A Computational Group Theoretic Symmetry Reduction Package for the SPIN Model Checker

Alastair F. Donaldson[⋆] and Alice Miller[⋆⋆]

Department of Computing Science
University of Glasgow, 17 Lilybank Gardens,
Glasgow, Scotland, G12 8QQ.
{ally,alice}@dcs.gla.ac.uk

Abstract. Symmetry reduced model checking is hindered by two problems: how to identify state space symmetry when systems are not fully symmetric, and how to determine equivalence of states during search. We present TopSPIN, a fully automatic symmetry reduction package for the SPIN model checker. TopSPIN uses the GAP computational algebra system to effectively detect state space symmetry from the associated Promela specification, and to choose an efficient symmetry reduction strategy by classifying automorphism groups as a disjoint/wreath product of subgroups. We present encouraging experimental results for a variety of Promela examples.

1 Introduction

Model checking concurrent systems comprised of replicated components can potentially be made easier by exploiting symmetries of a model of the system, induced by the replication. If such *component symmetries* can be identified before search then the model checking algorithm can be modified to consider a single state from each equivalence class of symmetric states. This results in reduced space requirements for verification by model checking.

However, symmetry reduction can only speed up model checking if an efficient procedure is available to determine whether or not a given state is equivalent to a previously reached state. A common approach to solving this problem for explicit state model checking is, given a total ordering on states and a symmetry group G, to convert a state s to $min[s]_G$—the *smallest* state in the equivalence class of s under G—before it is stored. Thus efficient algorithms are required to compute $min[s]_G$. This is the *constructive orbit problem*, which has been proved to be NP-hard [4]. Current implementations of symmetry reduction techniques for explicit state model checking, such as SymmSpin [2], are limited to dealing with full symmetry between components of a concurrent system—both symmetry detection and on-the-fly representative computation are easy for this special case.

⋆ Supported by the Carnegie Trust for the Universities of Scotland.
⋆⋆ Partially funded by The Universitiy of Glasgow John Robertson Bequest, award number JR05/14.

In previous work we proposed a framework for the automatic detection of arbitrary structural symmetry, with an implementation for the Promela specification language [6]. In this paper we present TopSpin, a symmetry reduction package for the Spin model checker which uses exact and approximate strategies for dealing with such arbitrary symmetries. The tool draws on theory and technology from computational group theory to efficiently compute equivalence class representatives. In particular, the Gap computational algebra system [10] is used both for symmetry detection, and for classifying an arbitrary group based on its structure as a direct/wreath product of basic subgroups, so that an appropriate symmetry reduction strategy may be chosen. For groups which cannot be classified in this way, TopSpin uses an approximate symmetry reduction strategy based on hillclimbing local search, which is sub-optimal in terms of memory requirements but fast and safe. We present experimental results which demonstrate the effectiveness of our techniques. TopSpin, together with Promela code for the specifications described in Sect. 5, can be found on our website [7]. Throughout the paper, we assume some basic knowledge of group theory.

2 Background and Notation

Spin [11] is the bespoke model checker for the Promela specification language, and provides several reasoning mechanisms: assertion checking, acceptance and progress states and cycle detection, and satisfaction of temporal properties, expressed in linear temporal logic (LTL). Spin translates each component defined in a Promela specification into a finite automaton and then computes the asynchronous interleaving product of these automata to obtain the global behavious of the concurrent system. This interleaving product is essentially a Kripke structure $\mathcal{M} = (S, s_o, R, L)$, where S is a finite set of states with initial state s_0, $R \subseteq S \times S$ a *total* transition relation, and $L : S \to 2^{AP}$ a labelling function. The set AP of atomic propositions refer to the values of local and global variables, and contents of buffered channels.

A bijection $\alpha : S \to S$ which satisfies, for all $(s,t) \in R$, $(\alpha(s), \alpha(t)) \in R$, is an *automorphism* or *symmetry* of $\mathcal{M}$, and all such symmetries form a group $Aut(\mathcal{M})$ under composition of mappings. If a subgroup G of $Aut(\mathcal{M})$ is known in advance then model checking can be performed over a *quotient* Kripke structure, $\mathcal{M}_G$, typically smaller than the original [12]. Kripke structure automorphisms induced by symmetry between components of the concurrent system, i.e. bijections of the component index set which give rise to automorphisms when lifted to act component-wise on states, are called *component symmetries* [9]. In this work we restrict our attention to component symmetries. If $G \leq Aut(\mathcal{M})$ and $s \in S$, then $[s]_G = \{\alpha(s) : \alpha \in G\}$ is the *orbit* of s under G.

3 An Overview of TopSpin

In order to check properties of a Promela specification, Spin first converts the specification into a C source file, `pan.c`, which is then compiled into an executable verifier. The state space thus generated is then searched. If the property being checked is proved to be false, a counterexample is given. TopSpin follows the approach used by the SymmSpin symmetry reduction package [2], where `pan.c` is generated as usual by Spin, and then converted to a new file, `sympan.c`, which includes algorithms for symmetry reduction. With TopSpin because we allow for arbitrary system topologies, symmetry must be detected before `sympan.c` can be generated. This is illustrated in Fig. 1.

First, the *static channel diagram* (SCD) of the Promela specification is extracted by the SymmExtractor tool [6]. The SCD is a graphical representation of potential communication between components of the specification. The group of symmetries of the SCD, $Aut(SCD)$, is computed using the *saucy* tool [5], which we have extended to handle directed graphs. The generators of $Aut(SCD)$ are checked against the Promela specification for validity (an assurance that they induce symmetries of the underlying state space). TopSpin uses Gap to compute, from the set of valid generators, the largest group $G \leq Aut(SCD)$ which can be safely used for symmetry-reduced model checking. Gap is then used to classify the structure of G in order to choose an efficient symmetry reduction strategy. The chosen strategy is merged with `pan.c` to form `sympan.c`, which can be compiled and executed as usual.

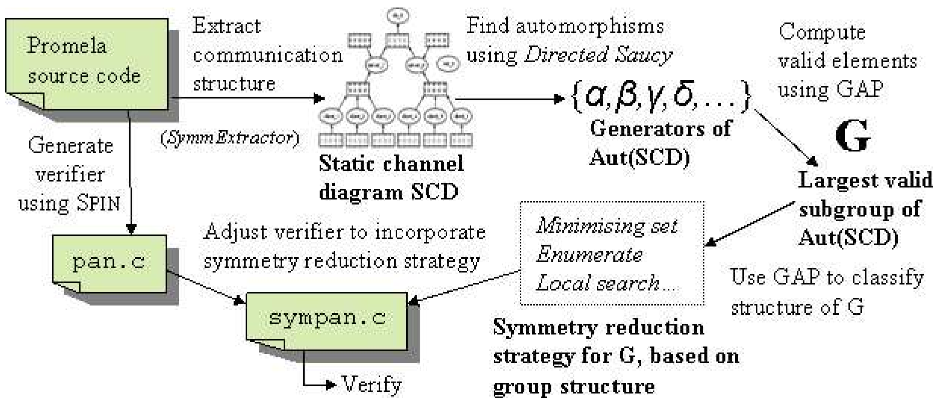
Promela source code
Extract communication structure
(SymmExtractor)
Static channel diagram SCD
Find automorphisms using Directed Saucy
{α, β, γ, δ, ...}
Generators of Aut(SCD)
Compute valid elements using GAP
G
Largest valid subgroup of Aut(SCD)
Use GAP to classify structure of G
Minimising set
Enumerate
Local search...
Symmetry reduction strategy for G, based on group structure
Generate verifier using SPIN
pan.c
Adjust verifier to incorporate symmetry reduction strategy
sympan.c
Verify

Fig. 1. The symmetry reduction process.

4 Symmetry Reduction Strategies

We refer to processes and channels of a Promela specification as *components*, and restrict our attention to Promela specifications with a fixed number of components. Throughout, we assume that $G \leq S_n$ is a nontrivial symmetry group for a Promela specification consisting of n components. In this section we outline various strategies which TopSpin uses to compute $min[s]_G$ for a state s and an *arbitrary* group G. An appropriate strategy for G is chosen based on analysis of the structure of G before search. Note that, during verification, the C function `memcmp` provides a total ordering on states.

4.1 The Strategies

Enumeration If G is a relatively small group ($|G| < 100$ say) then for a state s, $min[s]_G$ can be computed by *enumerating* the elements of G, and returning $min\{\alpha(s) : \alpha \in G\}$. TopSpin implements this approach with two optimisations, applied simultaneously. As the operation of applying a transposition to a state is less expensive than that of applying an arbitrary permutation, a group element α is expressed as a product of transpositions and $\alpha(s)$ is computed by applying these transpositions to s in order. TopSpin uses a *stabiliser chain* to enumerate the elements of G. Given a stabiliser chain $G = G^{(1)} \geq G^{(2)} \geq \cdots \geq G^{(k)} = \{id\}$ for some $k > 1$, every element of G can be uniquely expressed as a product $u_{k-1}u_{k-2} \dots u_1$, where, for $1 \leq i < k$, u_i is a representative of a coset of $G^{(i+1)}$ in $G^{(i)}$ [3]. Thus each $\alpha \in G$ need not be applied to a state s from scratch: partial images of s under the coset representatives may be re-used.
Minimising Sets Using terminology from [9], a group H is said to be *nice* if there is a small set $X \subseteq H$ such that $t = min[s]_H$ iff $\alpha(t) \geq t \; \forall \alpha \in X$. If H is nice with respect to a subset X then we call X a *minimising set* for H. Given a minimising set X for G, the element $min[s]_G$ can be computed by setting $t = s$, and applying elements of X to t until a fixpoint is reached. TopSpin uses this symmetry reduction strategy in cases where G is isomorphic to a fully symmetric group S_m, for some $m \leq n$, which simultaneously permutes several disjoint subsets of $\{1, 2, \dots, n\}$. (Such groups occur commonly in practice, e.g. a set of processes may have associated channels, so that any permutation of the processes must also permute the associated channels.) In this case, let $\alpha_{i,j}$ denote the permutation which simultaneously transposes the ith and jth elements of each subset. If G is generated by the set $X = \{\sigma_{j,k} : 1 \leq j < k \leq m\}$ then it can be shown that X is a minimising set for G, and $|X|$ is quadratic in n even though G may be very large. If TopSpin detects that G is isomorphic to S_m for some $m \leq n$ then it attempts to construct a minimising set of the above form.
Disjoint Products If G is the disjoint product of subgroups $H_1, H_2, \dots, H_k$ for some $k > 1$ then $min[s]_G = min[\dots min[min[s]_{H_1}]_{H_2} \dots]_{H_k}$ [4]. TopSpin constructs an equivalence relation on the generators of G to detect whether G is a disjoint product. The approach is very efficient, but not complete—it does not guarantee detection of the finest decomposition of G as a disjoint product. However, we have found it to work well in practice.
Wreath Products If G is a wreath product $H \wr K$ of two subgroups H and K then G contains r copies of H for some $r \geq 1$, denoted $H_1, H_2, \dots, H_r$, which each permute elements within a distinct "block" of components of the specification, and K permutes the blocks. In this case, it can be shown that $min[s]_G = min[min[\dots min[min[s]_{H_1}]_{H_2} \dots]_{H_r}]_K$ [4]. If reduction strategies can be found for H and K, then analogous strategies to that for H can easily be

obtained for each H_i, and $min[s]_G$ can be computed by applying the strategy for each H_i, followed by the strategy for K. To efficiently detect a wreath product decomposition for G, TopSpin identifies candidate blocks by using Gap to compute non-trivial *block systems* for G. Corresponding groups H and K are derived for the candidate blocks, and a check is made to see whether or not G is the wreath product of these groups.

Local Search If none of the above strategies are applicable then, since enumeration is very expensive, it may be infeasible to compute $min[s]_G$. TopSpin implements an *approximate* symmetry reduction strategy based on hillclimbing local search using the group generators, which does not guarantee unique representatives, but is safe to use when model checking as it guarantees storage of at least one state per equivalence class. Though not as space-efficient as enumeration, this strategy can work considerably faster.

4.2 Choosing a Reduction Strategy

TopSpin uses a top-down recursive algorithm to choose a symmetry reduction strategy for an arbitrary group G with respect to a set of n components. If G is isomorphic to a cyclic group and $|G| \leq n$, or to a dihedral group and $|G| \leq 2n$, then the enumeration strategy is selected. If $|G|$ is isomorphic to the group S_m for some $m \leq n$ then TopSpin attempts to construct a minimising set for G of the form described above, so that the minimising set strategy can be chosen. If G can be shown to decompose as a product of subgroups then a composite strategy is obtained by choosing a strategy for each subgroup. Otherwise, the local search strategy is chosen. In order to compare strategies it is possible to select the strategy used (rather than let TopSpin choose the most efficient).

5 Experimental Results

Table 1 gives experimental results applying our techniques to three families of Promela specifications. For each specification, we give the number of model states without symmetry reduction (**orig**), with full symmetry reduction (**red**), and using the strategy chosen by TopSpin (**best**). If the latter two are equal, '=' appears for the TopSpin strategy. The use of state compression, provided by Spin, is indicated by the number of states in italics. For each strategy (**basic** for enumeration without the optimisations described in Sect. 4.1, **enum** for optimised enumeration, and **best** for the strategy chosen by TopSpin), and when symmetry reduction is not applied (**orig**), we give the time taken for verification (in seconds). Verification attempts which exceeded available resources, or did not terminate within 5 hours, are indicated by '-'. All experiments were performed on a PC with a 2.4GHz Intel Xeon processor, 3Gb of available main memory, running Spin version 4.2.3. The first family of specifications model flow of control in a three-tiered architecture consisting of a database, a layer of p servers, and a layer of pq clients, where q clients are connected to each server (a D-S-C system). Here models exhibit wreath product symmetry: there is full symmetry between

system	**config.**	**states orig**	**time orig**	$\|G\|$	**states red**	**time basic**	**time enum**	**states best**	**time best**
D-S-C	2/3	103105	5	72	2656	7	4	=	2
D-S-C	2/4	1.1×10^6	37	1152	5012	276	108	=	2
D-S-C	3/3	$\mathit{2.54{\times}10^7}$	4156	1296	50396	4228	1689	=	19
D-S-C	3/4	-	-	82944	130348	-	-	=	104
R-C	3,3	16768	0.2	36	1501	0.9	0.3	=	0.1
R-C	4,4	199018	2	576	3826	57	19	=	0.4
R-C	5,5	2.2×10^6	42	14400	8212	4358	1234	=	2
R-C	4,4,4	2.39×10^7	1587	13824	84377	-	12029	=	17
R-C	5,5,5	-	-	1728000	254091	-	-	=	115
HC	3d	13181	0.3	48	308	0.6	0.3	468	0.2
HC	4d	380537	18	384	1240	58	34	6986	13
HC	5d	$\mathit{9.6{\times}10^6}$	2965	3840	3907	7442	5241	90442	946

Table 1. Experimental results for various configurations of the three-tiered (D-S-C), resource allocator (R-C) and hypercube (HC) specifications

the q clients in each block, and the blocks of clients, with their associated servers, are interchangeable. A configuration with p servers and q clients per server is denoted p/q. The second family of specifications model a resource allocator process which controls access to a resource by a competing set of prioritised clients (an R-C system). Models of these specifications exhibit disjoint product symmetry: there is full symmetry between each set of clients with the same priority level. A configuration with p_i clients of priority level i is denoted $p_1, p_2, \ldots, p_k$, where k is the number of priority levels. Finally, we consider specifications modelling message routing in an n-dimensional hypercube network (an HC system). The symmetry group here is isomorphic to the group of geometrical symmetries of an n-dimensional hypercube, which cannot be decomposed as a disjoint or wreath product of subgroups, and thus must be handled using either the *enumeration* or *local search* strategies. An n-dimensional hypercube specification is denoted nd. For all specifications, we verify deadlock freedom, and check the satisfaction of basic safety properties expressed using assertions.

In all cases, the basic enumeration strategy is significantly slower than the optimised enumeration strategy, which is in turn slower than the strategies chosen by TopSpin. For hypercube configurations, TopSpin chooses the local search strategy, which requires storage of more states than the enumeration strategy, but still results in a greatly reduced state space.

6 Related and Future Work

The SymmSpin symmetry reduction package avoids the problem of automatic symmetry detection by requiring symmetries to be specified using *scalarsets*, an approach proposed in [12]. Scalarsets can only specify full symmetry between identical components, thus the three-tiered architecture and hypercube examples of Sect. 5 could not be handled by SymmSpin. Multiple scalarset types could be used to specify symmetry between clients with the same priority level in the resource allocator example, but the automatic approach to symmetry detection provided by TopSpin is clearly preferrable.

Automatic symmetry detection by static channel diagram analysis is similar to an approach for deriving symmetry in a shared variable model of comunication [4]. However, this approach is not directly applicable to the specification language of a mainstream model checker such as Spin. Certain classes of groups for which orbit representatives can be efficiently computed are also presented in [4]. We extend this work by providing techniques to automatically determine whether a group belongs to one of these classes.

Future work includes extending TopSpin to allow symmetry-reduced verification of LTL properties under weak fairness, as described in [1]. This will involve combining strategies for representative computation with the nested depth first search algorithm employed by Spin [11]. The notion of *virtual* symmetry is suggested in [8] to deal with systems which are "almost" symmetric. The symmetry detection techniques which TopSpin uses could potentially be extended to handle virtual symmetry, allowing state-space reductions for examples with less symmetry than those which we present.

References

1. D. Bosnacki. A light-weight algorithm for model checking with symmetry reduction and weak fairness. In *SPIN'03*, LNCS 2648, pages 89–103. Springer, 2003.
2. D. Bosnacki, D. Dams, and L. Holenderski. Symmetric spin. *International Journal on Software Tools for Technology Transfer*, 4(1):65–80, 2002.
3. G. Butler. *Fundamental Algorithms for Permutation Groups*, volume 559 of *LNCS*. Springer-Verlag, 1991.
4. E.M. Clarke, E.A. Emerson, S. Jha, and A.P. Sistla. Symmetry reductions in model checking. In *CAV'98*, LNCS 1427, pages 147–158. Springer, 1998.
5. P.T. Darga, M.H. Liffiton, K.A. Sakallah, and I.L. Markov. Exploiting structure in symmetry detection for CNF. In *DAC'04*, pages 530–534. ACM Press, 2004.
6. A. F. Donaldson and A. Miller. Automatic symmetry detection for model checking using computational group theory. In *FM'05*, LNCS 3582, pages 418–496. Springer, 2005.
7. A. F. Donaldson and A. Miller. TopSpin Website: `http://www.dcs.gla.ac.uk/people/personal/ally/topspin/`.
8. E.A. Emerson, J. Havlicek, and R.J. Trefler. Virtual symmetry reduction. In *LICS'00*, pages 121–131. IEEE Computer Society Press, 2000.
9. E.A. Emerson and T. Wahl. Dynamic symmetry reduction. In *TACAS'05*, LNCS 3440, pages 382–396. Springer, 2005.
10. The Gap Group. *GAP–Groups, Algorithms, and Programming, Version 4.4*; 2006. `http://www.gap-system.org`.
11. G.J. Holzmann *The SPIN model checker: primer and reference manual.* Addison Wesley, 2003.
12. C. Ip and D. Dill. Better verification through symmetry *Formal Methods in System Design*, 9:41–75, 1996.