The Burden of High-Level Languages: Complicated Symbolic Model Checking

Sebastian Krings

Heinrich-Heine-University, Duesseldorf, Germany
krings@cs.uni-duesseldorf.de

Abstract

Symbolic model checking algorithms like IC3 have proven to be an effective technique for hardware model checking. Extensions to software model checking have been suggested and implemented and seem promising so far. However, using symbolic model checking algorithms for the specification languages B and Event-B is complicated. This is due to their high-level nature, which accounts for complex constraints. While some problems can be coped with by modifications to the algorithms, others relate to the heart of our tools: the constraint solver.

1 Introduction

At the software engineering and programming languages department at the Heinrich-Heine-University we are mainly concerned with developing tools for the B method, a rigorous development method for software and systems. The most important tool we develop is ProB [6, 5], an animator, constraint solver and model checker.

Within ProB, I am working on the development of symbolic techniques for the analysis of systems and model checking. These range from finding counter-examples over verifying refinement relations to automatic proof methods.

My PhD research centers around symbolic model checking techniques for models developed using the B method. In Section 2 I will describe my initial goals and the obstacles that occurred nearly immediately after taking on research. Following, in Sections 3 and 4 I will evaluate upon our approaches to overcome them. I will draw a conclusion in Section 5.

2 Research Goals

When I started working on my PhD, ProB's explicit state model checker could already be used to analyze the state spaces of very large software systems due to techniques like partial order reduction [2] or symmetry breaking [8]. State transitions and different B expressions are computed using a constraint solver written in SICStus Prolog.

As all this machinery was based on explicit state model checking. As a consequence, ProB could only be used to verify models involving finite state spaces. If necessary, finiteness was (and sometimes still is) enforced by limiting the cardinality of sets or the size of integers.

For my PhD, I set out to extend the existing techniques by a full-blown symbolic model checking algorithm. My goals were as follows:

- Identify symbolic model checking algorithms suited for B and Event-B.
- Strengthen ProB's constraint solver if necessary.
- Implement selected algorithms using the strengthened solver.
- Empirically evaluate by comparing to ProB's explicit state model checker.

Model	ProB	BMC	k-Ind.	IC3
Coloring			✓	✓
Counter			✓	✓
CounterWrong	✓	✓	✓	✓
f_m0	✓			✓
f_m1	✓			✓
Branching		✓	✓	✓
R0_GearDoor	✓		✓	✓
R1_Valve	✓		✓	✓
R2_Outputs	✓		✓	✓
R3_Sensors	✓		✓	✓
R4_Handle	✓			
Search				
SearchEvents		✓		✓
TravelAgency	✓			
# successful	9	3	8	11

Table 1: Symbolic MC Algorithms

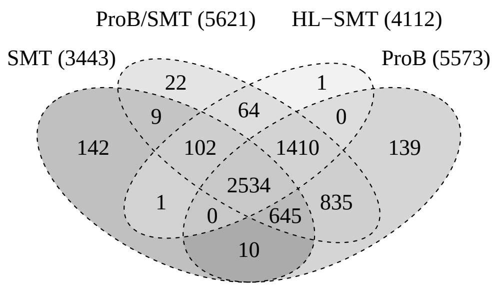

Figure 1: SMT Benchmarks

Initially, we focussed on symbolic model checking of safety properties and implemented three well-known algorithms: bounded model checking, k-induction and IC3. We applied them to a selection of models, but realized that occurring constraints were too complex for our solver.

From here on research split into two directions. On the one hand, algorithms had to be tuned towards B and Event-B. This will be explored in Section 3. On the other hand, constraint solving had to be improved to handle more complex constraints. See Section 4 for details.

3 Symbolic Model Checking Algorithms for B

As stated above, our first experiences with applying symbolic model checking algorithms to some benchmarks was disillusioning. Only a handful of models could be checked, with even slightly larger ones resulting in timeouts due to highly complex constraints.

We overcame the problem as outlined in [3]: When using the B method, one has to discharge different proof obligations used to verify the correctness of the model at hand. One such obligation is the *invariant preservation*. If proven, we know that some transition t can not lead to the violation of i_n, a conjunct of the system's invariant. To some degree, these proofs are performed automatically. We took the status of proof obligations and used it to ease solving the constraints occurring during symbolic model checking. Predicates known to hold are immediately asserted and given to the solver. Afterwards, we can the limit counter-example search to invariants that have not been proven.

Table 1 shows the results. As you can see, a number of models chosen to cover different characteristics could be checked by the extended algorithms. Among those, some could not be checked by ProB's explicit state model checker. Especially k-induction and IC3 seem promising when compared to explicit state model checking. Therefore, we now mainly focus on them: At the moment, we are trying to lift the abstraction technique outlined in [1] to B and Event-B.

4 Improving the Constraint Solver

The constraint solver of ProB was strong enough for animation and explicit state model checking but unable to cope with the constraints occurring in symbolic model checking. Thus, different ways of solving constraints have been explored:

- A translation from B to Kodkod and eventually to SAT was developed [7].
- We tried to strengthen our internal constraint solver. However, we soon realized that adding high-level reasoning to CLP(FD)-style constraint programming is cumbersome.
- Being unsatisfied by the former approaches, we looked into integrating ProB and Z3.

The last item was in fact the topic of our article submitted to iFM'16 [4]: We translate B and Event-B into a high-level SMT-LIB representation using Z3's set theory. This is particularly challenging, since B features many high-level constructs not available in Z3, such as set comprehensions and recursively defined operators.

Z3 can then be used alone (HL-SMT) or simultaneously with ProB (ProB/SMT), yielding a novel combination of DPLL(T) and CLP(FD). We compared to a low-level translation representing set-theory using predicate logic (SMT) and ProB alone. Figure 1 shows the results of releasing the four approaches onto different proof obligations. As can be seen, no approach is strictly superior. By enabling the Z3 integration, 197 additional obligations are discharged. However, 149 of them can not be discharged anymore.

5 Conclusions

In conclusion, symbolic model checking of B and Event-B specifications proves more challenging then expected. Existing algorithms are often too weak and have to be extended. This should be done keeping background knowledge in mind by considering characteristics and peculiarities of our input languages. However, this is not necessarily a bad thing. While B's high-level nature impedes the performance of algorithms, features like proof support come to our rescue.

Regarding constraint solving we can not solely rely on ourselves anymore. Instead, we need to reap what others sowed by integrating different solvers and technologies. Hopefully, making a combined effort will help overcome our current limitations.

References

[1] J. Birgmeier, A. R. Bradley, and G. Weissenbacher. Counterexample to induction-guided abstraction-refinement (CTIGAR). In *Proceedings CAV 2014*, volume 8559 of *LNCS*, pages 831–848. Springer, 2014.

[2] I. Dobrikov and M. Leuschel. Optimising the ProB model checker for B using partial order reduction. *Formal Aspects of Computing*, pages 1–29, 2016.

[3] S. Krings and M. Leuschel. Proof assisted symbolic model checking for B and Event-B. In *Proceedings ABZ 2016*, volume 9675 of *LNCS*. Springer, 2016.

[4] S. Krings and M. Leuschel. SMT solvers for validation of B and Event-B models. In *Proceedings iFM 2016*, volume 9681 of *LNCS*. Springer, 2016.

[5] M. Leuschel and M. Butler. ProB: A model checker for B. In *Proceedings FME 2003*, volume 2805 of *LNCS*, pages 855–874. Springer, 2003.

[6] M. Leuschel and M. Butler. ProB: an automated analysis toolset for the B method. *International Journal on Software Tools for Technology Transfer*, 10(2):185–203, 2008.

[7] D. Plagge and M. Leuschel. Validating B, Z and TLA+ using ProB and Kodkod. In *Proceedings FM 2012*, volume 7436 of *LNCS*, pages 372–386. Springer, 2012.

[8] C. Spermann and M. Leuschel. ProB gets Nauty: Effective symmetry reduction for B and Z models. In *Proceedings TASE 2008*, pages 15–22. IEEE, 2008.