Symmetry Reduction Methods for Model Checking

Alice Miller and Alastair Donaldson
*Department of Computing Science
University of Glasgow
alice,ally@dcs.gla.ac.uk

1 Introduction

Model checking is a widely used automatic method for system verification. Verification of even moderately large systems can be difficult due to an exponential growth in the number of states with the number of components. This phenomenon is known as *state space explosion*. Many systems, however, consist of clusters of sets of identical (up to process id) components, resulting in symmetry of the underlying state space. If this symmetry can be detected it can often be exploited to reduce the cost of model checking sufficiently to allow for full verification.

We describe what is meant for a model checker to be explicit state or symbolic, timed or probabilistic, and illustrate these terms via an overview of four of the most widely used model checkers: SPIN, SMV, PRISM and UPPAAL.

We then show, via a simple example, how symmetry reduction can be helpful to reduce the cost of model checking in the explicit state case, and describe two problems associated with symmetry reduced model checking, namely symmetry detection and the orbit problem. We show how we have addressed these problems with our own symmetry detection and reduction tools for SPIN: SymmExtractor and TopSPIN, and discuss the problems associated with symmetry reduction methods in the symbolic case.

2 Popular Model Checkers

Errors in software development often appear at the design stage, yet are not detected until the final testing stage. The later errors are found, the more expensive they are to correct. Model checking allows us to find errors very early on by building small logical models of a system which can be automatically checked. Model checking is most commonly used to verify finite state concurrent systems, like those associated with intricate communications protocols and sequential circuits.

All model checkers require a specification of a system, together with the requirements (properties to be checked). The model checker then creates a graph-like structure (usually a Kripke structure or Markov chain) which is checked for property correctness. If the property under consideration does not hold, a counter-example is provided. *Explicit state model checkers* typically use automata theoretic techniques to search the state-space (a Kripke structure combining the states of the original model with those of the property to be checked). In symbolic model checking the underlying model is expressed as a boolean formula which is stored as a reduced, ordered binary decision diagram (an ROBDD).

There are many commonly used model checkers (see (8) for references). The choice of model checker depends on the type of system behaviour to be analysed. SPIN and SMV are the most widely used explicit state and symbolic model checkers respectively which can be used to model behaviour of systems for which interleaving properties rather than timing aspects are of interest. If one is interested in modelling timing aspects, the real time symbolic model checker UPPAAL (in which models are expressed as timed automata) can be used. In order to capture the behaviour of complex, probabilistic protocols the symbolic model checker PRISM can be used to check quantitative properties, such as latency or QoS attributes.

3 Symmetry Reduction for Model checking

Symmetry reduction is a technique for tackling state space explosion. Consider the Kripke structure for a 2-component mutual exclusion protocol illustrated in Figure 1 (where N,T and C denote *non-critical*, *trying* and *critical* and state (N,T) (say) is the state in which components 1 and 2 are in the non-critical and trying states respectively). The Kripke structure clearly exhibits symmetry between the components. For example states (N, T) and (T, N) are identical, from a graphical point of view (we could impose one onto the other by reflecting about a line running vertically through the middle of the structure). Symmetry reduction in model checking involves replacing sets of symmetrically equivalent states in a model $\mathcal{M}$ by a single representative, $rep(s)$, from each equivalence class (orbit). The resulting Kripke structure $\mathcal{M}'$ is called a *quotient structure*. Providing a property ϕ is invariant under the symmetry used, $\phi \models \mathcal{M}$ if and only if $\phi \models \mathcal{M}'$.

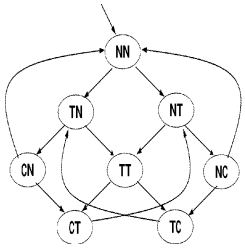

Figure 1: Kripke structure for 2-component mutual exclusion

Structures are reduced with respect to a group of symmetries G (specifically a group of automorphisms of the Kripke structure). In the n-component mutual exclusion protocol, G is the group consisting of all permutations of the set of ids $\{1, 2, \ldots, n\}$. In this case we say that our system is *fully symmetric*.

The idea is to not build the original model, but to build the quotient structure *on-the-fly* using knowledge of the group G. In explicit state model checking, instead of backtracking only when a state s is reached that has been reached previously, we now backtrack if $rep(s)$ has been reached previously. Two of the most difficult problems associated with symmetry reduction are therefore to statically identify G, and to evaluate $rep(s)$ for any state s. In most cases the lexicographically smallest element of the orbit containing s is chosen as $rep(s)$. Calculating $rep(s)$ in this way is known as the *constructive orbit problem*, and is NP-hard (2).

We have developed tools SymmExtractor (4) and TopSPIN (5) to allow us to perform automatic symmetry detection and reduction respectively. SymmExtractor extracts a graphical representation known as the *static channel diagram* $SCD(P)$ from a Promela specification P and uses the graph automorphism program *saucy* (3) to extract the automorphism group of $SCD(P)$. We then obtain G from this group.TopSPIN uses group theoretic results to solve the orbit problem efficiently for certain classes of group (for example, when the group can be decomposed into a product of smaller groups – a likely scenario when model checking systems consisting of clusters of sets of identical components).

Other existing implementations of symmetry reduction methods for explicit state model checking often rely on the existence of full symmetry identified by the user. A specific data type (scalarset) can be used to label states that can be permuted. Symmetry reduction is harder in the symbolic case as the BDD required to store the orbit relation is prohibitively large. As a consequence, symmetry reduction for symbolic probabilistic model checking is similarly thwarted – although we have had some success implementing symmetry reduction for PRISM (6) by extending an approcah based on *generic representatives* (7). In current work we are investigating the application of symmetry reduction methods to explicit state probabilistic model checkers, such as LiQuor (1).

References

[1] F. Ciesinski and C. Baier. LiQuor: A tool for qualitative and quantitative linear time analysis of reactive systems. In *Proceedings QEST'06*, IEEE Computer Society, pages 131–132, 2006.

[2] E. Clarke, E. Emerson, S. Jha, and A. Sistla. Symmetry reductions in model-checking. In *Proceedings of CAV '98*, vol 1427 of *Lecture Notes in Computer Science*, pages 147–158, 1998.

[3] P. Darga, M. Liffiton, K. Sakallah and I. Markov. Exploiting Structure in Symmetry Detection for CNF. In *Proceedings of DAC'03*, ACM, pages 530–534, 2004.

[4] A.F. Donaldson and A. Miller. Automatic symmetry detection for model checking using computational group theory. In *Proceedings of FM2005*, vol 3582 of *Lecture Notes in Computer Science*, pages 481–496, 2005.

[5] A.F. Donaldson and A. Miller. Exact and approximate strategies for symmetry reduction in model checking. In *Proceedings FM 2006*, vol 4085 of *Lecture Notes in Computer Science*, pages 541–556, 2006.

[6] A.F. Donaldson and A. Miller. Symmetry reduction for probabilistic model checking using generic representatives. In *Proceedings ATVA'06*, vol 4218 of *Lecture Notes in Computer Science*, pages 9–23, 2006.

[7] E. Emerson and T. Wahl. On combining symmetry reduction and symbolic representation for efficient model checking. In *Proceedings CHARME'03)*, volume 2860 of *Lecture Notes in Computer Science*, pages 216–230, 2003.

[8] A. Miller, A. Donaldson, and M. Calder. Symmetry in temporal logic model checking. *Computing Surveys*, 36(3), September 2006.