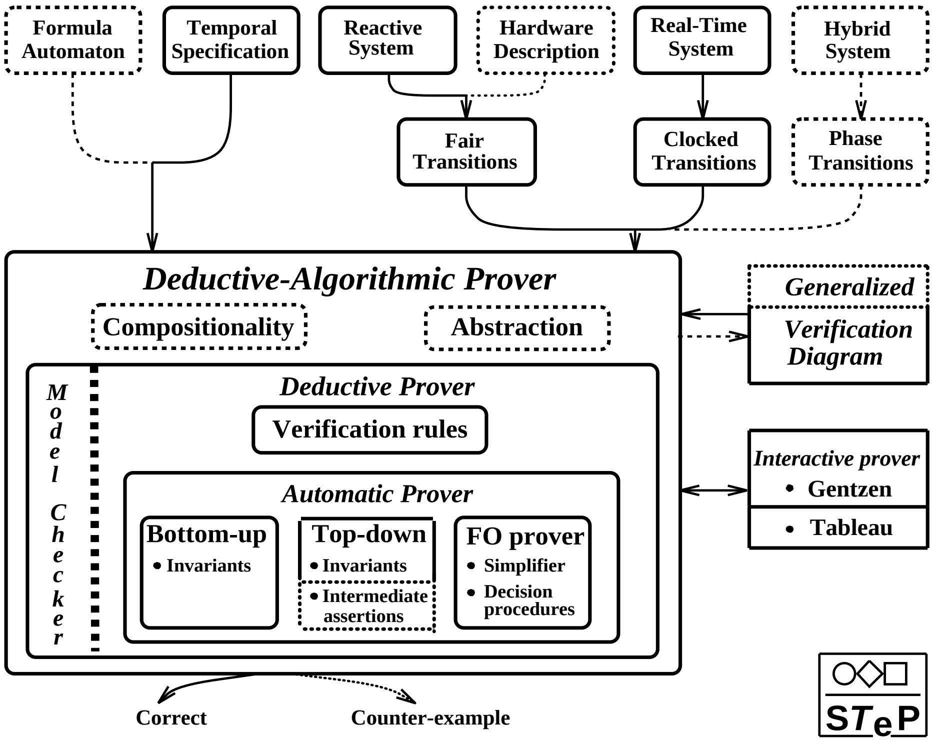

Fig. 1. An overview of the STeP system

System Structure: Fig. 1 presents an overview of STeP. Dotted lines indicate work in progress. The basic inputs are a reactive system (which can be a hardware or software description), expressed as a transition system, and a system property to be proved, represented by a temporal logic formula. Verification can be performed by the model checker or by deductive means. User guidance can be provided as intermediate assertions or verification diagrams. In either case, the system is responsible for generating and proving all of the required verification conditions. Tactics are available to automate parts of the high-level proof search by encoding long or repetitive sequences of proof commands. For a more extensive description of STeP and examples of verified programs, see [1, 6].

Interacting with STeP: STeP has three main interface components: the *Top-level Prover*, from which verification sessions are managed and verification rules are invoked; the *Interactive Prover*, used to prove the validity of first-order and temporal-logic formulas that are not proved automatically; and the *Verification Diagram Editor*, for the creation of Verification Diagrams. Fig. 2 shows these three interfaces, with a version of the Bakery algorithm loaded, together with a tree representing the ongoing proof process.

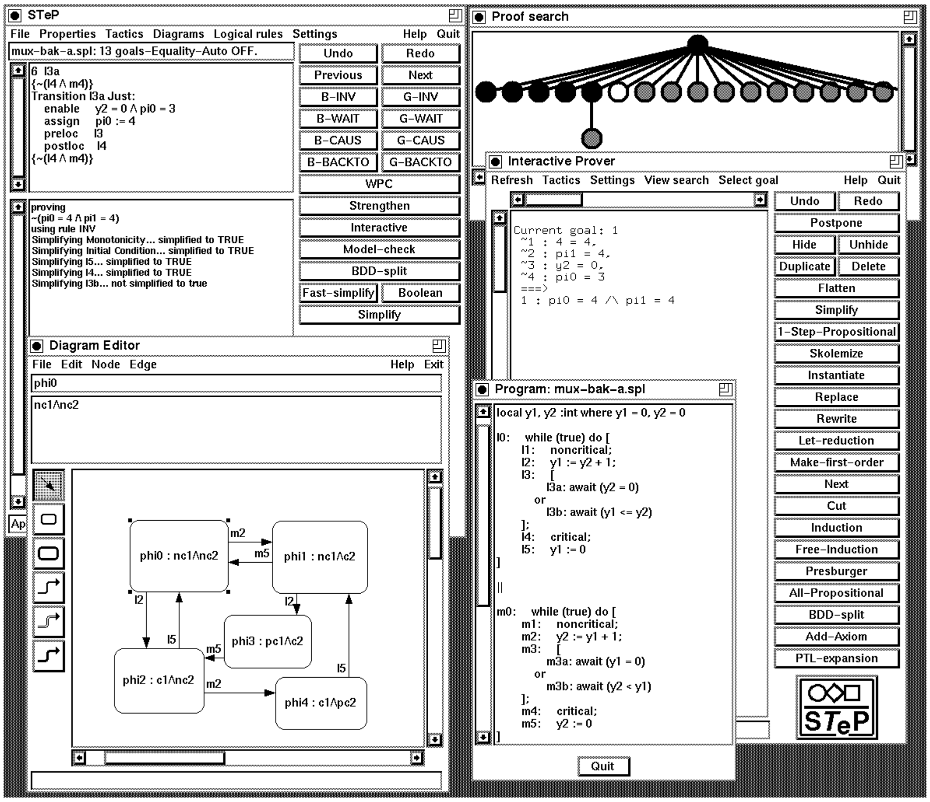

Fig. 2. Overview of STeP's interfaces

Real-time systems: STeP was recently extended to support the verification of safety properties of real-time systems, based on the computational model of *clocked transition systems* [9]. Systems described by timed transition systems or timed automata can be readily translated into this formalism. The specification language of linear-time temporal logic was extended, in turn, with real-valued clocks measuring progress of time.

Applications: STeP has been used to analyze a diverse number of systems, including: an infinite-state demarcation protocol used in distributed databases, a pipelined four-stage multiplication circuit, Ricart and Agrawala's mutual exclusion protocol, several (N-component) ring arbiters, Szymanski's N-process mutual-exclusion algorithm, and an industrial split-transaction bus protocol to coordinate access for six processors. Real-time systems analyzed include Fisher's mutual-exclusion protocol and a (parameterized) railroad gate controller [5].

STeP is being extended to support *deductive model checking* as described in [10], as well as modular verification diagrams [4].

Educational Version: An educational version of the system, which accompanies the textbook [8], is available. The distribution includes a comprehensive user manual [1] and a tutorial, as well as 40 example programs and their specifications, from the textbook, ready to be loaded. For many programs, ready-to-load verification diagrams are included as well.

STeP is implemented in Standard ML of New Jersey, using CML and eXene for its X-windows user interface. For information on obtaining the system, send e-mail to `step-request@cs.stanford.edu`.

References

1. Bjørner, N., Browne, A., Chang, E., Colón, M., Kapur, A., Manna, Z., Sipma, H., and Uribe, T. STeP: The Stanford Temporal Prover, User's Manual. Tech. Rep. STAN-CS-TR-95-1562, Computer Science Department, Stanford University, Nov. 1995.
2. Bjørner, N., Browne, A., and Manna, Z. Automatic generation of invariants and intermediate assertions. In 1^{st} *Intl. Conf. on Principles and Practice of Constraint Programming* (Sept. 1995), vol. 976 of *LNCS*, Springer-Verlag, pp. 589–623.
3. Browne, A., Manna, Z., and Sipma, H. Generalized verification diagrams. In *15th Conference on the Foundations of Software Technology and Theoretical Computer Science* (Dec. 1995), vol. 1026 of *LNCS*, pp. 484–498.
4. Browne, A., Manna, Z., and Sipma, H. Modular verification diagrams. Tech. rep., Computer Science Department, Stanford University, 1996.
5. Heitmeyer, C., and Lynch, N. The generalized railroad crossing: A case study in formal verification of real-time systems. In *Proc. ICCC Real-Time Systems Symposium* (1994), IEEE Press, pp. 120–131.
6. Manna, Z., Anuchitanukul, A., Bjørner, N., Browne, A., Chang, E., Colón, M., de Alfaro, L., Devarajan, H., Sipma, H., and Uribe, T. STeP: The Stanford temporal prover. Tech. Rep. STAN-CS-TR-94-1518, Computer Science Department, Stanford University, July 1994.
7. Manna, Z., and Pnueli, A. Temporal verification diagrams. In *Proc. Int. Symp. on Theoretical Aspects of Computer Software* (1994), vol. 789 of *LNCS*, Springer-Verlag, pp. 726–765.
8. Manna, Z., and Pnueli, A. *Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, New York, 1995.
9. Manna, Z., and Pnueli, A. Clocked transition systems. Tech. Rep. STAN-CS-TR-96-1566, Department of Computer Science, Stanford University, Apr. 1996.
10. Sipma, H., Uribe, T., and Manna, Z. Deductive model checking. In *Proc.* 8^{th} *Intl. Conference on Computer Aided Verification* (July 1996), Springer-Verlag.