Quantifiers, Decision Procedures, and the Verification of Reactive Systems

Nikolaj S. Bjørner, Tomás E. Uribe
Computer Science Department
Stanford University, Stanford, CA. 94305-9045
`nikolaj|uribe@CS.Stanford.EDU`

1 Introduction

In the last three years, we have worked on the development of the Stanford Temporal Prover, STeP, a tool of the verification of reactive systems [2, 1]. These are systems which have an ongoing interaction with their environment, including concurrent, real-time and hybrid systems. Their behavior is best expressed as temporal properties [8].

STeP is intended to verify temporal properties of infinite- as well as finite-state reactive systems, including both hardware and software. In the general case, *verification rules* are used to reduce the validity of a temporal property over a given system to the general validity of a set of first-order *verification conditions* [9]. General theorem-proving tools are required to establish these verification conditions, often in the presence of previously established properties, lemmas and invariants. Therefore, the following theorem-proving support is included in STeP:

- Automatic simplification of verification conditions, including contextual rewriting and decision procedures. This automatic simplifier is intended to be fast, rather than complete.
- An interactive environment for the analysis of general first-order formulas. This provides a complete set of rules for first-order logic.
- Facilities for temporal-logic reasoning: decision procedures for propositional linear-time temporal logic, temporal-tableau construction [7], and specialized interactive theorem-proving rules.

2 Specialized theorem-proving tools

2.1 Decision procedures and their combination

A number of particular theories are prevalent in formal verification: integer linear arithmetic, datatypes and finite domains occur in most systems to be verified; rationals and reals appear in the analysis of real-time and hybrid systems.

Efficient decision procedures for these theories can relieve much of the burden on the user, by discharging automatically many trivial verification conditions. To our knowledge, general-purpose theorem-proving systems cannot match the performance offered by these decision procedures for ground satisfiability, even when the decision procedures themselves are encoded as tactics or strategies of a general-purpose prover.

Like many other verification groups, we have found congruence closure [10] to be an essential component of the decision procedures; Shostak's combination of decision procedures based on congruence closure [11, 6] enhances the power of the basic equality reasoning by integrating *solvable* theories, such as datatypes and linear arithmetic.

2.2 First-order reasoning and decision procedures

Quantifiers appear in the axiomatization of theories for which there is no specialized support, as well as when analyzing *parameterized* systems, which have an arbitrary number of components, and (infinite-state) software and real-time systems [4]. Thus, the ground-level reasoning provided by most efficient decision procedures is no longer sufficient. Accordingly, we have developed a combination of first-order reasoning and decision procedures [5], intended to give a practical but not necessarily complete integration.

We found that such a procedure should satisfy the following requirements:

- It should handle large formulas with complex boolean structure, and allow for structure-sharing (e.g. as `let-` expressions). Thus, conversion into clausal form is not an option.
- It should be able to use "black box" decision procedures which can only decide the satisfiability of given sets of literals.
- The language should include quantifiers, mixed terms over combined theories, and uninterpreted function symbols.
- Whenever possible, $\mathcal{T}$-refuting substitutions suggested by the decision procedures should be used to instantiate quantifiers.

3 Other Practical Concerns

Our experience with verification of reactive systems has been that:

1. The verification conditions we encounter are mathematically trivial but rely on a large number of case distinctions.
2. Counterexamples are more important than detailed proof objects. Practical verification is often best regarded as a powerful debugging methodology. In this context, users can benefit most from counterexamples that invalidate conjectures, which can be used to correct the system or adjust the specification.
3. Static analysis of a program can automatically generate invariants [3], which serve as simple but useful auxiliary lemmas. This automation can generate a large database of axioms that must be efficiently used by the simplifier and first-order validity checker. This poses trade-offs of time versus power. Stronger automatic procedures will automatically establish more verification conditions, but can be significantly slower.

As a consequence, we make the following observations:

1. It seems inevitable to make the time vs. power trade-off visible to the user, who should be able to easily determine how much time is spent on the automatic simplification of verification conditions. We do this by presenting a range of simplification procedures of varying strength and scope.

2. Proof objects in pure logical frameworks (such as primitive recursive arithmetic or set-theoretic formalizations of mathematics) are of little interest, even though decision procedures can in principle be used to generate them. This would only be necessary, perhaps, in truly safety-critical verification efforts, if the decision procedures are not to be trusted, once the basic debugging phase is completed.

   On the other hand, a high-level proof containing the necessary system properties and their dependencies offers a transparent and rigorous documentation tool.

3. Standard benchmarks for automatic theorem provers for the pure first-order calculus are not in the scope of our applications. As pointed out earlier, our verification conditions are best established in theories admitting specialized and efficient decision procedures, and are not, in turn, in the scope of such provers.

Obtaining STeP: To obtain a copy of STeP, send email 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., Chang, E., Colón, M., Kapur, A., Manna, Z., Sipma, H., and Uribe, T. STeP: Deductive-algorithmic verification of reactive and real-time systems. In *Proc. 8^{th} Intl. Conference on Computer Aided Verification* (July 1996), vol. 1102 of *LNCS*, Springer-Verlag, pp. 415–418.

[3] Bjørner, N., Browne, A., and Manna, Z. Automatic generation of invariants and intermediate assertions. *Theor. Comp. Sci. 173*, 1 (Feb. 1997), 49–87.

[4] Bjørner, N., Manna, Z., Sipma, H., and Uribe, T. Deductive verification of real-time systems using STeP. In *AMAST Workshop on Real-Time Systems* (1997), LNCS, Springer-Verlag. To appear.

[5] Bjørner, N., Stickel, M., and Uribe, T. A practical combination of first-order reasoning and decision procedures. In *14th Intl. Conference on Automated Deduction* (July 1997), LNCS, Springer-Verlag. To appear.

[6] Cyrluk, D., Lincoln, P., and Shankar, N. On Shostak's decision procedure for combinations of theories. In *Proc. 13th Int. Conf. on Automated Deduction* (1996), vol. 1104 of *LNAI*, Springer-Verlag.

[7] Kesten, Y., Manna, Z., McGuire, H., and Pnueli, A. A decision algorithm for full propositional temporal logic. In *Proc. 5^{th} Intl. Conference on Computer Aided Verification* (1993), vol. 697 of *LNCS*, Springer-Verlag, pp. 97–109.

[8] Manna, Z., and Pnueli, A. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, New York, 1991.

[9] Manna, Z., and Pnueli, A. *Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, New York, 1995.

[10] Nelson, G., and Oppen, D. Fast decision procedures based on congruence closure. *J. ACM 27*, 2 (Apr. 1980), 356–364.

[11] Shostak, R. Deciding combinations of theories. *J. ACM 31*, 1 (Jan. 1984), 1–12.