Symmetry Reduction in the ProB Model Checker

Edd Turner and Michael Butler

Department of Electronics and Computer Science
University of Southampton
Highfield, Southampton, SO17 1BJ, UK
{ent03r,mjb}@ecs.soton.ac.uk

Abstract. Model checking suffers from the state space explosion problem. One method to alleviate this problem is to exploit symmetries in the system, such that duplicate symmetric components of the state space are not explored – saving time during the checking process. This paper identifies symmetries in typical structures of the formal language of B, including relations, powersets and elements of sets, and presents a method for finding them through the modification of the well known graph isomorphism program, NAUTY. This work has been implemented in the ProB model checker and preliminary experiments indicate the idea holds much potential for improving the performance of model checking for B.

1 Introduction

Model checking [1] uses an *automatic* procedure that searches every state in which a formal model of a system can be (its state space), and verifies whether some specification holds for each; any violations are reported to the user. Since systems may contain an infinite number of states, bounds are placed on the parameters of the system to make this number finite, and guarantee algorithmic termination. It is made difficult by the *state space explosion problem*, where the addition of small components to a system result in a combinatorial growth in the number of states, which the model checker must still search; thus, verification of large systems becomes intractable. Various methods aim to alleviate this problem, such as partial order reduction [2] and abstraction [3]. The subject of this paper concerns the technique of *symmetry reduction*, where one exploits symmetric components of a system, which are indistinguishable in terms of the global behaviour.

Moreover, this paper describes an approach for applying symmetry reduction to model checkers of systems written in B [4], whose findings have been implemented into the B model checker, ProB [5]. The B-Method is a theory and methodology used for the formal specification and development of computer software systems. It includes a concise language based on set theory, called B, and is used by industries in a range of critical domains, notably in railway control.

2 Approach to Symmetry Reduction

Intuitively, two states are symmetric if their state variables and values are semantically the same; a precise definition is given in definition 1.

Definition 1. *Given a system with specification ϕ and two states in its state space α and β, then α is symmetric to β when ϕ holds for α iff ϕ holds for β.*

In B, ϕ in definition 1 above, is an invariant that should be satisfied by all reachable states of a (finite state) system description, called a B *machine*.

The effect of symmetry reduction is to partition the state space into equivalence classes of symmetric states. A modified model checking algorithm that implements this partitioning could then search a smaller but equivalent quotient state space by ensuring that it reaches at least one *representative* state from each equivalence class. A major problem associated with this scheme is finding a computation that computes a representative, whose time saved by a constrained search is not outweighed by its computational inefficiency.

3 Symmetries in B Structures

Deferred sets are fundamental constructs in the B language. They contain abstract elements that have no specific meaning and cannot be individually referred to[1] i.e., the elements are symmetric. Therefore, such information can be used to find symmetries between other common B constructs. This work chooses to exploit the relation e.g., partial/total functions, providing their domain/codomain uses elements of deferred sets.

Current Progress and Goal: Finding symmetric relations is equivalent to finding isomorphic graphs, for which the most efficient general purpose program is NAUTY [6]. Given two *undirected* and *unlabelled* graphs, it computes a canonical label for each, where the labels match if and only if the graphs are isomorphic. The underlying algorithm used in this computation has been extended and integrated into the ProB model checker, to enable direct computation of canonical labels for *directed* and *labelled* graphs. This means symmetries can be found between multiple variables that are relations on deferred sets. Support is achieved for non-symmetric elements of deferred sets by colouring graphs before applying the algorithm; such cases arise if the deferred set includes machine constants. The goal is then to maximise the range of systems this symmetry reduction exploits, by translating most (if not all) B constructs into relations – so any state can be modelled as a single directed, labelled graph, whose canonical label can be found. Currently, a translation is used for powersets of arbitrary depth and elements of sets. Symmetry is exploited by reducing the state space search so that only one representative of each equivalence class is explored.

Empirical Results and Conclusions: Results obtained using the methods described above, which have been implemented into ProB, produce encouraging savings in time and state space required for the verification of B-machines.

[1] unless the specification has constants that use them.

For example, BRel models a single relation over two deferred sets, and has one operation to add an edge to the relation. Figures 1 and 2 show the savings made in ProB when model checking BRel, with varying sizes of deferred sets. Note the logarithmic scales on the y-axes, indicating that the savings can be exponential. Furthermore, experimentation is yet to find a verification that takes longer when symmetry reduction is used.

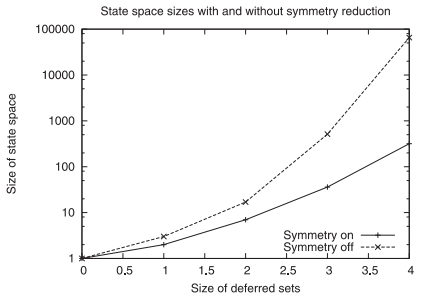

Fig. 1. State space size for BRel machine

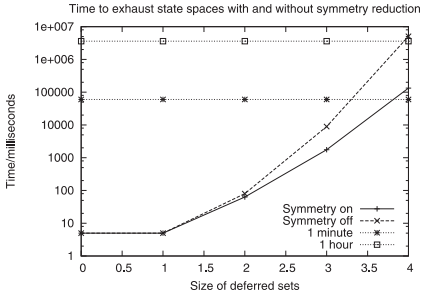

Fig. 2. Verification time for BRel machine

4 Future Work

Future work will increase the range of B constructs whose symmetries are exploited, through their translation to relations. Further improvements include programmatic optimisation of the canonical labelling algorithm.

An additional strategy is that of *symmetry breaking* [7], as used by the Alloy Analyser [8], whose reductions are often orders of magnitude. It prevents redundant symmetries being computed through the addition of predicates that constrain the search. Key ideas from [7, 8] appear applicable since computing the canonical label of a state generates automorphisms that could ensure symmetric 'next' states are never generated. Similar ideas may produce even greater reductions.

References

1. E.M. Clarke, O., Peled, D.: Model checking. The MIT Press (1999)
2. G.J. Holzmann, D.P.: Partial order reduction of the state space. In: First SPIN Workshop, Montrèal, Quebec (1995) position paper.
3. Clarke, E.M., Grumberg, O., Long, D.E.: Model checking and abstraction. ACM Trans. Program. Lang. Syst. **16**(5) (1994) 1512–1542
4. Abrial, J.R.: The B-Book: Assigning programs to meanings. Cambridge University Press, New York, NY, USA (1996)
5. Leuschel, M., Butler, M.J.: ProB: A model checker for B. In: FME. (2003) 855–874
6. McKay, B.: NAUTY user's guide (version 1.5), Technical report TR-CS-90-02. Australian National University, Computer Science Department, ANU. (1990)
7. Crawford, J.M., Ginsberg, M.L., Luks, E.M., Roy, A.: Symmetry-breaking predicates for search problems. In: KR. (1996) 148–159
8. Jackson, D.: Software Abstractions: Logic, Language, and Analysis. MIT Press (2006)