From Asymmetry to Full Symmetry: New Techniques for Symmetry Reduction in Model Checking[*]

E. Allen Emerson and Richard J. Trefler

Department of Computer Sciences and Computer Engineering Research Center
University of Texas, Austin, TX, 78712, USA

Abstract. It is often the case that systems are “nearly symmetric”; they exhibit symmetry in a part of their description but are, nevertheless, globally asymmetric. We formalize several notions of near symmetry and show how to obtain the benefits of symmetry reduction when applied to asymmetric systems which are nearly symmetric. We show that for some nearly symmetric systems it is possible to perform symmetry reduction and obtain a bisimilar (up to permutation) symmetry reduced system. Using a more general notion of “sub-symmetry” we show how to generate a reduced structure that is simulated (up to permutation) by the original asymmetric program.

In the symbolic model checking paradigm, representing the symmetry reduced quotient structure entails representing the BDD for the orbit relation. Unfortunately, for many important symmetry groups, including the full symmetry group, this BDD is provably always intractably large, of size exponential in the number of bits in the state space. In contrast, under the assumption of full symmetry, we show that it is possible to reduce a textual program description of a symmetric system to a textual program description of the symmetry reduced system. This obviates the need for building the BDD representation of the orbit relation on the program states under the symmetry group. We establish that the BDD representing the reduced program is provably small, essentially polynomial in the number of bits in the state space of the original program.

1 Introduction

Model checking [CE81] (cf. [QS82,LP85] [CES86]) is an algorithmic method for determining whether a finite state system, M, satisfies a temporal logic formula, f. Lichtenstein and Pnueli [LP85] have argued that in practice the complexity of model checking will be dominated by $|M|$, the size of M. Unfortunately, $|M|$ may be exponentially larger than the textual description of M. For example, a system comprised of n identical processes running in parallel, each of which has 3 local states, may have 3^n reachable states.

[*] The authors' work was supported in part by NSF grant CCR-980-4736 and SRC contract 98-DP-388.

L. Pierre and T. Kropf (Eds.): CHARME'99, LNCS 1703, pp. 142–157, 1999.

Symmetry reduction is an abstraction technique which endeavors to substantially ameliorate this state explosion problem by exploiting the fact that many computer systems are symmetric in their design and implementation (cf. [JR91,ID96,ES96,CE+96,HI+95,MAV96,ES97,GS97,ET98,AHI98]). Such symmetry can be seen to be a form of redundancy from the standpoint of model checking temporal logic formulae. The state graph, M, of many synchronization and coordination protocols which are the parallel composition of n processes identical up to renaming, often exhibits considerable symmetry. For example, the mutual exclusion protocol contains states (C_1, T_2) and (T_1, C_2) representing the states where process 1 is in its critical section and process 2 is attempting to reach its critical section and vice versa. These two states are related by the permutation (1 2) which drives process index 1 to 2 and 2 to 1; in general the permutation (1 2) when applied systematically to the states and transitions of M results in M again, that is (1 2) is an automorphism of M. Aggregating states which are equivalent up to permutation factors out the symmetry of a system and model checking is then performed on the symmetry reduced structure – a substantial, often exponential, savings can be achieved.

While symmetry reduction methods offer great potential, there are several obstacles to its more widespread application. Firstly, it is often the case that protocols are not symmetric; they may contain a high degree of symmetry in some part of their design but their global behavior is asymmetric. This can occur, for instance, in systems with processes which are identical up to renaming and the assignment of priorities. The readers–writers protocol, a refinement of the mutual exclusion protocol, is one such example. In the mutual exclusion algorithm the two processes competing for access to their critical sections are given equal priority; in the readers–writers protocol the writer is given priority. While the global state graph of the readers–writers protocol is asymmetric, it is symmetric in every aspect except the transition from the state where both processes are attempting to access their critical sections.

Secondly, BDD [Br86] based symbolic representation of the symmetry relation used in forming symmetry reduced quotient structures can be proved to be of exponential size. From this it has been argued that symmetry and symbolic representation [Mc92,BC+92] do not combine profitably [CE+96].

We describe solutions to both these problems in this paper. Previous work on symmetry reduction ([ES96] cf. [CE+96]) defined the symmetry reduced structure, $\overline{M}$, as the quotient structure of M induced by the equivalence relation, $\equiv_G$, on states. Two states, s and s', are equivalent, $s \equiv_G s'$, iff there is an automorphism π in G which drives s to s'. We relax this relationship by defining a permutation π to be a *near automorphism* iff for every transition $s \rightarrow t$ in M either $\pi(s) \rightarrow \pi(t)$ is in M or s is invariant under the automorphisms of S, the set of states of M. The equivalence relation on states induced by the group of near automorphisms defines a quotient structure that is bisimilar, up to permutation, to M. Therefore, even asymmetric structures can be near symmetry reduced.

Near automorphisms are, however, restrictive in the sense that whenever $\pi(s) \rightarrow \pi(t)$ is not a transition then s must be a highly symmetric state. By weakening the requirements for the preservation of transitions by permutations we can apply these ideas to a wider class of problems. Specifically, we define a notion of *rough symmetry* for multi-process systems whose processes are 'almost-symmetric'. Intuitively, a system is roughly symmetric if for every pair of processes i and j, the actions of process i from local state P, in global state s, can be mimicked by process j when j is the highest priority process in local state P, in the equivalent global state s'. We then show that the rough symmetry reduced system is bisimilar to the original system M.

By further weakening the restrictions on permutations applied to structures, we define a notion of *sub-symmetry* which allows for the creation of an abstract symmetry reduced structure that is simulated by the original program. A permutation π is a *sub-automorphism* of M if π drives certain "closed" subgraphs of M back into M. This notion of sub-automorphism induces a pre-order $\leq_H$ on states such that $s \leq_H t$ iff there is a sub-automorphism π which drives a closed subgraph containing s back into M and $\pi(s) = t$. We then use $\leq_H$ to define a sub-symmetry reduced structure, $\overline{M}_{\leq_H}$, which is simulated up to permutation by M, thereby showing that $\forall$CTL* [CGL94] formulae true of $\overline{M}_{\leq_H}$ are true of M.

Finally, we show how to successfully combine symmetry with BDD-based symbolic representations of systems. For many symmetry groups, including the full symmetry group, the BDD for the orbit relation, that is for determining equivalence of two states under group action, must always be of exponential size. This orbit BDD is used to permit designation of a *specific representative* state for each equivalence class in the quotient structure. The orbit BDD must recognize as equivalent, say, the states (N_1, N_2, T_3), (N_1, T_2, N_3), and (T_1, N_2, N_3). A specific, actual state is chosen as a representative. In the case of full symmetry, we can instead use *generic representatives*, for example, $(2N, 1T)$, which obviates the need for representation of the orbit relation. This is accomplished by compiling the program text of the fully symmetric program P into the program text of the symmetry reduced program $\overline{P}$ over generic states. $\overline{P}$ defines a structure $M(\overline{P})$ isomorphic (and bisimilar up to permutation) to the symmetry reduced structure $\overline{M}$ and model checking can then be performed on $M(\overline{P})$. Assuming that M is composed of n processes, this compilation process not only obviates the need for determining the equivalence of states under permutation but also reduces the number of bits used to represent a state in the symmetry reduced program from $\mathcal{O}(n)$ in the case of $\overline{M}$ to $\mathcal{O}(\log n)$ in the case of $M(\overline{P})$. A consequence is that the BDD representing $M(\overline{P})$ is always of polynomial size in the number of processes, in contrast to the exponential size BDD based on specific representatives.

The remainder of the paper is organized as follows: Section 2 contains some preliminaries, Section 3 discusses model checking asymmetric systems, compilation of fully symmetric programs into symmetry reduced programs is outlined in Section 4 and Section 5 contains a brief conclusion.

2 Preliminaries

Our model of computation, presented formally below, can be seen to represent the interleaved computations of a program composed of n communicating processes. States are n-tuples of local states, one for each process. Transitions represent the movement of one process from one local state to another. Permutations on the set $[1..n]$ can then be interpreted as permutations of process indices.

We denote the set of natural numbers by $\mathbb{N}$. Let $\mathcal{I}$ be a finite index set $[1..n]$ for some $n \in \mathbb{N}$, $n > 0$. LP is a finite set of local states. $Sym\ \mathcal{I}$ is the set of permutations on index set $\mathcal{I}$. $M = (S, R)$ is a structure where $S \subseteq LP^n$ and $R \subseteq S \times S$ is non-empty and total. We write both $(s, t) \in R$ and $s \to t \in R$ to mean that there is a transition from state s to state t in R. For $l \in LP$, $i \in [1..n]$ and $s \in S$ we write $(l, i) \in LP \times \mathcal{I}$ as l_i and $s(i) = l$ (l_i is true at s) iff the ith element of s is l.

A permutation $\pi \in Sym\ \mathcal{I}$ acts on a state $s \in S$ in the following way: $s(i) = l$ iff the $\pi(i)$th element of $\pi(s)$ is l. π is an automorphism of $M = (S, R)$ iff $S = \{\pi(s) \mid s \in S\}$ and $R = \{(\pi(s), \pi(t)) \mid (s, t) \in R\}$. Attention is usually restricted to such permutations because they preserve both the state space and the transition relation of the structure, M. A state s is said to be fully symmetric if for all $\pi \in Sym\ \mathcal{I}$, $\pi(s) = s$. The identity permutation is denoted by id. For any M, $Aut(M)$ the set of automorphisms of M, is a group. Similarly, for state s, $Aut(s)$ is the set of permutations, π, such that $\pi(s) = s$ and $Aut(S)$ is the set of permutations, π, such that $\pi(S) = S$.

Any subgroup G of $Aut(M)$, induces the following equivalence relation, $s \equiv_G t$ iff there exists a $\pi \in G$ such that $\pi(s) = t$. M's symmetry reduced structure, with respect to G, $\overline{M} = M/_{\equiv_G} = (\overline{S}, \overline{R})$ is defined as follows: $\overline{S} = \{\overline{s} \in S \mid \overline{s}$ is the unique representative of the equivalence class $[\overline{s}]_{\equiv_G}\}$ [1] and $(\overline{s}, \overline{t}) \in \overline{R}$ iff there exists $(\overline{s}, t) \in R$ for some $t \equiv_G \hat{t}$ [ES96,CE+96] (cf. [ES96] for more details).

In the sequel we will make use of the expressive branching time temporal logic CTL* [EH86] (cf. [Em90] for more details). Let $LP \times \mathcal{I}$ be the set of atomic propositions. A path formula is formed from boolean combinations ($\wedge, \vee, \neg$) and nestings of atomic propositions, state formulae and the usual temporal operators , G, F, U and V (the dual of U). State formulae are formed from boolean combinations of atomic propositions, state formulae and prefixing of path formulae by path quantifiers A and E. For example, the formula $\mathsf{AG}\neg(writerC \wedge readerC)$ says that along all computations it is never the case that both the *writer* and the *reader* are accessing their critical sections. We write $M, s \models f$ to denote that state s in structure M satisfies formula f and $M \models f$ to denote that there is a state, s, in M such that $M, s \models f$. A formula is in positive normal form (PNF) if the $\neg$ operator appears only in front of atomic propositions. ECTL* is the sub-logic of CTL* in which every formula, when put in PNF, contains only E path quantifiers. Similarly, ACTL* is the sub-logic of CTL* in which every formula, when put in PNF, contains only A path quantifiers [CGL94].

[1] $\overline{s}$ is the distinguished element of S and $[\overline{s}]_{\equiv_G}$ is the set of $s \in S$ such that $s \equiv_G \overline{s}$.

We define symmetric versions of CTL* and its sub-logics simply for ease of exposition – all our results can be restated to handle full CTL*[2]. The syntax of Symmetric CTL* (SCTL*) is the same as for CTL* except that the atomic formulae are restricted to the following: $\forall i : l_i$, $\exists i : l_i$, $\forall i : \neg l_i$, $\exists i : \neg l_i$ and $\exists i \neq j : l_i \wedge l_j$. For example, $\mathsf{AG}\neg(\exists i \neq j : C_i \wedge C_j)$ is a formula of SACTL*.

2.1 Simulation up to Permutation

Let $M = (S, R)$ and $M' = (S', R')$ be structures defined over LP and $\mathcal{I}$. $B \subseteq S \times S'$ is a simulation up to permutation (cf. [Mi71,Pa81] [HM85] [MAV96,ES96,CE+96]) iff for all $(s, s') \in B$

- there is a $\pi \in \mathit{Sym}\ \mathcal{I}$ such that $\pi(s) = s'$ and
- for all $(s, t) \in R$ there is a t' such that $(s', t') \in R'$ and $(t, t') \in B$.

$B \subseteq S \times S'$ is a bisimulation up to permutation iff for all $(s, s') \in B$ the above two conditions hold and

- for all $(s', t') \in R'$ there is a t such that $(s, t) \in R$ and $(t, t') \in B$.

Proposition 1. *([ES96,CE+96]) Let B be a bisimulation up to permutation. For all $(s, s') \in B$ and all SCTL* formulae f, $M, s \models f$ iff $M', s' \models f$.*

Proposition 2. *([ES96,CE+96]) Let B be a simulation up to permutation. For all $(s, s') \in B$ and all SACTL* formulae f, $M', s' \models f$ implies $M, s \models f$.*

Proposition 3. *([ES96,CE+96]) Let B be a simulation up to permutation. For all $(s, s') \in B$ and all SECTL* formulae f, $M, s \models f$ implies $M', s' \models f$.*

3 Symmetry Reduction and Asymmetric Systems

3.1 Near Automorphism Based Reductions

Let $M = (S, R)$ be a structure. A permutation π is a *near automorphism* of M if $\pi(S) = S$ and for all $(s, t) \in R$ either $Aut(S) \subseteq Aut(s)$ or $\pi(s) \rightarrow \pi(t) \in R$. Let $NAutM = \{\pi \in \mathit{Sym}\ \mathcal{I} \mid \pi$ is a near automorphism of $M\}$.

Theorem 1. *Given $M = (S, R)$ the set $NAutM$ is a group.*

Corollary 1. *$\overline{M}_{NAut} = M/\equiv_{NAutM} = (\overline{S}, \overline{R})$ is bisimilar up to permutation to $M = (S, R)$ and for all $(s, \overline{s})$ such that $s \equiv_{NAutM} \overline{s}$, and for all SCTL* formulae f, $M, s \models f$ iff $\overline{M}_{NAut}, \overline{s} \models f$.*

[2] When model checking formula f over $\overline{M} = M/\equiv_G$ it is required that for every maximal propositional sub-formula p of f, and every permutation $\pi \in G$, $\pi(p) \equiv p$ [ES96,CE+96]. SCTL*, SACTL*, and SECTL* all satisfy this requirement.

We can apply these ideas to the readers-writers problem as given in figure 1. The flip permutation (1 2) which drives index 1 to index 2 and vice versa is a near automorphism. This implies that the structure in the figure has the full symmetry group, $Sym\ \mathcal{I}$, as its group of near automorphisms and therefore the near symmetry reduced structure given in figure 2 is bisimilar up to permutation to the structure in figure 1. Model checking for safety formulae like $\mathsf{AG}\neg(\exists i \neq j : C_i \wedge C_j)$ and liveness formulae like $\mathsf{AG}[(\exists i : T_i) \Rightarrow \mathsf{AF}(\exists i : C_i)]$ – which says that along all computations it is always the case that if some process is trying to enter its critical section then it is inevitable that some process enters its critical section – can then be performed on the near symmetry reduced structure $\overline{M}_{NAut}$ instead of M.

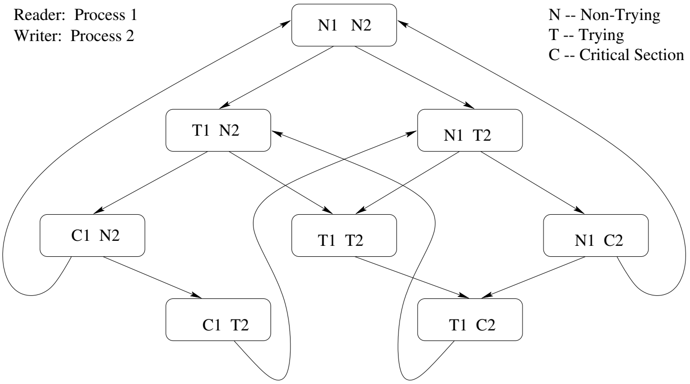

Fig. 1. Asymmetric Readers-Writers

Figure 3 contains the program skeletons which generate the structure M in figure 1. The near automorphisms for M can be generated directly from the program skeletons through the following observation. While the skeletons are not symmetric they are nearly symmetric in the following sense. Ignoring, for the moment, the transition $T_2 \to C_2$ that is enabled when T_1 is true, the two skeletons are symmetric – the flip permutations applied to the skeletons results in the same two skeletons. The asymmetry of the transition $T_2 \to C_2$ that is enabled when T_1 is true guarantees a near symmetry of the induced Kripke structure because this symmetry breaking transition is only enabled from the fully symmetric state (T_1, T_2). In the full paper we give a more detailed algorithm for determining near automorphisms from program skeletons.

Finally, we note that the near symmetry reduced quotient structure $\overline{M}_{NAut} = M/\equiv_{NAut}$ can be built directly from the program text without building M in a manner analogous to that used to build $\overline{M}$. Basically, the procedure works as follows, given a state $\overline{s} \in \overline{S}$ generate each of the states t such that $\overline{s} \to t \in R$ as described by the program text. For each t if t is equivalent to a state $\overline{t} \in \overline{S}$ then add an arc $\overline{s} \to \overline{t}$ to $\overline{R}$ otherwise add t to $\overline{S}$ and the arc $\overline{s} \to \overline{t}$ to $\overline{R}$ (see [ES96] for complete details).

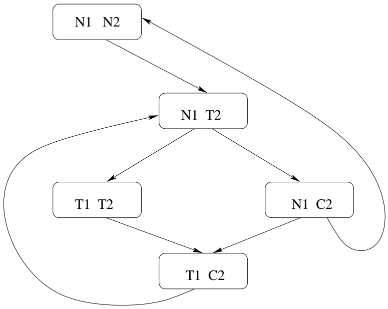

Fig. 2. Near Symmetry Reduced Asymmetric Readers-Writers

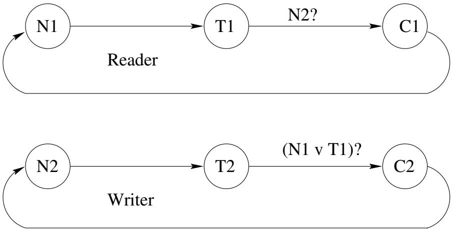

Fig. 3. Readers-Writers Program Skeletons

3.2 Rough Symmetry Based Reduction

The previous section showed how the definition of near automorphism was sufficient to justify applying symmetry reduction techniques to asymmetric systems such as the reader-writer problem. That technique required a permutation to act on the structure by preserving transitions: $s \rightarrow t$ implies $\pi(s) \rightarrow \pi(t)$ or s is a symmetric state. Requiring s to be symmetric, however, implies that the technique cannot be used to handle the more general readers-writers problems. We therefore seek a relationship which will allow extensive reduction and generality. Below we formulate a notion of equivalence based on roughly symmetric programs which again leads to a bisimulation between M and the rough symmetry reduced version of M. Within this framework we can show that a more general version of the readers-writers problem may be symmetry reduced.

The intuition behind our approach is as follows. We suppose that $M = (S, R)$ is the structure corresponding to a multi-process system in which the processes have been assigned priorities. Furthermore, we assume that some portion of M is highly symmetric. For instance, in the multiple readers-writers protocol, by ignoring the extra functionality of a writer over a reader, it is possible to see this protocol as 'fully symmetric'. We view this highly symmetric M' as a 'sub-structure' of M from which states and actions have either been removed or added in some systematic way to form M. Taking the group, G, of automorphisms from this sub-structure we then seek to show that for every pair, (s, s'), of G

equivalent states and every transition $s \to t$ of process i in M is preserved by some permutation π which drives s to s' and the transition $s \to t$ to an equivalent transition by the highest priority process, $\pi(i)$. We then say that M is roughly symmetric with respect to the symmetry group of M'.

Formally, let $M = (S, R)$ where $R = R_1 \cup \ldots \cup R_n$, each $R_i \subseteq S \times S$. R_i is the transition relation for process i. Let G be a sub-group of $\mathit{Sym}\ \mathcal{I}$. We say that R_i is covered (with respect to G) by R_j, iff for all transitions $s \to t \in R_i$ and all $s' \equiv_G s$, if $j = \max\{j' \mid s(i) = s'(j')\}$ then there is a permutation $\pi \in G$ such that $\pi(s) = s'$, $\pi(i) = j$, $s' \to \pi(t) \in R_j$. Then we say that M is roughly symmetric with respect to G iff for all $i, j \in \mathcal{I}$, R_i is covered by R_j.

Theorem 2. *Suppose $M = (S, R)$, $R = R_1 \cup \ldots, R_n$ is roughly symmetric with respect to Then $\overline{M} = M/\equiv_G \ = (\overline{S}, \overline{R})$ is bisimilar up to permutation to M.*

Proof: Let $B = \{(s, \overline{s}) \in S \times \overline{S} \mid s \equiv_G \overline{s}\}$. Let $s \equiv_G \overline{s}$ for some $s \in S$, $\overline{s} \in \overline{S}$. Then suppose $s \to t \in R_i$. Let $j = \max\{j' \mid \overline{s}(j') = s(i)\}$, then there is a $\pi \in G$ such that $\pi(s) = \overline{s}$, $\pi(i) = j$ and $\overline{s} \to \pi(t) \in R_j$. This implies $\overline{s} \to \overline{t} \in \overline{R}$ for some $\overline{t} \equiv_G \pi(t)$ which implies that $t \equiv_G \overline{t}$ and $(t, \overline{t}) \in B$. Suppose $\overline{s} \to \overline{t} \in \overline{R}$. Then for some i and $t' \equiv_G \overline{t}$, $\overline{s} \to t' \in R_i$. Let $j \max\{j' \mid s(j') = \overline{s}(i)\}$. Then there is some $\pi \in G$ such that $\pi(\overline{s}) = s$, $\pi(i) = j$ and $s \to \pi(t') \in R_j$. But then $\pi(t') \equiv_G t'$ which implies $\pi(t') \equiv_G \overline{t}$ and hence $(\pi(t'), \overline{t}) \in B$. □

We can apply these ideas to show that a general readers-writers system can be symmetry reduced by the full symmetry group to a bisimilar rough symmetry reduced quotient structure. We have $m < n$ identical readers and $n - m$ identical writers. The writers all have priority over all the readers but no two processes may access their critical sections at the same time. The generic process skeletons for these processes are identical to the ones in figure 3 except that for reader i the arc from state T_i to state C_i is labeled by $(N_1 \vee T_1) \wedge \ldots (N_{i-1} \vee T_{i-1}) \wedge (N_{i+1} \vee T_{i+1}) \wedge \ldots \wedge (N_m \vee T_m) \wedge N_{m+1} \wedge \ldots \wedge N_n$ and for writer j the arc from state T_j to state C_j is labeled by $(N_1 \vee T_1) \wedge \ldots \wedge (N_{j-1} \vee T_{j-1}) \wedge (N_{j+1} \vee T_{j+1}) \wedge \ldots \wedge (N_n \vee T_n)$. We now show that $M = (S, R)$ for this readers writers system is roughly symmetric with respect to $\mathit{Sym}\ \mathcal{I}$. Consider $s \to t \in R_i$ and $s \equiv_{\mathit{Sym}\ \mathcal{I}} s'$. Let $j = \max\{j' \mid s'(j') = s(i)\}$. Since there is a permutation mapping s to s' there is a permutation π such that $\pi(s) = s'$ and $\pi(i) = j$. Suppose $s \to t \in R_i$ follows from $N_i \to T_i$, then $\pi(s) \to \pi(t) \in R_j$. Similarly if $s \to t \in R_i$ is due to $C_i \to N_i$ then $\pi(s) \to \pi(t) \in R_j$. Suppose $s \to t \in R_i$ follows from either a writer or a reader entering its critical section. If i is a reader then $(N_1 \vee T_1) \wedge \ldots (N_{i-1} \vee T_{i-1}) \wedge (N_{i+1} \vee T_{i+1}) \wedge \ldots \wedge (N_{m-1} \vee T_{m-1}) \wedge N_m \wedge \ldots \wedge N_n$ or if i is a writer then $(N_1 \vee T_1) \wedge \ldots \wedge (N_{j-1} \vee T_{j-1}) \wedge (N_{j+1} \vee T_{j+1}) \wedge \ldots \wedge (N_n \vee T_n)$. Then consider that j is the largest index such that $s'_j = T$. If j is a writer it cannot be blocked. If j is a reader it cannot be blocked because no writer is in its T section. Therefore process j can enter it's critical section from state s' and we have that $\pi(s) \to \pi(t) \in R_j$. Therefore, the readers-writers system is roughly symmetric with respect to $\mathit{Sym}\ \mathcal{I}$ and an exponential savings may be achieved through rough symmetry reduction.

3.3 Simulation Based Reductions and Asymmetry

Given $M = (S, R)$, let S' be a subset of S. S' is closed (with respect to M) iff for all $s \in S'$ and all $(s, t) \in R$, $t \in S'$. Let $\pi \in Sym\ \mathcal{I}$ and $S' \subseteq S$ be closed. π is a sub-automorphism on S' iff $\{\pi(s) \mid s \in S'\} \subseteq S$ and for all $s, t \in S'$ if $s \to t \in R$ then $\pi(s) \to \pi(t) \in R$. Let H be the subset of $Sym\ \mathcal{I} \times 2^S$ such that $(\pi, S') \in H$ iff π is a sub-automorphism on the closed subset of S, S'. $s \leq_H t$ iff there is a $(\pi, S') \in H$ such that $s \in S'$ and $\pi(s) = t$.

Proposition 4. *$s \leq_H t$ and $t \leq_H u$ implies $s \leq_H u$.*

Proof: $s \leq_H t$ implies there is some closed $S' \subseteq S$ and π such that $(\pi, S') \in H$ and $\pi(s) = t$. Furthermore, there is some closed $S'' \subseteq S$ and ϕ such that $(\phi, S'') \in H$ and $\phi(t) = u$. Consider $T \subseteq S'$ such that T contains s and all the states reachable from s in S'. S' closed implies such a T exists and is a closed subset of S'. $\pi(T) \subseteq S''$ is straight forward. This implies that for all $s, t \in T$, $(\phi \cdot \pi)(s) \in S$ and if $(s, t) \in R$, $(\phi \cdot \pi)(s) \to (\phi \cdot \pi)(t) \in R$ which implies that $((\phi \cdot \pi), T) \in H$. Since $(\phi \cdot \pi)(s) = u$ it is the case that $s \leq_H u$. □

For, $\overline{s} \in S$, define $[\overline{s}]_{\leq H} = \{s \in S | s \leq_H \overline{s}\}$. Then a sub-symmetry reduced version of M is $\overline{M}_{\leq_H} = M/\leq_H = (\overline{S}, \overline{R})$ where

- $\overline{S} \subseteq S$ and
- for all $s \in S$ there is an $\overline{s} \in \overline{S}$ such that $s \in [\overline{s}]_{\leq H}$ and
- $(\overline{s}, \overline{t}) \in \overline{R}$ iff there is some $t \leq_H \overline{t}$ such that $(\overline{s}, t) \in R$.

Let $M = (S, R)$ and $\overline{M}_{\leq_H} = (\overline{S}, \overline{R})$ be structures as described above. Then let $B = \{(s, \overline{s}) \in S \times \overline{S} \mid s \in [\overline{s}]_{\leq H}\}$.

Theorem 3. *B is a simulation up to permutation.*

Proof: Suppose $(s, \overline{s}) \in B$ then $s \in [\overline{s}]$ and there is a $(\pi, S') \in H$, such that $\pi(s) = \overline{s}$. Suppose $(s, t) \in R$. This implies that $(\pi(s), \pi(t)) \in R$. By the structure of $\overline{M}_{\leq_H}$ this implies that there is some $\overline{t}$ such that $(\overline{s}, \overline{t}) \in \overline{R}$ and $\pi(t) \leq_H \overline{t}$. But this implies that $t \leq_H \overline{t}$ hence $(t, \overline{t}) \in B$. □

Corollary 2. *For all SACTL* formulae, f, and for all $(s, \overline{s}) \in B$, $\overline{M}_{\leq_H}, s \models f$ implies $M, s \models f$.*

In fact, this type of reduction is possible for any $H' \subseteq H$. $\leq_{H'}$ is defined as above and $\leq^*_{H'}$ is the reflexive, transitive closure of $\leq_{H'}$. The sub-symmetry reduced system, $\overline{M}_{\leq^*_{H'}} = M/\leq^*_{H'} = (\overline{S}, \overline{R})$ is then defined analogously.

Proposition 5. *If $\leq_H$ is symmetric then $\leq_H$ is an equivalence relation.*

Theorem 4. *Let $M = (S, R)$ and $\overline{M}_{\leq_H} = (\overline{S}, \overline{R})$ be as above and let $\leq_H$ be symmetric, then $B = \{(s, \overline{s}) \in S \times \overline{S} | s \in [\overline{s}]_H\}$ is a bisimulation up to permutation.*

Proof: Let $(s, \overline{s}) \in B$. Suppose $\overline{s} \rightarrow \overline{t} \in \overline{R}$, then there is some t such that $\overline{s} \rightarrow t \in R$ and $t \leq_H \overline{t}$. $s \leq_H \overline{s}$ implies $\overline{s} \leq_H s$ which implies there is some $(\phi, T') \in H$ such that $\overline{s} \in T'$ and $\phi(\overline{s}) = s$. Hence $s \rightarrow \phi(t)$ which implies $t \leq_H \phi(t)$ and therefore $\phi(t) \leq_H t$. This implies $\phi(t) \leq_H \overline{t}$ and therefore $(\phi(t), \overline{t}) \in B$. □

Corollary 3. *For all SCTL* formulae, f, and all $(s, \overline{s}) \in B$, $M, s \models f$ iff $\overline{M}_{\leq_H}, \overline{s} \models f$*

When $\leq_H$ (or $\leq_{H'}$) can be determined from the program text or is given *a priori* then it is possible to build the sub-symmetry reduced structure $\overline{M}_{\leq_H}$ directly from the program text without first constructing M. The procedure is analogous to building $\overline{M} = M/\equiv_{Aut(M)}$, however, it may require some backtracking as it is possible that a state s is generated in $\overline{M}_{\leq_H}$ which can then be replaced by a state $\overline{s}$ such that $s \leq_H \overline{s}$.

4 Symmetry Reduction on Fully Symmetric Programs

Representing symmetry reduced structures with BDD's is, typically, computationally intractable. The BDD representing the orbit relation of many groups, including the full symmetry group, is of size exponential in the number of processes or the number of bits in a state. In the sequel, we show that under the assumption of full symmetry, symmetry reduction can be done efficiently in the symbolic model checking paradigm without representation of the orbit relation. Let $k = |LP|$, be the number of local states of an individual process P_i. Given a program $P = //_{i \in [1..n]} P_i$, the parallel composition of n processes identical up to renaming, which defines a fully symmetric Kripke structure $M(P)$, we compile P into a program $\overline{P}$, in time linear in the size of P. $\overline{P}$ defines a symmetry reduced quotient structure $M(\overline{P})$ which is isomorphic to $\overline{M(P)}$. However, each specific representative in $\overline{M(P)}$ is replaced by its corresponding generic representative in $M(\overline{P})$. $M(\overline{P})$ can then be used to model check $M(P)$ without having to represent the orbit relation for the symmetry group on the states of $M(P)$. We have then reduced a problem of worst case size k^n which is exponential in n, to one of worst case size n^k which is polynomial for any fixed number k of local states. Furthermore, the number of bits required to symbolically represent a state has been decreased from $\mathcal{O}(n \log k)$ in $\overline{M(P)}$, the standard quotient, to $\mathcal{O}(k \log n)$ in $M(\overline{P})$ the generic quotient. We then show that in many cases the transitions in $\overline{P}$ can be represented by BDD's polynomial in the size of the text of $\overline{P}$.

The key idea is that a generic representative can be chosen for each of the equivalences classes of states under the assumption of full symmetry [ES96,CE+96]. Equivalence under full symmetry means that two states $s, t \in LP^n$ are equivalent iff they have exactly the same number of processes in local state l for each state $l \in LP$. Hence the generic representative needs only track the number of processes in each local state and not any information regarding which processes are in a particular local state.

Let a program $P = //_{i \in [1..n]} P_i$ be the parallel composition of processes $P_1, \ldots, P_n$ which are identical up to renaming. Each process is specified by a program skeleton similar to the ones in figure 3. The skeletons give rise to generic transitions of the processes which are specified by $l : g \rightarrow l'$ where $l, l' \in LP$ are local states and g is a guard. Guards are positive boolean combinations of the following elements: $\forall j : l_j$, $\forall j : \neg l_j$, $\exists j : l_j$, $\exists j : \neg l_j$ and $\exists j \neq j' : l_j \wedge l_{j'}$. Since the processes are identical up to renaming this syntax gives rise to fully symmetric structures.

The intended meaning of $l_i : g \rightarrow l'_i$ is that if P is in state s, where process i is in local state l_i and guard g is true of s then P may transit to the state t, everywhere the same as s, except that process i is in state l'_i. P executes the enabled transitions – there may be multiple enabled transitions for a single process – non-deterministically. We further stipulate that P defines an initial state s_0 of the form $l^n = (l_1, \ldots, l_n)$ for some $l \in LP$.

Given $P = //_{i \in [1..n]} P_i$ with initial state l^n, as above, P defines a Kripke structure $M(P) = (S, R, s_0)$ as follows: $s_0 = l^n$ is the initial state, $S = LP^n$ and $s \rightarrow t \in R$ iff there exists a generic transition statement $l : g \rightarrow l'$ such that $s(i) = l$, $t(i) = l'$, g is true at s and for all $i' \neq i$, $s(i') = t(i')$. For a Kripke structure M with an initial state s_0, we say that $M \models f$ iff $M, s_0 \models f$. $\overline{M(P)} = M(P)/\equiv_{Sym\ \mathcal{I}} \ = (\overline{S}, \overline{R}, s_0)$ is the symmetry reduced quotient structure.

Theorem 5. *[ES96] For any SCTL* formula f, $M(P), s_0 \models f$ iff $\overline{M(P)}$, $s_0 \models f$.*

We define the symmetry reduced program $\overline{P}$ as follows: $\overline{P}$ has variables $x_1, \ldots, x_k$ each of type $[0..n]$ and we assume the existence of a bijective function $\iota : LP \rightarrow [1..k]$. Suppose each process P_i has c different transitions of the form $l_i : g \rightarrow l'_i$ each generated by the generic transition $l : g \rightarrow l'$. Then $\overline{P}$ has c transitions of the form $x_{\iota(l)} > 0 \wedge \mathcal{T}(g) \rightarrow x_{\iota(l)}, x_{\iota(l')} := x_{\iota(l)} - 1, x_{\iota(l')} + 1$. The intended meaning being that if $\overline{P}$ is in a state $s \in [0..n]^k$ where the variable $x_{\iota(l)} \geq 0$ and the guard $\mathcal{T}(g)$ is true, then $\overline{P}$ may non-deterministically transit to a state $t \in [0..n]^k$ such that $x_{\iota(l)}$ has decreased by 1, $x_{\iota(l')}$ has increased by 1 and all other variables are unchanged.

The symmetry reduced guard $\mathcal{T}(g)$ is derived from g as follows: $\mathcal{T}(\forall j : l_j) =$'$x_{\iota(l)} = n$', $\mathcal{T}(\forall j : \neg l_j) =$ '$x_{\iota(l)} = 0$', $\mathcal{T}(\exists j : l_j) =$'$x_{\iota(l)} > 0$', $\mathcal{T}(\exists j : \neg l_j) =$ '$x_{\iota(l)} < n$', $\mathcal{T}(\exists j \neq j' : l_j \wedge l_{j'}) =$'$x_{\iota(l)} \geq 2$', $\mathcal{T}(g_1 \vee g_2) = \mathcal{T}(g_1) \vee \mathcal{T}(g_2)$ and $\mathcal{T}(g_1 \wedge g_2) = \mathcal{T}(g_1) \wedge \mathcal{T}(g_2)$. Finally, if the initial state of P is l^n then the initial state of $\overline{P}$ is $x_{\iota(l)} = n$ and $x_{\iota(l')} = 0$ for all $l' \neq l$.

$\overline{P}$ defines a Kripke structure $M(\overline{P}) = (S', R', s'_0)$ as follows: if $x_i = n$ and for all $i' \neq i$, $x_{i'} = 0$ is the initial state of $\overline{P}$ then $s'_0(i) = n$ and for all $i' \neq i$, $s'_0(i') = 0$, $S' = [0..n]^k$ and $R' \subseteq S' \times S'$ where $s \rightarrow t \in R'$ iff there is a transition in $\overline{P}$, of the form $x_{\iota(l)} \geq 0 \wedge \mathcal{T}(g) \rightarrow x_{\iota(l)}, x_{\iota(l')} := x_{\iota(l)} - 1, x_{\iota(l')} + 1$ where the $\iota(l)$th element of s is greater than 0, $\mathcal{T}(g)$ is true at s and for all $j \in [1..k]$, $j = \iota(l)$ implies $t(j) = s(j) - 1$, $j = \iota(l')$ implies $t(j) = s(j) + 1$ and otherwise $s(j) = t(j)$.

Theorem 6. *$M(\overline{P})$ is isomorphic to $\overline{M(P)}$.*

We can also show that $M(\overline{P})$ is bisimilar to $\overline{M(P)}$ by translating the labels of the states in $\overline{M(P)}$ into the generic state format, that is by representing only the number of processes in a particular local state. Similarly, by translating the formulae of SCTL* into this generic format we have the following result.

Corollary 4. *For all SCTL* formulae f, $M(\overline{P}), s'_0 \models f$ iff $M(P), s_0 \models f$*

In the sequel we describe how S' and R' can be succinctly represented by BDD's. States in S' are represented by tuples in $[0..n]^k$. Such a state space can be represented by $k{\cdot}(\log(n){+}1)$ boolean variables (for ease of explanation we assume that n is a power of two). Bits $b_0 \ldots b_{\log n}$ represent x_1, bits $b_{\log(n)+1} \ldots b_{2 \log n}$ represent the variable x_2, etc. Assuming that k is fixed, then generic states of S' can be represented in $\mathcal{O}(\log n)$ bits. It follows that, for any type of transition relation R' over S', the BDD representing R' is of size at most $poly(n)$. This should be contrasted with the size of the BDD representing the orbit relation in the conventional symmetry reduced quotient which has a lower bound $exp(\min(n,k))$ [CE+96]. But for this model of computation we can obtain better bounds as described below.

We now show that transitions of the form $x_{\iota(l)} \geq 0 \wedge \mathcal{T}(g) \rightarrow x_{\iota(l)}, x_{\iota(l')} := x_{\iota(l)} - 1, x_{\iota(l')} + 1$ can be represented succinctly when $\mathcal{T}(g)$ is of a particular form. Firstly, $x_{\iota(l)} \geq 0$ can be checked with a BDD of size $\mathcal{O}(log(n){+}1)$ since the BDD need only check that the bits $(\iota(l){-}1){\cdot}\log n \ldots [\iota(l){\cdot}\log n]{-}1$ are not all 0 (false). Consider the set of atomic boolean guards $\{x_j = n, x_j = 0, x_j > 0, x_j < n, x_j \geq 2\}$, for $j \in [1..k]$ and assume that $\mathcal{T}(g)$ is either a conjunction of atomic boolean guards or a disjunction of atomic boolean guards. For the case where $\mathcal{T}(g)$ is conjunctive, extend the set of atomic boolean guards to include $x_j > 0 \wedge x_j < n$ and $x_j < n \wedge x_j \geq 2$.

In a manner similar to the above it is possible to show that each of the extended atomic boolean guards is representable by a BDD polynomial in the number of bits used to represent the value of the variable which the guard restricts. Conjunctive guard $\mathcal{T}(g)$ can be rewritten so that it first mentions only those atomic boolean guards which mention variable x_1 then x_2 and so on. Consider the conjunctive portion of $\mathcal{T}(g)$ in which x_j occurs, $j \in [1..k]$. Under the assumption that $n \geq 1$, it is not hard to prove that any conjunctive combination of boolean atomic guards reduces to the constant 0 or a single instance of one of the extended set of conjunctive boolean guards. Since the BDD's for the separate variables in $\mathcal{T}(g)$ are independent, they can be put together to form the BDD for $\mathcal{T}(g)$ which is of size additive in the sizes of the BDD's for each of the separate variables and hence polynomial in the length of $\mathcal{T}(g)$.

A similar argument can be made for the case when $\mathcal{T}(g)$ is disjunctive. However, in that instance the set of atomic boolean guards is extended by $x_j = n \vee x_j = 0$ and $x_j = 0 \vee x_j \geq 2$. Furthermore, arbitrary disjunctions of the atomic boolean guards never result in the constant 0 (false) but they do result either in a single instance of the extended set of atomic boolean guards or the constant 1 (true). Finally, it is not hard to see that a BDD can be built to check whether two states are related by the assignments of the form $x_{\iota(l)}, x_{\iota(l')} := x_{\iota(l)} - 1, x_{\iota(l')} + 1$

which is of size polynomial in $k \cdot (\log(n) + 1)$. The bits representing the variable $x_{\iota(l)}$ $(x_{\iota(l')})$ increase (decrease) by 1 and all other variables remain unchanged. Finally, by combining all three sections of the BDD representing a transition we see that the BDD is at most cubic in $\mathcal{O}(k \cdot (\log(n) + 1))$ and hence polynomial in the size of the transition. These BDD's for individual program statements can be combined to get a BDD for R' of size $poly(n)$. However, they combine disjunctively which can be advantageous in terms of possible disjunctive partitioning.

When $P = ||_{i \in [1..n]} P_i$ is the synchronous composition of processes $P_1, \ldots, P_n$ a similar but slightly more complex translation is required. $\overline{P}_{||}$, the symmetry reduced program, contains two variables $x_{\iota(l)}$ and $x'_{\iota(l)}$ for each local state l. The generic transitions of the synchronous program P are translated in the same manner as the generic transitions in the asynchronous case except for the following: guards refer to unprimed variables while the assignments are made to the primed variables. Computation then proceeds in rounds. For each local state l, if the unprimed variable $x_{\iota(l)}$ has value b then up to b enabled transitions from place l – compiled transitions with $x_{\iota(l)} > 0$ in their guard – are executed. At the end of the round, each unprimed variable x_j is set to the value of the primed variable x'_j.

5 Conclusion

Many researchers have investigated the exploitation of symmetry in order to expedite verification but 'almost' symmetric designs have received little attention. A different type of partial symmetry has been explored in [HI+95], without precise formalization and only in relation to preservation of reachability properties of petri nets. Our formalizations of near and rough symmetry are new and our use of near and rough symmetries of M in the reduction of M to an abstract quotient structure is new. The term partial symmetry has been used for quite some time (cf. [Ko78]) in switching theory. There, however, a system is partially symmetric if its group of symmetries over index set $\mathcal{I}$ is isomorphic to the full symmetry group of an index set $\mathcal{I}' \subseteq \mathcal{I}$. This type of partial symmetry has been handled explicitly by [ES96] and [CE+96]. [AHI98] considers partial symmetry in a manner more analogous to our definition of sub-symmetry. However, they deal only with partial symmetries of the formula (or its automaton representation) to be model checked, rather than reduction of the structure itself. Abstraction of M, on the other hand, has the potential to be of greater benefit in ameliorating the state explosion problem [LP85]. We have shown that near automorphisms are sufficient for the preservation of temporal properties, a generalization of the results of [ES96,CE+96], we have extended these ideas to rough symmetries, and we have shown how to obtain simulated symmetry reduced quotient structures from asymmetric systems via sub-symmetries.

With respect to full symmetry, we have shown how to exploit the symmetry of program text without the need to represent the symmetry reduced Kripke structure or the orbit relation induced by the symmetry group. [ID96] deals with similar symmetry groups but they do so explicitly. That is, the state spaces are not represented by BDD's and they therefore a priori do not have to cope with the problem of representing the symmetry induced equivalence classes by a BDD. [CE+96] shows BDD's representing the orbit relation of the full symmetry group are of exponential size. They suggested a heuristic to mitigate this problem using multiple representatives, but did not prove it to yield a tractable representation in general. Our technique consists in compiling the symmetric program P with Kripke structure $M(P)$ to a symmetry reduced program $\overline{P}$ over generic states whose structure $M(\overline{P})$ is bisimilar up to permutation to $M(P)/\equiv_{Sym\ \mathcal{I}}$. This can be seen to be an example of the utility of compiling programs into Petri Nets to achieve an exponential reduction. We believe we are the first to show that it is just such a reduction strategy which can usefully combine symmetry reduction with BDD based state representation. Previous work on Petri Nets, BDD's and symmetry reduction has not dealt explicitly with the fact that BDD representation of the symmetry induced equivalence classes is a self-defeating proposition for many symmetry groups.

For the future, we are implementing a preprocessor front end to a symbolic model checking tool to take advantage of our results on full symmetry. We are also investigating extending our technique to a larger class of groups [CE+98] for which symmetry reduction can be applied directly to program text. With respect to near symmetry and full symmetry we are interested in exploring the applicability of our work here to symmetry reduction techniques which use the annotated symmetry reduced structure which preserves the truth of all CTL* (and μ-calculus) properties [ES96,ES97,ET98].

Acknowledgment

The authors would like to thank Bob Kurshan for many stimulating comments and discussions.

References

AHI98. Ajami, K., Haddad, S. and Ilie, J.-M., Exploiting Symmetry in Linear Time Temporal Logic Model Checking: One Step Beyond. In *Tools and Algorithms for the Construction and Analysis of Systems, 4th Interntational Conference, ETAPS98* LNCS 1384, Springer Verlag, 1998. 143, 154

BC+92. Burch, J. R., Clarke, E. M., McMillan, K. L., Dill, D. L. and Hwang, L. J., Symbolic Model Checking: 10^{20} states and beyond. In *Information and Computation*, 98(2):142-170, June, 1992. 143

Br86. Bryant, R. E., Graph-Based Algorithms for Boolean Function Manipulation. In *IEEE Transactions on Computers*, Vol. C-35, No. 8, Aug. 86, pp. 677-691. 143

CE81. Clarke, E. M., and Emerson, E. A., Design and Verification of Synchronization Skeletons using Branching Time Temporal Logic. In *Logics of Programs Workshop*, Springer, LNCS no. 131., pp. 52-71, May 1981. 142

CE+98. Clarke, E. M., Emerson, E. A., Jha, S. and Sistla A. P., Symmetry Reductions in Model Checking. In *Computer Aided Verification, 10th International Conference* LNCS 1427, Springer- Verlag, 1998. 155

CES86. Clarke, E. M., Emerson, E. A., and Sistla, A. P., Automatic Verification of Finite State Concurrent System Using Temporal Logic. In *ACM Trans. on Prog. Lang. and Sys.*, vol. 8, no. 2, pp. 244-263, April 1986. 142

CE+96. Clarke, E. M., Enders, R., Filkorn, T., and Jha, S., Exploiting Symmetry in Temporal Logic Model Checking. In *Formal Methods in System Design*, Kluwer, vol. 9, no. 1/2, August 1996. 143, 145, 146, 151, 153, 154, 155

CGL94. Clarke, E. M., Grumberg, O. and Long, D. E., Model Checking and Abstraction. In *Transactions on Programming Languages and Systems* ACM, vol 16, no. 5, 1994. 144, 145

Em90. E. Allen Emerson, Temporal and Modal Logic. In J. van Leeuwen editor *Handbook of Theoretical Computer Science* vol. B, Elsevier Science Publishing, 1990. 145

EH86. Emerson, E. A., and Halpern, J. Y., 'Sometimes' and 'Not Never' Revisited: On Branching versus Linear Time Temporal Logic, *JACM*, vol. 33, no. 1, pp. 151-178, Jan. 86. 145

ES96. Emerson, E. A. and Sistla, A. P., Symmetry and Model Checking. In *Formal Methods in System Design*, Kluwer, vol. 9, no. 1/2, August 1996. 143, 145, 146, 147, 151, 152, 154, 155

ES97. Emerson, E. A. and Sistla, A. P., Utilizing Symmetry when Model Checking under Fairness Assumptions. In *TOPLAS* 19(4): 617-638 (1997). 143, 155

ET98. Emerson, E. A. and Trefler, R. J., Model Checking Real-Time Properties of Symmetric Systems. In *Mathematical Foundations of Computer Science, 23rd International Symposium* LNCS 1450, Springer-Verlag, 1998. 143, 155

GS97. Gyuris, V. and Sistla, A. P., On-the-Fly Model checking under Fairness that Exploits Symmetry. In *Proceedings of the 9th International Conference on Computer Aided Verification, Haifa, Israel*, 1997. 143

HI+95. Haddad, S., Ilie, J. M., Taghelit, M. and Zouari, B., Symbolic Reachability Graph and Partial Symmetries. In *Application and Theory of Petri Nets 1995*, Springer-Verlag, LNCS 935, 1995. 143, 154

HM85. Hennessy, M., Milner, R., Algebraic Laws for Nondeterminism and Concurrency. In *Journal of the ACM*, Vol 32, no. 1, January, 1985, pp 137-161. 146

ID96. Ip, C-W. N., Dill, D. L., Better Verification through Symmetry. In *Formal Methods in System Design*, Kluwer, vol. 9, no. 1/2, August 1996. 143, 154

JR91. Jensen, K. and Rozenberg, G. (eds.), High-Level Petri Nets: Theory and Application, Springer- Verlag, 1991. 143

Ko78. Kohavi, Zvi, *Switching and Finite Automata Theory*, second edition, McGraw-Hill Book Company, New York, 1978. 154

LP85. Lichtenstein, O., and Pnueli, A., Checking That Finite State Concurrent Programs Satisfy Their Linear Specifications, POPL85, pp. 97-107, Jan. 85. 142, 154

MAV96. Michel, F., Azema, P. and Vernadat, F., Permutable Agents in Process Algebra. In *Tools and Algorithms for the Construction and Analysis of Systems, 96*, Springer Verlag, LNCS 1055, 1996. 143, 146

Mi71. Milner, R., An Algebraic Definition of Simulations Between Programs. In *Proceedings of the Second International Joint Conference on Artificial Intelligence*, British Computer Society, 1971, pp 481-489. 146

Mc92. McMillan, K. L., *Symbolic Model Checking: An Approach to the State Explosion Problem*, Ph.D. Thesis, Carnegie Mellon University, 1992. 143

Pa81. Park, D., Concurrency and Automata on Infinite Sequences. In *Theoretical Computer Science: 5th GI-Conference, Karlsruhe*, Springer-Verlag, LNCS 104, pp 167-183, 1981. 146

QS82. Queille, J. P., and Sifakis, J., Specification and verification of concurrent programs in CESAR, Proc. 5th Int. Symp. Prog., Springer LNCS no. 137, pp. 195-220, 1982. 142