Exploiting Symmetry In Temporal Logic Model Checking *

E. M. Clarke[1] and T. Filkorn[2] and S. Jha[1]

[1] Carnegie Mellon University, Pittsburgh, PA 15213
[2] Siemens AG, Corporate Research and Development, Otto-Hahn-Ring 6, W-8000 Muenchen 83, Germany

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 [13, 10, 17]. However, this research did not consider arbitrary temporal properties or the complications that arise when *BDD*s 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.

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. We are trying to exploit symmetry to reduce the size of the state space that must be explored by temporal logic model checking algorithms.

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 transition relation among the states, and L is a function that labels states with sets of

* 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.

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 $rep(\theta(s))$.

If $M = (S, R, L)$ is a Kripke Structure and G is 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 $S_G = \{\theta(s) | s \in S\}$, 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 $L_G(\theta(s)) = L(rep(\theta(s)))$.

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 the original state space S. We have developed techniques that build M_G without actually building M. We believe that our method will reduce the state space that must be searched considerably and are currently testing 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 [16, 6]. 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. The initial results that we have obtained are encouraging. 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 straightforward 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 approximating the quotient structure 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 [10, 17]. However, this work does not consider general temporal properties nor the complications that are caused by representing the state space using *BDDs*. The research that is closest to our own is that of Ip and Dill [13] who propose a data type ***scalarset*** which facilitates detection of symmetry in finite state systems. Their technique uses an explicit state representation rather than *BDDs*. Our results were obtained independently of their work; in fact, they reference some of our early results in their paper.

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 Section 6 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. 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 $(s_1, s_2) \in R$).
- $L : S \rightarrow 2^{AP}$ is a *labeling function* which associates with each state a set of atomic propositions that are true in the state.

Let G be a group of permutations, 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* for 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:

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

Let $\langle g_1, \ldots, g_k \rangle$ be the smallest permutation group containing all of 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.

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. $\pi \models Xg_1 \Leftrightarrow \pi^1 \models g_1$
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 $\neg f$.

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 $rep(\theta(s))$.

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

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

– The labeling function L_G is given by $L_G(\theta(s)) = L(rep(\theta(s)))$.

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 2. 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.

Definition 3. 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. 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.

Theorem 5. 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 6. 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.

5 Model Checking in the Presence of Symmetry

In this section we describe how to do model checking in the presence 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 [13]. 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. 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 $g_1, g_2, \cdots, g_r$, it is possible to prove that the orbit relation Θ is the least fixpoint of the equation given below:

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

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 $\xi : S \rightarrow S$, 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 [15]. 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 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 i-th 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 $\{1, 2, \ldots, n\}$, then σ acts on vectors in B^n in the following manner:

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

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 $\{1, 2, \cdots, n\}$ 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 7. Let G be a group acting on the set $\{1, 2, \cdots, n\}$. 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 8. 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 $f : V_1 \rightarrow V_2$ such that the following condition holds

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

Theorem 9. The *orbit problem* is as hard as the *Graph Isomorphism* problem.

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

Definition 10. Given a group G generated by r permutations $g_1, g_2, \cdots, g_r$ (the permutations act on the set $\{1, 2, \cdots, n\}$), 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 $\sigma = g_{i_1} g_{i_2} \ldots g_{i_m}, m \leq k$

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 from *EXACT COVER BY 3-SETS* [9].

Theorem 11. The ***bounded orbit problem*** is NP-complete.

Permutations $g_1, g_2, \ldots, g_r$ acting on the set $\{1, 2, \ldots, n\}$ are ***disjoint*** if and only if no element is moved by more than one permutation $g_1, \ldots, g_r$. If we restrict the set of generators of a group G to be disjoint, the orbit problem can be solved in polynomial time.

Theorem 12. Given a group G generated by disjoint generators $g_1, g_2, \cdots, g_k$ and two vectors $x \in B^n$ and $y \in B^n$, it can be decided in polynomial time whether there exists a permutation $\sigma \in G$ such that $y = \sigma x$.

Circuits are typically built from components and the state bits are grouped according to the hierarchical structure of the system. Because of this two types of symmetry groups occur frequently in practice:

- ***Rotation groups*** occur when equivalent components are ordered in a ring and can be rotated any number of steps. For example, the token ring protocol used in the solution to the distributed mutual exclusion problem exhibits rotational symmetry.
- ***Full symmetric groups*** occur when equivalent components are unordered and can be exchanged arbitrarily. Such groups 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.

For these two classes of symmetry groups we give results on the complexity of the BDD representations for the orbit relation.

Theorem 13. Let the state of a system be composed of N equivalent components each with k state variables. For a full symmetric permutation group G acting on the set $\{1, \ldots, N\}$ we have the following lower bound for the BDD representing the induced orbit relation Θ.

$$|\Theta| > 2^{K/8} \text{ with } K = \min(N, 2^k)$$

Theorem 14. Let the state of a system be composed of N equivalent components each with k state variables. For a rotation group G acting on the set $\{1, \ldots, N\}$ we have the following lower bound for the BDD representing the induced orbit relation Θ.

$$|\Theta| > 2^K \text{ with } K = \min(\sqrt{N}, 2^{k-1} - 1)$$

The BDD of the orbit relation induced by a full symmetric or rotation 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 in 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 7.

7 Approximating the Orbit Relation

In the approach described in Section 5, a function ξ was computed that mapped each state in an orbit to its unique representative. There are interesting symmetries where the *BDD* representation of the orbit relation is too big to compute. Although selecting unique representatives reduces the number of states in the quotient model, this is not the primary goal when using *BDDs*. Instead we want to choose the representatives so that the *BDDs* will be small. For this reason we have developed a method which uses an approximation of the quotient model.

Instead of having only one representative for an orbit we allow a set of representatives. Hence, we are using the set of representatives as an approximation to an orbit. For this purpose any subset of the orbit can be used, instead of only a unique representative as in Section 5 or the whole orbit as in the case of symbolic model checking without using symmetry. As an example consider a system of N processors where a specific processor is always the master. Suppose that the system is symmetric in the processors, i.e. we can exchange any two processors. As representatives we choose states where the first processor is the master. However, we might not be interested in further exchanging (i.e. sorting) the other processors to get a unique minimal representative, because this might introduce unnecessary dependencies which can result in larger *BDDs*.

We will still use $\xi(s)$ to denote the representative for s. However, now $\xi(s_1)$ can be different from $\xi(s_2)$ even if s_1 and s_2 are in the same orbit. We extend the functions ξ and Θ so that they map sets of states into sets of states in the usual manner. Thus if $M \subseteq S$, we have $\xi(M) = \{\xi(s) \mid s \in M\}$ and $\Theta(M) = \{\Theta(s) | s \in M\}$. Given a set of orbits O, we say that a set of representatives M is an *approximation* of O iff $\Theta(M) = O$.

The basic steps in model checking are the computation of the image ($\mathrm{Im}_R(M) = \{s' \mid \exists s \in M : R(s,s')\}$) and the preimage ($\mathrm{Im}_R^{-1}(M) = \{s \mid \exists s' \in M : R(s,s')\}$) of a set of states (see [3]). If M is a set of representatives, an approximation M' of the image or the preimage of $\Theta(M)$ can be computed as follows:

$$\mathrm{Im}_{R_G}(\Theta(M)) = \Theta(M') \text{ with } M' = \xi(\mathrm{Im}_R(M))$$
$$\mathrm{Im}_{R_G}^{-1}(\Theta(M)) = \Theta(M') \text{ with } M' = \xi(\mathrm{Im}_R^{-1}(M))$$

From the definitions given above we see that M' is an approximation of $\Theta(\mathrm{Im}_R(M))$.

Instead of using the transition relation of the quotient model, the image and preimage are computed in terms of representatives. This technique allows us to perform model checking on the quotient model without explicitly building the quotient model. The representatives at each step of the computation are used as an approximation for the set of orbits. This methodology can be easily extended to an arbitrary fixpoint computation by transforming the sets encountered to a set of representatives (using the ξ function).

The set *Rep* of representatives is defined as the image of the set of states S under ξ. If the set of initial states is a subset of *Rep*, then by induction it is clear that all the sets computed by the equations given above are subsets of *Rep*. Therefore the image or preimage computations are only performed on representatives, and one side of the transition relation can be restricted to the set *Rep* ($R' = R \cap (Rep \times S)$ or $R' = R \cap (S \times Rep)$). For our example this means that during a reachability analysis we need only consider transitions from states where the first processor is the master. This restriction

can reduce the size of the transition relation considerably, but this entails having to apply the mapping function ξ at each step.

8 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 [12]. 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 1).

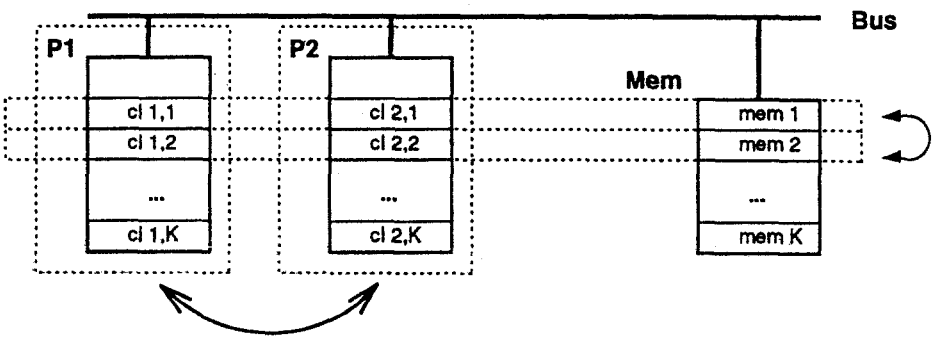

Fig. 1. System structure

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 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 1 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 orderings, which we call "concatenation" and "interleaving". The concatenation ordering is simply $P_1 \prec P_2 \prec \ldots \prec P_N$. 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. $p_{1,1} \prec p_{2,1} \ldots p_{N,1} \prec p_{1,2} \prec p_{2,2} \ldots$ where $p_{i,1}, \ldots, p_{i,K}$ are the state variables for processors 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.

As representatives we want to choose states where processor 1 is the master (cache line 1 is active on the bus). As generators we consider the permutations $g_1, \ldots, g_N$, where g_i exchanges processor 1 and i (cache lines 1 and i). Such a permutation $g_i : S \rightarrow S$ can be represented easily as a vector of boolean functions. We checked that each generator is a symmetry by comparing $R(s, s')$ and $R(g_i(s), g_i(s'))$, which is obtained by functional composition of R and g_i. For defining the mapping function we construct functions g_i' which exchange processor 1 and i only if processor i is the master and processor 1 is not. An analogous definition can be made for exchanging the cache lines. The mapping function, which is used as an approximation, is the composition of these functions g_i', i.e. $\xi(M) = g_1'(\ldots g_{N-1}'(g_N'(M))\ldots)$.

In the experiments we performed a reachability analysis using the approximation of the orbit relation. With the set of reachable states all safety properties of the form AG p where p is a propositional formula can be checked easily. We ran the experiments with various system configurations. The results are listed in Table 1. Each row $jPkC$ in the ta-

system config.	trans. relation	no symmetry		symmetry					
				processors		cache lines		combination	
	BDD nodes	BDD nodes	time sec.	BDD nodes	time sec.	BDD nodes	time sec.	BDD nodes	time sec.
2P4C c	1,911	32,655	28	9,721	11	9,941	15	3,401	12
2P4C i	1,819	4,942	6	2,660	4	2,177	8	1,142	7
4P4C c	7,675	586,479	665	133,967	161	176,266	174	35,359	108
4P4C i	12,469	52,480	57	14,820	20	24,907	34	6,773	22
2P8C c	6,699			> 995,027	-			310,124	2431
2P8C i	6,043	35,360	66	18,396	39	8,558	62	4,373	61
4P8C c	28,323	overflow	-	overflow					
4P8C i	43,765	413,741	1,151	110,305	277	103,028	434	26,934	288

Table 1. Empirical Results

ble gives the results for a configuration of j processors and k cache lines. The labels i and

c indicate which variable ordering (interleaved or concatenation) was used in the experiment. The first column gives the number of BDD nodes for representing the transition relation. The columns after that give the results for the symbolic reachability analyses. 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 for the reachability analysis the size of the largest intermediate BDD (number of BDD nodes) and the cpu time used are listed. All experiments where run on a Sun Sparc2 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 15 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 step in the reachability analysis. In the future, we hope to improve the efficiency of the mapping computation.

9 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 lower bounds on the size of the BDDs needed for the orbit relation for groups other than full symmetric and rotation groups. 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 [11]. 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.

Finally, we plan to check a number of liveness properties for the Futurebus protocol in Section 9 besides the safety properties that we have already tried. We also intend to try other hardware examples with more complicated topologies in addition to cache coherency protocols.

Acknowledgements

We would like to thank David Long and Ken McMillan for their help in writing this paper.

References

1. M. Browne, E. Clarke, and O. Grumberg. Characterizing finite kripke structures in propositional temporal logic. *Theoretical Comput. Sci.*, 59:115–131, 1988.
2. R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, C-35(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, editor. *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.*, 8(2):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. R. Enders. Note on the complexity of boolean representations of permutation and rotation functions. Technical report, Siemens, 1993. unpublished.
8. M. Furst, J. Hopcroft, and E. Luks. Polynomial-time algorithms for permutations groups. In *Proc. 21st Ann. Symp. on Found. of Comput. Sci.*, 1980.
9. M. Garey and D. Johnson. *Computers and Intractibility.* W.H. Freeman and Company, 1979.
10. P. Huber, A. Jensen, L. Jepsen, and K. Jensen. Towards reachability trees for high-level petri nets. *In G. Rozenberg, editor, Advances on Petri Nets*, pages 215–233, 1984.
11. S. L. Hurst, D. M. Miller, and J. C. Muzio. *Spectral Techniques in Digital Logic.* Academic Press, Inc., 1985.
12. IEEE Computer Society. *IEEE Standard for Futurebus+—Logical Protocol Specification*, Mar. 1992. IEEE Standard 896.1-1991.
13. C. Ip and D. Dill. Better verification through symmetry. To appear in Proc. 11th Int. Symp. on Comput. Hardware Description Lang. and their Applications, Apr. 1993.
14. R. Kannan and R. Lipton. Polynomial-time algorithm for the orbit problem. *J. ACM*, 33(4):808–821, 1986.
15. B. Lin and A. R. Newton. Efficient symbolic manipulation of equvialence relations and classes. In *Proc. 1991 Int. Workshop on Formal Methods in VLSI Design*, Jan. 1991.
16. K. L. McMillan and J. Schwalbe. Formal verification of the Gigamax cache consistency protocol. In N. Suzuki, editor, *Shared Memory Multiprocessing.* MIT Press, 1992.
17. P. Starke. Rechability analysis of petri nets using symmetries. *Syst. Anal. Model. Simul.*, 8(4/5):293–303, 1991.