Formal Methods in System Design 9, 77–104 (1996)

Exploiting Symmetry in Temporal Logic Model Checking*

E.M. CLARKE
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

R. ENDERS AND T. FILKORN
Siemens AG, Corporate Research and Development, Otto-Hahn-Ring 6, W-8000 Muenchen 83, Germany

S. JHA
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Received July 1993; Accepted January 1994

Abstract. In practice, finite state concurrent systems often exhibit considerable symmetry. We investigate techniques for reducing the complexity of temporal logic model checking in the presence of symmetry. In particular, we show that symmetry can frequently be used to reduce the size of the state space that must be explored during model checking. In the past, symmetry has been exploited in computing the set of reachable states of a system when the transition relation is represented explicitly [14, 11, 19]. However, this research did not consider arbitrary temporal properties or the complications that arise when BDDs are used in such procedures.

We have formalized what it means for a finite state system to be symmetric and described techniques for reducing such systems when the transition relation is given explicitly in terms of states or symbolically as a BDD. Moreover, we have identified an important class of temporal logic formulas that are preserved under this reduction. Our paper also investigates the complexity of various critical steps, like the computation of the orbit relation, which arise when symmetry is used in this type of verification. Finally, we have tested our ideas on a simple cache-coherency protocol based on the IEEE Futurebus + standard.

Keywords: model checking, symmetry, temporal-logic

1. Introduction

Finite state concurrent systems frequently exhibit considerable symmetry. It is possible to find symmetry in memories, caches, register files, bus protocols, network protocols—anything that has a lot of replicated structure. Generally, verification techniques do not take advantage of this fact. This work tries to exploit symmetry to reduce the size of the state space that must be explored by temporal logic model checking algorithms.

*This research was sponsored in part by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C-1465, ARPA Order No. 7597 and in part by the National Science Foundation under Grant No. CCR-8722633 and in part by the Semiconductor Research Corporation under Contract 92-DJ-294.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. government.

In *Temporal Logic Model Checking* we determine whether a temporal logic formula is valid in a finite state system $M = (S, R, L)$, where S is the state space, R is the state transition relation, and L is a function that labels states with sets of atomic propositions. Such a structure is usually called a *Kripke Model* and may have an enormous number of states. An efficient *Model Checking* procedure tries to reduce the number of states that are actually searched. In most cases the state space is expressed symbolically in terms of state variables, and the transition relation is represented by a *binary decision diagram* or BDD [2, 3].

Let G be a group of permutations acting on the state space S of the Kripke structure M. A permutation $\sigma \in G$ is said to be a *symmetry* of M if and only if it preserves the transition relation R. G is a *symmetry group* for the Kripke structure M if and only if every permutation $\sigma \in G$ is a *symmetry* for M. If s is an element of S, then the *orbit* of s is the set of states $\theta(s)$ obtained from s by applying permutations in G. From each orbit $\theta(s)$ we pick a representative that we call $\text{rep}(\theta(s))$.

If $M = (S, R, L)$ is a Kripke Structure and G a symmetry group acting on M, we can define a quotient model $M_G = (S_G, R_G, L_G)$ in the following manner:

- The state set is $S_G = \{\theta(s) \mid s \in S\}$, the set of orbits of the states in S;
- The transition relation R_G has the property that $(\theta(s_1), \theta(s_2)) \in R_G$ if an only if there exists two states s_3 and s_4 such that $s_3 \in \theta(s_1)$, $s_4 \in \theta(s_2)$, and $(s_3, s_4) \in R$;
- The labeling function L_G is given by $L_G(\theta(s)) = L(\text{rep}(\theta s)))$.

An atomic proposition is *invariant* under the action of a symmetry group G, if the set of states labeled by the proposition is closed under the application of the permutations in G. We prove that if h is a formula in the temporal logic CTL* and all of the atomic propositions in h are invariant under the symmetry group G, then h is true in M if and only if it is true in the quotient model M_G. This implies that we can determine the correctness of properties in the original model M by checking them in the quotient model M_G.

Since the quotient model M_G contains only one representative from each orbit, the state space S_G will, in general, be much smaller than the original state space S. We have developed techniques that build M_G without actually building M. We believe that our method will considerably reduce the state space that must be searched and we have tested our ideas on a simple cache coherency protocol based on the IEEE Futurebus + standard. Previous research on verification of cache coherence protocols has made the simplifying assumption that there is only one cache line in the system [6, 17]. This assumption is necessary because the BDDs that occur in verifying these protocols grow exponentially in the number of cache lines. By using symmetry, however, we are able to avoid this assumption and reason about systems with multiple cache lines. Since different cache lines behave almost independently, the ordering of the cache lines is relatively unimportant and this results in a small quotient model. We have obtained are encouraging results. The size of the BDDs that are needed to represent the model is, in some cases, reduced by an order of magnitude or more.

We also discuss the complexity of exploiting symmetry in model checking algorithms. The first problem that we consider is computing the *orbit relation*, i.e., determining whether two states are in the same orbit or not. This is an important problem because the direct method of computing the quotient model M_G uses this relation. We prove that this problem is at least as hard as the *graph isomorphism problem.* In addition, we show that a variant of this problem, called the *bounded orbit problem*, is *NP*-complete. We also give lower bounds on the size of the BDDs needed to encode the orbit relation. Because these bounds are exponential for some important symmetry groups that occur in practice, we develop a method of using multiple representatives from each orbit that does not require building the full orbit relation.

There has been relatively little research on exploiting symmetry in verifying finite state systems. Most of the work on this problem has been performed by researchers investigating the reachability problem for Petri nets [11, 19]. However, this work does not consider general temporal properties nor the complications that are caused by representing the state space using BDDs. In related research Ip and Dill [14] propose a data type *scalarset* which facilitates detection of symmetry in finite state systems. Their technique uses an explicit state representation rather than BDDs. The research closest to our own is of Emerson and Sistla [7], but their work does not investigate the complexity that arises while using BDDs. In [15] an approach to cut down the cost of protocol analysis using quotient structures induced by automorphism is proposed.

Our paper is organized as follows: The second section describes how symmetry groups act on Kripke models. The third section gives the syntax and semantics of the logic CTL* that we use for writing specifications. In the fourth section we show how to construct a BDD representation for the quotient model from the generators of its symmetry group. We also prove that a CTL* formula which is invariant under the symmetry group will be true in the original model if and only if it is true in the quotient model. Section 5 describes how the orbit relation can be used to reduce the size of the state space that must be searched in temporal logic model checking. In the next two sections we investigate the complexity of computing this relation and give lower bounds on the size of the BDDs needed to represent it. In the next section we show how to avoid constructing the full orbit relation during model checking. In Section 8 we demonstrate how symmetry can be used to verify a version of the Futurebus cache coherency protocol with multiple cache lines. In Section 9 we compare our techniques with bisimulation minimization. The final section contains a discussion of some directions for future research.

2. Symmetry groups

Let AP be a set of atomic propositions. A Kripke structure over AP is a triple $M = (S, R, L)$, where

- S is a finite set of *states*,
- $R \subseteq S \times S$ is a *transition relation*, which must be total (i.e., for every state s_1 there exists a state s_2 such that $(s_1, s_2) \in R$).
- $L : S \rightarrow 2^{\mathrm{AP}}$ is a *labeling function* which associates with each state a set of atomic propositions that are true in the state.

Let G be a permutation group, i.e., bijective mappings acting on the state space S of the Kripke structure M. A permutation $\sigma \in G$ is said to be a *symmetry* of M if and only if it preserves the transition relation R. More formally, σ should satisfy the following condition:

$$(\forall s_1 \in S)(\forall s_2 \in S)((s_1, s_2) \in R \Rightarrow (\sigma s_1, \sigma s_2) \in R)$$

G is a *symmetry group* for the Kripke structure M if and only if every permutation $\sigma \in G$ is a *symmetry* of M. Notice that our definition of a symmetry group does not refer to the labeling function L. Furthermore, since every $\sigma \in G$ has an inverse, which is also a symmetry, it can be easily proved that a permutation $\sigma \in G$ is a symmetry for a Kripke structure if and only if σ satisfies the following condition:

$$(\forall s_1 \in S)(\forall s_2 \in S)((s_1, s_2) \in R \Leftrightarrow (\sigma s_1, \sigma s_2) \in R)$$

Example 2.1. The transposition $\sigma = (S_1, S_2)$ exchanges the states S_1 and S_2 in the Kripke Structure shown in figure 1. The states S_0 and S_3 are not affected by the permutation σ, so the successors of all the states remain the same when σ is applied. Hence, σ is a symmetry of the Kripke Structure.

Let $\langle g_1, \ldots, g_k\rangle$ be the smallest permutation group containing all the permutations, $g_1, \ldots, g_k$. If $G = \langle g_1, \ldots, g_k\rangle$, then we say that the group G is *generated* by the set $\{g_1, \ldots, g_k\}$. It is easy to see that if every generator of the group G is a symmetry of M, then the group G is a symmetry group for M.

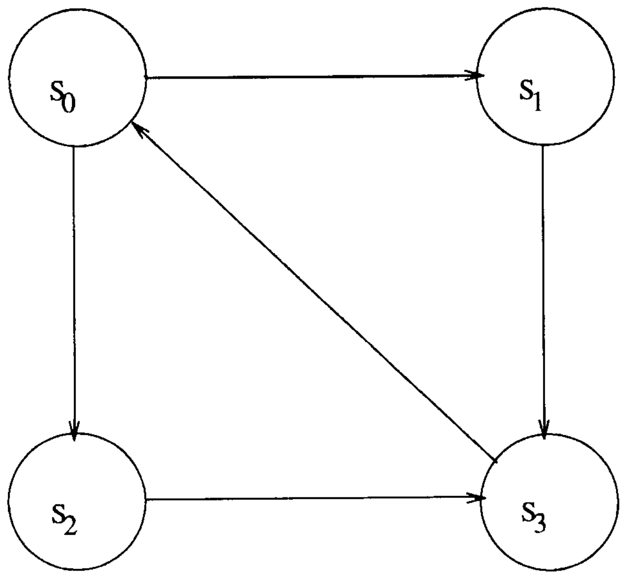

Figure 1. A Kripke structure.

3. The temporal logic CTL*

There are two types of formulas in CTL*: *state formulas* (which are true in a specific state) and *path formulas* (which are true along a specific path). Let AP be the a set of atomic propositions. A state formula is either:

- p, if $p \in$ AP
- if f and g are state formulas, then $\neg f$ and $f \vee g$ are state formulas;
- if f is a path formula, then $E(f)$ is a state formula.

A path formula is either:

- a state formula;
- if f and g are path formulas, then $\neg f$, $f \vee g$, Xf, and fUg are path formulas;

CTL* is the set of state formulas generated by the above rules.

We define the semantics of CTL* with respect to a Kripke structure $M = (S, R, L)$. A *path* in M is an infinite sequence of states $\pi = s_0, s_1, \ldots$, such that, for every $i \geq 0$, $(s_i, s_{i+1}) \in R$. π^i denotes the *suffix* of π starting at s_i. We use the standard notation to indicate that a state formula f holds in a structure. $M, s \models f$ means that f holds at the state s in the structure M. Similarly, $M, \pi \models f$ means that the path formula f is true along the path π. Assume that f_1 and f_2 are state formulas and g_1 and g_2 are path formulas, then the relation $\models$ is defined inductively as follows:

1. $s \models p \Leftrightarrow p \in L(s)$
2. $s \models \neg f_1 \Leftrightarrow s \not\models f_1$
3. $s \models f_1 \vee f_2 \Leftrightarrow s \models f_1$ or $s \models f_2$
4. $s \models E(g_1) \Leftrightarrow$ there exists a path π starting with s such that $\pi \models g_1$
5. $\pi \models f_1 \Leftrightarrow s$ is the first state of π and $s \models f_1$
6. $\pi \models \neg g_1 \Leftrightarrow \pi \not\models g_1$
7. $\pi \models g_1 \vee g_2 \Leftrightarrow \pi \models g_1$ or $\pi \models g_2$
8. $\pi \models X g_1 \Leftrightarrow \pi^1 \models g_1$
9. $\pi \models g_1 U g_2 \Leftrightarrow$ there exists $k \geq 0$ such that $\pi^k \models g_2$ and for all $0 \leq j < k, \pi^j \models g_1$

CTL is a subset of CTL* in which we restrict the path formulas to be:

- If f and g are state formulas, then Xf and fUg are path formulas.
- If f is a path formula, then so is $\neg f$.

The basic modalities of CTL are EFf, EGf, and $E(fUg)$, where f and g are again CTL formulas. The operators AGf, AFf and $A(fUg)$ can be expressed in terms of the basic modalities described above. Efficient procedures have been developed to determine if a CTL formula f is true in a Kripke Model M. In [5] a model checking algorithm is given that is linear in the size of the formula f and the model M. A symbolic model checking algorithm using BDDs that can handle models with more that 10^{20} states is discussed in [3].

4. Quotient models

Let G be a group acting on the set S and let s be an element of S, then the *orbit* of s is the set $\theta(s) = \{t \mid (\exists\sigma \in G)(\sigma s = t)\}$. From each orbit $\theta(s)$ we pick a representative which we call $\text{rep}(\theta(s))$ with the restriction that $\text{rep}(\theta(s)) \in \theta(s)$.

Definition 4.1. Let $M = (S, R, L)$ be a Kripke structure and let G be a symmetry group acting on M. We define the *quotient structure* $M_G = (S_G, R_G, L_G)$ in the following manner:

- The state set is $S_G = \{\theta(s) \mid s \in S\}$, the set of orbits of the states in S;
- The transition relation R_G is given by

$$R_G = \{(\theta(s_1), \theta(s_2)) \mid (s_1, s_2) \in R\}; \tag{1}$$

- The labeling function L_G is given by $L_G(\theta(s)) = L(\text{rep}(\theta(s)))$.

Next, we define what it means for a symmetry group G of a Kripke Structure M to be an *invariance group* for an atomic proposition p. Intuitively, G is an invariance group for an atomic proposition p if and only if the set of states labeled by p is closed under the application of all the permutations of G. More formally, a symmetry group G of a Kripke Structure $M = (S, R, L)$ is an invariance group for an atomic proposition p if and only if the following condition holds:

$$(\forall\sigma \in G)(\forall s \in S)(p \in L(s) \Leftrightarrow p \in L(\sigma s))$$

Lemma 4.1. *If G is an invariance group for an atomic proposition p and $p \in L(s)$, then $p \in L_G(\theta(s))$ in the quotient Kripke structure M_G.*

Proof: Let $s_1 = \text{rep}(\theta(s))$. By the definition of the orbit $s_1 = \sigma s$ for some $\sigma \in G$. If $p \in L(s)$, then $p \in L(\sigma s)$ because G is an invariance group for p. Therefore $p \in L(s_1)$, and since $L_G(\theta(s)) = L(s_1)$ we have that $p \in L_G(\theta(s))$. □

Definition 4.2. Given a Kripke structure $M = (S, R, L)$ and a symmetry group G, let $M_G = (S_G, R_G, L_G)$ be the quotient Kripke structure. Two paths $\pi = s_0, s_1, \ldots,$ in M and $\pi_G = \theta(t_0), \theta(t_1), \ldots,$ *correspond* if and only if $\forall i(s_i \in \theta(t_i))$.

Lemma 4.2. *For every path starting from s_0 in M there exists a corresponding path starting from $\theta(s_0)$ in M_G, and for every path starting from $\theta(s_0)$ in M_G there exists a corresponding path starting from s_0 in M.*

Proof:

($\Rightarrow$) Let $\pi = s_0, s_1, \ldots,$ be a path in M. The corresponding path in M_G is the path produced by taking the orbits of the states, or $\pi_G = \theta(s_0), \theta(s_1), \ldots.$ Notice that

π_G is a valid path in the quotient structure M_G because $(s_i, s_{i+1}) \in R$ implies that $(\theta(s_i), \theta(s_{i+1})) \in R_G$.

($\Leftarrow$) Let $\pi_G = \theta(s_0), \theta(s_1), \ldots,$ be a path in M_G. We show how to construct a path $\pi = t_0, t_1, \ldots,$ in M such that $t_0 = s_0$ and $t_i \in \theta(s_i)$. We will construct the path in an inductive manner. Initially, we let $t_0 = s_0$. We will maintain the invariant that $t_i \in \theta(s_i)$. Since $(\theta(s_0), \theta(s_1)) \in R_G$ there exists $u \in \theta(s_0)$ and $v \in \theta(s_1)$ such that $(u, v) \in R$. Since $u \in \theta(s_0)$ there exists a $\sigma \in G$ such that $s_0 = \sigma u$. By the definition of the symmetry group $(\sigma u, \sigma v) \in R$, or $(s_0, \sigma v) \in R$. Let $t_1 = \sigma v$. Notice that since $v \in \theta(s_1)$, we have that $t_1 \in \theta(s_1)$. Assume that we have constructed a path π up to t_k such that $t_k \in \theta(s_k)$. Using an argument similar to one given above we can find $t_{k+1} \in \theta(s_{k+1})$ such that $(t_k, t_{k+1}) \in R$. □

Theorem 4.1. *Let $M = (S, R, L)$ be a Kripke Structure, G be a* symmetry group *of M, and h be a CTL* formula. If G is an invariance group for all the atomic propositions p occurring in h, then*

$$M, s \models h \Leftrightarrow M_G, \theta(s) \models h \tag{2}$$

where M_G is the quotient structure corresponding to M.

This theorem is a direct consequence of the following lemma.

Lemma 4.3. *Let h be a either a state formula or a path formula such that G is an invariance group for all atomic propositions p occurring in h. Let $\pi = s, s_1, \ldots,$ be a path in M and $\pi_G = \theta(s), \theta(t_1), \ldots,$ be a corresponding path in M_G. Then*

- *$M, s \models h \Leftrightarrow M_G, \theta(s) \models h$ if h is a state formula, and*
- *$M, \pi \models h \Leftrightarrow M_G, \pi_G \models h$ if h is a path formula.*

Proof: This proof is similar to the Proof of Lemma 3.2 given in [1].

Basis: $h \in AP$. We have that G is an invariance group for h. In this case it is easy to see by Lemma 4.1 and the definition of an invariance group that $M, s \models h \Leftrightarrow M_G\theta(s) \models h$.

Induction: There are several cases.

- $h = \neg h_1$, *a state formula.* By the inductive hypothesis we have that $M, s \models h_1 \Leftrightarrow M_G, \theta(s) \models h_1$. Therefore $M, s \models h \Leftrightarrow M_G, \theta(s) \models h$. The same reasoning holds if h is a path formula.
- $h = h_1 \vee h_2$, *a state formula.*

$$\begin{aligned} M, s \models h &\Leftrightarrow M, s \models h_1 \text{ or } M, s \models h_2 \\ &\Leftrightarrow M_G, \theta(s) \models h_1 \text{ or } M_G, \theta(s) \models h_2 \\ &\Leftrightarrow M_G, \theta(s) \models h \end{aligned}$$

The second step uses the inductive hypothesis. We can also use this argument if h is a path formula.

- $h = E(h_1)$, *a state formula.* Suppose $M, s \models h$. There is a path π starting with s such that $M, \pi \models h_1$. By Lemma 4.2 there is a corresponding path π_G in M_G starting with $\theta(s)$. By the inductive hypothesis $M, \pi \models h_1 \Leftrightarrow M_G, \pi_G \models h_1$. Therefore, $M, s \models E(h_1) \Rightarrow M_G, \theta(s) \models E(h_1)$. A similar argument holds in the other direction.
- $h = h_1$, *where h is a path formula and* h_1 *is a state formula.* Although the lengths of h and h_1 are the same, we can imagine that $h = \mathit{path}(h_1)$, where *path* is an operator which converts a state formula into a path formula. Now we can apply the inductive step.
- $h = Xh_1$, *a path formula.* By the definition of the next time operator $M, \pi^1 \models h_1$. Since π and π_G correspond, so do π^1 and π_G^1. Therefore, by the inductive hypothesis, $M_G, \pi_G^1 \models h_1$, so $M_G, \pi \models h$. A similar argument proves the implication in the other direction.
- $h = h_1 U h_2$, *a path formula.* Suppose that $M, \pi \models h_1 U h_2$. By the definition of the *until* operator, there is a k such that $M, \pi^k \models h_2$ and for all $0 \leq j < k$, $M, \pi^j \models h_1$. Since π and π_G correspond, so do π^j and π_G^j for any j. Therefore, by inductive hypothesis M_G, $\pi_G^k \models h_2$ and $M_G, \pi_G^j \models h_2$ for all $0 \leq j < k$. Therefore, we have $M_G, \pi_G \models h$. We can use the same argument in the other direction. □

5. Model checking in the presence of symmetry

In this section we describe how to do model checking in the presence of symmetry. First, we discuss how to find the set of states in a Kripke structure that are reachable from a given set of initial states using an explicit state representation. In the explicit state case, a breadth-first or depth-first search starting from the set of initial states is performed. Typically, two lists, a list of reached states and a list of unexplored states are maintained. At the beginning of the algorithm, the initial states are put on both the lists. In the exploration step, a state is removed from the list of unexplored states and all its successors are processed. An algorithm for exploring the state space of a Kripke structure in the presence of symmetry is discussed in [14]. The authors introduce a function $\xi(q)$, which maps a state q to the unique state representing the orbit of that state. While exploring the state space, only the unique representatives from the orbits are put on the list of reached and unexplored states. Figure 2 gives the pseudo-code for exploring the state space under the presence of symmetry. This simple reachability algorithm can be extended to a full CTL model checking algorithm by using the technique described in [5]. To construct the function $\xi(q)$ it is important to compute the orbit relation efficiently. In the next section we will discuss the computational complexity of finding the orbit relation.

In the remainder of this section we focus on how to do symbolic model checking in the presence of symmetries. The straightforward method of computing the quotient model uses the BDD for the orbit relation $\Theta(x, y) \equiv (x \in \theta(y))$. Given a Kripke structure $M = (S, R, L)$ and a symmetry group G on M with r generators $g_1, g_2, \ldots, g_r$, it is possible to prove that the orbit relation Θ is the least fixpoint of the equation given below:

$$Y(x, y) \equiv (x = y) \vee (\exists z)(Y(x, z) \wedge (z = g_1 y \vee z = g_2 y \cdots \vee z = g_r y)) \tag{3}$$

In the next lemma we prove that the least fixpoint of the Eq. (3) is the orbit relation Θ.

```
$Reached = \emptyset$;
$Unexplored = \emptyset$;
For each initial state $s$ Do
        Append $\xi(s)$ to $Reached$;
        Append $\xi(s)$ to $Unexplored$;
Endforloop
While $Unexplored \neq \emptyset$ Do
        Remove a state $s$ from $Unexplored$;
        For each succesor state $q$ of $s$
                If $\xi(q)$ is not in $Reached$
                        Append $\xi(q)$ to $Reached$;
                        Append $\xi(q)$ to $Unexplored$;
                EndIf
        Endforloop
EndWhile
```

Figure 2. Exploring state space in presence of symmetry.

Lemma 5.1. *The least fixpoint of Eq.* (3) *is the orbit relation* Θ *induced by the group* G *generated by* $g_1, g_2, \ldots, g_r$.

Proof: First, we prove that Θ is a fixpoint of the equation given below:

$$Y(x, y) = (x = y \vee (\exists z)(Y(x, z) \wedge (z = g_1 y \vee z = g_2 y \cdots \vee z = g_r y)))$$

It is obvious by the transitivity and reflexivity of the orbit relation Θ that

$$\Theta(x, y) \subseteq (x = y \vee (\exists z)(\Theta(x, z) \wedge (z = g_1 y \vee z = g_2 y \cdots \vee z = g_r y)))$$

Suppose $\Theta(x, y)$, then by the definition of the orbit relation there exists $\sigma \in G$ such that $x = \sigma y$. Let us assume $x \neq y$ (if $x = y$, there is nothing to prove). This means there exists a generator g_k, $k \leq r$ such that $x = \sigma_1 g_k y$. Setting $z = g_k y$, we see that $\Theta(x, z)$ and $z = g_k y$. Since x and y are arbitrary boolean vectors we get the following inclusion:

$$\Theta(x, y) \subseteq (x = y \vee (\exists z)(\Theta(x, z) \wedge (z = g_1 y \vee z = g_2 y \cdots \vee z = g_r y)))$$

Hence Θ is a fixpoint of Eq. (3).

Next, we prove that if T is any fixpoint of the Eq. (3), then $\Theta \subseteq T$. We prove that $\Theta(x, y) \Rightarrow T(x, y)$. The definition of the orbit relation $\Theta(x, y)$ implies that there exists a $\sigma = g_{i_m} \cdots g_{i_2} g_{i_1}$, $1 \leq i_j \leq r$ such that $x = \sigma y$. Since T is a fixed point of Eq. (3), it can be proved by induction that for all $1 \leq l \leq m$, $T(g_{i_l} \cdots g_{i_1} y, y)$ holds. Using this result for $l = m$, we see that $T(x, y)$ holds. Since $\Theta(x, y) \Rightarrow T(x, y)$, we obtain that $\Theta \subseteq T$. Hence, Θ is the least fixpoint. □

If a suitable state encoding is available, this fixpoint equation can be computed using BDDs [3]. Once we have the orbit relation Θ, we need to compute a function $\xi\colon S \to S$, which maps each state s to the unique representative in its orbit. If we view states as vectors of values associated with the state variables, it is possible to choose the lexicographically smallest state to be the unique representative of the orbit. Since Θ is an equivalence relation, these unique representatives can be computed using BDDs by the method of Lin [16].

Definition 5.1. Let $B = \{0, 1\}$ and $\mathcal{R} \subseteq B^r \times B^n$ be a total relation. A function $\mathcal{F}\colon B^r \to B^n$ is *compatible* with $\mathcal{R}$ if it has the following properties:

- For all $x \in B^r$, we have $(x, \mathcal{F}(x)) \in \mathcal{R}$;
- For every u and v in B^r, if the possible mappings of u and v are the same, i.e., $(\forall y \in B^n)((u, y) \Leftrightarrow (v, y))$ then $\mathcal{F}(u) = \mathcal{F}(v)$;

Lin [16] also defines a *compatible projection operator* which is a compatible function that maps $x \in B^r$ to the least $y \in B^n$ (with respect to some norm N) such that $(x, y) \in \mathcal{R}$. Formally, the compatible projection is defined as follows:

Definition 5.2. Let $\mathcal{R} \subseteq B^r \times B^n$ be a binary relation. The *projection* of $\mathcal{R}$, denoted by projection ($\mathcal{R}$), is the function $\mathcal{F}$ defined as follows:

$$\mathcal{F} = \{(x, y) \mid (x, y) \in \mathcal{R} \wedge (\forall z)((x, z) \in \mathcal{R} \Rightarrow N(z) \geq N(y))\}$$

where the norm $N(x)$ is defined as follows:

$$N(x) = \sum_{i}^{n} x_i 2^{n-i}$$

which is the number whose binary representation is the vector x.

It can be shown that *projection* ($\mathcal{R}$) is a compatible function of $\mathcal{R}$. This function can be computed efficiently in a single bottom-up traversal of the BDD representation of the relation $\mathcal{R}$. Given the orbit relation Θ, the function *projection* (Θ) maps each state to the unique representative of its orbit. Notice that *projection* (Θ) is exactly the function ξ introduced before.

Assuming that we have the BDD representation of the mapping function ξ, the transition relation R_G of the quotient structure can be expressed as follows:

$$R_G(x, y) = (\xi(x) = x) \wedge (\exists y_1)(R(x, y_1) \wedge \xi(y_1) = y)$$

The formula $\xi(x) = x$ expresses the fact that x is the unique representative of its orbit.

6. Complexity of orbit calculations

The behavior of a sequential circuit or protocol is frequently determined by the values of a set of boolean state variables $x_1, x_2, \ldots, x_n$. For example, the behaviour of a bus arbitration protocol may be determined by the boolean state variables which encode the command on the bus and the identity of the master. When we extract a Kripke structure from a circuit or protocol, we treat these state variables as atomic propositions. The resulting Kripke model $M = (S, R, L)$ will have the following components:

- $S \subseteq B^n$, where each state can be thought of as a truth assignment to the n state variables.
- $R \subseteq S \times S$, where R is determined by the behavior of the circuit or protocol.
- The labeling function L is defined so that $x_i \in L(s)$ if and only if the ith component of s is 1.

It is often the case that the symmetry group is also given in terms of the state variables. For example, in a two bit adder with inputs x_1, x_2 and x_3, x_4, the permutation (13)(24) is a symmetry because we can exchange the inputs without affecting the result. If we have a permutation σ, which acts on the set $\{1, 2, \dots, n\}$, then σ acts on vectors in B^n in the following manner:

$$\sigma(x_1, x_2, \dots, x_n) = (x_{\sigma(1)}, x_{\sigma(2)}, \dots, x_{\sigma(n)})$$

Given two vectors x and y in B^n and a permutation σ, it is easy to see that $x \neq y$ implies $\sigma x \neq \sigma y$. Therefore, a group G acting on the set $\{1, 2, \dots, n\}$ induces a permutation group G_1 acting on the set B^n. In other words, a symmetry on the structure of a circuit induces a symmetry on the state space of the circuit.

Definition 6.1. Let G be a group acting on the set $\{1, 2, \dots, n\}$. Assume that G is represented in terms of a finite set of generators. Given two vectors $x \in B^n$ and $y \in B^n$, the *orbit problem* asks whether there exists a permutation $\sigma \in G$ such that $y = \sigma x$.

Let G induce the permutation group G_1 acting on B^n. The orbit problem asks if x and y are in the same orbit under the action of the group G_1. First, we prove that the *orbit problem* is as hard as the *Graph Isomorphism* problem.

Definition 6.2. Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ such that $|V_1| = |V_2|$, the *Graph Isomorphism problem* asks whether there exists a bijection $f: V_1 \to V_2$ such that the following condition holds

$$(i, j) \in E_1 \Leftrightarrow (f(i), f(j)) \in E_2$$

Theorem 6.1. *The* orbit problem *is as hard as the* Graph Isomorphism *problem.*

Proof: Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ we construct a group G and two $0 - 1$ vectors x and y such that x and y are in the same orbit under the action of the group G if and only if G_1 and G_2 are isomorphic. We assume that $|V_1| = |V_2| = n$. Let $A = \{a_{ij}\}$ and $B = \{b_{ij}\}$ be the adjacency matrices of the graph G_1 and G_2 respectively. Let $x \in B^{n^2}$ be defined as follows:

$$x_{n(i-1)+j} = a_{ij}, 1 \leq i \leq n, 1 \leq j \leq n$$

The vector $x \in B^{n^2}$ is a list of the elements of the matrix A in row order. The vector $y \in B^{n^2}$ is defined in a similar fashion using the adjacency matrix B. Let (ij) be a transposition acting on the set $\{1, 2, \ldots, n\}$. Intuitively, we can think of this transposition as exchanging the vertices i and j in the graph G_1. This corresponds to exchanging the rows i and j and columns i and j in the adjacency matrix and has exactly the same effect as applying the permutation σ given below to the vector x.

$$\begin{aligned}
\sigma_{\text{row}} &= (n(i-1)+1, n(j-1)+1) \cdots (n(i-1)+n, n(j-1)+n) \\
\sigma_{\text{col}} &= (i, j) \cdots ((n-1)n+i, (n-1)n+j) \\
\sigma &= \sigma_{\text{row}}\sigma_{\text{col}}
\end{aligned}$$

Each permutation acting on the set of size $n = \mid V_1 \mid$ corresponds to a bijection $f \colon V_1 \mapsto V_2$. We assume that the vertices are labeled by integers. If the bijection corresponding to the permutation (ij) is an isomorphism between G_1 and G_2, then exchanging rows i and j and columns i and j in the adjacency matrix A gives B. This implies that $y = \sigma x$ because x and y are just encodings of the adjacency matrix A and B respectively. Similarly, if $y = \sigma x$, then the bijection corresponding to the permutation (ij) is an isomorphism between the graph G_1 and G_2. Therefore, $y = \sigma x$ if and only if the bijection corresponding to the permutation (ij) is an isomorphism between G_1 and G_2. Every bijection $f : V_1 \mapsto V_2$ corresponds to some permutation in the full symmetric group S_n. Since the groups S_n acting on the set $\{1, 2, \ldots, n\}$ is generated by the transpositions (12), (13), ..., and $(1n)$ we have the result. We just have to code all these transpositions in the context of the $0-1$ vectors x and y. □

Example 6.1. Consider the two graphs G_1 and G_2 given in the figure 3. The vectors x and y given below encode the adjacency matrices of the graphs G_1 and G_2 respectively:

$$\begin{aligned}
x &= (011 \;\; 100 \;\; 100) \\
y &= (010 \;\; 101 \;\; 010)
\end{aligned}$$

The permutations are defined as follows:

$$\begin{aligned}
\sigma_{\text{row}} &= (1,4)(2,5)(3,6) \\
\sigma_{\text{col}} &= (1,2)(4,5)(7,8) \\
\sigma &= \sigma_{\text{row}}\sigma_{\text{col}}
\end{aligned}$$

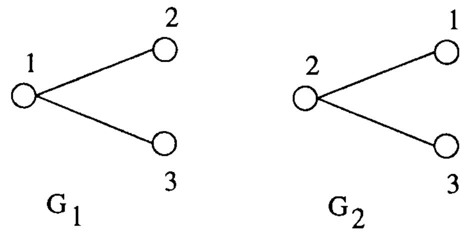

Figure 3. Two isomorphic graphs.

Notice that $y = \sigma x$ and the bijection corresponding to the permutation $(1, 2)$ is an isomorphism between G_1 and G_2. The permutation σ_{row} corresponds to exchanging rows 1 and 2.

A modified version of the *orbit problem* called the *bounded orbit problem* is defined as follows:

Definition 6.3. Given a group G generated by r permutations $g_1, g_2, \ldots, g_r$ (the permutations act on the set $\{1, 2, \ldots, n\}$), two vectors $x, y \in B^n$, and an integer k, does there exist a permutation σ obtained by at most k applications of the generators such that $x = \sigma y$? Formally, σ is of the following form $\sigma = g_{i_1} g_{i_2} \cdots g_{i_m}, m \leq k$.

Intuitively, in the *bounded orbit problem* we bound how many times we can apply the generators. Although the graph isomorphism problem is not known to be NP-hard, the bounded orbit problem can be shown to be NP-complete. The reduction is form EXACT COVER BY 3-SETS [10].

Definition 6.4. The EXACT COVER BY 3-SETS (X3C) is defined as follows:

INSTANCE: Set X with $| X |= 3q$ and a collection C of 3-element subsets of X.

QUESTION: Does C contain an exact cover for X, i.e., a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one member of C'.

Theorem 6.2. *The* bounded orbit problem *is NP-complete.*

Proof: The reduction will be from the $X3C$ problem. Consider an instance of $X3C$, where we are given a specific set X with $|X| = 3q$ and a collection C of 3-element subsets of X. We construct a group G acting on the set $\{1, 2, \ldots, 6q\}$. Let $n = 3q$. With an element $x_i \in X$ we associate the permutation $(i, n+i)$, $1 \leq i \leq n$. With each 3-element set in the collection C we associate a permutation which is the product of the permutation corresponding to its elements. For example, if we have the set $\{x_i, x_j, x_k\} \in C$, then the permutation associated with it is $(i, n+i)(j, n+j)(k, n+k)$. Let C_g be the set of permutations corresponding to the collection C. The group G is the group generated by the set of permutations C_g. Consider two $0-1$ vectors x and y of length $6q$ defined as follows: x has 1's in first $3q$ positions and 0's in last $3q$ positions, and y has 0's in first $3q$ positions and 1's in last $3q$ positions. We will show that there exists a permutation σ, which is the product of q generators from the set C_g, such that $y = \sigma x$ if and only if the instance of $X3C$ has a cover C' for the set X.

Notice that since we are dealing with 3-element subsets and X has $3q$ elements we have to use exactly q 3-element sets to cover X. Suppose there is a collection C' of q sets such that is covers X. Consider the permutation σ which is the product of all the permutations corresponding to the sets in the cover C'. Since C' is a cover for X, the transposition $(i, n+i)$ for each element x_i occurs in the permutation σ. Hence, it is obvious that $y = \sigma x$.

Suppose, on the other hand, there exists a permutation σ, which is the product of less than or equal to q generators from the set C_g, such that $y = \sigma x$. Let us assume that $\sigma = g_1 \cdots g_r$

where $r \leq q$ and $g_i \in C_g$ for $1 \leq i \leq r$. Since $x_i = 1$ and $y_{n+1} = 1$ (for $1 \leq i \leq 3q$), the permutation σ has to include each transposition of the form $(i, n+i)$ for $1 \leq i \leq 3q$. Since we need at least $3q$ transpositions of the form $(i, n+i)$ to transform the vector x to y, we will have to use exactly q generators. Moreover, the q generators have to be disjoint. It follows that the collection formed by the 3-element sets corresponding to the q generators which form σ is a cover for X. So the instance of the bounded orbit problem with the set of generators C_g, $0-1$ vectors x and y, and the integer bound q has a solution if and only if the instance of $X3C$ has solution.

The bounded orbit problem is obviously in NP because we can guess the string of k (the given integer) generators which generates the permutation σ such that $y = \sigma x$, were x and y are the specified $0-1$ vectors. □

Example 6.2. Consider the following instance of $X3C$ with $q = 2$. The various sets in this instance of $X3C$ are defined as follows:

$$X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$$
$$C = \{\{x_1, x_3, x_5\}, \{x_2, x_4, x_6\}, \{x_1, x_2, x_6\}\}$$

Now we define the instance of the bounded orbit problem. With each element x_i, $1 \leq i \leq 6$ we associate the permutation $(i, 6+i)$. With the collection C we associate the following set C_g of permutations:

$$C_g = \{(1, 6+1)(3, 6+3)(5, 6+5), (2, 6+2)(4, 6+4)(6, 6+6), \\ (1, 6+1)(2, 6+2)(6, 6+6)\}$$

For example, the permutation $(1, 6+1)(3, 6+3)(5, 6+5)$ corresponds to the set $\{x_1, x_3, x_5\}$. Consider the following vectors x and y in B^{12}.

$$x = (111111000000)$$
$$y = (000000111111)$$

Notice that $\{\{x_1, x_3, x_5\}, \{x_2, x_4, x_6\}\}$ is a cover for x, and the permutation σ which is the product of the permutations corresponding to these sets satisfies the equation: $y = \sigma x$.

At first glance, the bounded orbit problem might seem much weaker than the general orbit problem, but given a permutation group G with an arbitrary set of generators, we can always find a set of generators such that every permutation $\sigma \in G$ is the product of at most n of the new generators, where n is the size of the set on which G acts. This result follows immediately using an algorithm described in [9]. Given a group G acting on a set of size n and generated by $g_1, \ldots, g_k$, the paper shows how to construct a table T with n rows (labeled 0 to $n-1$) and n columns (labeled 1 to n), of permutations with the following property: $\sigma \in G$ if and only if σ can be expressed as $a_0 a_1 \cdots a_r$, were a_i is a member of the ith row. Using the permutations in the table T, the general orbit problem on G can be converted into an instance of the bounded orbit problem (with the bound n).

7. Complexity of the BDD for the orbit relation

We prove lower bounds on the BDD sizes for the orbit relation of transitive groups. The group of all permutations acting on the set $[n] = \{1, 2, \dots, n\}$ is called the *full symmetric group* and is denoted by S_n. The group acting on $[n]$ which is generated by $(1, 2, \dots, n)$ is called the *rotation group*. Transitive groups occur frequently in practice, e.g., both the full symmetric and the rotation group are transitive.

Definition 7.1. A group of G permutations acting on a set S is called *transitive*, if there exists an element i in the set S, such that for all elements $j \in S$ there exists an element $g_{i,j} \in G$ that maps i into j.

As a simple consequence for all $j, k \in S$ there is a $g_{j,k} \in G$ that maps j into k as we can define $g_{j,k} = g_{i,j}^{-1} g_{i,k}$. Let $G_{i,j} = \{g_{i,j} \in G \mid g_{i,j} \textit{ maps } i \textit{ into } j\}$, then $|G_{i,j}| = |G_{1,1}|$ as $G_{i,j} = g_{i,1} G_{1,1} g_{1,j}$ and $|G| = |S|\,|G_{1,1}|$ as $G = \bigcup_{i \in S} G_{1,i}$ and of course all $G_{1,i}$ are mutually disjoint.

A special case of the above situation is the following: the set $S = B^{nm}$ (i.e., a system composed of n components each with m state variables) and the group G of permutations is S is induced by a group G' of permutations acting on the set $\{1, \dots, n\}$ in the following way:

$$g(\langle x_{1,1}, \dots, x_{1,m}, \dots, x_{n,1}, \dots, x_{n,m} \rangle) = \langle x_{g'(1),1}, \dots, x_{g'(1),m}, \dots, x_{g'(n),1}, \dots, x_{g'(n),m} \rangle$$

with $g \in G$ and $g' \in G'$. Intuitively, the group G' tells us how to permute blocks where each block has m variables. Ring structures, where the components are ordered in a ring and can be rotated any number of steps, occur frequently in practice. The token ring protocol used in the solution to the distributed mutual exclusion problem uses this topology. In the terminology given above the symmetry group for the ring structure is induced by the rotational group. In star or bus topologies components are unordered and can be exchanged arbitrarily. Such situations occur, for example, in systems where components communicate via a common bus (e.g., multiprocessor systems), or in systems with broadcast and star-like communication structures. The symmetry group of these systems is induced by the full symmetric group, i.e., $G' = S_n$.

Definition 7.2. The orbit relation Θ of G is the set of pairs $\{\langle s, t \rangle \mid t \in \theta(s)\}$. The orbit relation *induced* by G' is the orbit relation of group G. If $S = B^{nm}$ the orbit relation can be represented by a characteristic boolean function Θ: $B^{nm} \times B^{nm} \to B$.

Theorem 7.1. *Let $S = B^{nm}$. For a transitive group G acting on the set $\{1, 2, \dots, n\}$ we can obtain the following lower bound for the BDD representing the induced orbit relation Θ:*

$$|\Theta| > 2^K \textit{ with } K = \min(\lfloor \sqrt{n} \rfloor, 2^{m-1} - 1)$$

Proof: We use unprimed variables $x_{i,j}$ for representing the first argument of the orbit relation and primed variables $x'_{i,j}$ for the second. For the proof we only consider the first variable of each component. First we determine a partition (L, R) of the variables—we go through all the variables in the given variable ordering, until we have K unprimed variables $x_{i,1}$ or K primed variables $x'_{i,1}$ in L and put the rest of the variables in R. Notice that all variables in L precede all the variables in R with respect to the variable ordering. Without loss of generality, we assume that L contains K unprimed variables with indices $I = \{i_1, \ldots, i_K\}$ and less than K primed variables $x'_{j,1}$ with indices $j \in J$.

With each pair $x_{i,1}, x'_{j,1}, i \in I, j \in J$ we associate a set of permutations $G_{i,j}$, i.e., the set of group elements of G that map the ith component into the jth component. Given a permutation $g \in G$ if there exists $i \in I$ and $j \in J$ such that g maps the ith component to the jth component, then $g \in G_{i,j}$. Since $|\bigcup_{i \in I, j \in J} G_{i,j}| \le |I||J||G_{1,1}| < K^2|G_{1,1}| \le |G|$, we can find a $g \in G$, such that all primed variables $x'_{g(i),1}$ for $i \in I$ are in R (otherwise $G = \bigcup_{i \in I, j \in J} G_{i,j}$). We construct an assignment in the following way: for $i_j \in I$, the variables $\langle x_{i_j,2}, \ldots, x_{i_j,m}\rangle$ and $\langle x'_{g(i_j),2}, \ldots, x'_{g(i_j),m}\rangle$ are instantiated with the binary representation of the number $\#j$ for $1 \le j \le K$. Note that since K has to be representable in $m - 1$ bits, we need that $K \le 2^{m-1} - 1$. The variables $x_{i,j}$ and $x'_{g(i),j}$ are instantiated with 0 for $i \notin I$. Let Θ' be the BDD obtained from Θ by the above instantiation. Instantiating variables with constant values in a BDD does not increase its size, so $|\Theta'| \le |\Theta|$. The BDD Θ' only depends on the variables $x_{i,1}$ and $x'_{g(i),1}$ for $i \in I$ and by our construction the only valid assignments are those, where the values of $x_{i,1}$ and $x'_{g(i),1}$ agree. In other words, Θ' is the BDD for the following proposition:

$$\bigwedge_{i \in I} x_{i,1} = x'_{g(i),1}$$

Since all variable $x_{i,1}$ precede all variables $x'_{goi,1}$, the BDD Θ' has atleast 2^K nodes. □

The above proof was originally obtained for the special cases of rotation and fully symmetric and subsequently generalized to the case of transitive groups of permutations by W. Büttner. It should be noted that in the above proof we instantiate most of the variables to get a BDD of exponential size. Therefore, we expect that the actual size is much worse than the above proved lower bound. For full symmetric groups $G' = S_n$, K can be proved to be the minimum of n and 2^m. The BDD of the orbit relation induced by a transitive group on the components is exponential in the minimum of the number of components and the number of states in one component. Consequently, exploiting these types of symmetries is symbolic model checking is restricted to examples with a small number of components or where each component has only a few states. An approach which avoids the computation of the orbit relation is described in Section 8.

8. Multiple representatives

Since the BDD for the orbit relation is large for some groups which occur frequently in practice, it is computationally expensive to use the single representative theory described earlier. For example, Lin's algorithm to extract an unique representative from each orbit would need the BDD for the orbit relation. Therefore, we now develop a scheme to use multiple representatives where the BDDs remain more manageable because the orbit relation for the symmetry group is never explicitly constructed.

Let Rep be the set of representative. Let $\xi \subseteq S \times$ Rep be the representative relation. We are assuming the most general setting where there can be multiple representatives from one orbit and a state can be related by ξ to more than one representative from its orbits. All the basic modalities of CTL (EFp, EGp, and $E(qUp)$) can be expressed as fix-points using the modality EX. The fix-point equations are described below and the correctness of the equations is demonstrated in [5].

$$\begin{aligned} \mathrm{EF}p &= \mu Y \cdot (p \vee \mathrm{EX}Y) \\ \mathrm{EG}p &= \nu Y \cdot (p \wedge \mathrm{EX}Y) \\ E(qUp) &= \mu Y \cdot (p \vee (q \wedge \mathrm{EX}Y)) \end{aligned}$$

It is clear from the above equations that it is enough to give the semantics for EXf. Let Im_ξ be the forward image under the representative relation, and Im_R^{-1} be the pre-image under the transition relation of the original structure $M = (S, R, L)$. The set of representatives satisfying EXf is $\mathrm{Im}_\xi(\mathrm{Im}_R^{-1}(K))$, where K is the set of representatives satisfying the state formula f. Let us consider an example. Consider the formula EFp such that p is an atomic proposition and G is an invariance group for p. Let $K_0 \subseteq$ Rep be the set of representatives such that p labels these representatives. Let K_i be the set of representatives at the ith iteration. The equations given below describe the set of representatives which satisfy the formula *EFp*.

$$\begin{aligned} K_0 &= \{r \mid r \in \mathrm{Rep}\,\mathrm{and}\; p \in L(r)\} \\ K_{i+1} &= \mathrm{Im}_\xi\big(\mathrm{Im}_R^{-1}(K_i)\big)\;\mathrm{for}\; i \geq 0 \end{aligned}$$

If K is a set of representatives, then $\mathrm{Im}_\xi(\mathrm{Im}_R^{-1}(K_i))$ again is a set of representatives. Therefore, in this new model checking algorithm we always maintain only subsets of representatives which can result in substantial savings. This section proves the correctness of this model checking procedure by placing some restrictions on the representative relation ξ. These conditions are quite general and we believe that they hold in most practical cases. There is an implicit assumption that for each orbit $\theta(s)$ of S under G there exists $r \in$ Rep such that $\theta(s) = \theta(r)$, i.e., every orbit is represented.

Definition 8.1. Let G be a symmetry group for a Kripke structure $M = (S, R, L)$. Let $\xi \subseteq S \times$ Rep be the representative relation such that $(s, r) \in \xi$ implies that $\Theta(s) = \Theta(r)$. Let $C \subseteq G$ be a subset of permutations. The set C is called *complete* for ξ if and only if

- The condition $(s, r) \in \xi$ implies that $\exists(\sigma \in C)$ such that $\sigma s = r$.
- For all $r \in$ Rep and $\sigma \in C$ we have that $(\sigma r, r) \in \xi$.

The intuition for this definition will become clear when we prove Lemma 8.1. It allows us to translate a path in the quotient model M_G to a corresponding path in the model M_ξ which is defined below.

Definition 8.2. Given a Kripke structure $M = (S, R, L)$ and a representative relation $\xi \subseteq S \times \text{Rep}$ define $M_\xi = (\text{Rep}, R_\xi, L_\xi)$ as follows:

$$\begin{aligned} R_\xi &= \{(r_1, r_2) \mid \exists(s \in S)((s, r_1) \in \xi \wedge (s, r_2) \in R)\} \\ L_\xi(r) &= L(r) \end{aligned}$$

Notice that $\text{Im}_{R_\xi} = \text{Im}_R \circ \text{Im}_\xi^{-1}$ and $\text{Im}_\xi^{-1} = \text{Im}_\xi \circ \text{Im}_R^{-1}$. Therefore, the pre-image of a set of representatives K in the structure M_ξ is $\text{Im}_\xi(\text{Im}_R^{-1}(K))$. Hence, proving that the model checking procedure described at the beginning of the section is correct is equivalent to proving that M_ξ and M satisfy the same invariant CTL* formulas. Since we have already proved the equivalence between M and the quotient model M_G, it suffices to prove the equivalence between M_ξ and M_G.

Lemma 8.1. *If ξ has a complete set of permutations $C \subseteq G$, then the corresponding path theorem holds for M_ξ and M_G.*

Proof: Let $\pi_\xi = (r_0, r_1, \ldots)$ be a path in M_ξ. From the definitions it is easy to see that $(r_1, r_2) \in R_\xi \Rightarrow (\Theta(r_1), \Theta(r_2)) \in R_G$, so $\pi_G = (\Theta(r_0), \Theta(r_1), \ldots)$ is a path in M_G.

Now we handle the other direction. Let $\pi_G = (\Theta(s_0), \Theta(s_1), \ldots)$ be a path in M_G. Let r_0 be an arbitrary representative from the orbit $\Theta(s_0)$. Since G is a symmetry group for M and $(\Theta(s_0), \Theta(s_1)) \in R_G$, there exists $t_1 \in \Theta(s_1)$ such that $(r_0, t_1) \in R$. Let $r_1 \in \text{Rep}$ such that $(t_1, r_1) \in \xi$. Since C is a complete set for ξ, there exists $\sigma \in C$ such that $\sigma t_1 = r_1$. Since G is a symmetry group, $(\sigma r_0, \sigma t_1) \in R$. Since C is a complete set for ξ, $(\sigma r_0, r_0) \in \xi$. By definition of R_ξ, $(r_0, r_1) \in R_\xi$. Continuing the argument we can show that for every natural number i there exists $r_i \in \text{Rep}$ such that $\Theta(r_i) = \Theta(s_i)$ and $(r_i, r_{i+1}) \in R_\xi$. The path $\pi_\xi = (r_0, r_1, \ldots)$ is a corresponding path in M_ξ. □

Theorem 8.1. *Let $M = (S, R, L)$ be a Kripke Structure, G be a symmetry group of M, and h be a CTL* formula. Let $\xi \subseteq S \times \text{Rep}$ be a representative relation and $C \subseteq G$ be the corresponding complete set. If G is an invariance group for all the atomic propositions p occurring in h, then*

$$M_G, \theta(s) \models h \Leftrightarrow M_\xi, \theta(s) \models h \tag{4}$$

where M_G is the quotient structure corresponding to M and M_ξ is defined above.

Proof: The proof of theorem is very similar to that of Lemma 4.3 using the lemma proved above. □

Theorem 8.2. *Let G be a permutation group acting on a finite set S. Let H be a sub-group of G. Let the set of representatives $\text{Rep} \subseteq S$ be the union of some orbits of S under H. Given the conditions above these exists a representative relation ξ and a corresponding complete set C.*

Proof: Let $G = H + H\psi_1 + \cdots + H\psi_r$ be the complete right traversal of H in G (each ψ_i is in a different right coset of $G \backslash H$). Define $C = \{e, \psi_1, \psi_1^{-1}, \ldots, \psi_r, \psi_r^{-1}\}$ (e is the identity permutation). We define the representative relation as follows: $(s, r) \in \xi$ if and only if there exists $\sigma \in C$, $r \in \text{Rep}$, and $\sigma s = r$. The first condition for C to be complete for ξ follows from the definition. We now prove that the second condition holds. Consider σr for $\sigma \in C$. Since $\sigma^{-1} \in C$, we have that $(\sigma r, r) \in \xi$. Hence, C is a complete set for ξ, but we have to prove a *consistency* condition, i.e., for every $s \in S$ there exists $r \in \text{Rep}$ such that $(s, r) \in \xi$ and $\Theta(s) = \Theta(r)$. The condition states that every state is related to some representative in its orbit.

The way we have defined ξ means that proving *consistency* is equivalent to proving that for all $s \in S$ there exists a $\sigma \in C$ such that $\sigma s \in \text{Rep}$. Notice that since $C \subseteq G$, for all $\sigma \in C$ the states σs and s are in the same orbit of S under G. Let $\Theta(s)$ be the orbit of $s \in S$ under G. Since H is a sub-group of G, the orbits of S under H are a refinement of orbits of S under G. Let $\theta_H \subseteq \Theta(s)$ be an orbit of S under H such that $\theta_H \subseteq \text{Rep}$. Let $r \in \theta_H$ be an arbitrary representative. Since r and s belong to the same orbit of S under G, there exists $\sigma \in G$ such that $\sigma s = r$. Using the right traversal of H in G we can write $\sigma = \sigma_1 \psi$, were $\sigma_1 \in H$ and $\psi \in C$. Since $\sigma s = r$, $\psi s = \sigma_1^{-1} r$. Since $\sigma_1^{-1} \in H$, $\sigma_1^{-1} r \in \theta_H$. Therefore $\psi s \in \text{Rep}$. □

The theorem given above assumes that Rep is the union of some orbits of S under H. Therefore, if the orbits of H are large, then the set of representatives could be large. In practice, Rep is provided and then the group H is determined. Suppose the state set S is given by the assignment to n boolean state variables $x_1, \ldots, x_n$. Let Rep be those states such that $x_1 = 1$. If G is the symmetry group of the underlying structure, then G^1 (the subgroup of G which fixes the index 1) serves the purpose of H in the theorem given above. This fact will be shown later. Hence, in some ways the choice of H is fixed by the choice of Rep.

A permutation σ acting on the set S is said to *stabilize* a set $Y \subseteq S$ iff the following condition holds:

$$\forall (y \in Y)(\sigma(y) \in Y)$$

Notice that the condition given above is equivalent to the condition $\sigma(Y) = Y$. Let G be a permutation group acting on the set S. Given $Y \subseteq S$, the subgroup G_Y of G is defined as follows:

$$G_Y = \{\sigma \mid (\sigma \in G) \wedge (\sigma(Y) = Y)\}$$

Now we prove a useful generalization of the theorem given above.

Theorem 8.3. *Let $M = (S, R, L)$ be a Kripke structure and G be its symmetry group. Let $Rep \subseteq S$ be the set of representatives. Let H be a subgroup of G such that for all $\sigma \in H \sigma(Rep) = Rep$ (notice that that this is another way of stating that Rep is the union of some orbits of S under H), i.e., H stabilizes Rep. Let $C \subseteq G$ be a set which satisfies the following conditions*

1. *For each coset of $G \backslash H$ we have a permutation $\psi \in C$ which belongs to that coset.*
2. *The set C is inverse closed, i.e., $\psi \in C$ implies that $\psi^{-1} \in C$.*

Let the representative relation ξ be defined as follows: $(s, r) \in \xi$ iff *$s \in S$, $r \in Rep$ and there exists a $\psi \in C$ such that $\psi s = r$. Then ξ is valid representative relation and C is the corresponding complete set.*

Proof: It is exactly same as the Proof of Theorem 8.2. □

In the theorem given below S is the set of states given by assignments to the boolean variables $x_1, x_2, \ldots, x_n$. Each state is a $0 - 1$ vector of size n.

Theorem 8.4. *Let the set of representatives Rep be given by the propositional formula $p(x_1, x_2, \ldots, x_n)$, i.e., a state $s = (y_1, \ldots, y_n)$ is a representative iff $p(y_1, \ldots, y_n) = 1$. Let G be the symmetry group for the Kripke structure $M = (S, R, L)$. Given these assumptions there exists a representative relation $\xi \subseteq S \times Rep$ and a corresponding complete set C.*

Proof: Let G_p be the invariance group of the propositional formula p. Let $H = G_p \cap G$. We prove that if $s \in S$ is a representative (this means that $p(s) = 1$), then $\Theta_H(s) \subseteq$ Rep($\Theta_H(s)$ is the orbit of s under H). Since every permutation $\sigma \in H$ is an invariant for p, $p(\sigma s) = 1$ for all $\sigma \in H$. Hence $\Theta_H(S) \subseteq$ Rep. Therefore, Rep is the union of orbits of H. Now we can use Theorem 8.2 to get a representative relation ξ and a complete set C. □

We give examples illustrating the utility of the result.

Example 8.1. Let $x_1, x_2, \ldots, x_n$ be the list of boolean state variables. Let G be a permutation group acting on the set $\{1, 2, \ldots, n\}$. G induces a permutation group $B(G)$ on the set $\{0, 1\}^n$, but we will work with G. Quite frequently representatives are given by an assignment to variables whose index is in a certain set $Y \subseteq \{1, 2, \ldots, n\}$. For example, $Y = \{1, 2\}$ and $x_1 = 0, x_2 = 1$ might describe the representatives (in this case the proposition $\bar{x}_1 \wedge x_2$ describes the representatives). Let $\mathcal{A}$ be the assignment to the variables x_i such that $i \in Y$. The assignment $\mathcal{A}$ defines the set of representatives Rep. We use $\mathcal{A}_i$ to denote the value of the variable $x_i (i \in Y)$ in the assignment. Let $Y_1 = \{i \mid \mathcal{A} = 1\}$ and $Y_0 = Y - Y_1$. In this case H (the invariance group of the proposition describing the representatives) is $(G_{Y_1})_{Y_0}$. Intuitively, all variables which are assigned the same value by the assignment $\mathcal{A}$ can be permuted freely. Now we can use Theorem 8.4 to get a representative relation ξ and a corresponding complete set C.

Example 8.2. We give an even more concrete example. Take the cache-coherence protocol with n processes. Each process has k local variables. The variables $x_{k(i-1)+1}, \ldots, x_{k(i-1)+k}$ are the local variables corresponding to process i. Let $x_{k(i-1)+1}$ correspond to the variable which indicates whether process i is the master, i.e., $x_{k(i-1)+1} = 1$ means process i is the master. Assume that we can switch the context of processes i and j, which corresponds to the permutation.

$$\sigma_{ij} = (k(i-1)+1, k(j-1)+1) \cdots (k(i-1)+k, k(j-1)+k)$$

The symmetry group G of the Kripke structure M is generated by σ_{1i} $(2 \leq i \leq n)$. Suppose we choose the set of representatives as the states where process 1 is the master, i.e., $x_1 = 1$. The invariance group H of the proposition x_1 is G^1, where G^1 is the subgroup of G which fixes in index 1. Two permutations σ_1 and σ_2 are in the same coset of $G \backslash G^1$ iff $\sigma_1(1) = \sigma_2(1)$. Notice that for $k \neq j$ σ_{1j} and σ_{1k} lie in different coset of $G \backslash G^1$ because they map 1 to different positions. Therefore, the complete right traversal of G^1 in G is given by the following equation.

$$G = G^1 + G^1\sigma_{12} + \cdots + G^1\sigma_{1n}$$

Using Theorem 8.3 we get a representative relation ξ and the complete set $\{e, \sigma_{12}, \ldots, \sigma_{1n}\}$ (notice that in this case $\sigma_{1i}^{-1} = \sigma_{1i}$).

The next lemma will be used to prove the correctness of our experiments performed on a simple version of the Futurebus + cache-coherence protocol. The experimental results are given in the next section. First, we extend the definition of the orbit of a state to a orbit of a set of states. Orbit of a set $Y \subset \{1, 2, \ldots, n\}$ is defined as follows:

$$\Theta(Y) = \{Y' \mid \exists(\sigma \in G)(\sigma(Y) = Y')\}$$

The group G acts on the set $\{1, 2, \ldots, n\}$.

Lemma 8.2. *Let G, Y, Y_0, Y_1 be as in Example* 8.1. *Let $C \subseteq G$ be a set which is inverse closed and such that for every pair of sets $Y_0' \in \Theta(Y_0)$ and $Y_1' \in \Theta(Y_1)$ there exists a $\sigma \in C$ such that $\sigma(Y_0) = Y_0'$ and $\sigma(Y_1) = Y_1'$. Let $\xi \subseteq S \times Rep$ be defined as follows: $(s, r) \in \xi$ iff there exist $\sigma \in C$ such that $\sigma(s) = r$. In this case ξ is a valid representative function and C is the corresponding complete set.*

Proof: Let $H = (G_{Y_1})_{Y_0}$ be as defined in Example 8.1. Two permutations σ_1 and σ_2 of G are in the same coset of H iff $\sigma_1(Y_0) = \sigma_2(Y_0)$ and $\sigma_1(Y_1) = \sigma_2(Y_1)$. Therefore, C has a permutation from every coset of $G \backslash H$ (the argument is very similar to the one given in the example before). Applying Theorem 8.2 we get the result. □

9. Empirical results

To test our ideas we have chosen a simple cache coherence protocol for a single-bus multiprocessor system based on the Futurebus + IEEE standard [13]. The verification of a more detailed version of the protocol with multiple buses is described in [6]. The system has a bus over which the processors and the global memory communicate. Each processor contains a local cache which consists of a fixed number of cache lines (see figure 4).

In each bus cycle the bus arbiter chooses one processor to be the master. The master processor selects a cache line address and a command it wants to put on the bus. The other processors and the memory respond to the bus command and change their local context.

The reaction of the components is described in the protocol standard, which enforces the coherence of the cache lines among the different processors, i.e., only valid data values

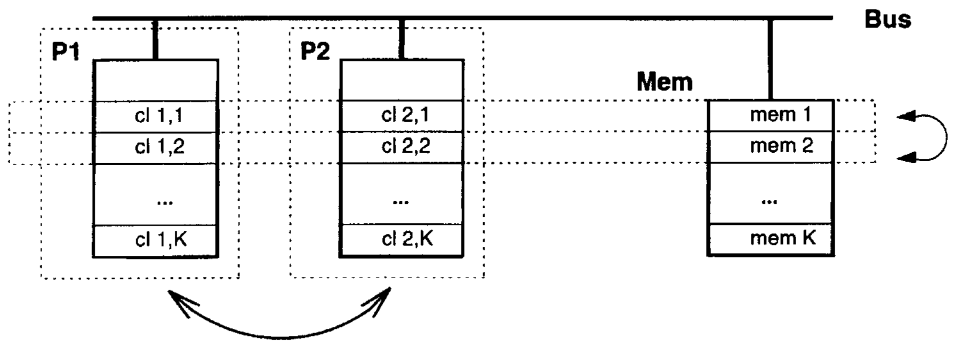

Figure 4. System structure.

are read by the processors and no writes are lost. For the verification task the protocol is formalized, and cache coherence and other important system properties are expressed in temporal logic.

The behavior of the processors, the bus and the memory can be described by finite state machines. The state of the processor P_i is a combination of the states of each cache line in the processor cache and the state of the bus interface. The global bus is represented by the command on the bus, the active cache line address and other bus control signals, e.g., for bus snooping and arbitration.

There are two obvious symmetries in the system. First, processors are symmetric, i.e., we can exchange the context of any two processors in the system. Second, cache lines are symmetric, i.e., any two cache lines can be exchanged simultaneously in all processors and the memory. To maintain consistency, along with applying the symmetries mentioned above all the cache lines and processor addresses in the system must be renamed. Both symmetries are indicated in figure 4 by arrows.

The complete system is the synchronous composition of all the components and is described by a Kripke structure $M = (S, R, L)$. Since domains can be encoded in binary, a state is just a binary vector, and the transition relation R can be represented by a BDD. We experimented with two variable ordering which we call "concatenation" and "interleaving". The concatenation ordering is simply $P_1 \prec P_2 \prec \cdots \prec P_N$. The variables of processor i are ordered before the variables of processor $i+1$. In the interleaved ordering the processor variables are interleaved, i.e., $p_{1,1} < p_{2,1} < \cdots < p_{N,1} < p_{2,2}, \ldots,$ where $p_{i,1}, \ldots, p_{i,K}$ are the state variables of processor P_i. The variables of the bus and the memory are ordered before all other variables in both orderings. In both orderings, each next state variable is placed immediately after the corresponding state variable.

Throughout this discussion we use N to denote the total number of processors and M to denote the total number of cache lines. Let π_{1j} be the permutation which exchanges the context of processor 1 with processor j. Let ψ_{1k} be the permutation which exchanges cache line 1 with cache line k. The symmetry group which uses processor (cache) symmetry alone is generated by the set of permutations $\Pi = \{\pi_{1j} \mid 1 \leq j \leq N\}(\Psi = \{\psi_{1k} \mid 1 \leq k \leq M\})$. The symmetry group which uses processor and cache symmetry is generated by the permutations $\Pi \cup \Psi$. When we used only processor (cache) symmetry, we used as the set of representatives the states where processor 1 is the master (cache line 1 is active). When we used both symmetries we used the set of states where processor 1 is the master and cache line 1 is active as the set of representatives. The representative relation for all the three cases is given below:

1. In the case of processor symmetry $(s, r) \in \xi_p$ iff there exists π_{1j} $(1 \leq j \leq N)$ such that $\pi_{1j}(s) = r$.
2. In the case of cache symmetry $(s, r) \in \xi_c$ iff there exists ψ_{1k} $(1 \leq k \leq M)$ such that $\pi_{1j}(s) = r$.
3. In the case when both symmetries are used $(s, r) \in \xi_{pc}$ iff there exists $1 \leq j \leq N$ and $1 \leq k \leq M$ such that $\psi_{1k}(\pi_{1j}(s)) = r$.

We will prove that the set $C = \{\psi_{1k} \circ \pi_{1j} \mid k \leq M \wedge j \leq N\}$ is complete for the representative relation ξ_{pc}. The cases when we use only cache and processor symmetry are very similar to Example 2 in the section on multiple representatives. Let master$_i$ be the index of the variable which tells that processor i is the master, i.e., $x_{\text{master}_i} = 1$ means processor i is that master. Let active$_j$ be the index which indicates that cache line j is active. Using the notation of Lemma 8.2 we have that $Y_1 = \{\text{master}_1, \text{active}_1\}$. The orbit of Y_1 is the set of pairs of indices of the form $\{\text{master}_j, \text{active}_k\}$. Consider a typical element $Y' = \{\text{master}_j, \text{active}_k\}$ in $\Theta(Y_1)$. It is clear that $\psi_{1k}(\pi_{1j}(Y_1)) = Y'$. Therefore, by Lemma 8.2 C is a complete set for ξ_{pc}.

Consider the following properties which can be represented by a propositional formula:

1. Property p is that for all cache lines it is true that if one processor is in EM (exclusive modified) state, then all other processors are in I (invalid) state.
2. Property q is that for all cache lines it is valid that if memory has valid data, then either all processors are in SU (shared unmodified) or I (invalid) state or one processor is in EU (exclusive unmodified) state.
3. Property m asserts that all cache lines in memory are valid.
4. Property c says that the command on the bus is either *read-modified* or *invalidate*.

We checked that the initial state doesn't belong to EF$\neg p$ and EF$\neg q$. These properties could be checked by doing reachability, but in order to test our theory we check them by computing EF$\neg p$ because this involves finding preimages. We also tested that the initial state satisfies the property AG$(m \rightarrow A(mUc))$ which asserts that if memory has valid data then it remains valid until an appropriate command is issued. This property turned out be false because there are reachable states where m is true and there exists a path where the command is never *read-modified* or *invalidate*. We also tested that from all the reachable states it is possible to get to a state where memory has valid data for all the cache lines, i.e., we checked that initial state satisfies the property AG(EFc). The BDD sizes for the property EF$\neg q$ were the largest. We ran the experiments with various system configurations. The results are summarized below. The BDD sizes in the case of the interleaved ordering were much smaller than the BDD sizes in the case of the concatenation ordering. Therefore, in the table given below we have only included the data for the interleaved ordering. The results are listed in Table 1. Each row *jPkC* in the table gives the results for a configuration of j processors

Table 1. Empirical results.

	Trans. relation	No symmetry		Symmetry					
				Processors		Cache lines		Combination	
System config.	BDD nodes	BDD nodes	Time sec.	BDD nodes	Time sec.	BDD nodes	Time sec.	BDD nodes	Time sec.
2P2C	631	920	2	668	1	518	1	368	1
4P2C	8,534	6,048	11	2,573	4	2,855	6	1,309	3
2P4C	1,519	6,166	36	3,917	18	1,458	9	1,178	6
4P4C	22,154	42,595	231	14,626	62	6,831	47	4,266	27
2P8C	3,295	17,446	756	10,837	407	2,618	152	2,338	98
4P8C	49,394	121,475	5,911	40,466	1,400	11,551	678	8,986	424
2P12C	5,071	28,726	5,136	17,757	2,884	3,778	841	3,498	577
4P12C	76,686	—	—	—	—	16,271	3,808	13,706	2,300

and k cache lines. The first column gives the number of BDD nodes for representing the transition relation. The columns after that gives the results for model checking of the properties described above. First, no symmetry was used. Next, we give the results for symmetry between processors and symmetry between cache lines. In the last case, we used the combination of both symmetries. In each case the size of the largest intermediate BDD (number of BDD nodes) encountered while checking the four temporal properties and the cpu time used are listed. All experiments were run on a Sun Sparc10 workstation and the size of the largest BDD gives a tight bound for the maximal memory usage.

Exploiting the symmetry between processors or cache lines reduces the BDD size by a factor that is linear in the number of processes or cache lines. The combination of these two symmetries reduces the size of the largest BDD by the product of the number of processors and cache lines. This is due to the fact that the two symmetries are independent. So by exploiting symmetry the memory usage is reduced considerably, e.g., by a factor of 13.5 in the case of 4 processors and 8 cache lines. The cpu times are not reduced by the same factor. For exploiting the symmetry we need additional time for the mapping of states onto the representatives after each pre-image step in the model checking procedure. In the future, we hope to improve the efficiency of the mapping computation.

10. Bisimulation experiments

A relation $B \subseteq S \times S$ is called a *bisimulation relation* for a Kripke structure $M = (S, R, L)$ iff $(s, s_1) \in B$ implies that:

- for all s' such that $(s, s') \in R$ there exists a s'_1 such that $(s_1, s'_1) \in R$ and $(s', s'_1) \in B$

and an equivalent condition holds for s_1.

Given a symmetry group G for the Kripke structure $M = (S, R, L)$ the orbit relation induced by G is a bisimulation relation. A natural question is, why we don't use bisimulation minimization algorithms to obtain a quotient structure?

Bisimulation minimization algorithms require that a transition graph must be constructed (partially in "on-the-fly" algorithms), which is then minimized using the fixpoint characterization of bisimulation. In contrast symmetry can be already used during the construction of the transition graph (see Algorithm 2), and therefore even infinite systems can be handled, if the quotient structure is finite.

The orbit relation corresponding to a symmetry group is not the largest bisimulation, i.e., two orbits might be bisimular and will be merged by the coarsest bisimulation. So the quotient structure obtained by symmetry has more states than the one obtained using bisimulation minimization. However when using BDDs the number of states is not the crucial parameter, but the "structure" of the state space is. For example, using multiple representatives we increased the number of representatives, but got much better performance when working with BDDs. Similarly, our experiments described below indicate that we get better reductions using symmetry (with multiple representatives) instead of bisimulation minimization.

Let R be the transition relation of a Kripke structure and E an equivalence relation on the set of states. B_E is the largest bisimulation contained in E and can be computed by the following fixpoint computation:

```
$B_0(p, q) := E(p, q)$
**repeat**
    $B_{i+1}(p, q) := \forall p' : (R(p, p') \rightarrow \exists q' : (R(q, q') \wedge B_i(p', q'))) \wedge$
                     $\forall q' : (R(q, q') \rightarrow \exists p' : (R(p, p') \wedge B_i(p', q')))$
**until** $B_i == B_{i+1}$
$B_E := B_i$
```

When implementing such an iterative scheme using BDDs the variable ordering is very important. There are two variable orderings to consider:

- The variable ordering of the p variables (similarly q variables). This ordering was the *interleaved ordering* defined in Section 8.
- The variable ordering of q variables with respect to the p variables.

We performed our experiments with three different orderings between p and q: *concatenated*, i.e., all p variables precede all q variables; *interleaved*, i.e., $p_0, q_0, p_1, q_1, \ldots$; and *reordered* when we used dynamic variable reordering [18, 8] during the iteration.[1] In Table 2 the column *bisim order* denotes the used order. Experiments with the interleaved ordering always behaved much worse with respect to time and space requirements than the concatenated ordering. Therefore, we have only included the data for the concatenated ordering and reordering in the table.

Results for various configurations of the cache-coherence protocol are listed in the following table. The computations were aborted if a specified time bound was exceeded. The column *inv* denotes the set used for E. The symbol *none* means that E = true and *inv* means that two states are in E iff they agree on the property *conflict*, which is true if a cache

Table 2. Bisimulation results.

Example	Inv	Reach (sec.)	Bisim. order	Time (sec.)	Nodes
2p2c	inv	no	c	5.4	935
2p2c	inv	0.5	c	1.2	506
2p2c	inv	0.6	r	3.5	405
2p2c	none	no	c	0.9	180
2p2c	none	0.6	c	1.1	506
2p2c	none	0.6	r	3.5	405
2p4c	inv	no	c	—	—
2p4c	inv	1.4	c	7.7	1986
2p4c	inv	1.5	r	21.8	1203
2p4c	none	no	c	4.0	344
2p4c	none	1.5	c	7.8	1986
2p4c	none	1.5	r	21.9	1203
4p2c	inv	no	c	—	—
4p2c	inv	2.2	c	—	—
4p2c	inv	2.0	r	—	—
4p2c	none	no	c	10.6	528
4p2c	none	2.1	c	10.9	1904
4p2c	none	2.1	r	40.0	1484

line exists, for which the caches of two different processors are in conflict, e.g., both are in "exclusive-modified" states. Notice that the property *conflict* is basically the property $\neg p$ where p is defined in Section 8. We experimented with computing the reachable states in advance and performing bisimulation minimization only for the reachable states. For these experiments column *reach* gives the time for reachability computation. If reachability computation was not done, there is a *no* in that column. The column *time* gives the time for the bisimulation iteration (without reachability) and *nodes* the number of BDD nodes for the BDD for the relation B_E. For all larger configurations, like 4p4c, not listed in the table the time bound was always exceeded.

In summary, we can see from the experiments that in this example bisimulation minimization is more complex than performing model checking using multiple representatives. Using symmetry we showed that model checking can be made more efficient. So we conclude that in our class of applications exploiting symmetry is superior to performing bisimulation minimization.

11. Directions for future research

There are a number of directions for future research. Perhaps the most interesting (and difficult) problem is to determine the exact complexity of the orbit computation. This problem seems fundamental with many applications other than verification. It may be possible to show that the problem has exactly the same complexity as the graph isomorphism problem or even that it is NP-complete.

It would also be nice to have tight lower bounds on the size of the BDDs needed for the orbit relation for symmetry groups occurring in practice. This type of information would be useful in determining if it is feasible to construct the quotient model directly using the orbit relation or whether it is necessary to develop special techniques (like the *approximation procedure* in Section 8) for mapping states onto representatives.

An automatic procedure for identifying symmetries in circuits would definitely be useful. Techniques based on the Walsh transform have been tried for this purpose in the past [12]. However, we suspect that this will always be a hard problem and that some information from the designer of the circuit will usually be required.

We also intend to try other hardware examples with more complicated topologies in addition to cache coherency protocols.

Acknowledgments

We would like to thank David Long and Ken McMillan for their help in writing this paper. We would also like to thank Manish Pandey for giving us many helpful comments.

Note

1. We used window and sifting techniques in the order algorithms.

References

1. M. Browne, E. Clarke, and O. Grumberg, "Characterizing finite Kripke structures in propositional temporal logic," *Theoretical Comput. Sci.*, Vol. 59, pp. 115–131, 1988.
2. R.E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Comput.*, Vol. C-35, No. 8, 1986.
3. J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and J. Hwang, "Symbolic model checking: 10^{20} states and beyond," in *Proc. 5th Ann. Symp. on Logic in Comput. Sci.*, IEEE Comp. Soc. Press, June 1990.
4. L. Claesen (Ed.), *Proc. 11th Int. Symp. on Comput. Hardware Description Lang. and their Applications*, North-Holland, Apr. 1993.
5. E.M. Clarke, E.A. Emerson, and A.P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Trans. Prog. Lang. Syst.*, Vol. 8, No. 2, pp. 244–263, 1986.
6. E.M. Clarke, O. Grumberg, H. Hiraishi, S. Jha, D.E. Long, K.L. McMillan, and L.A. Ness, "Verification of the Futurebus + cache coherence protocol," to appear in *Proc. 11th Int. Symp. on Comput. Hardware Description Lang. and their Applications*, Apr. 1993.
7. E.A. Emerson and A.P. Sistla, "Symmetry and model checking," in *Proc. Fifth Workshop on Comput.-Aided Verification*, C. Courcabetis (Ed.), June 1993.
8. E. Felt, G. York, R. Brayton, and A.S. Vincentelli, "Dynamic variable reordering for bdd minimiation," in *Proc. EuroDAC*, pp. 130–135, Sept. 1993.
9. M. Furst, J. Hopcroft, and E. Luks, "Polynomial-time algorithms for permutations groups," in *Proc. 21st Ann. Symp. on Found. of Comput. Sci.*, 1980.
10. M. Garey and D. Johnson, *Computers and Intractibility*, W.H. Freeman and Company, 1979.

11. P. Huber, A. Jensen, L. Jepsen, and K. Jensen, "Towards reachability trees for high-level Petri nets," in *Advances on Petri Nets*, G. Rozenberg (Ed.), pp. 215–233, 1984.
12. S.L. Hurst, D.M. Miller, and J.C. Muzio, *Special Techniques in Digital Logic*, Academic Press, Inc., 1985.
13. IEEE Computer Society, *IEEE Standard for Futurebus +—Logical Protocol Specification*, Mar. 1992. IEEE Standard 896.1–1991.
14. C. Ip and D. Dill, "Better verification through symmetry," to appear in *Proc. 11th Int. Symp. on Compuct. Hardware Description Lang. and their Applications*, Apr. 1993.
15. R.P. Kurshan, "Testing containment of ω-regular languages," Technical Report 1121-861010-33-TM, Bell Laboratories, 1986.
16. B. Lin and A.R. Newton, "Efficient symbolic manipulation of equivalence relations and classes," in *Proc. 1991 Int. Workshop on Format Methods in VLSI Design*, Jan. 1991.
17. K.L. McMillan and J. Schwalbe, "Formal verification of the Gigamax cache consistency protocol," in *Shared Memory Multiprocessing*, N. Suzuki (Ed.), MIT Press, 1992.
18. R. Rudell, "Dynamic variable reordering for ordered binary decision diagrams," in *Proc. IEEE ICCAD*, pp. 42–47, Nov. 1993.
19. P. Starke, "Reachability analysis of petri nets using symmetries," *Syst. Anal. Model. Simul.*, Vol. 8, Nos. 4/5, pp. 293–303, 1991.