Expose: Boolean Encoding of Statically Finite Sets in B Machines

Jan Roßbach

August 1, 2023

Contents

1 Introduction

The B-Method [1] is a formal methods that allows one to specify system requirements in very high level terms like first order logic can guarantee the correctness of the final implementation. This is especially useful in safety critical environments.

A problem these systems face regularly is that of state space explosion. Even for relatively small systems, the amount of states the program can be in can get too large for a computer to check in a reasonable amount of time [2]. One method trying to alleviate this problem is partial order reduction (POR) [2]. It attempts to reduce the state space by exploiting symmetries in the interleaving of operations. And while the method has proven to be very effective for low level formalisms, such results have yet to be achieved in B machines.

In this work we assume, that this is due to the high level nature of B and that the analysis is being slowed down by its high level data structures. We therefore hypothesize, that refining the high level expression in the machine will decrease computation time and time-out for POR in B machines. To test this hypothesis we aim to implement an algorithm for automatically translating high level relational expressions in B machines into a bitvector representation. For this an experimental Clojure interface for B machines, that builds on ProB [3], called lisb is used.

2 Related Work

Similar translations from first-order formula into quantifier-free boolean formula have been done by Daniel Jackson [4, 5] and implemented in model finding tools like Alloy [6] and Kodkod [7].

A translation from ProB to Kodkod has also been done [8]. These works use a relational user language and do the boolean translation behind the scenes in order to provide it to a SAT solver. The exact nature of the translation is hidden from the user and can not be used to do further calculations like feeding it to POR.

In this work, we aim to do the translation on the backend and then provide a rewritten form of the machine to the user. This sort of translation can be considered a form of automatic data refinement [4] which has been implemented before by tools like *BART* [9]. Another difference to previous translations is, that normal boolean representations only have to be compatible with other boolean variables and logic predicates.

In our translation we are still on the level of a B machine, which supports high level constructs like relations and functions. We will address this point later in section 3.

3 Problem Statement

In this work we will introduce a way to translate high level relational statements in B machines into low level boolean statements and a corresponding implementation of the algorithm. Then we try the algorithm on a number of B machines and test if this translation has measurable benefits for POR in ProB. This work should at least

- Provide a translation algorithm for enumerated sets up to a maximum size and deferred sets of a provided size, to boolean expressions.

- Provide a translation that unrolls single operations, that depend on rewritten variables, into many different operations that can easily be checked for independence.
- Provide a working implementation for the algorithm in lisb

Optionally, this can be extended with the following points.

- Experimentation on selected examples
- Support for functions and relations

4 Example Translation

Consider as an example this for brevity simplified B machine representing a CPU scheduler. The target for the translation is the enumerated set PID. We aim to translate all variables whose Type is based on PID, here active, ready and waiting. For each we will create new variables of type boolean, which will serve as the bitvector representation for the machine. We first analyze the code for variable definitions that depend on our target set. Here we find 3 such variables with the type POW(PID). Representatively, consider the variable active. Since it is defined as a subset, of PID, the variable has $2^3 = 8$ different possible states, meaning we need three boolean variables to represent these states. For example, the state active={PID1, PID3} would be encoded as activePID1=TRUE, activePID2=FALSE, activePID3=TRUE. We can then rewrite the invariant and initialization in terms of this boolean encoding.

We also need to deal with operations. For this we consider the inputs to each operation. If there is only one input and it comes from the target set, we can generate an equivalent operations, that deal with each one of the possible target elements individually. In our case we need to expand the existing operation into three new and different operations, that each deal with one of the possible inputs.

These new operations each read and write different sets of variables and are therefore automatically independent for the purpose of POR.

5 Structure of the Thesis

1. Introduction
2. Related Work
3. Translation
4. Implementation
5. Experiments
6. Conclusion
7. References

6 Timetable

The project is a Bachelor's Thesis and has a time limit of twelve weeks. There will be bi-weekly check ins on progress. Here we give a rough estimate of what should be done by what point.

- 1-2. week: research on translation / finish section's 1 & 2
- 3-4. week: implement parts of the translation & finish section 3 & 4
- 5-6. week: complete first draft & finish implementation
- 7-8. week: second draft & experiments on selected examples
- 9-10. week: reflect on results & finishing the thesis
- 11-12. week: refinement/buffer

References

[1] Jean-Raymond Abrial. *The B-Book*. Cambridge University Press, 2005.

[2] Antti Valmari. *The State Explosion Problem*, pages 429–528. Springer Berlin Heidelberg, 1998.

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

[4] Daniel Jackson. Automating first-order relational logic. *ACM SIGSOFT Software Engineering Notes*, 25(6):130–139, 2000.

[5] Daniel Jackson. An intermediate design language and its analysis. *SIGSOFT Softw. Eng. Notes*, 23(6):121–130, November 1998.

[6] Daniel Jackson. Alloy: A lightweight object modelling notation. *ACM Trans. Softw. Eng. Methodol.*, 11(2):256–290, apr 2002.

[7] Emina Torlak and Daniel Jackson. Kodkod: A relational model finder. In *Proceedings TACAS*, volume 4424 of *LNCS*, pages 632–647. Springer, 2007.

[8] Dominik Hansen and Michael Leuschel. Translating TLA+ to B for validation with ProB. In *Proceedings IFM*, volume 7321 of *LNCS*, pages 24–38. Springer Berlin Heidelberg, 2012.

[9] Antoine Requet. BART: A tool for automatic refinement. In Egon Börger, Michael Butler, Jonathan P. Bowen, and Paul Boca, editors, *Abstract State Machines, B and Z*, pages 345–345. Springer Berlin Heidelberg, 2008.