Symmetry and Model Checking*

E. Allen Emerson[1] and A. Prasad Sistla[2]

[1] Department of Computer Sciences, The University of Texas at Austin, USA
[2] Electrical Engineering and Computer Science Department, The University of Illinois at Chicago, USA

"Whenever you have to do with a structure-endowed entity Σ try to determine its group of automorphisms"
– Hermann Weyl
in *Symmetry*

Abstract. We show how to exploit symmetry in model checking for concurrent systems containing many identical or isomorphic components. We focus in particular on those composed of many isomorphic processes. In many cases we are able to obtain significant, even exponential, savings in the complexity of model checking.

1 Introduction

In this paper, we show how to exploit symmetry in model checking. We focus on systems composed of many identical (isomorphic) processes. The global state transition graph $\mathcal{M}$ of such a system exhibits a great deal of symmetry, characterized by the group of graph automorphisms of $\mathcal{M}$. The basic idea underlying our method is to reduce model checking over the original structure $\mathcal{M}$, to model checking over a smaller quotient structure $\overline{\mathcal{M}}$, where symmetric states are identified. In the following paragraphs, we give a more detailed but still informal account.

More precisely, the symmetry of $\mathcal{M}$ is reflected in the group, *Aut* $\mathcal{M}$, of permutations of process indices defining graph automorphisms of $\mathcal{M}$. Similarly, any specification formula f intended to capture correctness of $\mathcal{M}$ in a particular Temporal Logic (say, CTL*) exhibits a certain degree of symmetry reflected in the group, *Aut* f, of permutations of process indices that leave f invariant, utiliizing commutativity and associativity of $\wedge$, $\vee$, etc.

We show that for any G contained in *Aut* $\mathcal{M}$, we can define $\overline{\mathcal{M}} = \mathcal{M}/G$ to be the quotient structure obtained by identifying any two states s, t of $\mathcal{M}$ that are in the same orbit (or equivalence class) of the state space of $\mathcal{M}$ induced by G in the usual way: there exists a permutation π in G such that $\pi(s) = t$. In

* The work of the first author was supported in part by ONR contract N00014-89-J-1472, and Texas Advanced Technology Program Grant 003658-250. The work of the second author was supported in part by NSF grant CCR-9212183.

other words, s and t are the same except for a permutation of their indices. (For example: $s = (N_1, T_2, C_3), t = (N_2, T_3, C_1)$).

We next show that such a quotient structure $\overline{\mathcal{M}}$ corresponds in a coarse sense to the original structure $\mathcal{M}$, so that if there is a path in $\overline{\mathcal{M}}$ there is an analogous path in $\mathcal{M}$, and conversely. However, the correspondence may not be sufficiently precise to (directly) model check a specification f. If we further stipulate that G be contained in $Aut\ \mathcal{M} \cap Aut\ f$ then we get a precise correspondence enabling us to establish

$$\mathcal{M}, s \models f \quad \text{iff} \quad \overline{\mathcal{M}}, \overline{s} \models f$$

where f is a formula of CTL* or Mu-Calculus, and $\overline{s}$ indicates the equivalence class of s.

Determination of a suitable G is potentially a difficult problem. We note that $G = Aut\ \mathcal{M}$ is a possibility, and the fundamental problem of computing $Aut\ \mathcal{M}$ appears to be a computationally difficult one, that is polynomial time equivalent to graph isomorphism (cf. [Ho82]). Fortunately, since $\mathcal{M}$ is derived from a concurrent system $\mathcal{P} = //_i K_i$ consisting of many isomorphic processes K_i, we are able to show that $Aut\ \mathcal{M} = Aut\ CR$, where CR is the process communication graph for $\mathcal{P}$. Since CR often follows a simple, standard pattern, $Aut\ CR$ is often known as a basic fact of group theory. Moreover, for massively parallel architectures $Aut\ \mathcal{M} = Aut\ CR$ is likely to be a large group reflecting a high degree of symmettry.

We emphasize here that any subgroup G of $Aut\ \mathcal{M} \cap Aut\ f$ is sufficient. The largest one possible is desirable for maximal compression. For many of the automorphism groups G determined in practice we can efficiently and incrementally compute $\mathcal{M}/G$, thereby circumventing the construction of $\mathcal{M}$. Of course, we then accrue the advantage of model checking over the smaller structure $\overline{\mathcal{M}} = \mathcal{M}/G$.

One common and advantageous case occurs when $G = Sym\ [1:n]$, the set of all permutations on indices $[1:n]$. For a system with n processes each with l local states, the orginal structure can have on the order of l^n states, while $\mathcal{M}/G$ has on the order of n^l states. When l is fixed and relatively small, while n is large, then $n^l \ll l^n$. We can thus realize exponential savings.

A complication can occur when f is a complex formula with little symmetry. Then $Auto\ f$ and G may be small, resulting in little compression using just the techniques described above. We argue that it is frequently beneficial to decompose f into smaller constituent subformulae and check those individually. We also show how the symmetry of individual states can be exploited for further gains in efficiency.

Finally, we give an alternative, uniform method that permits use of (essentially) the single quotient $\overline{\mathcal{M}} = \mathcal{M}/Aut\ \mathcal{M}$ for model checking any specification f, without computing and intersecting with $Auto\ f$. The idea is to annotate the quotient with "guides", indicating how coordinates are permuted from one state to the next in the quotient. An automaton for f designed to run over paths through $\mathcal{M}$, can be modified into another automaton run over $\overline{\mathcal{M}}$ using the guides to keep track of shifting coordinates.

The remainder of the paper is organized as follows: in section 2 we give preliminary definitions and terminology. In section 3 we give the main technical results establishing correspondences between the quotient structures and the orginal structures. In section 4 we show how the correspondences are applied to reduce model checking over $\mathcal{M}$ to $\mathcal{M}/G$ using the simplification provided by $Aut\ \mathcal{M} = Aut\ CR$. We also discuss optimizations based on formula decomposition and state symmetry. An alternative approach is discussed in Section 5. Examples are considered in section 6. In the concluding section 7 we discuss related work.

2 Preliminaries

2.1 Model of Computation

We deal with *structures* of the form $\mathcal{M} = (\mathcal{S}, \mathcal{R})$ where

- $\mathcal{S} = L^I \times D^V$ is the finite set of *states*, with L a finite set of individual process *locations*, I the set of process indices, D is a finite *data domain*, and V is a finite set of shared *variables*. We remark that D and V are optional, in which case we define $\mathcal{S} = L^I$. When present, D and V can have their own additional internal organization. In particular, they can depend on I.
- $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ which represents the moves of the system.

Notation: For convenience, each state $s = (s', s'') \in \mathcal{S}$ can be written in the form $(\ell_1, \ldots, \ell'_n, v = d, \ldots v' = d')$ indicating that processes $1, \ldots, n$ are in locations $\ell, \ldots, \ell'$, respectively and the shared variables $v \ldots v'$ are assigned data values $d \ldots d'$, respectively.

As usual, a *path* through $\mathcal{M}$ is a finite or infinite sequence of states such that every consecutive pair of states is in $\mathcal{R}$. We denote it by $s_0, s_1, s_2, \ldots$ or by $s_0 \longrightarrow s_1 \longrightarrow s_2, \ldots$ not bothering to explicitly indicate the last state for finite paths. A *fullpath* is a maximal path, i.e., either an infinite path or a finite one whose last state lacks an $\mathcal{R}$-successor.

In practice, for ordinary model checking, $\mathcal{M}$, is the global state transition graph of a finite state concurrent program $\mathcal{P}$ of the form $//_i K_i$ consisting of processes $K_1, \ldots, K_n$ running in parallel. Each K_i may be viewed as a finite state transition graph with node set L. An arc from node ℓ to node ℓ' may be labelled by a guarded command $B \longrightarrow A$. The guard B is a predicate that can inspect shared variables and local states of "acessible" processes The action A is a set of simultaneous assignments to shared variables $v := d \mid\mid \cdots \mid\mid v' := d'$. When process K_i is in local state ℓ and the guard B evaluates to *true* in the current global state, the global system can nondeterministically choose to advance by firing this transition of K_i which changes the local state of K_i to be ℓ' and the shared variables in V according to A. Thus the arc from ℓ to ℓ' in K_i represents a *local transition* of K_i that we denote: $\ell : B \longrightarrow A : \ell'$.

In this paper, we will further stipulate that (i) All the K_i's are isomorphic, i.e., identical up to renaming the index i; and (ii) There is a *communication relation* CR describing the network topology of which processes can communicate with each other. Pair $(i, j) \in CR$ iff K_i can communicate with K_j iff K_j can communicate with K_i. Formally, $CR \subseteq I \times I$ is an irreflexive, symmetric binary relation on I.

The structure $\mathcal{M}$ corresponding to $\mathcal{P}$ is thus defined using the obvious operational semantics. First, the set of all possible states $\mathcal{S}$ is determined from $\mathcal{P}$ because it provides us with the set of local locations L, process indices I, variables V, and data domain D. We define $s \longrightarrow t \in \mathcal{R}$ iff
$\exists i \in I$ process K_i causes s to move to t, denoted $s \longrightarrow_i t \in \mathcal{R}$ iff
$\exists i \in I$ $\exists$ local transition $\tau_i = \ell_i : B_i \longrightarrow A_i : m_i$ of K_i which *drives* $s = (s', s'')$ to $t = (t', t'')$ in $\mathcal{M}$; this means the i-th component of s' equals ℓ_i, the i-th component of s'' equals m_i, predicate $B_i(s) = \mathit{true}$, and $t'' = A_i(s'')$.

We are often interested in just the set of states reachable by executing $\mathcal{P}$ starting in a particular start state s_0. It is often most natural to consider execution of a program appropriately intialized. Moreover, the set of states reachable from s_0 can be much smaller than the set of all possible states. It is thus important to note that we can incrementally generate the structure $\mathcal{M}$ corresponding to $\mathcal{P}$ starting in state s_0. We use the notation $K_i(s)$ to denote the set of states reachable from state s by a single step of process K_i. We begin with s_0, propagate it by adding in the members of the various $K_i(s_0)$'s, and then propagate the K_i's of those members, and so on until closing off. See section 4.2 for a helpful generalization of this idea.

2.2 Logics of Programs

We assume a familiarity with basic aspects of temporal and modal logics of programs. (cf. [Em90], [MP92], [St91]). We use the usual linear temporal operators F (sometime), G (always), X (nexttime), and U (strong until). We get branching time logic by combining these with the path quantifiers A (for all fullpaths) and E (for some fullpath). The *basic modalities* of CTL are of the form A or E followed by a single F, G, X, or U applied to pure propositional arguments. The formulae of CTL are compositions and boolean combinations of its basic modalities. CTL* is defined analogously but with basic modalities of the form A or E followed by an arbitrary formula of propositional linear temporal logic with F, G, X and U. The propositional Mu-calculus is another very powerful logic built up from $\vee$, $\wedge$, $\neg$, atomic propositions and variables, AX, EX, and the extremal fixpoint operators μ (least fixpoint) and ν (greatest fixpoint). It can encode, often succinctly, most other logics of interest.

2.3 Applicable Group Theory

We sumarize the essential notions from group theory needed here. We refer the reader to one of the many standard texts discussing this topic (cf. [He64]) for additonal information. A group $\mathcal{G}$ is a set G together with a binary operation on G, called the group multiplication, that is associative, has an identity, and has an inverse for each group element. In practice, we write just G for $\mathcal{G}$ and multiplication may be indicated by concatenation. $H \leq G$ denotes H is a subgroup of G.

A *permutation* π on a finite set of obects I is a 1-1, onto mapping $\pi : I \longrightarrow I$. The set of all permutations on I, denoted $\mathit{Sym}\ I$, forms a group under functional composition: if permutations $\pi', \pi'' \in \mathit{Sym}\ I$ then $\pi = \pi'' \circ \pi' \in \mathit{Sym}$. Here the order of functional composition in $\pi'' \circ \pi'$ is to first apply π' then apply π''. If $J \subseteq I$ then $\mathit{Stab}\ J$ denotes $\{\pi : \forall\, j \in J\ \pi(j) = j\}$. Id is the *identity* permutation or relation on I.

Given an indexed object b, i.e. one whose description depends on I, we can defined a notion of ***permutation*** π being ***applied to*** b, denoted $\pi(b)$. In general, $\pi(b)$ is obtained from b by simultaneously replacing every occurrence of index $i \in I$ by $\pi(i)$ We consider some examples of specific types of objects.

Given state $s = (N_1, T_2, C_3, turn = 1)$, where $\{N, T, C\} \subseteq L$, $turn$ is a shared variable, and $\pi : 1 \mapsto 2, 2 \mapsto 1, 3 \mapsto 3$, we have $\pi(s) = (N_{\pi(1)}, T_{\pi(2)}, C_{\pi(3)}, turn = \pi(1)) = (N_2, T_1, C_3, turn = 2) = (T_1, N_2, C_3, turn = 2)$. We define $Aut\ s = \{\pi \in Sym\ I : \pi(s) = s\}$.

For a formula f of propositional or temporal logic (over an alphabet of atomic propositions indexed by I (cf. [CG89]) such as $f = P_1 \wedge P_2$ with π as above, we have $\pi(f) = P_{\pi(1)} \wedge P_{\pi(2)} = P_2 \wedge P_1 \equiv P_1 \wedge P_2 = f$. However, for $g = P_1 \wedge \neg P_2$, we get the inequivalent $\pi(g) = P_{\pi(1)} \wedge \neg P_{\pi(2)} = P_2 \wedge \neg P_1$. We define $Aut\ f = \{\pi \in Sym\ I : \pi(f) \equiv f\}$. The determination of $Aut\ f$ is discussed further in section 4.1.

If f is a CTL* formula with $q_1, \ldots, q_m$ all the maximal propositional subformulae of f w.r.t. the subformula relation, it is also useful to define $Auto\ f = Aut\ q_1 \cap \ldots \cap Aut\ q_m$. $Auto\ f$ consists of those permutations respecting the symmetry not only of f but also of its major constituent subformulae q_i. Similarly for the Mu-calculus.

We will also define a notion of automorphism h of structure $\mathcal{M}$ into itself. By analogy with the usual definition of graph automorphism for labelled, directed graphs we say the following:

An *automorphism* h of structure $\mathcal{M} = (\mathcal{S}, \mathcal{R})$ is a mapping $h : \mathcal{S} \longrightarrow \mathcal{S}$ that
1. is 1-1, onto on $\mathcal{S}$,
2. preserves edge structure: $s \longrightarrow t \in \mathcal{R}$ implies $h(s) \longrightarrow h(t) \in \mathcal{R}$, and
3. preserves "labelling" of states up to a permutation: $h(s) = \pi_{h,s}(s)$ for some $\pi_{h,s} \in Sym\ I$.

Observe, in particular, that a permutation π on I, viewed a mapping $\mathcal{S} \longrightarrow \mathcal{S}$, may be an automorphism of $\mathcal{M}$, as it fulfills the 1st and 3rd criteria. If it also fulfulls the 2nd criterion it is an automorphism of $\mathcal{M}$. We define $Aut\ \mathcal{M} = \{\pi : \pi$ defines an automorphism of $\mathcal{M}\}$.

Finally, let G be any subgroup of $Sym\ I$. Then we can define an equivalence relation $\equiv_G$ on states in $\mathcal{S}$ where $s \equiv_G t$ iff $\exists\ \pi \in G$ such that $t = \pi(s)$. The equivalence class of s, denoted $[s]_G$, is also referred to as the ***G-orbit*** of s. In the sequel, our task will be to find a subgroup G of $Sym\ I$ that is a subgroup of $Aut\ \mathcal{M}$ thus respecting the symmetry of $\mathcal{M}$ and also is a subgroup of $Auto\ f$, thus respecting the symmetry of f. We then collapse ***G-equivalent*** states to get a "quotient structure" as defined below. ***We emphasize that any subgroup G of Auto $\mathcal{M}$ $\cap$ Aut f is sufficient for our application. The largest one possible is desirable for maximal compression.***

2.4 Quotient Construction

Let $\mathcal{M} = (\mathcal{S}, \mathcal{R})$ be a structure and let $\equiv$ be an equivalence relation on $\mathcal{S}$. Let $\overline{\mathcal{S}}$ be a *set of representatives* of the partition of $\mathcal{S}$ into equivalence classes induced by $\equiv$: for each $s \in \mathcal{S}$ there exists a unique *representative* $\overline{s}$ of s such that $\overline{s} \in [s] \cap \overline{\mathcal{S}}$. Then the *quotient of* $\mathcal{M}$ *modulo* $\equiv$, as specified by the set of representatives $\overline{\mathcal{S}}$, is $\overline{\mathcal{M}} = \mathcal{M}/\equiv\, = (\overline{\mathcal{S}}, \overline{\mathcal{R}})$ where $\overline{s} \longrightarrow \overline{t} \in \overline{\mathcal{R}}$ iff there exists $s' \equiv \overline{s}$ and there exists $t' \equiv \overline{t}$ such that $s' \longrightarrow t' \in \mathcal{R}$. When $\equiv$ is $\equiv_G$ we write just $\mathcal{M}/G$ or $\overline{\mathcal{M}}$.

3 Correspondence Results

Correspondence Lemma 3.1

There is a bidirectional correspondence between paths of the original structure $\mathcal{M}$ and the quotient structure $\overline{\mathcal{M}} = \mathcal{M}/G$ for any $G \leq Aut\ \mathcal{M}$:
(i) From the structure $\mathcal{M}$ to the quotient $\overline{\mathcal{M}}$: if $x = s_0, s_1, s_2, \ldots$ is a path in $\mathcal{M}$, then there is an image path $\overline{x} = \overline{s}_0, \overline{s}_1, \overline{s}_2, \ldots$ of corresponding representatives $\overline{s}_i \equiv_G s_i$ in $\overline{\mathcal{M}}$.
(ii) From the quotient $\overline{\mathcal{M}}$ to the structure $\mathcal{M}$: if $\overline{x} = \overline{s}_0, \overline{s}_1, \overline{s}_2, \ldots$ is a path in $\overline{\mathcal{M}}$, then for every state $s'_0 \equiv_G \overline{s}_0$ in $\mathcal{M}$ there exists a corresponding path $x' = s'_0, s'_1, s'_2, \ldots$ in $\mathcal{M}$ of states $s'_i \equiv_G \overline{s}_i$.

Proof The direction (i) is immediate from the definition of quotient structure.

For direction (ii) let $\overline{x} = \overline{s}_0, \overline{s}_1, \overline{s}_2, \ldots$ be a path in $\overline{\mathcal{M}}$. Choose an arbitrary $s'_0 \equiv_G \overline{s}_0$. By definition of quotient structure and since $\overline{s}_0 \longrightarrow \overline{s}_1 \in \overline{\mathcal{M}}$, there exists $s''_0 \equiv_G \overline{s}_0$ and there exists $s''_1 \equiv_G \overline{s}_1$ such that $s''_0 \longrightarrow s''_1 \in \mathcal{M}$.

Thus, by transitivity $s'_0 \equiv_G s''_0$ and $s'_0 = \pi(s''_0)$ for some permutation $\pi \in G$. Let $s'_1 = \pi(s''_1)$. Now, $s'_0 \longrightarrow s'_1 = \pi(s''_0) \longrightarrow \pi(s''_1) \in \mathcal{M}$ since $s''_0 \longrightarrow s''_1 \in \mathcal{M}$ and $\pi \in G \leq Aut\ \mathcal{M}$. Moreover, $s'_1 = \pi(s''_1) \equiv_G \overline{s}_1$ as desired.

The first edge of x' is thus defined by $s'_0 \longrightarrow s'_1$. Continuing with s'_1 the same argument can be applied to exhibit s'_2 such that $s'_1 \longrightarrow s'_2 \in \mathcal{M}$ and $s'_2 \equiv_G \overline{s}_2$. Proceeding, in this fashion we see that there is $s'_i \longrightarrow s'_{i+1} \in \mathcal{M}$ corresponding to each $\overline{s}_i \longrightarrow \overline{s}_{i+1}$ of $\overline{x}$ in $\overline{\mathcal{M}}$. The process continues for all natural numbers i or until the terminal i of $\overline{x}$ if it is finite. Let $x' = s'_0 \longrightarrow s'_1 \longrightarrow s'_2 \longrightarrow \ldots$ be the resulting path in $\mathcal{M}$. By construction, it corresponds to $\overline{x}$ in the desired way. □

Remark If the Correspondence Lemma is restricted to paths consisting of a single transition, it amounts to saying that there is a bisimulation between $\mathcal{M}$ and $\overline{\mathcal{M}}$ defined by $\equiv_G$. □

The Correspondence Lemma makes it easy to prove the following fundamental results showing that model checking over $\mathcal{M}$ can be reduced to model checking over $\overline{\mathcal{M}}$.

Theorem 3.2 $\mathcal{M}, s \models f$ iff $\mathcal{M}/G, \overline{s} \models f$ for any $G \leq Aut\ \mathcal{M} \cap Auto\ f$ where f is of one of the following forms with p being a propositional formula:

- $f = p$; or
- $f = EXp$; or
- $f = EFp$; or
- $f = EGp$.

Proof. We note $Auto\ f = Aut\ f = Aut\ p$ and argue by cases.

$f = p$: $\mathcal{M}, s \models p$ iff $\mathcal{M}, \overline{s} \models p$ since $s, \overline{s}$ are in the same G–orbit and $G \leq Aut\ p$. The later holds iff $\mathcal{M}/G, \overline{s} \models p$ since the truth of a propositional formula depends only the current state $\overline{s}$ which is present in both structures $\mathcal{M}$ and $\mathcal{M}/G$.

$f = EXp$: The $\Rightarrow$ direction is immediate. The $\Leftarrow$ direction is as follows. Assume $\mathcal{M}/G, \overline{s} \models EXp$. Then there is a path of length 1, i.e., edge in $\mathcal{M}/G$ of the form $\overline{s} \longrightarrow \overline{t}$ such that $\mathcal{M}/G, \overline{t} \models p$. Since p is propositional, $\mathcal{M}, \overline{t} \models p$. By direction (ii) of the Correspondence Lemma, there is a corresponding path x' in M of the form $s = s' \longrightarrow t'$ with $\overline{t} \equiv_G t'$, and since $G \leq Autp$, it must be that $\mathcal{M}, t' \models p$. Thus, $\mathcal{M}, s \models EXp$.

$f = EGp$: The $\Rightarrow$ direction is immediate from the Correspondence Lemma direction (i). The $\Leftarrow$ direction is as follows. Suppose $\mathcal{M}/G, \overline{s}_0 \models EGp$. Then there is a fullpath x of the form $\overline{s}_0 \longrightarrow \overline{s}_1 \longrightarrow \overline{s}_2 \cdots$ in $\mathcal{M}/G$ such that, for each i, $\mathcal{M}/G, \overline{s}_i \models p$. By the Correspondence Lemma, part (ii), there is a corresponding path x' of the form $s_0 = s'_0 \longrightarrow s'_1 \longrightarrow s'_2 \ldots$ in $\mathcal{M}$ such that, for all i, we have $s'_i \equiv_G \overline{s}_i$. Since $G \leq Aut\ p$, the truth value of the propositional formula p must be the same for s'_i and $\overline{s}_i$ for each i. Hence, for all i, we have $\mathcal{M}, s'_i \models p$ and $\mathcal{M}, x' \models Gp$. By virtue of x', we conclude $\mathcal{M}, s_0 \models EGp$.

$f = EFp$: The $\Rightarrow$ direction is immediate from part (i) of the Correspondence Lemma. The $\Leftarrow$ direction is as follows. Suppose $\mathcal{M}/G, \overline{s}_0 \models EFp$. Then there is a path $\overline{x}$ in $\mathcal{M}/G$ of the form $\overline{s}_0, \overline{s}_1, \ldots, \overline{s}_k$ such that $\mathcal{M}/G, \overline{s}_k \models p$ and, since p is propositional, $\mathcal{M}, \overline{s}_k \models p$. By direction (ii) of the Correspondence Lemma, there is a corresponding path x' in M of the form $s_0 = s'_0, s'_1, s'_2, \ldots s'_k$. In particular, $\overline{s}_k \equiv_G s'_k$, and since $G \leq Autp$, it must be that $\mathcal{M}, s'_k \models p$. Thus, $\mathcal{M}, s_0 \models EFp$.
$\square$

The above theorem can be generalized to

Theorem 3.3 $\mathcal{M}, s \models f$ iff $\mathcal{M}/G, \overline{s} \models f$ for any $G \leq Aut\ \mathcal{M} \cap Auto\ f$ where f is any formula CTL*.

A still stronger generalization is possible

Theorem 3.4 $\mathcal{M}, s \models f$ iff $\mathcal{M}/G, \overline{s} \models f$ for any $G \leq Aut\ \mathcal{M} \cap Auto\ f$ where f is any formula of the propositional Mu-calculus.

The above results are directly applicable to arbitrary, composite formulae of CTL* or the Mu-calculus. However, it is sufficient to handle just the basic modalities of a logic like CTL* or the Mu-calculus for model checking the entire logic (cf. [Em90]). In some cases it is advantageous to decompose a formula into subformulae like basic modalities before applying the above results (cf. section 4.3).

4 Applications

We wish to determine whether $\mathcal{M}, s_0 \models f$, where $\mathcal{M}$ is the global state transition graph of $\mathcal{P} = //_i K_i$ and f is an arbitrary CTL* or Mu-calculus formula, without incurring the potentially enormous cost of constructing $\mathcal{M}$. By Theorems 3.3 and 3.4, it suffices to instead construct $\mathcal{M}/G$, where G is a subgroup of $Aut\ \mathcal{M} \cap Auto\ f$, and then check whether $\mathcal{M}/G, s_0 \models f$. If G is large, reflecting a good deal of symmetry common to $\mathcal{M}$ and f, then we should realize a significant savings.

4.1 Determination of the Group G

It should be noted, however, that to calculate G we must also determine

(i) *Auto* f, (ii) *Aut* $\mathcal{M}$, and (iii) the intersection of (i) and (ii). Each of these appears to be a difficult problem equivalent to Graph Isomorphism in general. Fortunately, with certain reasonable restrictions on $\mathcal{M}$ and f the computations of (i)-(iii) become much easier.

4.1.1 Determination of the Automorphisms of Formulae For a propositional logic formula f, there are classically known algorithms to compute *Aut* f (cf. [Ko78]). If f is a temporal formula that is a "monadic" basic modality f of the form EXp, EFp, or EGp where p is propositonal, then *Aut* f = *Aut* p. We also note the following. Let $f = Eh$ be a basic modality of CTL*. Then f of the form $Eh(q_1, \ldots, q_m)$ where each q_i is a maximal, with respect to the subformula relation, propositional subformula of f. We have the useful approximation $Auto\ f = Aut\ q_1 \cap \ldots \cap Aut\ q_m \leq Aut\ f$. Similarly for the Mu-calculus.

For a temporal or propositional formula f, *Aut* f can often be determined by inspection, in practice, since f is often short or has a simple "indexed" representation. We have the following rules:

(i) If $f = g_i$ is a formula with only (occurrences of) the single index symbol i appearing, then $Aut\ f = Stab\ \{i\}$.
(ii) If $f = g_{ij}$ is a formula with only (occurrences of) the two index symbols i, j appearing, then $Aut\ f = Stab\ \{i, j\}$.
(iii) If $f = \bigwedge_{i \in I} g_i$ then $Aut\ f = Sym\ I$.
If $f = \bigvee_{i \in I} g_i$ then $Aut\ f = Sym\ I$.
(iv) If $f = \bigwedge_{i \neq j \in I} g_{ij}$ then $Aut\ f = Sym\ I$.
If $f = \bigvee_{i \neq j \in I} g_{ij}$ then $Aut\ f = Sym\ I$.

Here are some specfic example formulae and their automorphism groups:

$f = GP_1$	$Aut\ f = Stab\ \{1\}$
$f = GP_1 \wedge GP_2$	$Aut\ f = Sym\ [1:2] \cdot Stab\ \{1,2\}$
$f = G(P_1 \Rightarrow P_2)$	$Aut\ f = Stab\ \{1,2\}$
$f = AG(T_1 \Rightarrow AFC_1)$	$Aut\ f = Stab\ \{1\}$
$f = \bigwedge_i AG(T_i \Rightarrow AFC_i)$	$Aut f = Sym\ I$
$f = \bigwedge_{i \neq j} AG(\neg C_i \vee \neg C_j)$	$Aut\ f = Sym\ I$

To compute *Auto* f, by definition, involves intersections (cf. section 4.1.3), but can often be determined by inspection. For monadic modalities f of CTL*, $EXp, EFp, EGp, EGFp, EFGp$, we have $Auto\ f = Aut\ = p$.

4.1.2 Determination of the Automorphisms of Structures For many systems we can also determine *Aut* $\mathcal{M}$ by inspection of program $\mathcal{P}$. In the case when $\mathcal{P} = //_i K_i$ and all K_i are isomorphic, we are aided by the proposition below which intuitively asserts that under reasonable assumptions, $Aut\ \mathcal{M} = Aut\ CR$, reflecting the communication topology of $\mathcal{P}$.

Theorem 4.1 If $\mathcal{M}$ is the global state transition graph of $\mathcal{P} = //_i K_i$ where all K_i are isomorphic and each transition in K_i is of the form:

$$\ell_i : \bigwedge_{j \in CR(i)} B_{ij} \longrightarrow ||_{j \in CR(i)}\ A_{ij} : m_i$$

then $Aut\ \mathcal{M} = Aut\ CR$.

Proof *Aut CR* $\subseteq$ *Aut* $\mathcal{M}$: Pick $\pi \in$ *Aut CR*. Assume $s \longrightarrow_i t \in \mathcal{M}$. Then in process K_i there is some local transition

$$\tau_i = \ell_i : \wedge_{j \in CR(i)} B_{ij} \longrightarrow \|_{j \in CR(i)} A_{ij} : m_i$$

driving s to t in $\mathcal{M}$. Since K_i and $K_{\pi(i)}$ are isomorphic there is a local transition $\pi(\tau_i)$ in $K_{\pi(i)}$ of the following form:

$$\ell_{\pi(i)} : \wedge_{\pi(j) \in CR(\pi(i))} B_{\pi(i)\pi(j)} \longrightarrow \|_{\pi(j) \in CR(\pi(i))} A_{\pi(i)\pi(j)} : m_{\pi(i)}$$

Since $\pi(j)$ fulfills the roles of a bound variable, $\pi(\tau_i)$ can be written more simply as

$$\ell_{\pi(i)} : \wedge_{j' \in CR(\pi(i))} B_{\pi(i)j'} \longrightarrow \|_{j' \in CR(\pi(i))} A_{\pi(i)j'} : m_{\pi(i)}.$$

This local transition, $\pi(\tau_i)$ of $K_{\pi(i)}$ ensures that $\pi(s) \longrightarrow_i \pi(t) \in \mathcal{M}$. Hence, $\pi \in$ *Aut* $\mathcal{M}$.

Aut $\mathcal{M} \subseteq$ *Aut CR*: Let π in *Aut M* be chosen arbitrarily. Consider $s \longrightarrow_i t \in \mathcal{M}$. It must be caused by some local transition

$$\tau_i = \ell_i : \wedge_{j \in CR(i)} B_{ij} \longrightarrow \|_{j \in CR(i)} A_{ij} : m_i$$

in K_i. The local transition τ_i is unique, because our stipulation on $\mathcal{P}$ that there is at most a single local transition between two locations in each process K_i. Because $\pi \in$ *Aut* $\mathcal{M}$, $\pi(s) \longrightarrow_{\pi(i)} \pi(t) \in \mathcal{M}$, which could ony be accounted for by the local transition

$$\pi(\tau_i) = \ell_{\pi(i)} : \wedge_{j' \in CR(\pi(i))} B_{\pi(i)j'} \longrightarrow \|_{j' \in CR(\pi(i))} A_{\pi(i)j'} : m_{\pi(i)}$$

which must be present in $K_{\pi(i)}$ as all processes are isomorphic. But $\pi(\tau_i)$ is well-defined only if $\pi \in$ *Aut CR*. □

The communication relation CR of $\mathcal{P}$ may be viewed as a graph on the process indices where there is an edge between i and j iff K_i and K_j communicate. Since in designing the program $\mathcal{P}$, the choice of CR is (one hopes!) explicitly and carefully considered, and often chosen from a standard pattern, determination of *Aut CR* is often easy in practice, and frequently is just a well-known fact of graph theory. We have for example:

- If $CR = I \times I \setminus Id$ so that the communication topology is the complete graph on I, then *Aut* $\mathcal{M}$ = *Aut CR* = *Sym I*.
- If the processes $K_1, \ldots, K_n$ of $\mathcal{P}$ are arranged in a ring then $CR = \{(i, i \oplus_n 1), (i, i \ominus n1) : i \in I\}$, where $\oplus_n$ denotes wrap-around addition where $n \oplus_n 1 = 1$, and analogously for subtranction. This indicates that each process can only communicate with its two neighbors in the ring. Thus, *Aut* $\mathcal{M}$ = *Aut CR* = D_n, the dihedral group of order $2n$.

4.1.3 Determination of the Intersection To determine the intersection of *Aut f* and *Aut* $\mathcal{M}$, we can again often proceed by inspection. In practice, it is likely to turn out that one or both of *Aut f* or *Aut* $\mathcal{M}$ is large, for example *Sym I* or *Sym* $I \setminus \{i\}$, or at least a well-known permutation group which simplifies our task.

4.2 Constructing the Quotient Structure

We can construct $\mathcal{M}/G$ from $\mathcal{P}$ incrementally, without building $\mathcal{M}$ itself, as shown in figure 1 (cf. [ID92], [LY92], [CEJ93]). Here we assume $\mathcal{P}$ has start state s_0.

```
Let $\overline{\mathcal{S}} := \emptyset$
Let $\overline{s}_0 := s_0$
Add $\overline{s}_0$ to $\overline{\mathcal{S}}$
While unprocessed($\overline{\mathcal{S}}$) $\neq$ do
    Remove some unprocessed $\overline{s}$ from $\overline{\mathcal{S}}$
        For each $i \in [1:n]$ do
            For each $t \in K_i(\overline{s})$ do
                Ensure $\overline{t}$ ends up in $\overline{\mathcal{S}}$:
                    If $\exists \overline{u} \in \overline{\mathcal{S}}\ t \equiv_G \overline{u}$ then
                        Note $\overline{t} = \overline{u} \in \overline{\mathcal{S}}$ already
                    Else
                        Let $\overline{t} := t$
                        Add $\overline{t}$ to unprocessed($\overline{\mathcal{S}}$)
                Add $\overline{s} \longrightarrow \overline{t}$ to $\overline{\mathcal{R}}$
            End
        End
    Mark $\overline{s}$ processed
End
```

Fig. 1. Incremental Construction of Quotient

An important part of the above procedure is the test whether $t \equiv_G \overline{u}$. Since G may in the worst case be *Aut* $\mathcal{M}$, this could conceivably be intractable (cf. [Ho82], [CEJ93]). However, in practice $\mathcal{M}$ has special structure derived from $\mathcal{P}$, which can simplify matters. In some cases, the test is particularly simple. For example, if $\mathcal{S} = L^I$ and $L = \{\ell_1, \ldots, \ell_m\}$ and $G = Sym\ I$, then $s \equiv_G t$ iff for each $i \in [1:m]$ the number of processes in local state ℓ_i is the same for both global states s and t.

4.3 Decomposing Formulae

In some instances G may be very small essentially because f is a large composite formula. Consider, for example, $f = \wedge_i AG(N_i \Rightarrow AFC_i)$. We see that *Auto* f = *Stab* 1 $\cap \ldots \cap$ *Stab* $n = Id$, so that no compression is possible in forming the quotient $\overline{M} = M/G$, since $G \leq$ *Auto* f is required. It is often possible to ovecome this problem by breaking down the composite formula into its basic modalities (or other appropriate subformule) and checking them individually. While this may entail computing multiple quotients, it can still be more efficient. In the example formula, we can check for each conjunct $f_i = AG(N_i \Rightarrow AFC_i)$ in turn. Since *Auto* f_i = *Stab* i = *Sym* $I \setminus \{i\}$ is of exponential size, any G obtained from such an *Auto* f is likely to be large. Thus computing n different exponentially smaller quotients can be more efficient than computing one large quotient, actually equal to the full, original structure. See the following section

4.4 on state symmetry for further optimizations.

4.4 State Symmetry

Suppose s_0 is a state that is fully symmetric in a fully symmetric structure $\mathcal{M}$, viz. $Aut\ s_0 = Aut\ \mathcal{M} = Sym\ I$. For example, s_0 could be the start state $(N_1, \ldots, N_n)$ for a solution to the mutual exclusion problem with each process in its noncritical region (cf. [AE89], [EC82], section 6).

Note we have that $\mathcal{M}, s_0 \models \wedge_i\ g_i$ iff $\mathcal{M}, s_0 \models g_1$ The $\Rightarrow$ direction is obvious. To see the $\Leftarrow$ direction, choose an arbitrary $i \in I$. Then pick some $\pi \in Aut\ s_0 = Sym\ I$ such that $\pi(1) = I$. The right-hand-side implies for all permutations π' that $\mathcal{M}, \pi'(s_0) \models g_{\pi'(1)}$. For $\pi' = \pi$ this simplifies to the desired $\mathcal{M}, s_0 \models g_i$. The left-hand-side follows.

Thus, in reference to the previous section 4.3, in checking a formula such as $\wedge_i AG(N_i \Rightarrow AFC_i)$ evaluation of multiple conjuncts over multiple quotients is not required.

This idea can be generalized to states and systems with somewhat less symmetry. $Aut\ s \cap Aut\ \mathcal{M}$ induces an equivalence relation on I: $i \equiv j$ iff $i = \pi(j)$ for some $\pi \in Aut\ s \cap Aut\ \mathcal{M}$. There is an associated partition of I, call it $Part$, induced by $\equiv$. Let Rep be a set of representatives, one from each equivalence class in $Part$.

Theorem 4.2 $\mathcal{M}, s \models \wedge_i\ g_i$ iff $\mathcal{M}, s \models \wedge_{j \in Rep}\ g_j$

Proof The $\Rightarrow$ direction is obvious. To see the $\Leftarrow$ direction, assume the left hand side holds. Choose an arbitrary $i \in I$. Let j be the representative equivalent to i. For some $\pi \in Aut\ s \cap Aut\ \mathcal{M}$ we have $i = \pi(j)$. Moreover, $\mathcal{M}, s \models g_j$. So $\mathcal{M}, \pi(s) \models \pi(g_j)$ because $\pi \in Aut\ \mathcal{M}$. Because $\pi(s) = s$ and $\pi(j) = i$, this simplifies to $\mathcal{M}, s \models g_i$. □

Thus, instead of checking all $n = |I|$ conjuncts g_i, it suffices to check $|Rep|$ constructs which may be significantly smaller. In the extreme case, as above, only one conjunct need be checked. If $Aut\ \mathcal{M} = Sym\ I$ then matters simplify so that at most $|L|$, the number of distinct local states, need be checked. Typically $|L| \ll n = |I|$. If $Aut\ s = Sym\ I$, so that s is a start state with all process in the same local state, then if $Aut\ \mathcal{M} = Aut\ CR$ is nontrivial, some equivalence class on I has 2 or more members, $|Rep| < n$, and some savings is obtained. It many practical cases $Aut\ CR$ may yield a small $|Rep|$. Even a "sparse" connectivity graph like a ring yields only a single equivalence class.

5 Alternative Approach

We can give an alternative, uniform method that permits use of (essentially) the single quotient $\overline{\mathcal{M}} = \mathcal{M}/Aut\ \mathcal{M}$ for model checking any specification f, without computing and intersecting with $Auto\ f$. The idea is to annotate the quotient with "guides", indicating how coordinates are permuted from one state to the next in the quotient. An automaton for f designed to run over paths through $\mathcal{M}$, can be modified into another automaton run over $\overline{\mathcal{M}}$ using the guides to keep track of shifting coordinates.

We elaborate the above method here. We define $\overline{\mathcal{M}} = (\overline{\mathcal{S}}, \mathcal{AR})$ where $\overline{\mathcal{S}}$ is the set of representative states as before, and $\mathcal{AR}$ is an annotated relation. It is a set of triples of the form $(\overline{s}, \pi', \overline{t})$ where π' is a member of $Aut\ \mathcal{M}$ such that if $(\overline{s}, t) \in \mathcal{R}$ then there exists a single triple $(\overline{s}, \pi, \overline{t}) \in \mathcal{AR}$ with $\overline{t} = \pi(t)$. Note that all transitions in the original structure from a representative state to another representative state are included in $\mathcal{AR}$. Also, all transitions from a representative state to a non-representative state are included in $\mathcal{AR}$. Transitions from a non-representative state to a non-representative state in the original structure are not included in $\overline{\mathcal{M}}$. Due to this, many times, the size of the structure $\overline{\mathcal{M}}$ can be much smaller than that of $\mathcal{M}$.

This augmented representation of the quotient structure allows us to model check indexed CTL* properties efficiently. For example, consider the problem of checking the correctness of the formula $\wedge_i\ Ag_i$ where g_i is a formula in Propositional Linear Temporal Logic over the propositions corresponding to process i. To do this, it is enough to verify that the formula $\vee_i\ E\neg g_i$ is not satisfied. For this we construct the automaton $\mathcal{A}$ corresponding to the formula g_i and check that there is no path in $\mathcal{M}$ that is accepted. The input alphabet of $\mathcal{A}$ is the set of subsets of local propositions. To check the above condition, we take the cross product of the augmented structure with automaton $\mathcal{A}$ and show that there is no accepting path in the cross product. The states of the cross product are going to be triples of the form $(\overline{s}, q, j)$ where q is a state of the automaton $\mathcal{A}$ and j is a process index. The initial states of the product are going to be $(\overline{s_0}, q_0, i)$ where $\overline{s_0}$ and q_0 are the initial states of the structure and automaton respectively and i is any process index. For each transition of the form $(\overline{s}, \pi, \overline{t}) \in \mathcal{AR}$ and for each automaton state q and process index j, there is going to be a transition in the product automaton from the state $(\overline{s}, q, j)$ to the state $(\overline{t}, r, \pi(j))$ where r is any state to which there is a transition of $\mathcal{A}$ from state q on input which is the set of local propositions satisfied in the process j's component of $\overline{t}$. The accepting states of the product automaton are exactly those states of the form $(\overline{s}, q_f, j)$ where q_f is an accepting state of A.

The above automata method can be extended to check properties specified by automata. In this case, the automata checks global (mulit-index) properties, not simply the local (single index) properties. If this automaton is also symmetric with respect to the permutations in $Aut\ \mathcal{M}$ then we can give an efficient method based on the automata-theoretic approach. This is described in the full paper.

6 Examples

We first consider a simple example. A solution $\mathcal{P}$ to the mutual exclusion problem is given in Figure 2. Each process K_i has a noncritical section, corresponding to location N_i, and a critical section, represented by location C_i. The transition from N_i to C_i is guarded by the predicate $\wedge_{j \neq i} \neg C_j$. Hence, each process cycles through its two sections preserving the property of mutual exclusion: that no two processes are ever in their critical section at the same time. This can be expressed in CTL by (a formula of the form) $AG(\wedge_{i \neq j} \neg(C_i \wedge C_j))$. Thus the solution is safe. The starting condition can be captured as $\wedge_i N_i$.

To verify mutual exclusion, for a system with n processes, for any fixed n, we could build its global state transition graph $\mathcal{M}$, with $n + 1$ states, as in Figure 3. However, since the communication relation for $\mathcal{P}$ is the complete graph on n nodes, $Aut\ \mathcal{M} = Sym\ [1 : n]$. Our rules also tell us that $Aut\ f = Sym\ [1 : n]$

Thus we can take $G = Sym\ [1:n]$. Using $(N_1, N_2, \ldots, N_n)$ and $(C_1, N_2, \ldots, N_n)$ as representatives we obtain a quotient $\mathcal{M}/G$ shown in Figure 4. We can now model check over the quotient using Theorem 3.3.

Additional examples will be given in the full paper including one based on a multi-process extension (cf. [AE89]) of the 3 local state mutual exclusion problem from [EC82], showing (i) how auxillary variables can be handled (ii) treatment of liveness properties, and (iii) a case where true exponential savings is obtained.

7 Conclusions and Related Work

We have described a framework for expediting model checking by forming the quotient structure modulo a subgroup of the group of automorphisms of the original structure and the specification. The resulting reduction in size can be dramatic when the degree of symmetry is high. The group of automorphisms of the structure depends on process network topology, which is possibly a crucial factor here. For massively parallel systems with with high connectivity and high symmetry like hypercubes, we should get a very good savings. For rings, we would get much less. We have also shown how to improve the efficiency by decomposing large formulae into smaller subformulae. We have further shown that it is possible to exploit the symmetry of individual states to avoid redundant computation. An alternative approach using automata to track shifting indices was also given

It should be noted that, while we have focused on systems with many isomorphic processes, this is more in the nature of a restriction on the "systems" terminology. Excepting, for example, Theorem 4.1 showing $Aut\ \mathcal{M} = Aut\ CR$, the basic mathematical machinery here is applicable to systems containing multiple isomorphic "components". All that is really essential is symmetry in the state space, whatever its "physical, systems" source.

At present, we have a method, that is not fully automated. Obviously, we could mechanize it by using naive algorithms to compute automorphism groups, but this in general would not be efficient. Thus important open problems seem to us to be to identify useful special cases for when $Aut\ b$, for various objects b can provably be calculated efficiently, and the related problem of testing $\equiv_G$ efficiently (cf. [CEJ93]). Of course, these are largely group-theoretic in nature. There is a vast literature in computational group theory which should be helpful (cf. [Ho82]). In the interim, we are compiling a catalog of helpful special cases.

The telling quote from Herman Weyl [We52] in the introduction shows that the basic idea of exploiting the group of automorphisms of a structure in order to understand its basic properties, symmetry and otherwsie, is a rather old one in mathematics. However, its application to temporal logic model checking seems to be quite new. In the realm of program verification symmetry seems to have first been utilized, with varying degrees of formality, in the realm of reachability analysis for petri nets (cf. [JR91]). Here, however, the work seems to have centered around simple reachability (AGp) rather than the full range of temporal correctness properties. Ip and Dill also [ID93] consider the problem of verifying reachabiltiy only, rather than essentially arbitrary correctness properties. Their system provides a new, somewhat more abstract than usual programming language, to facilitate identifications of symmetrys. It has been implemented as the MurΦ system and applied to examples. In [AGS83] and [Ku86] an algebraic approach to reducing the cost of protocol analysis based on the use of quotient structures induced by automorphisms is proposed. For example, the symmetry between 0 and 1 in the alternating bit protocol is factored out to reduce the size of the state space by one half.

The most directly related work is that of Clarke, Filkorn, and Jha [CEJ93] who have independently reported correspondence results similar to those of our section 3 and follow a somewhat similar overall strategy (cf. [CGB89], [St93]). Moreover, they have implemented their ideas using BDD's, provided an analysis of the complexity of BDD-based manipulations of permutation groups showing that testing $\equiv_G$ is graph isomorphism hard for BDD-based representations, and done practical examples.

Our work may be distinguished by the most general explicit correspondence results, including CTL* and the Mu-Calculus, and by focussing on the symmetry induced by having many identical processes, which allows us to reduce the difficult problem of computing *Aut* $\mathcal{M}$ to *Aut CR*. We also permit auxillary variables, exploit formula decomposition and state symmetry, and provide an alternative automata-theoretic approach.

Acknowledgements and Historical Remark We have been thinking about this problem for some time. Actually, we had the Correspondence Lemma 3.1 in 1988 but encountered other difficulties. In any event, we would also like to thank Paul Attie and Steve Kaufman for valuable suggestions. We also thank Bob Kurshan for his comments.

8 References

[APS83] Aggarwal S., Kurshan R. P., Sabnani K. K., "A Calculus for Protocol Specification and Validation", in Protocol Specification, Testing and Verification III, H. Ruden, C. West (ed's), North-Holland 1983, 19-34.

[AE89] Attie, P. C., and Emerson, E. A., Synthesis of Many Processes, POPL89

[CE81] Clarke, E. M., and Emerson, E. A., Design and Verification of Synchronization Skeletons using Branching Time Temporal Logic, Logics of Programs Workshop 1981, Springer LNCS no. 131.

[CFJ93] Clarke, E. M., Filkorn, T., Jha, S. Exploiting Symmetry in Temporal Logic Model Checking, 5th CAV, June 1993 (this proceedings).

[CGB88] Clarke, E. M., Grumberg, O., and Brown, M., Characterizing Kripke Structures in Temporal Logic, Theor. Comp. Sci., 1988

[CGB89] Clarke, E. M., Grumberg, O., and Brown, M., Reasoning about Many Identical Processes, Inform. and Comp., 1989

[EC82] Emerson, E. A., and Clarke, E. M., Using Branching Time Temporal Logic to Synthesize Synchronization Skeletons, Science of Computer Programming, Dec. 1982.

[Em90] Emerson, E. A., Temporal and Modal Logic, in Handbook of Theoretical Computer Science, (J. van Leeuwen, ed.), Elsevier/North-Holland, 1991.

[He64] Herstein, I, Topics in Algebra, Xerox 196?

[Ho82] Hoffmann, C., Graph Isomorphism and Permutation Groups, Springer LNCS, 1992.

[Ku86] Kurshan, R. P., "Testing Containment of omega-regular Languages", Bell Labs Tech. Report 1121-861010-33 (1986); conference version in R. P. Kurshan, "Reducibility in Analysis of Coordination", LNCIS 103 (1987) Springer-Verlag 19-39.

[JR91] Jensen, K., and Rozenberg, G. (eds.), High-level Petri Nets: Theory and Application, Springer-Verlag, 1991.

[Ko78] Kohavi, Z., Switching and Finite Automata Theory, McGraw-Hill, 1978.

[ID93] Ip, C-W. N., Dill, D. L., Better Verification through Symmetry, CHDL, April 1993.

[MP92] Manna, Z. and Pnueli, A., Temporal Logic of Reactive and Concurrent Systems: Specification, Springer-Verlag, 1992

[St93] Stirling, C., Modal and Temporal Logics. in Handbook of Logic in Computer Science, (D. Gabbay, ed.) Oxford, 1993

[We52] Weyl, H., Symmetry, Princeton Univ. Press, 1952

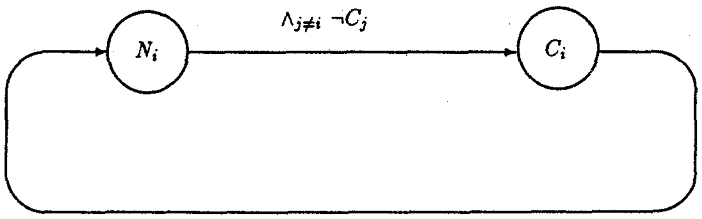

Figure 2: Skeleton for Two State n Process Mutual Exclusion

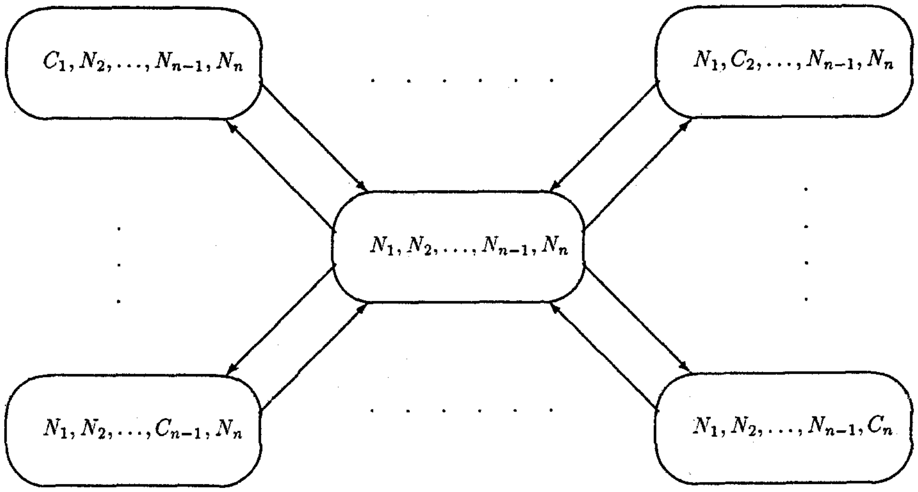
$C_1, N_2, \ldots, N_{n-1}, N_n$
$N_1, C_2, \ldots, N_{n-1}, N_n$
$N_1, N_2, \ldots, N_{n-1}, N_n$
$N_1, N_2, \ldots, C_{n-1}, N_n$
$N_1, N_2, \ldots, N_{n-1}, C_n$

Figure 3: Model for Two State n Process Mutual Exclusion

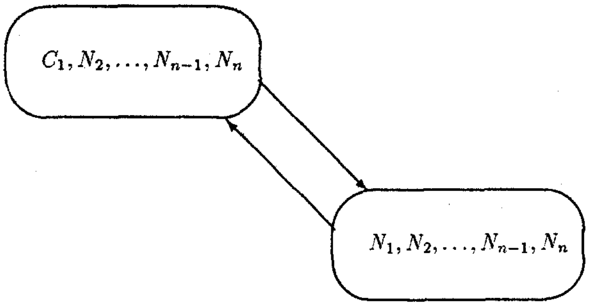
$C_1, N_2, \ldots, N_{n-1}, N_n$
$N_1, N_2, \ldots, N_{n-1}, N_n$

Figure 4: Quotient of Model for Two State n Process Mutual Exclusion