Symmetry Reduction of Partially Symmetric Systems

Christopher Power
Department of Computing Science
University Of Glasgow
power@dcs.gla.ac.uk

Alice Miller
Department of Computing Science
University Of Glasgow
alice@dcs.gla.ac.uk

Abstract

Previous research into symmetry reduction techniques have shown them to be successful in combatting the state-space explosion problem. We provide a brief overview of a fully automated technique for the application of symmetry reduction to partially symmetric systems.

1 Introduction

As software becomes more complex the need for development techniques capable of uncovering errors at design time is critical. A model checker [2] accepts two inputs: a specification $\mathcal{P}$, described in a high level formalism and a set of testable properties, ϕ. A model checker generates and exhaustively searches a finite state model $\mathcal{M}(\mathcal{P})$ to confirm if a property holds, or conversely, report a violation of the system specification. The intuition being, bugs found in the model of the system will reveal bugs in the system design. However, the application of model checking is limited as the state-space of even moderately sized concurrent systems can be too large for state-of-the art machines to exhaustively search. Although verification algorithms have a linear run time complexity, this is offset as the number of states in a model grows exponentially as parameters are added. Consequently, research often focuses on techniques to reduce the impact of the state-space explosion.

One such technique is symmetry reduction [1]. If a concurrent system is comprised of many replicated processes then checking a model of the system may involve redundant search over equivalent, or symmetric, areas of the state-space. Symmetry reduction is concerned with exploiting these underlying regularities by only storing one representative of a structure. For highly symmetric systems, this can result in a reduction factor exponential to the number of system components. However, standard symmetry reduction techniques often discount many symmetries. These symmetries are discounted due to a process having a transition that distinguishes it from other processes while all other behaviour is shared [4].

1.1 Partial Symmetry Reduction

A system can be considered partially symmetric if the majority of updates can be performed by most processes in a specific context. Therefore, a large degree of similarity can be observed between the processes in a system. In an attempt to exploit these symmetries it is better to represent all similar behaviour between processes as a single program and annotate the exceptions.

An annotation is a partition of the set of process indices [6]. A state is marked with an annotation if it lies on a path containing a transition that distinguishes two components. The annotation places the violating process indices into separate partitions. Upon further exploration of the path only processes in the same partition may be permuted. This can lead to the situation where a state is reached along two paths: one with many asymmetric transitions and the other containing only symmetric transitions. In order to make the largest reduction in the state space, it is assumed the state was reached through the symmetric path. The asymmetric state is said to have been subsumed by the symmetric one. This leads to a reduction technique only suitable for reachability analysis as it creates a quotient structure that is not bisimulation-equivalent to the original.

1.2 Detecting Partial Symmetries

A system comprised of n processes can be abstractly represented as a state transition diagram where local states are nodes and local transitions edges. These transitions have the form

$$A \xrightarrow{\phi, \mathcal{Q}} B$$

The transition predicate ϕ takes two variables as input: a state s defining the context in which the edge is to be executed and a process id i. The predicate $\phi(s,i)$ returns the value true if in state s, process i is allowed to make a transition from local state A to local state B. Predicate ϕ can be written in any logical formalism that allows for basic arithmetic operation to be conducted on state and index variables. $\mathcal{Q}$ defines the partitioned set of permutations that preserve predicate ϕ. It is critical for an adaptive symmetry reduction algorithm dealing with a partially symmetric system to be capable of finding $\mathcal{Q}$, thus enabling states to be annotated.

Current approaches to the reduction of partially symmetric systems suggest symmetries be specified by the user, manually or with the aid of a proposition solver [6]. This hinders reduction by adding the overhead of repetitive computation and requires the user to be an expert in the technique. For symmetry reduction to be viable it must be possible to calculate the state annotations without explicitly building the state space or placing the onus on the user. Therefore, we propose a technique capable of calculating $\mathcal{Q}$ directly from the source text of a program.

It has been establish [1] that there is a correspondence between symmetries in a model's underlying communication structure and those in the subsequently generated Kripke structure. To determine the symmetry present in a system, a structure called a static channel diagram [3], a graphical representation of potential communication within the system, can be generated directly from a model specification. Typically the static channel diagram of a system is a small graph and symmetries of this graph can be efficiently found.

Symmetry of the static channel diagram induces symmetry in the model corresponding to the system, provided certain conditions are satisfied. These are generally conditions on assignments to process ID sensitive variables and can be efficiently checked. The set of strictly valid symmetries can therefore be calculated and tagged to each individual update in the high level system description [3]. These tags represent the values of i for which predicate $\phi(s,i)$ returns true. If the update is enabled in the context of state s

This information can be used by an adaptive symmetry reduction algorithm [6] to explore a partially symmetric state space. From an initial state s the algorithm attempts to generate and explore the reachable part of the state space. Successor states of s are found by iterating through all edges valid in the context of state s. Due to the tags, information regarding the solution to $\phi(s,i)$ is available. However, as we are dealing with a partially symmetric system, two slightly differing processes may have the same available update. The update common to these processes will be tagged with a different set of *personal symmetries*. Therefore, at each state, tags can be concatenated for identical edge updates regardless of the process that initiates them. This allows for the largest possible value of $\mathcal{Q}$ to be returned and the coarsest annotation appended to the state. The algorithm continues by reconciling any symmetry violations so only states not subsumed by others are explored.

2 Conclusion

The use of an adaptive symmetry reduction algorithm allows for potentially larger reductions in the state space of a partially symmetric system to be achieved. Encouraging experimental results for this technique have been shown using the SMC model checker [5]. However, as with all current approaches, information regarding partial symmetries is assumed to be known prior to reduction. The proposed technique would alleviate this issue by enabling annotations to be calculated directly from the source text. This in turn allows the application of an adaptive symmetry reduction algorithm to be fully automated.

References

[1] E.M. Clarke, E.A. Emerson, S. Jha, and A.P. Sistla. Symmetry reductions in model checking. In *Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 147–158. Springer, 1998.

[2] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model checking*. Springer, 1999.

[3] A.F. Donaldson, A. Miller, and M. Calder. Finding symmetry in models of concurrent systems by static channel diagram analysis. *Electronic Notes in Theoretical Computer Science*, 128(6):161–177, 2005.

[4] E.A. Emerson and R.J. Trefler. From asymmetry to full symmetry: New techniques for symmetry reduction in model checking. In *Conference on Correct Hardware Design and Verification Methods*, pages 142–156. Springer, 1999.

[5] A.P. Sistla, V. Gyuris, and E.A. Emerson. SMC: a symmetry-based model checker for verification of safety and liveness properties. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 9(2):133–166, 2000.

[6] T. Wahl. Adaptive symmetry reduction. In *Computer Aided Verification*, volume 4590 of *Lecture Notes in Computer Science*, page 393. Springer, 2007.