Utilizing Symmetry when Model Checking under Fairness Assumptions: An Automata-theoretic Approach

E. A. Emerson
University of Texas at Austin
and
A. P. Sistla
University of Illinois at Chicago

One useful technique for combating the state explosion problem is to exploit symmetry when performing temporal logic model checking. In previous work it is shown how, using some basic notions of group theory, symmetry may be exploited for the full range of correctness properties expressible in the very expressive temporal logic CTL*. Surprisingly, while fairness properties are readily expressible in CTL*, these methods are not powerful enough to admit any amelioration of state explosion, when fairness assumptions are involved. We show that it is nonetheless possible to handle fairness efficiently by trading some group theory for automata theory. Our automata-theoretic approach depends on detecting fair paths subtly encoded in a quotient structure whose arcs are annotated with permutations, by using a threaded structure that reflects coordinate shifts caused by the permutations.

Categories and Subject Descriptors: D.2.4 [**Software Engineering**]: Program Verification; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*Mechanical Verification*; *Logics of Programs*; D.1.3 [**Programming Techniques**]: Concurrent Programming

General Terms: Verification, Model Checking, Temporal Logic, Abstraction

Additional Key Words and Phrases: Algorithm, Automata, Symmetry, State Explosion

---

Authors' addresses: E. A. Emerson, Department of Computer Sciences and Computer Engineering Research Center, The University of Texas at Austin, Austin TX 78712, USA;
A. P. Sistla, Electrical Engineering and Computer Science Department, The University of Illinois at Chicago, 851 S. Morgan Street, Chicago, IL 60607, USA.
Emerson's work was supported in part by NSF grant CCR-9415496 and Semiconductor Research Corporation Contract 96-DP-388.
Sistla's work was supported in part by NSF grants CCR-9212183, CCR-9623229 and CCR-9633536.

1. INTRODUCTION

Recently there has been much interest in using various techniques to combat the state explosion problem in the automatic verification of finite state concurrent systems [Cleaveland 1993; Dams et al. 1993]. One valuable technique (cf. [Ip and Dill 1993; Clarke et al. 1993; Emerson and Sistla 1993]) is to exploit the symmetry inherent in systems with many similar processes or subcomponents when performing temporal logic model checking [Clarke and Emerson 1981; Clarke et al. 1983]. In [Ip and Dill 1993] the focus is on reasoning about a simple but basic type of correctness, viz., safety properties expressible in the temporal logic CTL by an assertion of the form $AG\neg error$. The works of Clarke, Filkorn, & Jha [Clarke et al. 1993] and Emerson & Sistla [Emerson and Sistla 1993] show how, using some basic notions of group theory, symmetry may be exploited for the full range of correctness properties expressible in the very expressive temporal logic CTL*. Surprisingly, while fairness properties are readily expressible in CTL*, the methods of [Clarke et al. 1993] and [Emerson and Sistla 1993] are not powerful enough to admit any amelioration of state explosion, when fairness assumptions are involved.

Fairness is necessary for faithfully modeling concurrency with the usual interleaving semantics. Here, concurrent execution of multiple sequential processes is treated as the nondeterministic interleaving of the execution of sufficiently small atomic steps of the individual processes. In order to reflect the basic requirement that each process should be running at some positive but indefinite speed, fairness in its simplest form amounts to assuming that each process is scheduled infinitely often. Much work in the literature has been devoted to the topic of reasoning under fairness constraints (cf. [Francez 1986]), because of its importance in proving liveness and progress properties.

In this paper, we will explain why fairness is unexpectedly problematic. We will then present a solution that permits us to efficiently handle the full range of fairness properties, including strong fairness, weak fairness, and unconditional fairness.

In [Clarke et al. 1993; Emerson and Sistla 1993] model checking a CTL* specification f over a large structure M is reduced to model checking f over a (usually) smaller quotient structure $\overline{M}$. The structure M is potentially intractably large, of size exponential in that of the underlying program whose state graph it represents. The quotient structure $\overline{M}$ is often exponentially smaller than M.

$\overline{M}$ is derived from (the program defining) M and f by identifying "$\mathcal{G}$-symmetric" states, where $\mathcal{G}$ is a subgroup of permutations on process indices. For example, states (N_1, C_2) and (C_1, N_2) might be identified in a solution to the Critical section problem. The group $\mathcal{G}$ reflects the global symmetry of M but must also respect the internal symmetry of the specification f. The latter requirement of so respecting f is crucial but is also the source of the problem (as explained in detail in Section 2.6).

Interestingly, our solution depends on trading some group theory for automata theory. The dependence of $\overline{M}$ on the symmetries of f in the "purely group-theoretic" approaches of [Clarke et al. 1993] and [Emerson and Sistla 1993], only serves to limit the amount of compression obtained. We give a new automata-theoretic method (cf. [Vardi and Wolper 1986]) that makes it possible to exploit all the symmetry inherent in M, irrespective of f. An earlier automata-theoretic approach to exploit symmetry was described in [Emerson and Sistla 1993]. But that earlier method could *not* handle fairness efficiently. An important additional advantage of the use of automata is that it allows the use of a single quotient structure $\overline{M}$ that depends only on M and can be used with any specification f.

Let Φ be a fairness constraint. The crux of our new, automata-theoretic method is showing how to efficiently detect the existence of fair paths in the large global state graph M, i.e., testing $M, s_0 \models E\Phi$, using an *annotated* quotient structure $\overline{M}$ defined with respect to a group $\mathcal{G}$ that respects the symmetry of M but does not depend on Φ. In the annotated $\overline{M}$ arcs are labeled by permutations indicating how the meaning of coordinates shift as $\overline{M}$ is uncompressed to obtain M. These permutations are what make it possible for $\overline{M}$ to succinctly encode M and yet provide enough information to model check over $\overline{M}$ (even though $\overline{M}$ does not respect the symmetry of the specification f or Φ). But they also scramble the meaning of the propositional labeling of states in $\overline{M}$ making it difficult to check for the existence of fair paths. Nonetheless, we show how to efficiently search $\overline{M}$ for any possible "subtly fair" strongly connected subgraphs $\overline{C}$ of $\overline{M}$ that can be unwound into strongly connected subgraphs C of M that contain a path satisfying Φ. To facilitate this search process, we effectively resolve $\overline{M}$ into a *threaded* structure M^* which in essence physically reflects the coordinate shifts in $\overline{M}$ caused by the permutations on arcs. When we find the appropriate "plainly fair" strongly connected subgraphs of M^* then we can conclude that $M, s_0 \models E\Phi$. We can now check whether $M, s_0 \models E(\Phi \wedge f)$ where f is a linear time formula using the automata-theoretic approach of [Emerson and Sistla 1993]. This in turn makes it possible to efficiently model check $M, s \models g$ where g is a Fair Indexed CTL* formula.

The model checking algorithm described in this paper has been implemented, and tested on some examples giving encouraging results.

This paper is organized as follows. Section 2 presents preliminaries. The main results are described in Section 3. Section 4 briefly discusses the implementation and experimental results. Some concluding remarks are given in Section 5. Technical proofs of the main lemmas and theorems are given in the Appendix.

2. PRELIMINARIES

2.1 Systems and Structures.

Let $I = [1 : n]$ be a set of n process *indices*. We have a system $\mathcal{P} = //_{i \in I} K_i$ of n processes K_i running in parallel, and using a set V of (local and/or shared) *variables* including location counters, and other data objects. We assume also that there is a set AP of *atomic propositions*, which amount to boolean variables. Letting $V' = V \cup AP$, then a *global state* s of the system is an assignment of values (over appropriate domains) to each variable in V'. We write $s(v')$ for the value of variable v' in state s.

We assume that all the variables $u, v, \ldots$ in V are indexed, and the indices of each variable indicate the processes that share the variable. For example, $u_{i,j}$ denotes a variable shared between processes i and j, whereas v_i is a local variable of process i. In practice, the local variables can be used to denote "location counters" of processes. For example, in a solution to the mutual exclusion problem we may have a state of the form (C_1, T_2) indicating that process 1 is in its critical section while process 2 is in its trying region. if

The formal semantics of such a system is given by a *structure* $M = (S, R, L, S_0)$ where S is the set of global states, R is a (total) binary *transition relation* specifying single steps of the system, $L(s) = \{P \in AP : s(P) = true\}$ is the *labeling* of each state s with true propositions, and S_0 is the set of *initial states.*

2.2 Automorphisms.

Let $Sym\ I$ be the set of all permutations π on I. $Sym\ I$ forms a group with functional composition ($\circ$) being the group operation. Our convention is that $\pi_b \circ \pi_a$ is evaluated right-to-left: first apply π_a, then π_b. Let Id denote the identity permutation and π^{-1} the inverse of π.

For any indexed object b, such as a state, a variable, or a formula, whose definition depends on I, we can define the notion of permutation π acting on b, by simultaneously replacing each occurrence of index $i \in I$ by $\pi(i)$ in b to get the result $\pi(b)$.

For a structure M, the set of (permutations on I defining graph) *automorphisms* of M, is $Aut\ M = \{\pi \in Sym\ I : \pi(M) = M\}$. Less tersely, this means that $\pi \in Aut\ M$ precisely when (a) the mapping $\pi : S \longrightarrow S : s \mapsto \pi(s)$ is one-one and onto; (b) $s \rightarrow t \in R$ iff $\pi(s) \rightarrow \pi(t) \in R$; and (c) for each s, $s \in S_0$ iff $\pi(s) \in S_0$.

For example, in Figure 1, $Aut\ M = \{Id, Flip\}$ where $Flip$ is the transposition of 1 and 2. For state (C_1, T_2), we have $Flip((C_1, T_2)) = (T_1, C_2)$.

$Aut\ M$ forms a subgroup of $Sym\ I$. Let $\mathcal{G}$ be any subgroup of $Aut\ M$. Two states s and t in S are *$\mathcal{G}$-equivalent*, written $s \equiv_{\mathcal{G}} t$ iff there exists a $\pi \in \mathcal{G}$ such that $t = \pi(s)$. Since $\mathcal{G}$ is a group, $\equiv_{\mathcal{G}}$ is an equivalence relation.

2.3 Quotients.

Given structure $M = (S, R, L, S_0)$ and subgroup $\mathcal{G}$ of $Aut\ M$, an *annotated quotient structure* (AQS) for M modulo $\mathcal{G}$, $\overline{M} = M/\mathcal{G}$, is a structure of the form $(\overline{S}, \overline{R}, \overline{L}, \overline{S}_0)$ where

—$\overline{S}$ – is a set of designated *representative states*, exactly one representative $\overline{s}$ for each $\mathcal{G}$-equivalence class $[s]$ of states in S,

—$\overline{R}$ – is a set of triples $\overline{s} \xrightarrow{\pi} \overline{t}$ denoting edges between representative states annotated with permutations from $\mathcal{G}$. To define $\overline{R}$ formally, with each state $s \in S$, we associate a canonical permutation π_s such that $\pi_s(s) = \overline{s}$. Now, we define $\overline{R} = \{\overline{s} \xrightarrow{\pi_t^{-1}} \overline{t} : \overline{s} \rightarrow t \in R\}$. Intuitively, the edge $\overline{s} \rightarrow t$ in M is represented by the annotated edge $\overline{s} \xrightarrow{\pi_t^{-1}} \overline{t}$,

—$\overline{L}(\overline{s}) = L(\overline{s})$ for each $\overline{s} \in \overline{S}$, and

—$\overline{S}_0$ – the set of representatives of states in S_0.

The intuition is that $\overline{M}$ encodes M, often succinctly. [1] M itself maybe recovered by "unwinding" $\overline{M}$. (See Figure 2, the annotated quotient of Figure 1.) $\overline{M}$ can be

[1] $\overline{M}$ can have an exponentially reduced number of states and transitions than M, but it may also have multiple edges with different permutation annotations between the same pair of nodes. This multiplicity of edges between the same pair of nodes can be limited as follows. If the underlying program defining M is the parallel composition of n *sequential* processes, where each process exhibits bounded non-determinism of degree k, then the maximal outdegree of each node in M is

constructed incrementally from the program text in essentially the same fashion as that discussed in [Emerson and Sistla 1993; Clarke et al. 1993; Ip and Dill 1993] for the unannotated quotient. [2]

2.4 Temporal Logics.

PLTL is the standard propositional linear temporal logic built up from atomic propositions, boolean connectives, and the usual linear time operators G (always), F (sometime), X (next time), and U (until). CTL* extends PLTL by also allowing the use of the path quantifiers A (for all infinite paths) and E (for some infinite path). The *basic modalities* of CTL* are formulae of the form Ef where f is a pure PLTL formula. All CTL* formulae can be obtained by taking boolean combinations and nestings of the basic modalities (e.g., $AGEFP$ has basic modalities of the form AG and EF). CTL is a restricted version of CTL* with basic modalities of the form: A or E, followed by a single F,G, X, or U.

Indexed CTL* is built up from basic modalities of the form $\vee_i Ef_i$ and $\wedge_i Ef_i$ where f_i is a PLTL formula that uses only singly indexed *local* atomic propositions such as P_i and/or *global* propositional formulas q.[3] The latter are unindexed (global) atomic propositions such as Q, or propositional formulae of the form $\vee_i p_i$, $\wedge_i p_i$, $\vee_{i\neq j} p_{i,j}$, $\wedge_{i\neq j} p_{i,j}$. Here, $\vee_i$, $\wedge_i$ act as existential and universal process quantifiers ranging over all individual process indices, $\vee_{i\neq j}$, $\wedge_{i\neq j}$ range over all distinct pairs of process indices, p_i is a propositional formula involving only global atomic propositions and local propositions of index i, and $p_{i,j}$ is a propositional formula involving only global atomic propositions and local propositions of index i and/or j. Formulas of Indexed CTL* are inductively built up from the basic modalities using boolean connectives and nesting (an Indexed CTL* formula may be substituted for a global proposition in another Indexed CTL* formula).[4] The logic can specify that each (or some) of the individual processes satisfies a local property such as absence of individual starvation, and can also specify many important global properties, such as mutual exclusion, by using global propositional formulas appropriately (see [Clarke et al. 1993]).

The following are some examples of ICTL* formulas. For a mutual exclusion problem, the ICTL* formula $\wedge_i\ (T_i \ \Rightarrow\ AF(C_i))$ asserts that, for every $i \in I$, if process i is in trying region (indicated by the atomic proposition T_i) then on all paths starting from that state process i will eventually reach the critical region (indicated by C_i). Another example of an ICTL* formula is $AG\ ((\vee_i T_i) \Rightarrow AF\ (\vee_i C_i))$. This formula asserts that in any reachable global state, if at least one

---

kn. As each annotated edge $\overline{s} \xrightarrow{\pi_t^{-1}} \overline{t} \in \overline{M}$ is induced by some edge $\overline{s} \rightarrow t \in M$, there can be at most kn distinct edges in $\overline{M}$ from $\overline{s}$.

[2]The incremental algorithm needs to check if the next generated state is equivalent to a previously generated state under the group of permutations $\mathcal{G}$. The complexity of performing this check depends on the group $\mathcal{G}$. It is, in general, as hard as graph isomorphism [Clarke et al. 1993], a problem for which there is no known polynomial time algorithm, but which is not typically viewed as intractable in practice. Moreover, for many commonly occurring $\mathcal{G}$ associated with real systems such as full symmetry or rotational symmetry, it can be done very simply and efficiently [Emerson and Sistla 1993; Clarke et al. 1993; Ip and Dill 1993].

[3]We assume f_i, f_j, etc. are identical up to reindexing.

[4]This formulation of Indexed CTL* is slightly different from ICTL* in [Clarke et al. 1989].

of the processes is in trying region then on all paths starting from that state there will be a future state in which at least one of the processes is in critical region. Note that, we treat the propositional formulas $(\vee_i T_i)$ and $(\vee_i C_i)$ as global propositions. This is because their truth value is same in all equivalent global states.

Fair Indexed CTL* is just like Indexed CTL* but uses the path quantifiers E_Φ and A_Φ where path quantification ranges only over fair paths [Emerson and Lei 1987]. Thus $\wedge_i E_\Phi f_i$ means that, for each $i \in I$, the CTL* formula $E(\Phi \wedge f_i)$ holds, where Φ is (expanded to be) a fairness constraint as (specified in PLTL) below.

The formal semantics of these logics is defined in the usual way [Emerson 1991] and we write, e.g., $M, s \models g$ to mean that in structure M and at state s, formula g holds true. Observe that for any $\pi \in$ *Aut* M and any $s \in S$, $M, s \models q$ iff $M, \pi(s) \models q$ for global q, while $M, s \models P_i$ iff $M, \pi(s) \models P_{\pi(i)}$.

2.5 Fairness.

We use special local propositions, en_i and ex_i, for each $i \in I$. Proposition en_i holds in state s iff process i is enabled in s; proposition ex_i holds in state s iff all transitions leading to s are due to the execution of a single step of process i. Plainly, for any $\pi \in$ *Aut* M, $M, s \models ex_i$ iff $M, \pi(s) \models ex_{\pi(i)}$, and similarly for en_i.

An infinite path x of M is *strongly fair* if it satisfies $\Phi = \wedge_{i=1\dots n}(GFen_i \Rightarrow GFex_i)$, meaning that each process that is enabled infinitely often is executed infinitely often. Similarly a path x is *weakly fair* if it satisfies $\Phi = \wedge_{i=1\dots n}(FGen_i \Rightarrow GFex_i)$ meaning that any process that is continuously enabled is executed infinitely often. A path is *unconditionally fair* if it satisfies $\Phi = \wedge_{i=1\dots n} GFex_i$, meaning simply that each process is executed infinitely often.

For example, Figure 1 is the global state graph of a two process solution to the mutual exclusion problem. Each process cycles from its Noncritical to its Trying to its Critical region and back. It is free from individual starvation under strong fairness, satisfying, e.g., $\wedge_i A_\Phi(G(T_i \Rightarrow FC_i))$; but not under pure nondeterministic scheduling, i.e., violating $\wedge_i A(G(T_i \Rightarrow FC_i))$. Mutual exclusion is maintained, so it also satisfies $\wedge_i A_\Phi G(\wedge_{i \neq j} \neg(C_i \wedge C_j))$.

2.6 Background.

In [Emerson and Sistla 1993] and [Clarke et al. 1993] it is shown that

(*) $\qquad M, s_0 \models f$ iff $\overline{M}, \overline{s}_0 \models f$

where f is any CTL* formula and $\overline{M}$ is the (unannotated) quotient $M/\mathcal{G}$ – provided $\mathcal{G}$ is a subgroup of *Aut* $M \cap$ *Auto* f, where *Auto* f is the group of permutations[5] that leave semantically invariant all maximal propositional subformulas of f. For

[5]For the interested reader, we explain in detail the crucial role played by *Auto* f in the proof of (*). (We also emphasize that *Auto* f is required, rather than *Aut* f, despite 3 typo's near the beginning of [Emerson and Sistla 1993].) Assume that $M, s_0 \models f$, where $f = E(GFex_1 \wedge GFex_2)$. We wish to show that $\overline{M}, \overline{s}_0 \models f$. Then there is a path $x = s_0, s_1, s_2, \dots$ in M such that $M, x \models GFex_1 \wedge GFex_2$. Moreover, there is (it turns out) a corresponding path $\overline{x} = \overline{s}_0, \overline{s}_1, \overline{s}_2, \dots$ in $\overline{M}$ such that, for each i, $s_i \equiv_{\mathcal{G}} \overline{s}_i$. Thus, s_i and $\overline{s}_i$ are not in general identical, but they do match up to a permutation in $\mathcal{G}$. Since $s_i \equiv_{\mathcal{G}} \overline{s}_i$ and in particular since $\mathcal{G}$ is a subgroup of *Auto* f, it follows from the definitions that $M, s_i \models p$ iff $\overline{M}, \overline{s}_i \models p$ for the maximal propositional subformulas ex_1 and ex_2. Intuitively, this means that the indexing of the representatives in $\overline{M}$ preserves the truth of ex_1 and ex_2. Hence, $\overline{x}$ will be "decorated" with ex_1 and ex_2 exactly as is x. We conclude $\overline{M}, \overline{x} \models GFex_1 \wedge GFex_2$ and $\overline{M}, \overline{s}_0 \models f$.

example, if $I = [1:4]$ and $f = E(GFex_1 \wedge GFex_2)$, then *Auto* f consists of those permutations that leave fixed both 1 and 2.

If we consider $f = E\Phi = E(GFex_1 \wedge \ldots \wedge GFex_n)$ where $I = [1:n]$, then *Auto* f is the set of permutations leaving invariant each of $ex_1, \ldots, ex_n$. The only such permutation is the identity Id. Hence, *Auto* $f = \mathcal{G} = \{Id\}$, and $\overline{M} = M$. *Thus, no compression is possible under fairness. Since the uncompressed* $\overline{M} = M$ *is in general of size exponential in* n*, the "purely group-theoretic" methods of [Clarke et al. 1993] and [Emerson and Sistla 1993] are not adequate for dealing with fairness.*

3. CHECKING CORRECTNESS UNDER FAIRNESS ASSUMPTIONS

In this section we give an efficient algorithm for model checking formulas of Fair Indexed CTL* under strong fairness assumption $\Phi = \wedge_{i=1\ldots n}(GFen_i \Rightarrow GFex_i)$. This also permits us to handle unconditional and weak fairness since they are special cases of strong fairness [Emerson and Lei 1987]. Through out the remainder of this section, we let M denote a structure and $\mathcal{G}$ denote any subgroup of *Aut* M.

3.1 Automata-theoretic Approach

The crucial step in the model checking problem is to handle the basic modalities [Emerson and Lei 1987]. We will describe in detail how to handle $\vee_i E_\Phi f_i$ and $\wedge_i E_\Phi f_i$. By the following lemma, it is therefore enough to check satisfaction of such basic modalities at representative states. The proof of the lemma depends on the "top-level" symmetry inherent in Fair Indexed CTL* formulas due to the process quantifiers.

Lemma 3.1. Two ($\mathcal{G}$-)equivalent states in M satisfy the same set of Fair Indexed CTL* formulas.

We initially consider the formula $E_\Phi f_i$, abbreviating the CTL* formula $E(\Phi \wedge f_i)$. The classical automata-theoretic approach [Vardi and Wolper 1986] to model checking can be used. Apply the usual tableau construction to form the (Buchi) ω-string automaton $\mathcal{A}$ equivalent to f_i which accepts an infinite sequence of (labels of) states of M by guessing a run which goes through GREEN (accepting) automaton states infinitely often. Next construct essentially the product graph (automaton) B of the structure M with automaton $\mathcal{A}$, which, intuitively, guesses a path through M that is also accepted by $\mathcal{A}$. Then test B for nonemptiness by finding reachable and accepting strongly connected components in the transition graph of B. Using complemented pairs acceptance for B [Streett 1981], we get the standard result (Theorem 3.2) below, which effectively solves our problem in time $O(|M| \cdot |\mathcal{A}| \cdot n^2)$ (cf. [Litchtenstein and Pnueli 1985], [Emerson and Lei 1987]). But it requires construction of the global state graph M, which may very well be prohibitively large, and is often of size exponential in the size of the original program text.

Now, we present a detailed and formal definition of the product graph B. For each $i \in I$, corresponding to the formula f_i, there is a Buchi automaton $\mathcal{A}_i$, which

If it happened that $\mathcal{G}$ did not leave invariant some such propositional p, the proof would break down. For example, if representatives had been chosen so that in $\overline{M}$ index 1 were uniformly replaced by 3, say, then $\overline{M}, \overline{s}_0 \not\models E(GFex_1 \wedge GFex_2)$ since 1 does not even appear in the quotient. This problem is, in general, unavoidable.

has input symbols that are sets of global propositions Q and local propositions of the form P_i. All of these various $\mathcal{A}_i$ are isomorphic up to reindexing as are the formulas f_i. Hence, it is convenient to define a single, *generic automaton* $\mathcal{A}$ which is, intuitively, the same as any $\mathcal{A}_i$ but the indices have been stripped off the local propositions P_i. Technically, there is a transition from state q to state r in $\mathcal{A}$ on input $\{Q, \ldots, Q', P, \ldots, P'\}$ iff there is a transition from q to r in $\mathcal{A}_i$ on input $\{Q, \ldots, Q', P_i, \ldots, P_i'\}$ (any i will do). The generic automaton $\mathcal{A}$ can used for any coordinate j, in lieu of $\mathcal{A}_j$, using the operator $s|j$ which, in effect, projects the label of state s onto coordinate j and then strips off the subscripts.

We will now define the product graph B to be essentially $M \times \mathcal{A} \times I$, which will be used in the proof but is not actually constructed by the algorithm. Intuitively, B denotes the simulation of the automaton $\mathcal{A}$ on the execution of different processes indicated by the process index j in each node. It is to be noted that all edges in B are between nodes with the *same* process index.

Definition. Formally, $B = (V_B, E_B)$ where $V_B = S \times Q \times I$ is the set of nodes (s, q, j) and E_B is the set of edges of B; edge $(s, q, j) \longrightarrow (s', q', j') \in E_B$ iff $j = j'$, $(s, s') \in R$, and $q' \in \delta_{\mathcal{A}}(q, s|j)$ (i. e., there is a transition of the automaton $\mathcal{A}_j$ from state q to the state q' on the input symbol comprised of the local propositions of index j as well as the global propositions that label state $\overline{s}$.)

For any permutation π on I and any node (s, q, j) in V_B, we let $\pi((s, q, j))$ denote the node $(\pi(s), q, \pi(j))$. For every $\pi \in$ *Aut* M, we have $(s, q, j) \to (t, r, j)$ is an edge in E_B iff $\pi((s, q, j)) \to \pi((t, r, j))$ is an edge in E_B. Hence every $\pi \in$ *Aut* M is also an automorphism of B.

Definition. Given a graph B', an *scsg* C' of B' is a nonempty set of nodes of B' such that the subgraph of B' induced by C' is strongly connected and total (i.e. has at least one edge). An *mscsg* C' of graph B' is a maximal scsg, i.e., an scsg such that no proper superset is an scsg of B'. [6]

Definition. An scsg C of B is *obviously fair* provided that, for each $j \in [1 : n]$ and each node s of C, if s is labeled with en_j then there is some node t in C labeled with ex_j. An scsg C is *green* iff it contains a node with automaton component in the GREEN accepting set for the Buchi automaton $\mathcal{A}$.

THEOREM 3.2. *$M, \overline{s}_0 \models E(\Phi \wedge f_i)$ iff B contains an obviously fair, green scsg C reachable starting from node $(\overline{s}_0, q_0, i)$ where q_0 is the start state of the automaton $\mathcal{A}$.*

We now develop a method to work with the annotated quotient structure $\overline{M}$, which is typically much smaller than M. An automata-theoretic approach based on annotated quotients – that cannot handle fairness efficiently – was described in [Emerson and Sistla 1993].[7] The idea is that, to check (the "unfair") $M, s_0 \models Ef_i$, we form a product graph (automaton) $\overline{B}$ of the form $\overline{M} \times \mathcal{A} \times I$ that succinctly encodes B, just as $\overline{M}$ encodes M. (In particular, each node u of B has a representative node $\overline{u}$ in $\overline{B}$; the technical definition of $\overline{B}$ is given in the next subsection.)

[6] Such mscsgs C' are total, maximal, strongly connected components of B', which will facilitate their calculation.

[7] This is in addition to the "purely" group-theoretic approach shown not to work in Section 2.

Intuitively, $\overline{B}$ guesses a path $\overline{x}$ through $\overline{M}$ and simulates the automaton $\mathcal{A}$ for formula f_i along the path x in M obtained by "unwinding" $\overline{x}$, using the permutations along $\overline{x}$ to keep track of the shifting position of index i. $M, \overline{s}_0 \models Ef_i$ iff the language accepted by $\overline{B}$ is nonempty; we can test nonemptiness of $\overline{B}$ in time $O(|\overline{B}|) = O(|\overline{M}| \cdot |\mathcal{A}| \cdot n)$. We thus have an efficient model checking algorithm that avoids construction of the global state graph M.

This approach can be easily extended to check $M, \overline{s}_0 \models Eg$, where g is any linear time formula in which k process indices $h_1, \ldots, h_k$ are used. Construct the automaton $\hat{B}$ of the form $\overline{M} \times \mathcal{A} \times I^k$, where $\mathcal{A}$ is the ω-automaton for g. Each state of $\hat{B}$ is of the form $(\overline{s}, q, j_1, \ldots, j_k)$ where $\overline{s} \in \overline{M}$, $q \in \mathcal{A}$, and $j_1, \ldots, j_k \in I$ store a list of "current values" for indices $h_1, \ldots, h_k$, respectively. $\hat{B}$ is nonempty iff $M, s_0 \models Eg$. We can test nonemptiness of $\hat{B}$ in time $O(|\hat{B}|) = O(|\overline{M}| \cdot |\mathcal{A}| \cdot n^k)$. However, when $g = \Phi \wedge f_i$, since the fairness constraint Φ involves n coordinates, we have that $k = n$ and the state space of $|\hat{B}|$ is of size exponential in n. As M itself is typically of size exponential in n, this approach defeats our purpose. *It is, thus, not possible to efficiently handle fairness by applying the automata-theoretic approach of [Emerson and Sistla 1993] directly.*

3.2 Handling Fairness

We now present a nontrivial generalization of the automata-theoretic approach that does efficiently cater for fairness. First, we formally define the annotated product graph $\overline{B}$ which is of the form $\overline{M} \times \mathcal{A} \times I$; however, it is the product of the annotated quotient structure $\overline{M}$ with the automaton $\mathcal{A}$ for *permuted* process indices. Thus, in general, the edges in $\overline{B}$ can be between nodes with *different* process indices, while this is not the case in B.[8]

Definition. Formally, $\overline{B} = (\overline{V}, \overline{E})$, where $\overline{V} = \overline{S} \times Q \times I$ is the set of nodes $(\overline{s}, q, j)$, and $\overline{E}$ is the set of edges. Each edge is of the form (x, π, y), also denoted $x \xrightarrow{\pi} y$, where $x, y \in \overline{V}$, and $\pi \in \mathcal{G}$; edge $(\overline{s}, q, j) \xrightarrow{\pi} (\overline{t}, r, k) \in \overline{E}$ iff $\overline{s} \xrightarrow{\pi} \overline{t}$ is an edge in $\overline{M}$, $\pi^{-1}(j) = k$, and $r \in \delta_{\mathcal{A}}(q, \overline{s}|j)$ (i.e., there is a transition of the automaton $\mathcal{A}_j$ from state q to state r on the input symbol comprised of the local propositions of index j and the global propositions that label state s.)

Recall that for any state $s \in S$, $\overline{s}$ denotes the unique representative of the $\equiv_{\mathcal{G}}$-equivalence class containing s. We extend the notion of representative to nodes of B.

Definition. For any node $u = (s, q, j) \in V_B$, let $\overline{u}$ denote the *representative* node $(\overline{s}, q, \pi_s(j))$ where π_s is the designated permutation, associated with $\overline{M}$, that maps s to $\overline{s}$, i.e. $\pi_s(s) = \overline{s}$. For any set of nodes $C \subseteq V_B$, let $\overline{C} = \{\overline{u} : u \in C\}$. The subgraph of $\overline{B}$ induced by C has $\overline{C}$ as its set of nodes and $\{\overline{c} \xrightarrow{\pi} \overline{d} : c, d \in C$ and $\overline{c} \rightarrow \pi(\overline{d}) \in E_B\}$ as its set of edges.

[8] $\overline{B}$ would be built by a straightforward implementation of the algorithm. The space requirements may be reduced by constructing the threaded graph B^* directly, rather than $\overline{B}$. The idea is that (the salient information from the) collection of annotated edges $\{\overline{u} \xrightarrow{\pi_a} \overline{v}, \ldots, \overline{u} \xrightarrow{\pi_c} \overline{v}\}$ from $\overline{u}$ to $\overline{v}$ in $\overline{B}$ is represented by a collection of at most n^2 edges in B^* of the form $\{(\overline{u}, i) \rightarrow (\overline{v}, \pi^{-1}(i)) : i \in I$ and π ranges over $\pi_a, \ldots, \pi_c\}$.

Definition. For any finite path $\overline{x}$ in $\overline{B}$, we define $\pi_{\overline{x}}$ to be the product of permutations appearing on the path from left to right. Let $\overline{C}$ be a scsg in $\overline{B}$ and let $\overline{c}$ be a node in $\overline{C}$. We define the subgraph of B, denoted by $unwind(\overline{C},\overline{c})$, obtained by unwinding $\overline{C}$ starting from $\overline{c}$ as follows. The set of nodes in $unwind(\overline{C},\overline{c})$ is exactly the set of nodes $\pi_{\overline{x}}(\overline{u})$ where $\overline{x}$ is a path (entirely) in (the subgraph induced by) $\overline{C}$ starting from $\overline{c}$ to a node $\overline{u}$. The set of edges are of the form $(\pi_{\overline{x}}(\overline{u}), \pi_{\overline{y}}(\overline{v}))$ where $\overline{x}$ and $\overline{y}$ are paths starting in $\overline{C}$ starting from $\overline{c}$ and ending with $\overline{u}$ and $\overline{v}$ respectively, and such that $\overline{y}$ is an extension of $\overline{x}$ by a single edge from $\overline{u}$ to $\overline{v}$ in $\overline{C}$.

Figure 3 shows a scsg $\overline{C}$ in $\overline{B}$ for a system with three processes. The edge from $\overline{s}$ to $\overline{t}$ is labeled with the permutation $flip_{12}$ that interchanges processes 1 and 2, while the edge from $\overline{t}$ to $\overline{s}$ is labeled with the identity permutation. The subgraph $unwind(\overline{C},\overline{s})$ consists of the four node cycle, as shown in figure 5, consisting of the nodes $\overline{s}$, t, s, $\overline{t}$. Here $t = flip_{12}(\overline{t})$ and $s = flip_{12}(\overline{s})$.

We use the key result (Theorem 3.3) below which will ultimately lead to an efficient algorithm for $E(\Phi \wedge f_i)$, based on the notion of a "subtly" fair scsg $\overline{C}$ of $\overline{B}$ that encodes an obviously fair scsg C of B.

Definition. An scsg $\overline{C}$ of $\overline{B}$ is *subtly fair* provided that, for each $j \in [1:n]$ and $\overline{u} \in \overline{C}$, if $\overline{u}$ is labeled with en_j, then there is a path in $\overline{C}$ starting at $\overline{u}_0 = \overline{u}$, $\overline{x} = \overline{u}_0 \overset{\pi_1}{\to} \overline{u}_1 \ldots \overset{\pi_k}{\to} \overline{u}_k$, such that $\overline{u}_k$ is labeled with $ex_{\pi_{\overline{x}}^{-1}(j)}$, where $\pi_{\overline{x}} = \pi_1 \circ \ldots \circ \pi_k$ (and $\overline{u}_k$ there by represents a reachable state labeled with ex_j). An scsg $\overline{C}$ is *green* iff its automaton component is in the set GREEN.

Consider the scsg $\overline{C}$ shown in figure 3. The node $\overline{s}$ is labeled with en_1. Now consider the cycle $\overline{x}$ from $\overline{s}$ back to $\overline{s}$. Now $\pi_{\overline{x}}$ is the permutation $flip_{12}$ and $\pi_{\overline{x}}^{-1}(1) = 2$. Further more, $\overline{s}$ is labeled with ex_2. Thus, according to the definition, $\overline{C}$ is subtly fair.

THEOREM 3.3. $M, \overline{s}_0 \models E(\Phi \wedge f_i)$ *iff*
$\overline{B}$ contains a subtly fair, green scsg $\overline{C}$ reachable from start state $(\overline{s}_0, q_0, i)$.

The proof of Theorem 3.3 depends on establishing precise correspondences between B and $\overline{B}$, and between their respective scsgs.

Lemma 3.4. (Correspondence of paths in B and $\overline{B}$)
Let N be a nonempty subgraph of B, and $\overline{N}$ the subgraph of $\overline{B}$ induced by the representatives of N.
(a) If $\overline{x} = \overline{u}_0 \overset{\pi_1}{\to} \overline{u}_1 \overset{\pi_2}{\to} \overline{u}_2 \ldots$ is a path in $\overline{N}$, then the path obtained by unwinding $\overline{x}$,

$unwind(\overline{x}) \overset{\text{def}}{=} \overline{u}_0 \to \pi_1(\overline{u}_1) \to \pi_1 \circ \pi_2(\overline{u}_2) \ldots$

is a path in B.
(b) If $x = \overline{u}_0 \to u_1 \to u_2 \ldots$ is a path in N starting at representative node $\overline{u}_0$, then there exists an path in $\overline{N}$

$\overline{x} \overset{\text{def}}{=} \overline{u}_0 \overset{\pi_1}{\to} \overline{u}_1 \overset{\pi_2}{\to} \overline{u}_2 \ldots$

such that $x = unwind(\overline{x})$.

Lemma 3.5. If C is an obviously fair, green scsg of B, then the subgraph of $\overline{B}$ induced by the representatives of nodes in C, $\overline{C}$, is a subtly fair, green scsg of $\overline{B}$.

Lemma 3.6. If $\overline{C}$ is a subtly fair, green scsg of $\overline{B}$ and $\overline{c}$ is *any* node of $\overline{C}$, then $unwind(\overline{C},\overline{c})$ is an obviously fair, green scsg of $\overline{B}$.

3.3 Threaded Graphs

In the previous section, we have, by Theorem 3.3, reduced the problem of checking the satisfaction of $E(\Phi \wedge f_i)$ at state $\overline{s}_0$ in M to that of detecting subtly fair scsgs $\overline{C}$ of $\overline{B}$, which may not be straightforward due to their scrambled nature. However, by resolving $\overline{C}$ into a "threaded" subgraph C^* which physically realizes the permutation annotations, the difficulty is overcome easily.

Definition. The *threaded graph* associated with (the subgraph of $\overline{B}$ induced by) scsg $\overline{C}$ is the graph $C^* = (V_C^*, R_C^*, L_C^*)$ where
$V_C^* = \{(\overline{u}, j) : \overline{u} \in \overline{C}$ and $j \in [1:n]\}$,
$R_C^* = \{(\overline{u}, j) \rightarrow (\overline{v}, k) : \overline{u} \xrightarrow{\pi} \overline{v} \in \overline{C}$ and $k = \pi^{-1}(j)\}$, and
$L_C^*((\overline{u}, j))$ contains en (respectively, ex) provided en_j (respectively, ex_j) is in the label of $\overline{u}$.

Note. π is the "decoding" function, while π^{-1} is the "encoding" function. In unwinding or decoding $\overline{M}$ to get M we use π, while encoding $\overline{C}$ into C^* requires π^{-1}.

Definition. Threaded graph C^* is *plainly fair* provided that, for each node $(\overline{s}, j)$ in C^*, if $(\overline{s}, j)$ is labeled with en, then there is in C^* a path to some node $(\overline{t}, k)$ labeled with ex. Threaded graph C^* is *green* iff it contains a node whose automaton component is in the GREEN set of automaton states.

Figure 4 shows the threaded subgraph C^* corresponding to the scsg $\overline{C}$ shown in figure 3. Note that the nodes $(\overline{s}, 1)$ and $(\overline{s}, 2)$ are labeled with en and ex respectively. From the definition, it is easy to see that C^* is plainly fair.

Lemma 3.7. (Correspondence of paths in $\overline{C}$ and C^*)
Let $\overline{C}$ be a scsg of $\overline{B}$ and C^* the threaded graph associated with $\overline{C}$.
(a) If $\overline{x} = \overline{u}_0 \xrightarrow{\pi_1} \overline{u}_1 \xrightarrow{\pi_2} \overline{u}_2 \ldots$ is a path in $\overline{C}$, then for each $j \in [1:n]$, the path
$(\overline{u}_0, j) \rightarrow (\overline{u}_1, \pi_1^{-1}(j)) \rightarrow (\overline{u}_2, \pi_2^{-1} \circ \pi_1^{-1}(j)) \ldots$
is a corresponding *thread* in C^*.
(b) If $x^* = (\overline{u}_0, j_0) \rightarrow (\overline{u}_1, j_1) \rightarrow (\overline{u}_2, j_2) \ldots$ is a thread in C^*, then there is a corresponding path in $\overline{C}$
$\overline{x} = \overline{u}_0 \xrightarrow{\pi_1} \overline{u}_1 \xrightarrow{\pi_2} \overline{u}_2 \ldots$ one of whose threads is x^*.

Using this correspondence we can now prove the following crucial lemma relating subtle fairness of $\overline{C}$ to plain fairness of C^*.

Lemma 3.8. Let $\overline{C}$ be a scsg of $\overline{B}$ and C^* the threaded graph associated with $\overline{C}$. Then,
$\overline{C}$ is subtly fair and green iff C^* is plainly fair and green.

3.4 Main Algorithm

Putting all the results of the previous subsections together, we can get an efficient algorithm that determines ("marks") all representative states $\overline{s}$ such that $M, \overline{s} \models E(\Phi \wedge f_i)$. We assume that an earlier algorithm has constructed the annotated quotient structure $\overline{M}$.

Algorithm.

(1) Construct the automaton $\mathcal{A}$ corresponding to the formula f_i. Construct $\overline{B}$ from $\overline{M}$ and $\mathcal{A}$. Save a copy of $\overline{B}$ as the graph $\overline{H}$.

(2) Repeat the following procedure $n+1$ times:
    Compute the mscsgs of $\overline{B}$
    For each mscsg $\overline{C}$ of $\overline{B}$
        /*Check if $\overline{C}$ is subtly fair: */
        Resolve $\overline{C}$ into C^*
        Check if C^* is plainly fair
            if so mark all nodes of $\overline{C}$
            if not, delete from $\overline{B}$ each node $\overline{s}$ of $\overline{C}$ for which
                some node $(\overline{s}, j)$ in C^* is "bad",
                meaning that it is labelled with en,
                but it cannot reach in C^* a node labeled ex

(3) Using $\overline{H}$, propagate the marking to all nodes that can reach an already marked node.

THEOREM 3.9. *The above algorithm correctly marks exactly those representative states $\overline{s}$ such that $M, \overline{s} \models E(\Phi \wedge f_i)$. It runs in time $O(|\overline{B}| \cdot n^2) = O(|\overline{M}| \cdot |\mathcal{A}| \cdot n^3)$.*

Note that the above theorem actually applies to any index $i \in [1:n]$ and that the states of $\overline{B}$ include $(\overline{s}, q_0, i)$ for *all* $i \in [1:n]$. Thus, the above algorithm is, in effect, checking *simultaneously* correctness of all n formulas $E(\Phi \wedge f_1), \ldots, E(\Phi \wedge f_n)$, yielding trivially an algorithm for the basic modalities:

Corollary 3.10. (Model checking for basic modalities $\wedge_i E_\Phi f_i$ and $\vee_i E_\Phi f_i$)
(a) $M, \overline{s} \models \wedge_i E_\Phi f_i$ iff all n states $(\overline{s}, q_0, 1), \ldots, (\overline{s}, q_0, n)$ are marked.
(b) $M, \overline{s} \models \vee_i E_\Phi f_i$ iff one or more of the n states $(\overline{s}, q_0, 1), \ldots, (\overline{s}, q_0, n)$ are marked.

THEOREM 3.11. *(Model Checking for Fair Indexed CTL*) The above procedure provides an algorithm which checks $M, \overline{s}_0 \models g$ for any Fair Indexed CTL* formula g under strong fairness, in time $O(|\overline{M}| \cdot n^3 \cdot |g| \cdot a)$ where $|\overline{M}|$ is the size of $\overline{M}$, $n = |I|$ is the number of processes, $|g|$ is the length of g, and a is the maximum size of the automaton $\mathcal{A}$ for any basic modality of g.*

It is interesting to compare the complexity bound of $O(|\overline{M}| \cdot n^3 \cdot |g| \cdot a)$ for our algorithm with the complexity bound of the traditional algorithm that works directly on the original structure M, which can be shown to be $O(|M| \cdot n^3 \cdot |g| \cdot a)$ in the terminology of theorem 3.11.[9] Thus, the complexity bounds for our algorithm and the traditional algorithm have the same form except that our bound is over the likely succinct annotated quotient $\overline{M}$, while that of the traditional algorithm is over the entire structure M which may be intractably large. The "Litchtenstein-Pnueli Thesis" [Litchtenstein and Pnueli 1985] suggests that it is the linear complexity in

[9]This is seen by noting that for any fixed i, the complexity of the traditional algorithm for model checking the formula $E_\Phi f_i$ on M is $O(|M||\mathcal{A}|n^2)$ where $\mathcal{A}$ is the automaton referred to in theorem 3.9. The complexity when we use this algorithm for model checking a basic modality $\wedge_i E_\Phi f_i$ is $O(|M||\mathcal{A}|n^3)$, since we need to repeat the previous step for each $i = 1, ..., n$. The claimed bound follows by noting that there are at most $|g|$ basic modalities in g.

the structure size that is most important for applications, since structures tend to be extremely large while correctness properties tend to be short.

For Fair Indexed CTL the factor a is a small constant and the bound simplifies to $O(|\overline{M}| \cdot n^3 \cdot |g|)$. Moreover, under weak and unconditional fairness, it turns out that there is no need to perform the iteration in step 2 of the core algorithm. Hence, the bound simplifies to $O(|\overline{M}| \cdot n^2 \cdot |g| \cdot a)$ for Fair Indexed CTL* and $O(|\overline{M}| \cdot n^2 \cdot |g|)$ for Fair Indexed CTL.

4. IMPLEMENTATION

We have developed a prototype of the model checker implementing the above presented algorithm for weak fairness and strong fairness. Our implementation constructs both $\overline{M}$ and $\overline{B}$ explicitly. Permutations are represented as arrays. The edges in $\overline{M}$ and $\overline{B}$ contain pointers to permutations that label the edges. All such permutations are stored in a separate data structure. If multiple edges are labeled with the same permutation then all of them contain pointers to the same permutation array. The exploration of the threaded graph C^*, corresponding to each strongly connected component $\overline{C}$ of $\overline{B}$ is carried out using an implicit representation as follows. With each state $\overline{u}$ of $\overline{B}$, we maintain an array of n bits corresponding to the n processes. The i^{th} bit is used to denote if the C^*-node $(\overline{u}, i)$ is visited during the search operation. The actual exploration of C^* is carried out using breadth first search, and by inferring the edges of C^* from the edges of $\overline{B}$ as and when needed, i.e. on the fly. Thus, our implementation does not use an explicit representation of the threaded graphs.

We have used the implemented system to check for the correctness of a resource controller example. In this example, we have a resource controller C and many user processes U. Each user process communicates with the controller C through shared variables. Typically, a user process requests for a resource by setting a shared variable to 1 and waits for the controller process to grant the request. If the resource is available the controller process non-deterministically grants the resource to one of the requesting users; it does this by setting another shared to value 1. When a user sees that its request is granted, it uses the resource and releases it by resetting a global $busy$ flag which was set by the controller. For this system, we were able to check many properties including the liveness property that every user process that requests a resource will eventually access the resource.

We ran our model checker on the resource controller example with six user processes on a SparcCenter 1000 machine. For this case, we ran the model checker without using symmetry and with symmetry. We found a dramatic improvement in all performance measures when we used symmetry as compared to the case when we did not use symmetry. The performance statistics we obtained are given below. Each statistic is given as a/b where a, b are the numbers for the cases when we ran the model checker without and with symmetry respectively. The numbers of states in the structure is 512/26; the number of edges in the structure is 1984/112; the amount of memory used for storing structure is 38K/2K bytes; the time for constructing the structure .733/.08 seconds; the time for constructing the product graph is .566/.07 seconds; the total time is 1.3/.15 seconds. The difference in the performances, without and with symmetry, grows exponentially with the number of processes. It is to be noted that performance measures such as the amount of memory and over all time taken are highly implementation dependent, where as measures such as number of states, transitions etc. are implementation independent.

We also ran the model checker using symmetry on the resource controller example for cases when the number of processes is 10,50,100 and 150. The performance statistics for these cases are given in table I. This table gives the number of states and transitions in the Annotated Quotient Structure (AQS), the CPU time for construction of the AQS, the number of states and transitions in $\overline{B}$, the time taken for construction of $\overline{B}$, the time taken for exploring $\overline{B}$ and detecting the existence of subtly fair strongly connected subgraphs in $\overline{B}$, and finally the over all CPU time taken. The time for exploring the threaded graphs is included as part of the time for exploring $\overline{B}$. Each column in the table corresponds to a fixed number of processes which is indicated at the top of the column.

	10	50	100	150
AQS states/ & transitions	38/ 235	198/ 6175	398/ 24850	598/ 56 K
AQS constr. time in secs	0	11	192	730
$\overline{B}$ states/ & transitions	521/3.8K	14.6 K/ 545 K	59.2 K/ 4.43 M	134 K/15 M
$\overline{B}$ constr. time in secs	0	18	467	2500
time taken for exploring $\overline{B}$	0	7	41	515
over all CPU time	0	36	700	3745

Table I. Statistics for the model checker

For the above example, it can analytically be shown that the number of states in the AQS should be about $4n-2$ and the number of transitions should be about $n(5n-3)/2$ where n is the number of processes. On the other hand, the number of states in the original structure (i.e. the reachability graph in which equivalent states are not identified) is exponential in n. In fact, the original reachability graph for a system with 150 processes has about 10^{47} reachable states, and we were able to check its liveness properties using our symmetry based model checker.

5. CONCLUSIONS

Symmetry reduction has previously been shown to be a powerful tool for reasoning about reactive systems in a number of contexts. For example, there has been a good deal of work in the Petri-net community using symmetry reduction to ameliorate state explosion when reasoning about reachability, boundedness and other Petri-net related properties (cf. [Jensen and Rozenberg 1991], [Jensen 1994]). Symmetry reduction has been successfully applied to facilitate protocol verification in [Aggarwal et al. 1983] and [Kurshan 1986], and to facilitate hardware verification in [Ip and Dill 1993]. Such previous works were not intended to cater for the full range of temporal correctness properties, but symmetry reduction techniques for arbitrary CTL* properties were developed in [Clarke et al. 1993] and [Emerson and

Sistla 1993]. Unfortunately, as explained in sections 2.6 and 3 of this paper, the techniques of [Clarke et al. 1993] and [Emerson and Sistla 1993] are, in effect, inapplicable when trying to reason under fairness assumptions. [Kurshan 1994] shows how to reason about general temporal properties under certain types of fairness-like assumptions in the framework of L-processes and L-automata; however, fairness is not handled in full generality and the approach is limited to cyclic symmetry.

To our knowledge, ours is the first approach that permits symmetry reduction for essentially arbitrary temporal properties under (general) fairness constraints. Our particular contribution here is to broaden the range of applicability of symmetry reduction techniques by showing how to efficiently handle Fair Indexed CTL* formulas. This permits efficient reasoning about many important liveness properties that depend on fairness assumptions. It is also worth noting that this could not be done using the essentially group-theoretic approaches of [Clarke et al. 1993; Emerson and Sistla 1993], but depends crucially on the automata-theoretic framework. This seems to testify to the power of automata [Vardi and Wolper 1984; 1986] (cf. [Kurshan 1994]).

Finally, it would be interesting to compare the symmetry reduction approach of collapsing M according to symmetry group $\mathcal{G}$ to get quotient $\overline{M}$ and the approach of computing $\hat{M}$, the quotient of M modulo its "coarsest bisimulation". Actually, there are several unsettled points regarding this latter notion. First, a precise definition of "the" coarsest bisimulation in this context must be given; it would seem to inherently involve itself in some group-theoretic considerations relating to symmetries, considering that, for example, states (T_1, N_2) and (N_1, T_2) should be bisimilar. Thus, such a bisimulation would itself be a type of symmetry reduction. Another, more important, problem may be computation of the coarsest bisimulation incrementally. Although there are some recent results for incremental computation of the standard coarsest bisimulation [Bouajjani et al. 1992; Lee and Yannakakis 1992], they require symbolic representation of equivalence classes and "oracles" for performing certain operations on these equivalence classes efficiently. As a consequence, they are of uncertain general applicability, and matters seem even less certain for the notion of a symmetry based bisimulation. In any event, this is a topic that might merit further attention.

APPENDIX

In this appendix, we provide proofs omitted from the main sections.

Lemma 3.1. Two ($\mathcal{G}$-)equivalent states in M satisfy the same set of Fair Indexed CTL* formulas.

PROOF. We can argue by induction on formula structure that for any formula f of Fair Indexed CTL*, and any permutation π in *Sym I*, f and $\pi(f)$ are equivalent, i.e. express the same property. Now, assume $s \equiv_{\mathcal{G}} t$ so that $t = \pi(s)$ for some $\pi \in \mathcal{G}$. We have $M, s \models f$ iff $M, \pi(s) \models \pi(f)$ (since $\pi \in \mathcal{G} \subseteq$ *Aut M*) iff $M, t \models f$ (since $t = \pi(s)$ and $\pi(f)$ is equivalent to f). □

THEOREM 3.2. *$M, \overline{s}_0 \models E(\Phi \wedge f_i)$ iff B contains an obviously fair, green scsg C reachable starting from node $(\overline{s}_0, q_0, i)$ where q_0 is the start state of the automaton $\mathcal{A}$.*

PROOF. To prove the forward direction, assume that $M, \overline{s}_0 \models E(\Phi \wedge f_i)$. Then there is an infinite path $p = \overline{s}_0, s_1, s_2, \ldots$ in M starting from the state $\overline{s}_0$, which satisfies the fairness constraint Φ and the formula f_i. Hence, there exists an accepting run $\rho = q_0, q_1, q_2, \ldots$ of the automaton $\mathcal{A}$ along the input string obtained by projecting the path p onto coordinate i. When we combine p with this run ρ, we get an infinite path $(\overline{s}_0, q_0, i), (s_1, q_1, i), (s_2, q_2, i), \ldots$ in B and the set of nodes appearing infinitely often on this path defines an obviously fair green scsg C reachable from $(\overline{s}_0, q_0, i)$.

To prove the reverse direction, assume that an obviously fair green scsg C is reachable from $(\overline{s}_0, q_0, i)$ in B. From this we can get a path in B starting from $(\overline{s}_0, q_0, i)$ and visiting each node of C infinitely often; this path satisfies the fairness constraint Φ and contains an accepting node of $\mathcal{A}$ infinitely often. This path, when projected on to the state components of M, gives us a path p in M starting from $\overline{s}_0$ which satisfies the formula f_i and the fairness constraint Φ. $\square$

THEOREM 3.3. *$M, \overline{s}_0 \models E(\Phi \wedge f_i)$ iff*
$\overline{B}$ contains a subtly fair, green scsg $\overline{C}$ reachable starting from node $(\overline{s}_0, q_0, i)$.

PROOF. To prove the forward direction, assume that $M, \overline{s}_0 \models E(\Phi \wedge f_i)$. By theorem 3.2, we see that B contains an obviously fair, green scsg C reachable from $(\overline{s}_0, q_0, i)$. Using lemma 3.5 below, we see that $\overline{C}$ is a subtly fair, green scsg of $\overline{B}$. Furthermore, using lemma 3.4 below, we see that there is a path in $\overline{B}$ from $(\overline{s}_0, q_0, i)$ to the scsg $\overline{C}$.

To prove the reverse direction, assume that $\overline{B}$ contains a subtly fair, green scsg $\overline{C}$ reachable from $(\overline{s}_0, q_0, i)$. Using lemma 3.6 below, we see that $unwind(\overline{C}, \overline{c})$ is an obviously fair scsg of B where $\overline{c}$ is any node in $\overline{C}$. ¿From lemma 3.4 we see that, if p is an path in $\overline{B}$ from the node $(\overline{s}_0, q_0, i)$ to $\overline{c}$ in $\overline{C}$ then $unwind(p)$ is a path in B from $(\overline{s}_0, q_0, i)$ to some node in $unwind(\overline{C}, \overline{c})$. Hence $unwind(\overline{C}, \overline{c})$ is a green obviously fair scsg in B which is reachable from $(\overline{s}_0, q_0, i)$. By theorem 3.2, we conclude that $M, \overline{s}_0 \models E(\Phi \wedge f_i)$. $\square$

Lemma 3.4. (Correspondence of paths in B and $\overline{B}$)
Let N be a nonempty subgraph of B, and $\overline{N}$ the subgraph of $\overline{B}$ induced by N.
(a) If $\overline{x} = \overline{u}_0 \xrightarrow{\pi_1} \overline{u}_1 \xrightarrow{\pi_2} \overline{u}_2 \ldots$ is a path in $\overline{N}$, then the path obtained by unwinding $\overline{x}$,

$x = unwind(\overline{x}) \stackrel{\text{def}}{=} \overline{u}_0 \rightarrow \pi_1(\overline{u}_1) \rightarrow \pi_1 \circ \pi_2(\overline{u}_2) \ldots$

is a path in B.
(b) If $x = \overline{u}_0 \rightarrow u_1 \rightarrow u_2 \ldots$ is a path in N starting at representative state $\overline{u}_0$, then there exists a path in $\overline{N}$

$\overline{x} \stackrel{\text{def}}{=} \overline{u}_0 \xrightarrow{\pi_1} \overline{u}_1 \xrightarrow{\pi_2} \overline{u}_2 \ldots$

such that $x = unwind(\overline{x})$.

PROOF. To prove (a), it is enough if we show that, for each $j \geq 1$, $\pi_1 \circ \pi_2 \circ \ldots \circ \pi_j(\overline{u}_j) \rightarrow \pi_1 \circ \pi_2 \circ \ldots \circ \pi_{j+1}(\overline{u}_{j+1})$ is a transition in B. Since $\mathcal{G}$ is a group, we know that $\pi_1 \circ \pi_2 \circ \ldots \circ \pi_j$ is a permutation in $\mathcal{G}$, and an automorphism of B. Furthermore, from the definition of $\overline{B}$, we know that $\overline{u}_j \rightarrow \pi_{j+1}(\overline{u}_{j+1})$ is a transition of B. The desired result follows.

To prove (b), we note that we can write any such x in the form $\overline{u}_0 \rightarrow \phi_1(\overline{u}_1) \rightarrow \phi_2(\overline{u}_2) \rightarrow \ldots$ where, for each $j \geq 1$, ϕ_i is any permutation in $\mathcal{G}$ such that $\phi_i(\overline{u}_i) = u_i$. We will argue by induction on j that we can take ϕ_j to be a permutation of the form $\pi_1 \circ \ldots \circ \pi_j$ as in the statement of the lemma. For $j = 1$, since $\overline{u}_0 \rightarrow \phi_1(\overline{u}_1) \in B$, by definition of $\overline{B}$, there is some $\pi_1 \in \mathcal{G}$ and $\overline{u}_0 \xrightarrow{\pi_1} \overline{u}_1 \in \overline{B}$ such that $\pi_1(\overline{u}_1) = \phi_1(\overline{u}_1)$. Thus, we can take ϕ_1 to be π_1. Inductively, we can take $\phi_j = \pi_1 \circ \ldots \circ \pi_j$. Because $\phi_j(\overline{u}_j) \rightarrow \phi_{j+1}(\overline{u}_{j+1}) \in B$, $\overline{u}_j \rightarrow (\pi_1 \circ \ldots \circ \pi_j)^{-1} \circ \phi_{j+1}(\overline{u}_{j+1}) \in B$, by induction hypothesis and since ϕ_j^{-1} is an automorphism of B. Hence, there is some $\pi_{j+1} \in \mathcal{G}$ and some $\overline{u}_j \xrightarrow{\pi_{j+1}} \overline{u}_{j+1} \in \overline{B}$ such that $\pi_{j+1}(\overline{u}_{j+1}) = (\pi_1 \circ \ldots \circ \pi_j)^{-1} \circ \phi_{j+1}(\overline{u}_{j+1})$. Thus, we can take $\phi_{j+1} = \pi_1 \circ \ldots \circ \pi_j \circ \pi_{j+1}$, thereby completing the induction step. Then, the path $\overline{x} = \overline{u}_0 \xrightarrow{\pi_1} \overline{u}_1 \xrightarrow{\pi_2} \overline{u}_2 \ldots$ is such that $unwind(\overline{x}) = x$. Moreover, since x is in N, then all the representatives $\overline{u}_j$ are in $\overline{N}$, and $\overline{x}$ is itself in $\overline{N}$. □

Lemma 3.5. If C is an obviously fair, green scsg of B, then the subgraph $\overline{C}$ of $\overline{B}$, induced by the representatives of nodes in C, is a subtly fair, green scsg of $\overline{B}$.

PROOF. Assume that C is an obviously fair, green scsg of B. Let u', v' be any two nodes in $\overline{C}$. There exist nodes u, v in C such that $u' = \overline{u}$ and $v' = \overline{v}$. Because C is an scsg of B, there exists a path in C from u to v. Now, since u' is the representative of u, $u' \equiv_{\mathcal{G}} u$, and there is a permutation $\pi_u \in \mathcal{G}$ such that $\pi_u(u) = u'$. As π_u is an automorphism of M and also of B, there also exists a path from $\overline{u} = u' = \pi_u(u)$ to $\pi_u(v)$ in $\pi_u(C)$, which is an isomorphic copy of C in B whose set of representatives coincides with that of C thereby inducing the same subgraph $\overline{C}$ of B as does C. Using lemma 3.4, it follows that there exists a path in $\overline{C}$ from $u' = \overline{u}$ to $v' = \overline{v}$. Hence $\overline{C}$ is strongly connected and is a scsg.

To see that $\overline{C}$ is subtly fair, consider any node u' in $\overline{C}$ and any index j such that u' satisfies en_j.[10] There exists a node u in C such that u' is a representative of u. Let $\pi \in \mathcal{G}$ be a permutation such that $\pi(u) = u'$. Since u' satisfies en_j, it is the case that u satisfies $en_{\pi^{-1}(j)}$. Since C is obviously fair, there exists another node v in C such that v satisfies $ex_{\pi^{-1}(j)}$. Let y be a path in C from u to v. Since π is an automorphism of B, the sequence $x = \pi(y)$ is also a path in B, from $u' = \pi(u)$ to $\pi(v)$. By lemma 3.4, there exists a path $\overline{x}$ in $\overline{C}$ from u' to $\overline{v}$, such that $unwind(\overline{x}) = x$. Hence $\pi(v) = \pi_{\overline{x}}(\overline{v})$ (Recall that $\pi_{\overline{x}}$ is the product of permutations appearing on the edges in the path $\overline{x}$). Since v satisfies $ex_{\pi^{-1}(j)}$, it is the case that $\pi(v)$, and hence $\pi_{\overline{x}}(\overline{v})$, satisfies ex_j.[11] Since $\pi_{\overline{x}}(\overline{v})$ satisfies ex_j, it is the case that $\overline{v}$ satisfies $ex_{\pi_{\overline{x}}^{-1}(j)}$. From this, it follows that $\overline{C}$ is subtly fair. Since $\mathcal{G}$ leaves states of $\mathcal{A}$ invariant, it should be clear that $\overline{C}$ is green. □

Lemma 3.6. If $\overline{C}$ is a subtly fair, green scsg of $\overline{B}$ and $\overline{c}$ is any node of $\overline{C}$, then $unwind(\overline{C}, \overline{c})$ is an obviously fair, green scsg of B.

PROOF. Assume that $\overline{C}$ is a subtly fair, green scsg of $\overline{B}$ and $\overline{c}$ is any node of $\overline{C}$. We first show that $unwind(\overline{C}, \overline{c})$ is a scsg in B. By definition $\overline{c}$ is in

[10] We say that node $u' = (s', q', i')$ of B *satisfies* a proposition P provided that its state component s' satisfies P, i.e., contains P in its label. Here P may be a global proposition or a local one Q_j; in the latter case there need be no relationship between j and i'.

[11] Here, we use this principle. If u satisfies P then $\pi(u)$ satisfies $\pi(P)$ for any π in $\mathcal{G}$. The justification is that we are simply reindexing both sides of "satisfies". The restriction to $\mathcal{G}$ a subgroup of $Aut\ M$ is to ensure that $\pi(u)$ is a well-defined node of B.

$unwind(\overline{C},\overline{c})$. Let u be any node in $unwind(\overline{C},\overline{c})$. We will show that there is a cycle in $unwind(\overline{C},\overline{c})$ which contains both $\overline{c}$ and u. This will ensure that all the nodes in $unwind(\overline{C},\overline{c})$ are strongly connected. ¿From the definition of $unwind(\overline{C},\overline{c})$, we see that there exists a path $\overline{x}$ from $\overline{c}$ to $\overline{u}$ in $\overline{C}$ such that $\pi_{\overline{x}}(\overline{u}) = u$. Since $\overline{C}$ is a scsg, it follows that there exists a cycle $\overline{y}$ which is an extension of $\overline{x}$ (i.e., $\overline{x}$ is a prefix of $\overline{y}$), such that $\overline{y}$ starts and ends with $\overline{c}$. The permutation $\pi_{\overline{y}}$ is in $\mathcal{G}$. Hence there exists an integer $l > 0$ such that $(\pi_{\overline{y}})^l$ is the identity permutation. Now consider the cycle $\overline{y}^l$, i.e. the cycle obtained by repeating $\overline{y}$ exactly l times. Using lemma 3.4, it is easy to see that $unwind(\overline{y}^l)$ is a cycle in $unwind(\overline{C},\overline{c})$ that contains both $\overline{c}$ and u.

Now, we prove that $unwind(\overline{C},\overline{c})$ is obviously fair. Let u be any node in $unwind(\overline{C},\overline{c})$ and j be any index such that u satisfies en_j. Then, there is a path $\overline{x}$ from $\overline{c}$ to $\overline{u}$ in $\overline{C}$ such that $unwind(\overline{x})$ is a path from $\overline{c}$ to u in C and $u = \pi_{\overline{x}}(\overline{u})$. Let $k = \pi_{\overline{x}}^{-1}(j)$. Since u satisfies en_j, it is the case that $\overline{u}$ satisfies en_k. Since $\overline{C}$ is subtly fair, there exists a path $\overline{y}$ from $\overline{u}$ to a node $\overline{v}$ in $\overline{C}$ such that $\overline{v}$ satisfies $ex_{\pi_{\overline{y}}^{-1}(k)}$. Now $\overline{x}\cdot\overline{y}$, the fusion of $\overline{x}$ with $\overline{y}$, is a path from $\overline{c}$ to $\overline{v}$ in $\overline{C}$. Hence $unwind(\overline{x}\cdot\overline{y})$ is a path from $\overline{c}$ to u to $\pi_{\overline{xy}}(\overline{v})$ in $unwind(\overline{C},\overline{c})$. Since $\overline{v}$ satisfies $ex_{\pi_{\overline{y}}^{-1}(k)}$, $k = \pi_{\overline{x}}^{-1}(j)$, and $\pi_{\overline{x}\cdot\overline{y}} = \pi_{\overline{x}}\pi_{\overline{y}}$, we calculate that $\pi_{\overline{xy}}(\overline{v})$ satisfies ex_j. This establishes that $unwind(\overline{C},\overline{c})$ is obviously fair. As $\mathcal{G}$ does not act on the states of $\mathcal{A}$, it is clear that $unwind(\overline{C},\overline{c})$ is green. □

Proof of lemma 3.7 is straightforward from the definition of C^* and lemma 3.4, and is left to the reader.

Lemma 3.8. Let $\overline{C}$ be a scsg of $\overline{B}$ and C^* the threaded graph associated with $\overline{C}$. Then,
$\overline{C}$ is subtly fair and green iff C^* is plainly fair and green.

PROOF. To prove the forward direction, assume that $\overline{C}$ is subtly fair and green. Since $\mathcal{G}$ does not act on $\mathcal{A}$, it is clear that C^* is green. We prove that C^* is plainly fair. Consider any node $(\overline{u}, j)$ in C^* which is labeled with en. This means that the state $\overline{u}$ is labeled with en_j in $\overline{C}$. Since $\overline{C}$ is subtly fair, there exists a path $\overline{x}$ from $\overline{u}$ to a node $\overline{v}$ such that $\overline{v}$ is labeled with $ex_{\pi_{\overline{x}}^{-1}(j)}$. From lemma 3.7, it is seen that there is a thread in C^* from $(\overline{u}, j)$ to $(\overline{v}, \pi_{\overline{x}}^{-1}(j))$. Clearly, the node $(\overline{v}, \pi_{\overline{x}}^{-1}(j))$ in C^* is labeled with ex. Hence C^* is plainly fair. Now we prove the lemma in the other direction. Assume that C^* is plainly fair and green. It is trivial to see that $\overline{C}$ is green. Now consider any node $\overline{u}$ which is labeled with en_j for some j. In C^* the node $(\overline{u}, j)$ is labeled with en. Since C^* is plainly fair, there exists a node $(\overline{v}, k)$, labeled with ex, which is reachable from $(\overline{u}, j)$. ¿From lemma 3.7, it should be easy to see that there exists a path $\overline{x}$ in $\overline{C}$ from $\overline{u}$ to $\overline{v}$ such that $k = \pi_{\overline{x}}^{-1}(j)$. Clearly, $\overline{v}$ is labeled with ex_k. This proves that C^* is plainly fair. □

THEOREM 3.9. *The algorithm correctly marks exactly those representative states $\overline{s}$ such that $M, \overline{s} \models E(\Phi \wedge f_i)$. It runs in time $O(|\overline{B}| \cdot n^2) = O(|\overline{M}| \cdot |\mathcal{A}| \cdot n^3)$.*

PROOF. To prove the theorem, it is enough if we show that in the second step of the algorithm, the node $\overline{u}$ is marked iff it belongs to a subtly fair green scsg in $\overline{B}$. From all our previous lemmas, it is easy to see that if a node $\overline{u}$ is marked then it must belong to a subtly fair green scsg in $\overline{B}$. To prove the other direction we need the following claim.

Claim. If $\overline{C}$ is a scsg in $\overline{B}$ then, C^* is plainly fair iff all the maximal scsgs in C^* are plainly fair (i.e. if each maximal scsg in C^* contains a node labeled with en then it also contains a node labeled with ex).

PROOF. We prove the claim by simply showing that all the maximal scsgs in C^* are disconnected, i.e. C^* does not have edges that connect nodes in different scsgs. It is enough to show that if there is an edge connecting two nodes in C^* then both the nodes belong to the same scsg. Assume that there is an edge in C^* from some node $(\overline{u}, j)$ to some other node $(\overline{v}, k)$. This means that there is an edge e in $\overline{B}$ from $\overline{u}$ to $\overline{v}$ which is labeled with a permutation π such that $\pi^{-1}(j) = k$. Since $\overline{u}, \overline{v}$ belong to the scsg $\overline{C}$, there exists a path $\overline{x}$ starting and ending with the node $\overline{u}$ and using the edge e for the first transition. Clearly, $\pi_{\overline{x}}$, which is the product of permutations along $\overline{x}$, is in $\mathcal{G}$. Since $\mathcal{G}$ is a group, it is easily seen that there exists an integer $l > 0$ such that $(\pi_{\overline{x}})^l$ is the identity permutation. Clearly, $(\pi_{\overline{x}})^l$ is the permutation associated with the path $\overline{x}^l$. Using lemma 3.8, it is easily seen that there is a thread in C^*, corresponding to the path $\overline{x}^l$, which is a cycle and which uses the edge from $(\overline{u}, j)$ to $(\overline{v}, k)$. Hence both the nodes $(\overline{u}, j)$ and $(\overline{v}, k)$ belong to the same scsg in C^*. □

Now, assume that $\overline{u}$ is a node in a subtly fair green scsg $\overline{D}$ in $\overline{B}$. It is not difficult to see that in step 2 of the algorithm none of the nodes in $\overline{D}$ are ever going to be deleted. Now, we show that with in the $n + 1$ iterations of the outermost loop in step 2, $\overline{u}$ (in fact, all the nodes in $\overline{D}$) is going to be marked. In an outermost iteration at step 2, assume that some nodes in $\overline{B}$ belonging to the mscsg containing $\overline{u}$ are deleted; assume that this deletion occurs due to the presence of bad nodes in the mscsg of the threaded graph containing $(\overline{u}, j)$ for some j (in this case, the mscsg in the threaded graph containing $(\overline{u}, j)$ does not have any nodes labeled with ex and the bad nodes are all the nodes in it that are labeled with en). It is to be noted that, in all future instances after this iteration, the mscsg in the threaded graph containing $(\overline{u}, j)$ will have no more bad nodes, and it will be plainly fair. Hence with in at most n iterations, for each $j = 1, ..., n$, all the mscsgs in the threaded graph containing $(\overline{u}, j)$ will be plainly fair; at this time, the mscsg in $\overline{B}$ containing $\overline{u}$, call it $\overline{C}$, will be subtly fair; this is because C^* will be plainly fair (due to the above claim). It should be now be clear that in the next iteration of step 2 all the nodes in $\overline{D}$ will be marked.

It should be easy to see that steps 2 dominates the time complexity. Each iteration of the outermost loop of step 2 can be implemented in time proportional to the size of the threaded version B^* of $\overline{B}$, i.e., $O(|\overline{B}| \cdot n)$ time. Hence, step 2 can be implemented in time $O(|\overline{B}| \cdot n^2) = O(|\overline{M}| \cdot |\mathcal{A}| \cdot n^3)$.

THEOREM 3.11. *(Model Checking for Fair Indexed CTL*) The above procedure provides an algorithm which checks $M, \overline{s}_0 \models g$ for any Fair Indexed CTL* formula g under strong fair semantics, in time $O(|\overline{M}| \cdot n^3 \cdot |g| \cdot a)$ where $|\overline{M}|$ is the size of $\overline{M}$, $n = |I|$ is the number of processes, $|g|$ is the length of g, and a is the maximum size of the automaton $\mathcal{A}$ for any basic modality of g.*

Proof. We can handle any single basic modality in g in time $O(|\overline{M}| \cdot a \cdot n^3)$ by Theorem 3.9 and the definition of a. Then we can handle all of g, which is composed of boolean combinations and nesting of basic modalities, by recursive descent in time at most $O(|\overline{M}| \cdot n^3 \cdot |g| \cdot a)$ (cf. [Emerson and Lei 1987]). $\square$

ACKNOWLEDGMENT

We are grateful to many students, specifically Leo Milliades, for their help in the implementation of the system. We are also grateful to the anonymous referees for their valuable comments.

REFERENCES

Aggarwal, S., Kurshan, R. P., and Sabnani, K. K. 1983. *Protocol Specification, Testing and Verification III, H. Ruden, C. West Eds.* North-Holland, Chapter A Calculus for Protocol Specification and Validation, 19–34.

Bouajjani, A., Fernandez, J., Halbwichs, N., Raymond, P., and Ratel, C. 1992. Minimal state generation. *Science of Computer Programming*.

Clarke, E. M. and Emerson, E. A. 1981. Design and verification of synchronization skeletons using branching time temporal logic. In *Logics of Programs Workshop, Springer LNCS no. 131.*

Clarke, E. M., Emerson, E. A., and Sistla, A. P. 1983. Automatic verification of finite state concurrent programs using temporal logic: A practical approach. In *Proceedings of the ACM Symposium on Principles of Programming Languages* (Austin, Texas). ACM, NewYork. Journal version appeared in ACM TOPLAS, April 1986.

Clarke, E. M., Filkorn, T., and Jha, S. 1993. Exploiting symmetry in temporal logic model checking. In *5th International Conference on Computer Aided Verification* (Crete, Greece).

Clarke, E. M., Grumberg, O., and Brown, M. 1989. Reasoning about many identical processes. *Inform. and Comp.*.

Cleaveland, R. 1993. Analyzing concurrent systems using the concurrency workbench. In *Functional Programming, Concurrency, Simulation, and Automated Reasoning.* 129–144. Springer LNCS no. 693.

Dams, D., Grumberg, O., and Gerth, R. 1993. Generation of reduced models for checking fragments of ctl. In *Proceedings of Intl. Conf. on Computer Aided Verification.* also in Springer-Verlag LNCS no. 697.

Emerson, E. A. 1991. *Handbook of Theoretical Computer Science, J. van Leeuwen, Eds.* Elsevier/North-Holland, Chapter Temporal and Modal Logic.

Emerson, E. A. and Lei, C. L. 1987. Modalities for model checking: Branching time strikes back. *Science of Computer Programming 8*, 275–306.

Emerson, E. A. and Sistla, A. P. 1993. Symmetry and model checking. complete version to appear in the Journal Formal Methods in System Design.

Francez, N. 1986. *Fairness.* Springer-Verlag, New York.

Ip, N. C. and Dill, D. L. 1993. Better verification through symmetry. In *CHDL.*

Jensen, K. 1994. *Colored Petri Nets: Basic Concepts, Analysis Methods, and Practical Use, vol. 2: Analysis Methods, EATCS Monographs.* Springer-Verlag.

Jensen, K. and Rozenberg, G. 1991. *High-level Petri Nets: Theory and Application.* Springer-Verlag.

Kurshan, R. P. 1986. Testing containment of omega-regular languages. In *Bell Labs Tech. Report 1121-861010-33 (1986).* appeared in LNCS 103 (1987) Springer-Verlag 19-39.

Kurshan, R. P. 1994. *Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Approach.* Princeton University Press, Princeton, New Jersey.

Lee, D. and Yannakakis, M. 1992. On-line minimization of transition systems. In *Proceedings of ACM SYmp. on Theoretical Computer SCience.*

Litchtenstein, O. and Pnueli, A. 1985. Checking that finite state concurrent programs satisfy their linear specifications. In *Proceedings of ACM Symp. on Priciples of Prog. Languages.*

Streett, R. 1981. Propositional dynamic logic of looping and converse. Ph.D. thesis, Mass. Inst. of Technology, Cambridge, Mass., USA.

Vardi, M. and Wolper, P. 1984. An automata-theoretic framework for modal logics of programs. In *the Proceedings of ACM Symp. on Theory of Computing.*

Vardi, M. and Wolper, P. 1986. An automata-theoretic framework for automatic program verification. In *Proceedings of Intl. Symp. on Logic in Computer Science.*

Received July 1996; accepted September 1996.

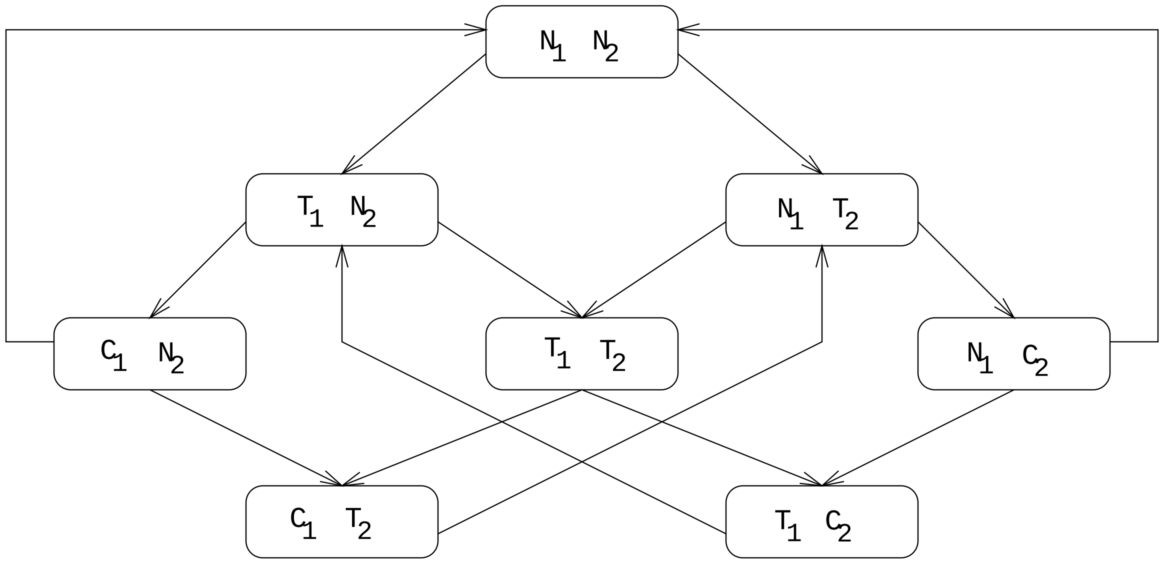

FIG 1 : Global Transition Graph

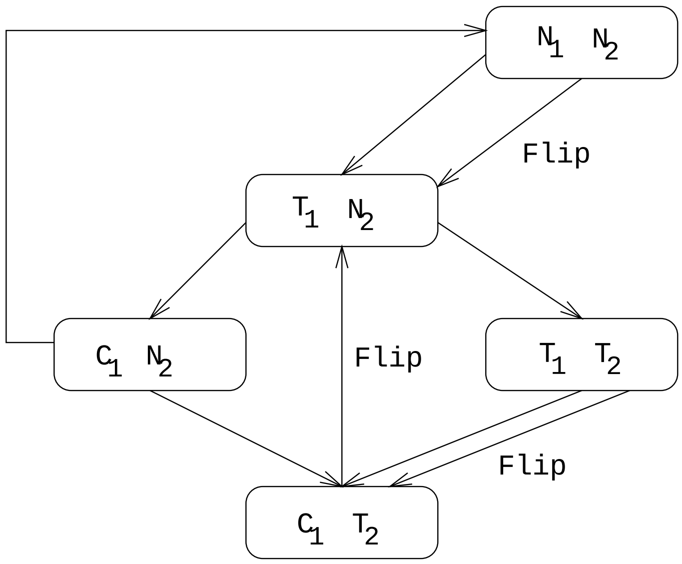

FIG 2 : The Reduced Graph

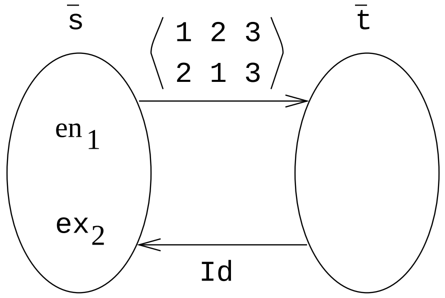

FIG 3 : A scsg $\bar{C}$

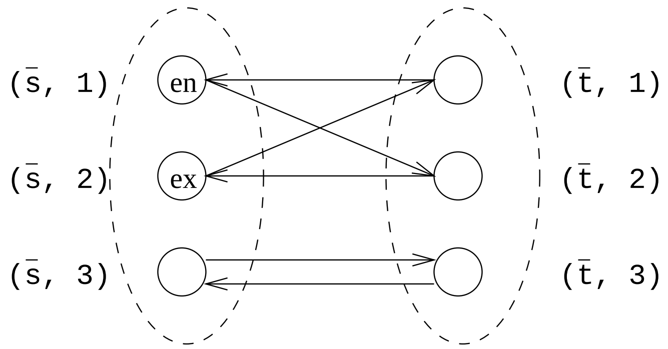

FIG 4 : The threaded graph C*

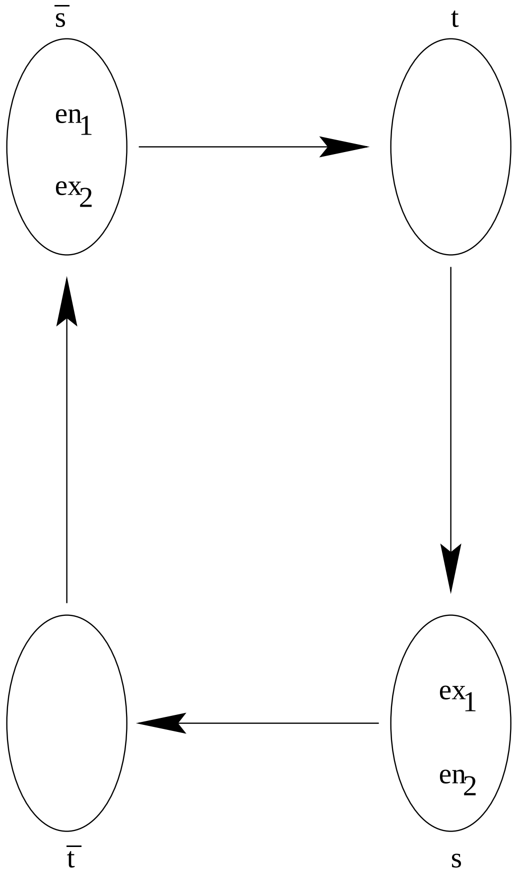

FIG 5: unwind ($\bar{C}$, $\bar{s}$)