Efficient Generation of Set of Support for Safety Properties using UNSAT Cores and Induction [⋆]

Elaheh Ghassabani[1], Andrew Gacek[2], and Michael W. Whalen[1]

[1] Department of Computer Science and Engineering
University of Minnesota, 200 Union Street, Minneapolis, MN 55455,USA
{whalen,ghassaba}@cs.umn.edu
[2] Rockwell Collins Advanced Technology Center
400 Collins Rd. NE, Cedar Rapids, IA, 52498, USA
{andrew.gacek}@rockwellcollins.com

Abstract. Sequential model checkers can construct proofs of very complex models. However, the results reported by the tool when a proof succeeds do not generally provide much insight to the user. It is often useful for users to have traceability information: which portions of the model were necessary to the proof. This traceability information can be used to diagnose a variety of modeling problems such as overconstrained axioms and underconstrained properties, and can also be used to measure *completeness* of a set of requirements over a model. In this paper, we present a new algorithm to efficiently compute the set of support within a model necessary for inductive proofs of safety properties for sequential systems. The algorithm is based on the UNSAT core support built into current SMT solvers and a novel encoding of the inductive problem to try to generate a minimal set of support. We prove our algorithm correct, and describe its implementation in the jkind model checker for Lustre models. We then present an experiment in which we benchmark the algorithm in terms of speed, robustness, and minimality, with promising results.

Keywords: Auto-traceability, Set of Support, Completeness, Requirement Engineering

1 Introduction

- Overview of the problem: sequential model checkers do not provide much insight into proofs.
- Section should roughly follow the structure of ”Finding Minimal Unsatisfiable Cores of Declarative Specifications” paper by Torlak et al (with Dan Jackson).
- UNSAT Cores have been used for a variety of analysis tasks
- we want to generalize this idea for sequential systems

[⋆] This work has been supported by XXX

2 Motivating Example

- Not sure if this should go before or after the background section with a description of Lustre.
- Need a small but interesting example. Andrew, do any of the models that you use as jkind tests function in this way? It would be nice to look at what we have lying around; we need something that requires invariants.
- It would also be good to have a few points of interest with the model-requirement pairing:
- vacuity due to an overconstrained environment
- definitions within the model that are irrelevant to the proof.
- Explain the model and the proof process.

3 Preliminaries

- Symbolic transition systems (use material from Sheeran's "Induction using a SAT Solver" paper?)
- Lustre language
- UNSAT cores
- jkind
- more here?

4 Set of Support

- What is it?
- How do we formalize it in terms of symbolic transition systems?
- How do we prove it correct?

5 Implementation

- This section will probably be fairly short.

6 Experiments

- Research questions (informally):
- RQ1: How much overhead does computation of set of support add to inductive proofs?
- RQ2: How stable is the computed set of support given different solvers and verification algorithms?
- RQ3: How close to minimal is the computed set of support compared to a much slower, but guaranteed minimal approach?
- Experimental setup: What did we use for benchmark models? Where did they come from? Why is this a good set?
- Tables related to each research question.

7 Related work

Alloy is a framework for describing high-level design of various systems, whose analyzer is a fully automatic constraint solver. Constraints are translated into propositional logic solved by a SAT solver; hence, the analysis considers only a finite number of values for each type. For this reason, even for a set of simple constraints, the analyzer is never able to prove the correctness of a property.

- MUS's
- Work on Alloy
- Work that Teme pointed us to.
- Anything else Elaheh has found.

8 Conclusions and Future Work

- Write this at the end.

Acknowledgments: We thank XXXX

References