On Combining State Space Reductions with Global Fairness Assumptions*

Shao Jie Zhang[1], Jun Sun[2], Jun Pang[3], Yang Liu[1], and Jin Song Dong[1]

[1] National University of Singapore
{shaojiezhang@,liuyang@comp.,dongjs@comp.}nus.edu.sg
[2] Singapore University of Technology and Design
sunjun@sutd.edu.sg
[3] University of Luxembourg
jun.pang@uni.lu

Abstract. Model checking has established itself as an effective system analysis method, as it is capable of proving/dis-proving properties automatically. Its application to practical systems is however limited by state space explosion. Among effective state reduction techniques are symmetry reduction and partial order reduction. Global fairness often plays a vital role in designing self-stabilizing population protocols. It is known that combining fairness and symmetry reduction is nontrivial. In this work, we first show that global fairness, unlike weak/strong fairness, can be combined with symmetry reduction. We extend the PAT model checker with the technique and demonstrate its usability by verifying recently proposed population protocols. Second, we show that partial order reduction is not property-preserving with global fairness.

1 Introduction

In the area of system verification and model checking, liveness means something good must eventually happen. A counterexample to a liveness property is typically a loop (or a deadlock state which can be viewed as a trivial loop) during which good things never occur. Fairness, which is concerned with a fair resolution of non-determinism, is often necessary and important to prove liveness properties. Fairness is an abstraction of the fair scheduler in a multi-threaded programming environment or the relative speed of the processors in distributed systems. Without fairness, verification of liveness properties often produces unrealistic infinite system executions during which one process or event is unfairly favored. It is important to systematically rule out those unfair counterexamples so as to identify real bugs.

The population protocol model has recently emerged as an elegant computation paradigm for describing mobile ad hoc networks [2]. A number of population protocols have been proposed and studied [2, 16]. Fairness plays an important role in these protocols. For instance, it was shown that the self-stabilizing population protocols for the complete network graphs only work under *weak fairness*, whereas the algorithm

* This research was partially supported by a grant "SRG ISTD 2010 001" from Singapore University of Technology and Design.

for network rings only works under *global fairness* [14]. Different from weak/strong fairness, global fairness requires that a transition (instead of an event or process) must be infinitely often taken if infinitely often enabled. It has been further proved that with only *strong fairness* or weaker, uniform self-stabilizing leader election in rings is impossible [14]. In order to verify (implementations of) those algorithms, model checking techniques must take the respective fairness constraints into account.

In our previous work [27], we developed a unified approach to model checking concurrent systems with a variety of fairness constraints. It was later applied to recently proposed population protocols [19] and previously unknown bugs are detected successfully. Nonetheless, it is limited by the state space explosion problem, like any model checking algorithm. Previous work has identified and solved the problem combining weak/strong fairness with state space reduction techniques like symmetry reduction [12] and partial order reduction [6]. In this work, we examine a combination of model checking with global fairness with symmetry reduction and partial order reduction. The contributions are stated below:

First, we investigate the problem of model checking with global fairness and symmetry reduction. Symmetry reduction is a natural choice to population protocols, or network protocols, which in general often contain many behaviorally similar or identical network nodes. Symmetry reduction has been investigated by many researchers for many years [8, 11, 3]. In [12, 15], it has been shown that combining weak/strong fairness with symmetry reduction is non-trivial. In this paper, we prove that different from weak/strong fairness, symmetry reduction and global fairness can be integrated without extra effort. Adding symmetry reduction slightly changes the algorithm for model checking with global fairness. We present the combined reduction algorithm based on Tarjan's strongly connected component algorithm [29].We extend our home-grown PAT model checker with symmetry reduction and show its scalability by verifying recently proposed population protocols.

Second, partial order reduction is an effective state reduction technique for concurrent systems with independent transitions. It is shown that partial order reduction preserves the behaviors with weak fairness, but not with strong fairness [22, 6]. In this paper, we examine the combination of partial order reduction and global fairness, and show that partial order reduction is not property preserving with global fairness.

2 Preliminaries

We present our work in the setting of Labeled Kripke structures (LKS) [7].

Definition 1 (LKS). *An LKS is a 6-tuple $\mathcal{L} = (S, init, \Sigma, \rightarrow, AP, L)$ where: S is a finite set of states; $init \in S$ is the initial state; Σ is a finite set of events; AP is a finite set of atomic state propositions; $\rightarrow$: $S \times \Sigma \times S$ is a transition-labeling relation with events; $L : S \rightarrow 2^{AP}$ is a state-labeling relation with atomic propositions.*

For simplicity, we write $s \overset{e}{\rightarrow} s'$ to denote that (s, e, s') is a transition in $\rightarrow$; $s \rightarrow s'$ to denote there exists some e in Σ such that $s \overset{e}{\rightarrow} s'$. Figure 1 shows an LKS, where transitions are labeled with event names and states are denoted by numbers, and 0 is the initial state. The dash-lined circles will be explained later.

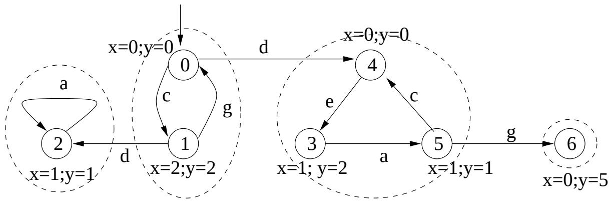


Fig. 1. Labeled Kripke system

We say that $\mathcal{L}$ is finite if and only if S is finite. A run of $\mathcal{L}$ is a finite or infinite sequence of alternating states and events $\langle s_0, e_0, s_1, e_1, \cdots \rangle$ such that $s_0 = init$ and $s_i \overset{e_i}{\rightarrow} s_{i+1}$ for all i. Because fairness affects infinite not finite system behaviors, we focus on infinite system runs in this paper. A state s is *reachable* if and only if there exists a finite run that reaches s. Throughout the paper, we assume that LKSs are always *reduced*, *i.e.*, all states are reachable.

We assume properties are stated in the form of state/event linear temporal logic (SE-LTL) formulae [7]. Given an LKS $\mathcal{L} = (S, init, \Sigma, \rightarrow, AP, L)$, an SE-LTL formula ϕ can be constituted by not only atomic state propositions but also events.

$$\phi ::= p \mid a \mid \neg\phi \mid \phi \wedge \phi \mid \mathbf{X}\phi \mid \mathbf{F}\phi \mid \mathbf{G}\phi \mid \phi\mathbf{U}\phi, \ where\ p \in AP\ and\ a \in \Sigma.$$

Definition 2. *Let $\pi = \langle s_0, e_0, s_1, e_1, \cdots \rangle$ be a run in $\mathcal{L}$ and π_i the suffix of π starting at s_i. The path satisfaction relation is defined as follows:*

- *$\pi \models p$ iff s is the first state of π and $p \in L(s)$.*
- *$\pi \models a$ iff a is the first event of π.*
- *$\pi \models \neg\phi$ iff $\pi \not\models \phi$.*
- *$\pi \models \phi_1 \wedge \phi_2$ iff $\pi \models \phi_1$ and $\pi \models \phi_2$.*
- *$\pi \models \boldsymbol{X}\phi$ iff $\pi_1 \models \phi$.*
- *$\pi \models \boldsymbol{F}\phi$ iff there exists a $k \geq 0$ such that $\pi_k \models \phi$.*
- *$\pi \models \boldsymbol{G}\phi$ iff for all $i \geq 0$ such that $\pi_i \models \phi$.*
- *$\pi \models \phi_1\boldsymbol{U}\phi_2$ iff there exists a $k \geq 0$ s.t. $\pi_k \models \phi_2$ and for all $0 \leq j < k, \pi_j \models \phi_1$.*

An example is $\mathbf{G}(d \Rightarrow \mathbf{F}(x > 1))$ where d is an event and $x > 1$ is an atomic proposition. The formula states that event d is always followed by a run such that $x > 1$ is eventually satisfied.

3 Model Checking with Fairness

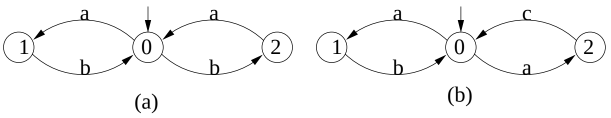
A fairness constraint restricts the set of system behaviors to only those fair ones. Without fairness constraints, a system may behave freely as long as it starts with an initial state and conforms to the transition relation. There are a variety of fairness constraints, *i.e.*, event-level or process-level weak fairness, event-level or process-level strong fairness, global fairness, etc. In the following, we briefly review weak and strong fairness and then focus on global fairness. For simplicity, we focus on event-level fairness.

3.1 Fairness

Event-level weak fairness [17] states that if an event becomes enabled *forever* after some steps, then it must be engaged infinitely often. An equivalent formulation is that every run should contain infinitely many positions at which the event is disabled or has occurred. Given the LKS presented in Figure 1, the run $\langle 0, c, 1, g\rangle^\omega$ where the superscript ω indicates an infinite number of repetitions does not satisfy event-level weak fairness because event d is always enabled (*i.e.*, at both state 0 and 1) but never occurs during the run. The run which loops through state 3, 4 and 5 satisfies weak fairness as no event is enabled forever. Event-level strong fairness states that if an event is *infinitely often* enabled, it must infinitely often occur. This type of fairness is particularly useful in the analysis of systems that use semaphores, synchronous communication, and other special coordination primitives. It has been identified by different researchers [18, 14, 24]. Given the LKS presented in Figure 1, the run which loops through state 3, 4 and 5 does not satisfy strong fairness because event g is infinitely often enabled but never occurs. It can be shown that strong fairness implies weak fairness. Model checking with weak or strong fairness, or combing weak/strong fairness with state space reduction techniques has been well investigated [10, 25–28].

Definition 3 (Global fairness). *Let $E = \langle s_0, e_0, s_1, e_1, \cdots\rangle$ be a run of an LKS $\mathcal{L}$. E satisfies global fairness if and only if, for every s, e, s' such that $s \xrightarrow{e} s'$, if $s = s_i$ for infinitely many i, then $s_i = s$ and $e_i = e$ and $s_{i+1} = s'$ for infinitely many i.*

Global fairness[4] was proposed by Fischer and Jiang in [14]. It is in fact a restricted form of extreme fairness proposed by Pnueli [23]. Global fairness states that if a *step*[5] (from s to s' by engaging in event e) can be taken infinitely often, then it must actually be taken infinitely often. Many population protocols rely on global fairness [2, 14]. Compared to event-level strong fairness, global fairness requires that an infinitely enabled event must be taken infinitely often in *all* contexts, whereas event-level strong fairness only requires the enabled event to be taken in *one* context. Thus, global fairness is stronger than strong fairness. Their difference is illustrated in the following figure.


(a)
(b)

Under event-level strong fairness, state 2 in (a) may never be visited because all events occur infinitely often if the left loop is taken infinitely. With global fairness, all states in (a) must be visited infinitely often. Their difference when there is non-determinism is illustrated in (b). Both transitions labeled a must be taken infinitely with global fairness, which is not necessary with event-level strong or weak fairness. *It can be shown that global fairness coincides event-level strong fairness when every transition is labeled with a different event.* This observation implies that we can uniquely label all transitions

[4] In [14], it is called strong global fairness and defined for unlabeled transition systems. We slightly changed it so as to suit the setting of LKS.

[5] Step and transition are used interchangeably in this paper.

with different events and then apply model checking algorithm for strong fairness to deal with global fairness. We show however, model checking with global fairness can be solved using a more efficient approach. In contrast to nontrivial combination of strong fairness and symmetry reduction [12], we show that using our approach model checking with global fairness can be straightforwardly combined with symmetry reduction.

3.2 Model Checking with Fairness

Given an LKS $\mathcal{L}$ and a liveness property ϕ, model checking is to search for a run of $\mathcal{L}$ which fails ϕ. In automata-based model checking, the negation of ϕ is translated to an equivalent Büchi automaton $\mathcal{B}$. Model checking with fairness is to search for a system run which is accepting by $\mathcal{B}$ *whilst satisfying the fairness constraints*. In the following, we write $\mathcal{L} \vDash \phi$ to mean that $\mathcal{L}$ satisfies the property (without fairness assumption) and write $\mathcal{L} \vDash_{gf} \phi$ to mean that $\mathcal{L}$ satisfies the property with global fairness, *i.e.*, every run of $\mathcal{L}$ which satisfies global fairness also satisfies ϕ. We define a *loop* in the product of $\mathcal{L}$ and B is a sequence of alternating states/events:

$$\langle (s_0, b_0), e_0, (s_1, b_1), e_1, \cdots, (s_{n-1}, b_{n-1}), e_n, (s_n, b_n) \rangle^\omega$$

such that for all $0 \leq i \leq n$, s_i is a state of $\mathcal{L}$, b_i is a state of $\mathcal{B}$, (s_0, b_0) is reachable, $s_n = s_0$ and $b_n = b_0$. A loop is *accepting* if and only if there exists at least one accepting state of $\mathcal{B}$ in $\langle b_0, b_1, \cdots, b_n \rangle$. Furthermore, we define the following sets for a loop l whose projection on $\mathcal{L}$ is $l_{\mathcal{L}} = \langle s_0, e_0, s_1, e_1, \cdots, s_{n-1}, e_{n-1}, s_0 \rangle$.

$$\begin{aligned}
&\mathit{onceStep}(l) &&= \textstyle\bigcup_{k=0}^{n-1} \mathit{enabled}(s_k) \\
&\mathit{engagedStep}(l) &&= \textstyle\bigcup_{k=0}^{n-1} \mathit{engaged}(s_k, l) \\
&\mathit{enabled}(s) &&= \{(s, e, s') \mid s \xrightarrow{e} s'\} \\
&\mathit{engaged}(s_k, l) &&= \{(s_k, e_k, s_{k+1}) \mid \langle s_k, e_k, s_{k+1} \rangle \text{ is a subsequence of } l_{\mathcal{L}}\}
\end{aligned}$$

Intuitively, $\mathit{onceStep}(l)$ is the set of steps which are enabled at least once during the loop, and $\mathit{engagedStep}(l)$ is the set of steps which are engaged during the loop. By definition, the proposition follows immediately.

Proposition 1. *Let $E = m(l^\omega)$ be a run in $\mathcal{L}$ where m is a finite run. E satisfies global fairness if and only if $\mathit{onceStep}(l) = \mathit{engagedStep}(l)$.* □

3.3 Algorithm for Model Checking with Global Fairness

Model checking with fairness can often be reduced to search for strongly connected components (SCC). In graph theory, an SCC is defined as a *maximum* subgraph such that every pair of vertices in the subgraph is connected by a path in the subgraph. A terminal SCC is an SCC such that all of its edges lead to vertices contained in the SCC. Naturally, an LKS can be viewed as a directed graph and therefore the concept of SCC can be extended to LKS. For instance, the LKS presented in Figure 1 contains four SCCs, indicated by dash-lined circles. Among the four, the one containing state 2 is terminal, whereas the one containing state 0 and 1 is not. For simplicity, we refer to a set of states of an LKS as an SCC if the subgraph containing the states and the transitions among them forms an SCC. We write that 'an SCC fails a liveness property ϕ' as equivalent to that a run which reaches any state in the SCC and infinitely often traverses through all states and transitions of the SCC fails ϕ. For instance, the SCC containing state 2 fails the property $\mathbf{G}(d \Rightarrow \mathbf{F}(x > 1))$.

In our previous work [27], we proved that the problem of model checking with global fairness can be reduced to the problem of searching for a terminal SCC which fails the given property. Formally, it can be stated as the following theorem.

Theorem 1. *Let $\mathcal{L}$ be an LKS; ϕ be a property. $\mathcal{L} \vDash_{gf} \phi$ if and only if there does not exist a terminal SCC S in $\mathcal{L}$ such that S fails ϕ.* □

The theorem implies that we can use a simple procedure to find a counterexample by enumerating all terminal SCCs and then testing each one of them. The approach implemented in the PAT model checker is based on Tarjan's algorithm for on-the-fly identification of SCCs. Its complexity is linear in the number of edges in the graph. Given the LKS presented in Figure 1 with the property $\mathbf{G}(d \Rightarrow \mathbf{F}(x > 1))$, the SCC containing state 2 is identified as a counterexample with global fairness. Note that the SCC containing state 3, 4, and 5 is a counterexample only with no fairness or weak fairness. It is not a counterexample with global fairness because it does not satisfy global fairness, *i.e.*, the step from state 5 to 6 by performing g is enabled infinitely often but never occurs.

4 Model Checking with Symmetry Reduction

Distributed/concurrent systems, especially communicating protocols, often exhibit considerable symmetry. Symmetry reduction aims to explore the symmetry in order to reduce state space. Intuitively, the idea is that states which are symmetric exhibit similar or even identical behaviors and therefore exploring one representative would suffice in proving/dis-proving a property. In the following, we briefly introduce symmetry reduction (refer to Chapter 14 of [9] for details), using the following running example.

Example 1. In [14], a self-stabilizing leader election protocol is proposed for complete networks. The system contains multiple network nodes which interact with each other following a number of simple rules. The system is modeled in the following form.

$$System = Controller \parallel Node(0) \parallel Node(1) \parallel \cdots \parallel Node(N)$$

where $Controller$ is a controlling process distinguished from the network nodes; $Node(i)$ models a network node with a unique identity i; $\parallel$ denotes parallel composition. A node is marked as either a leader or not. Two nodes can interact according to the rules and start/quit being a leader. For instance, one of the rules states that if two interacting nodes are both leaders, then one of the nodes quits being a leader. The network nodes (*i.e.*, process $Node(i)$) are indistinguishable in the protocol and therefore they are all symmetric. One essential property of the protocol is that all nodes must eventually converge to the correct configuration. That is, eventually always there is one and only one leader in the network, *i.e.*, $\mathbf{FG}\ one\ leader$.

A permutation σ on a finite set of objects is a bijection (*i.e.*, a function that is one-to-one and onto). For instance, a permutation of the process identities in the above example is: $\sigma_0 = 0 \mapsto 1, 1 \mapsto 2, \cdots, N-1 \mapsto N, N \mapsto 0$ where $0 \mapsto 1$ reads as '0 maps to 1'. A permutation group is a group of permutations. For instance, the group containing all permutation of process identities in the leader election example is a permutation group. Given an LKS $\mathcal{L} = (S, init, \Sigma, \rightarrow, AP, L)$, let G be a permutation group of process identities acting on S. *We first assume any event in Σ is not allowed to be permuted.* A permutation σ is said to be an automorphism of $\mathcal{L}$ if and only if it preserves the transition relation and initial state. Formally, σ satisfies the following condition.

$$(\forall\, s_1, s_2 \in S;\ \ e \in \Sigma.\ s_1 \overset{e}{\rightarrow} s_2 \Leftrightarrow \sigma(s_1) \overset{e}{\rightarrow} \sigma(s_2)) \wedge \sigma(init) = init$$

A group T is an automorphism group of $\mathcal{L}$ if and only if every $\sigma \in T$ is an automorphism of $\mathcal{L}$. A permutation σ is said to be an invariance of an SE-LTL formula ϕ if and only if $\sigma(\phi) \equiv \phi$ where $\equiv$ denotes logical equivalence under all propositional interpretations [11]. For instance, given any permutation of process identities in the leader election example, the truth value of proposition $one\ leader$ remains the same and therefore the permutation is an invariance of **FG** $one\ leader$. A permutation σ is said to be an invariance of $\mathcal{L}$ and property ϕ if and only if it is an automorphism of $\mathcal{L}$ and it is an invariance of ϕ. G is an invariance group of $\mathcal{L}$ and ϕ if and only if every $\sigma \in G$ is an invariance of $\mathcal{L}$ and ϕ.

Given a state $s \in S$, the orbit of s is the set $\theta(s) = \{t \mid \exists \sigma \in G.\ \sigma(s) = t\}$, *i.e.*, the equivalence group which contains s. From each orbit of state s, a unique representative state $rep(s)$ can be picked such that for all s and s' in the same orbit, $rep(s) = rep(s')$. Intuitively, if σ is an invariance of ϕ, states of the same orbit are behaviorally indistinguishable with respect to ϕ. For instance, the states of the 0-node being the only leader and the 1-node being the only leader in the leader election protocol are indistinguishable to the property **FG** $one\ leader$. Based on this observation, an LKS can be turned into a *quotient* LKS where states in the same orbit are grouped together. Formally, a quotient LKS is defined as follows.

Definition 4. *Let $\mathcal{L} = (S, init, \Sigma, \rightarrow, AP, L)$ be an LKS; G be an automorphism group. The quotient LKS $\mathcal{L}_G = (S_G, init_G, \Sigma, fun_G, AP, L)$ is defined as follows:*

- $S_G = \{rep(s) \mid s \in S\}$ *is the set of representative states of orbits.*
- $init_G = \{rep(init)\}$ *is the initial representative state.*
- $(r, e, r') \in \rightarrow_G$ *iff there exists $r'' \in S$ such that $r \overset{e}{\rightarrow} r''$and $rep(r'') = r'$.*

It has been proved [9] that if G is an invariance group of $\mathcal{L}$ and ϕ, then $\mathcal{L}$ satisfies ϕ if and only if $\mathcal{L}_G$ satisfies ϕ. Formally, it is stated as the following theorem. It is proved by showing that the relation $(s, \theta(s))$ is a bi-simulation relation between $\mathcal{L}$ and $\mathcal{L}_G$.

Theorem 2. *Let $\mathcal{L} = (S, init, \Sigma, \rightarrow, AP, L)$ be an LKS; ϕ be an SE-LTL formula. If G be an invariance group of $\mathcal{L}$ and ϕ, then $\mathcal{L} \vDash \phi$ if and only if $\mathcal{L}_G \vDash \phi$.* □

5 Symmetry Reduction with Global Fairness

In the following, we prove that global fairness is orthogonal with symmetry reduction by showing that there is a run which satisfies global fairness and fails ϕ in $\mathcal{L}$ if and only if there is a run which satisfies global fairness and fails ϕ in $\mathcal{L}_G$. For convenience, we fix that ϕ is an $SE\text{-}LTL$ formula to be checked, $\mathcal{B}$ is the Büchi automaton constructed by the negation of ϕ, $\mathcal{L}$ is LKS of the original system, G is invariance group of $\mathcal{L}$ and ϕ and $\mathcal{L}_G$ is LKS of the abstract system after applying symmetry reduction.

Lemma 1. *There exists a run $p = \langle s_0, a_0, s_1, a_1, \cdots \rangle$ in $\mathcal{L}$ if and only if there exists a run $q = \langle r_0, a_0, r_1, a_1, \cdots \rangle$ in $\mathcal{L}_G$ such that $r_i = rep(s_i)$ for all i.*

Proof It follows from the proof of Lemma 3.1 in [11]. □

Theorem 3. *There exists an accepting loop in the product of $\mathcal{L}$ and $\mathcal{B}$ which satisfies global fairness if and only if there also exists an accepting loop in the product of $\mathcal{L}_G$ and $\mathcal{B}$ which satisfies global fairness.*

Proof: (***Sufficient condition***) We first prove the sufficient condition. The proof is divided into two parts. In the first part, we prove (1) if there exists an accepting loop l' in the product of $\mathcal{L}_G$ and $\mathcal{B}$, then there exists an accepting loop l in the product of $\mathcal{L}$ and $\mathcal{B}$. Then we prove (2) if l' satisfies global fairness, so does l.

Let $l' = \langle (r_0, b_0), a_0, (r_1, b_1), a_1, \cdots, (r_{n-1}, b_{n-1}), a_{n-1}, (r_0, b_0) \rangle$ be an accepting loop. Without loss of generality we assume that b_0 is an accepting state. Then there exists in the product of $\mathcal{L}_G$ and $\mathcal{B}$ a path arriving at (r_0, b_0). By Lemma 1 there exists a corresponding path in the product of $\mathcal{L}$ and $\mathcal{B}$ to state (s_0, b'_0) where $r_0 = rep(s_0)$. Because G is the invariance group of L and ϕ, $b'_0 = b_0$ which is also an accepting state. By Lemma 1 again, for l' there exists in the product of $\mathcal{L}$ and $\mathcal{B}$ a path $p^0 = \langle (s_0, b_0), a_0, (s_1, b_1), a_1, \cdots, (s_{n-1}, b_{n-1}), a_{n-1}, (s_0^1, b_0) \rangle$ such that for all i in p^0 we have $r_i = rep(s_i)$. Notice that p^0 is not necessarily a loop. Since $r_0 = rep(s_0^1)$, we can unfold l' again according to Lemma 1, but this time beginning at s_0^1, which will produce the path $p^1 = \langle (s_0^1, b_0), a_0, (s_1^1, b_1), a_1, \cdots, (s_{n-1}^1, b_{n-1}), a_{n-1}, (s_0^2, b_0) \rangle$, and for all i in p^1 we still have $r_i = rep(s_i^1)$. We can repeat this unfolding arbitrary many times which will give us a sequence of path $p^0, p^1, p^2, \cdots$ with the corresponding end states $(s_0^1, b_0), (s_0^2, b_0), (s_0^3, b_0), \cdots$ which are all accepting. As the orbit of the states $s_0, s_0^1, s_0^2, \cdots$ is finite, $s_0^i = s_0^j$ for some i and j. Obviously, the concatenation of the paths p^i to p^{j-1}, say l, is an accepting loop in the product of $\mathcal{L}$ and $\mathcal{B}$.

Because l' satisfies global fairness, $onceStep(l') = engagedStep(l')$. We define a function $recover$ such that given $(s, e, s') \in \rightarrow_G$ and some permutation $\sigma \in G$, $recover((s, e, s'), \sigma) = (t, e, t')$ such that $s\sigma^{-1} = t \wedge t \xrightarrow{e} t'$. Intuitively, $recover$ returns the corresponding transition of (s, e, s') in $\mathcal{L}$ with respect to a specific permutation σ. For $0 \leq m \leq n$, r_m in loop l' corresponds to s_m^t (*i.e.*, $r_m = s_m^t \sigma_m^t$) in each path p^t ($i \leq t < j$). Then

- $enabled(s_m^t) = recover(enabled(r_m), \sigma_m^t)$;
- $engaged(s_m^t, p^t) = recover(engaged(r_m, l'), \sigma_m^t)$.

Thus, $onceStep(l) = \{recover(enabled(r_m), \sigma_m^t),\ 0 \leq m < n,\ i \leq t < j\}$ and $engagedStep(l) = \{recover(engaged(r_m, p^t), \sigma_m^t),\ 0 \leq m < n,\ i \leq t < j\}$. Since $onceStep(l') = engagedStep(l')$, $onceStep(l') = \{enabled(r_m), 0 \leq m \leq n, i \leq t < j\}$ and $engagedStep(l') = \{engaged(r_m, p^t), 0 \leq m \leq n, i \leq t < j\}$, we have $onceStep(l) = engagedStep(l)$.

(Necessary condition) Let $l = \langle(s_0, b_0), a_0, (s_1, b_1), a_1, \cdots, (s_{n-1}, b_{n-1}), a_{n-1}, (s_0, b_0)\rangle$ be an accepting loop in the product of $\mathcal{L}$ and $\mathcal{B}$. There exists a path arriving at (s_0, b_0). Assume b_0 is an accepting state in $\mathcal{B}$. By Lemma 1 there exists a path in the product of $\mathcal{L}_G$ and $\mathcal{B}$ leading to state $(rep(s_0), b_0)$. By Lemma 1, there exists in the product of $\mathcal{L}_G$ and $\mathcal{B}$ a corresponding loop $l' = \langle(s_0\sigma_0, b_0), a_0, (s_1\sigma_1, b_1), a_1, \cdots, (s_{n-1}\sigma_{n-1}, b_{n-1})$, $a_{n-1}, (s_0\sigma_0, b_0)\rangle$ such that $\sigma_i \in G$ and $rep(s_i) = s_i\sigma_i$ for all $0 \leq i < n$.

Because l satisfies global fairness, $onceStep(l) = engagedStep(l)$. We define a function $twist$ such that given $s \xrightarrow{e} s'$, $twist(s, e, s') = rep(s) \xrightarrow{e}_G rep(s')$. Intuitively, $twist$ returns the corresponding transition in $\mathcal{L}_G$ of (s, e, s'). For all $0 \leq i < n$, s_i in loop l corresponds to $rep(s_i)$ in l'. Then

- $enabled(rep(s_i)) = twist(enabled(s_i))$;
- $engaged(rep(s_i), l') = twist(engaged(s_i, l))$.

Thus, $onceStep(l') = \{twist(enabled(s_i)), 0 \leq i < n\}$ and $engagedStep(l') = \{twist(engaged(s_i, l')), 0 \leq i < n\}$. Because $onceStep(l) = engagedStep(l)$, we have $onceStep(l) = \{enabled(s_i), 0 \leq i < n\}$ and $engagedStep(l) = \{engaged(s_i, l)$, $0 \leq i < n\}$. Thus, we have $onceStep(l') = engagedStep(l')$. □

Note that we did not allow the events to be permuted in the definition of permutation given at the beginning of this section, which seems too restrictive. Now we relax the definition of permutation to permute states and events simultaneously. It is proved in [13] that the new definition is equivalent to the one given before. By a simple argument, it can be shown that Theorem 3 still holds.

Based on Theorem 3, we present a practical algorithm for searching the reduced state space for accepting globally fair loops, based on Tarjan's SCC algorithm. Underlining shows the differences compared with the usual algorithm for model checking with global fairness. Assume that G is a permutation group of process identities which is an invariance group of $\mathcal{L}$ and ϕ. Let rep be a function which, given a state, returns a unique representative. Using function rep, we can tell whether two states are in the same orbit or not. Note that identifying an optimal representative function rep can be nontrivial. We adopt the automata-theoretic approach and perform the following. Firstly, a Büchi automaton $\mathcal{B}$ is generated from the negation of ϕ. Next, the synchronous product of $\mathcal{B}$ and $\mathcal{L}$ is computed on-the-fly. Tarjan's SCC algorithm is used to identify SCC in the product along the construction. Note that a state of the product is a pair (s, b) where s is a state of $\mathcal{L}$ and b is a state of $\mathcal{B}$. Assume that the initial state of the product is $(init_s, init_b)$ where $init_s$ is the initial state of $\mathcal{L}$ and $init_b$ is the initial state of $\mathcal{B}$[6].

The detailed algorithm is presented in Figure 2. It resembles the standard Tarjan's SCC algorithm [29]. Note that we use the iterative version of Tarjan's SCC algorithm in the practice implementation for performance reason. Three data structures are used to identify SCCs: $path$ is a stack containing states along a path from the initial state to the current one; $index$ and $lowlink$ are hash tables which assign two numbers to a state. A state is a root of an SCC if and only if the two numbers are equivalent. To apply symmetry reduction, instead of working with concrete states, Tarjan's algorithm is applied to representatives of orbits. For instance, $path$ contains only $rep(v)$ (line

[6] For simplicity, we assume there is only one initial state in $\mathcal{B}$.

10) and $lowlink$ and $index$ map $rep(v)$ to numbers (line 7 and 8). Whenever an SCC is identified (line 17), we check whether the SCC is terminal in $\mathcal{L}$ and accepting. If it is, then we prove existence of at least one counterexample. We skip the details on generating a concrete counterexample. Note that an SCC is terminal in $\mathcal{L}$ if and only if, for every state (s, b) in the SCC, if $s \rightarrow s'$, then there exists (s', b') in the SCC. An SCC is accepting if and only if it contains a state (s, b) such that b is an accepting state in $\mathcal{B}$. The algorithm terminates when all states have been checked. The correctness of the algorithm follows from the theorems presented in previous sections. It is always terminating because the number of un-explored states are monotonically decreasing and the number of states are finite. Its complexity is linear in the edges of transitions in the product of $\mathcal{L}$ and $\mathcal{B}$.

The following claims establish the correctness of the algorithm.

Lemma 2. *In the product of $\mathcal{L}$ (resp. $\mathcal{L}_G$) and $\mathcal{B}$, there exists an accepting loop which satisfies global fairness if and only if there exists an accepting SCC which is also a terminal SCC in $\mathcal{L}$ (resp.$\mathcal{L}_G$).*

Proof: ***(Necessary Condition)*** Suppose l is an accepting loop which satisfies global fairness. so $onceStep(l) = engagedStep(l)$. The states in l forms a strongly connected subgraph S in the product and S is a terminal SCC in $\mathcal{L}$. Let S' be the SCC that contains the states in S. Suppose l' be the loop which traverses all transitions in S. Because S is a terminal SCC in $\mathcal{L}$, $onceStep(l') = engagedStep(l') = onceStep(l)$. So S' is also a terminal SCC in $\mathcal{L}$. On the other hand, because l is accepting, there is an accepting state in S'.
(Sufficient Condition) Suppose S is an accepting SCC in the product of $\mathcal{L}$ and $\mathcal{B}$, and it is a terminal SCC in $\mathcal{L}$. Let l be the loop which traverses all transitions in S. We get $onceStep(l) = engagedStep(l)$. so l is a globally fair loop. Since there is an accepting state in l, l is an accepting loop which satisfies global fairness.

Using same argument one can show the lemma holds for product of $\mathcal{L}_G$ and $\mathcal{B}$. □

Theorem 4. *Let ϕ be an SE-LTL formula. If G is an invariance group of $\mathcal{L}$ and ϕ, then $\mathcal{L} \models_{gf} \phi$ if and only if $\mathcal{L}_G \models_{gf} \phi$.*

Proof By Theorem 1, $\mathcal{L} \not\models_{gf} \phi$ if and only if there exists an accepting SCC in the product of $\mathcal{L}$ and $\mathcal{B}$ which is also a terminal SCC in $\mathcal{L}$. Similarly, $\mathcal{L}_G \not\models_{gf} \phi$ if and only if there exists an accepting SCC in the product of $\mathcal{L}_G$ and $\mathcal{B}$ which is also a terminal SCC in $\mathcal{L}_G$. By Theorem 3 and Lemma 2, there exists an accepting SCC S such that S is a terminal SCC in $\mathcal{L}$ if and only if there exists an accepting SCC S' such that S' is a terminal SCC in $\mathcal{L}_G$, which proves the theorem. □

6 Partial Order Reduction with Global Fairness

In this section, we show that partial order reduction is not property-preserving with global fairness, which means that partial order reduction cannot be applied in our setting.

We begin by fixing notations and terminology. Given an LKS $\mathcal{L} = (S, init, \Sigma, \rightarrow, AP, L)$, the function $\alpha(s)$ returns the set α-successors of s in $\mathcal{L}$. That is, $s' \in \alpha(s)$ iff $s \xrightarrow{\alpha} s'$. Two fundamental relations are first defined for partial order reduction.

```
int counter := 0;
stack path := an empty stack;
hashtable index := an empty hash table;
hashtable lowlink := an empty hash table;
TarjanModelChecking($(init_s, init_b)$);

procedure TarjanModelChecking(v)
    $index[\underline{rep(v)}] := counter$;
    $lowlink[\underline{rep(v)}] := counter$;
    $counter := counter + 1$;
    push $\underline{rep(v)}$ into $path$
    forall $v \rightarrow v'$ do
        if ($\underline{rep(v')}$ is not in index)
            TarjanModelChecking($v'$)
            $lowlink[\underline{rep(v)}] = min(lowlink[\underline{rep(v)}], lowlink[\underline{rep(v')}])$;
        else if ($rep(v')$ is in $path$)
            $lowlink[\underline{rep(v)}] = min(lowlink[\underline{rep(v)}], index[\underline{rep(v')}])$;
        endif
    endfor
    if ($lowlink[\underline{rep(v)}] = index[\underline{rep(v)}]$)
        set $scc$ := an empty set;
        repeat
            pop an element $v'$ from $path$ and add it into $scc$;
        until ($v' = v$)
        if ($scc$ forms a terminal SCC in $\mathcal{L}$ and $scc$ is accepting)
            generate a counterexample and return false;
        endif
    endif
endprocedure
```

Fig. 2. Tarjan's algorithm with symmetry reduction

Definition 5. *An independence relation $I \subseteq \rightarrow \times \rightarrow$ is a symmetric, antireflexive relation, satisfying the following two conditions for each state $s \in S$ and for each $(\alpha, \beta) \in I$: (1) If $\alpha, \beta \in enabled(s)$, then $\alpha \in enabled(\beta(s))$. (2) If $\alpha, \beta \in enabled(s)$, then $\alpha(\beta(s)) = \beta(\alpha(s))$. The dependency relation is the complement of I.*

Definition 6. *Let $L : S \rightarrow 2^{AP}$ be the function that labels each state with a set of atomic propositions. A transition $\alpha \in T$ is invisible with respect to a set of propositions $AP' \subseteq AP$ if for each pair of states $s, s' \in S$ such that $s' = \alpha(s), L(s) \cap AP' = L(s') \cap AP'$.*

The state space reduction is achieved by only exploring a subset of $enabled(s)$, called $ample(s)$ for any visiting state s. The following conditions on $ample(s)$ are used to preserve properties to be verified [9].

C0 $ample(s) = \varnothing$ iff $enabled(s) = \varnothing$.

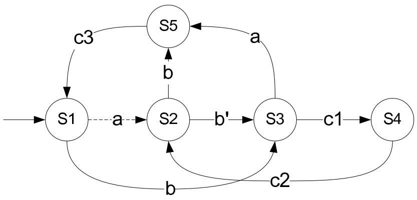


Fig. 3. Model and its reduction

C1 Along every path in the full state space starting from s, a transition that is dependent on a transition in $ample(s)$ cannot occur without one in $ample(s)$ occurring first.
C2 If $enabled(s) \neq ample(s)$, then every $\alpha \in ample(s)$ is invisible.
C3 A cycle is not allowed if it contains a state in which some transition α is enabled, but is never included in $ample(s)$ for any state s on the cycle.

It is proved in [9] that when satisfying the above four conditions, the following holds.

Theorem 5. *The original state space and reduced state space are stuttering equivalent.*

Based on Theorem 5, for any globally fair path in the full state space, there is a stuttering equivalent path in the reduced state space. Unfortunately, this path may be not globally fair. Figure 3 shows a part of the full state graph. The transition from $s1$ to $s2$ is not present in the reduced state graph. Let transitions labeled with a and b be independent and all other transitions be mutually dependent. Further let b, b' be invisible and a, c_1, c_2, c_3 visible. For the globally fair path $\lambda = (abc_3bc_1c_2b'ac_3)^\omega$ in the full state space, there is no stuttering equivalent globally fair path in the reduced state space. Because any globally fair path π in the reduced one has to traverse the transition labeled with b from state $s2$ to $s5$, π must include a segment stuttering-equivalent path to c_2c_3 whereas λ does not have such segment.

7 Implementation and Evaluation

In the following, we evaluate the effectiveness of our combined method. We extend the PAT[7] model checker with our algorithms for model checking with global fairness and symmetry reduction.

Previously in [19], PAT has been applied to model checking population protocols with global fairness without symmetry reduction. It is evidenced that only small networks can be checked. In the population protocol model, one protocol consists of N nodes, numbered from 0 to $N - 1$. A protocol is usually described by a set of interaction rules between an initiator u and a responder v. Such rules have conditions on the state and the input of the initiator and the responder, and specify the state of the initiator and the responder if a transition can be taken. Interested readers are referred to [19] for protocol details. Note that many of the protocols are designed for network rings. It has been noticed that protocols designed for network rings often require global fairness. All relevant experiment information is provided online [1].

[7] http://pat.comp.nus.edu.sg

Model	Network Size	Without Reduction		With Reduction		
		States	Time (Sec)	States	Time (Sec)	Gain
two-hop coloring	3	122856	36.7	42182	16.7	54.5%
orienting rings (prop 1)	3	19190	2.27	6398	0.53	76.7%
orienting rings (prop 2)	3	19445	2.23	6503	0.97	56.5%
orienting rings (prop 1)	4	1255754	267.2	313940	70.5	73.6%
orienting rings (prop 2)	4	1206821	267.1	302071	63.6	79.6%
orienting rings (prop 1)	5	11007542	9628.1	2201510	1067.4	88.9%
orienting rings (prop 2)	5	10225849	8322.6	2045935	954.5	88.5%
leader election (complete)	3	6946	0.87	2419	0.51	41.4%
leader election (complete)	4	65468	11.6	16758	5.00	56.9%
leader election (complete)	5	598969	176.1	120021	45.9	73.9%
leader election (odd)	3	55100	6.27	18561	2.56	59.2%
leader election (odd)	5	–	–	6444097	5803.96	×
token circulation	3	728	0.12	244	0.09	25.0%
token circulation	4	4466	0.35	1118	0.19	45.7%
token circulation	5	24847	1.86	4971	0.77	58.6%
token circulation	6	129344	10.7	21559	3.03	71.7%
token circulation	7	643666	77.2	91954	16.2	79.0%
token circulation	8	3104594	740.8	388076	97.1	86.9%

Table 1. Experiment Results

8 Related Work

This work is related to research on combining fairness and symmetry reduction. A solution for applying symmetry reduction under weak/strong fairness was discussed in [12]. Their method works by finding a candidate weak/strong fair path in the abstract transition system and then using annotations of state permutation details for each transition, in order to resolve the abstract path to a threaded structure which then determines whether there is a corresponding fair path in the concrete transition system. A similar approach was presented in [15]. Another close work is a nested depth first search algorithm that combines symmetry reduction with weak fairness [5]. Unfortunately, the combined algorithm cannot guarantee to preserve all behaviors under weak fairness and thus may produce false positives.

We compare our algorithm with the one which handles strong fairness in [12]. Since global fairness can be regarded as a kind of strong fairness, the algorithm is applicable to global fairness. It is the only algorithm for combining strong fairness and symmetry reduction that we could find in literature. First, Theorem 3.11 in [12] shows its time complexity is $O(\mid \overline{M} \mid \times n^3 \times \mid g \mid \times a)$, where $\mid \overline{M} \mid$ is the size of the reduced graph $\overline{M}$, n is the number of processes, $\mid g \mid$ is the length of the checked property g, and a is the maximum size of the automaton for any basic modality of g. Our algorithm is almost identical to Tarjan's SCC algorithm except for adding line 24, 25 in Figure 2. For a found SCC c the condition checking in line 24 can be implemented in time linear in the number of edges in c. As a result our algorithm can be implemented in time

$O(\mid \overline{M} \mid \times \mid g \mid \times a)$. Second, in our approach it is not necessary to record permutations appearing on each path (unless unwinding an abstract counterexample) and to construct threaded structure for each strong connected subgraph B, of which the size is $O(\mid \overline{B} \mid \times n)$. Hence our algorithm outperforms theirs in space and time. Further, an important practical advantage of our algorithm, unlike [12], is that our algorithm reuses the original algorithm for model checking with global fairness with slight changes.

This work is also related to our previous work on combining weak/strong fairness with counter abstraction [28]. The idea is to show that model checking with process-level weak/strong fairness is feasible even if process identities are abstracted away. It is achieved by systematically keeping track of the local states from which actions are enabled/executed within any infinite loop of the abstract state space. Different from the above work, our approach works with global fairness and we show that global fairness and symmetry reduction can be integrated in a relatively easy way. Additionally, this work is remotely related to work on combining state reduction techniques and fairness, evidenced in [20, 30, 4]. Our work explores one kind of state reduction and shows that it works with global fairness.

Closest to our work on combination of partial order reduction and fairness is that of Peled [22, 21] and Brim et al. [6]. Peled proposes equivalence robust property to guarantee that all behaviors under certain fair assumption remain in the reduced state space. However, since only weak fair is equivalence robust, stronger fair assumption need to add more dependency relations to achieve equivalence robustness. In his later work [21], he presents on-the-fly reduction algorithms with/without fairness assumptions. The authors in [6] define two partial order reduction strategies, safe and aggressive reduction, and demonstrate that each weakly fair behavior is preserved in safe reduction but not in aggressive one, while not all strongly fair behaviors are preserved in either reductions.

9 Conclusion and Future Work

The contribution of this work is threefold. First, we show that unlike weak/strong fairness, global fairness can be combined with symmetry reduction. Next, we present a practical fair model checking algorithm with symmetry reduction. Lastly, we prove that classic partial order reduction can not guarantee to preserve properties with global fairness. An interesting line of future work is to identify sufficient condition that allows combination of fairness and abstraction in general. In the current implementation, symmetry relationships are assumed to be known or easily detected. In the future, we plan to develop symmetry detection technique (as well as reduction techniques) for hierarchical complex systems.

References

1. http://www.comp.nus.edu.sg/%7Epat/fm/sym.
2. D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang. Self-stabilizing Population Protocols. In *OPODIS*, LNCS, pages 103–117, 2005.
3. D. Bosnacki, D. Dams, and L. Holenderski. Symmetric Spin. In *SPIN*, pages 1–19, 2000.
4. D. Bosnacki, N. Ioustinova, and N. Sidorova. Using Fairness to Make Abstractions Work. In *SPIN*, LNCS, pages 198–215. Springer, 2004.

5. D. Bošnački. A Light-weight Algorithm for Model Checking with Symmetry Reduction and Weak Fairness. In *SPIN*, pages 89–103, 2003.
6. L. Brim, I. Cerná, P. Moravec, and J. Simsa. On Combining Partial Order Reduction with Fairness Assumptions. In *FMICS/PDMC*, LNCS, pages 84–99, 2006.
7. S. Chaki, E. M. Clarke, J. Ouaknine, N. Sharygina, and N. Sinha. State/Event-Based Software Model Checking. In *IFM*, LNCS, pages 128–147. Springer, 2004.
8. E. M. Clarke, T. Filkorn, and S. Jha. Exploiting Symmetry In Temporal Logic Model Checking. In *CAV*, LNCS, pages 450–462. Springer, 1993.
9. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 2000.
10. G. Delzanno. Automatic Verification of Parameterized Cache Coherence Protocols. In *CAV*, LNCS, pages 53–68. Springer, 2000.
11. E. A. Emerson and A. P. Sistla. Symmetry and Model Checking. *Formal Methods in System Design*, 9(1-2):105–131, 1996.
12. E. A. Emerson and A. P. Sistla. Utilizing Symmetry when Model-Checking under Fairness Assumptions: An Automata-Theoretic Approach. *ACM Transactions on Programming Languages and Systems*, 19(4):617–638, 1997.
13. E. Allen Emerson, Somesh Jha, and Doron Peled. Combining partial order and symmetry reductions. In *TACAS*, pages 19–34, London, UK, 1997.
14. M. J. Fischer and H. Jiang. Self-stabilizing Leader Election in Networks of Finite-state Anonymous Agents. In *OPODIS*, LNCS. Springer, 2006.
15. V. Gyuris and A. P. Sistla. On-the-Fly Model Checking Under Fairness That Exploits Symmetry. In *CAV*, LNCS, pages 232–243. Springer, 1997.
16. H. Jiang. *Distributed Systems of Simple Interacting Agents*. PhD thesis, Yale Uni., 2007.
17. L. Lamport. Proving the Correctness of Multiprocess Programs. *IEEE Transactions on Software Engineering*, 3(2):125–143, 1977.
18. L. Lamport. Fairness and Hyperfairness. *Distributed Computing*, 13(4):239–245, 2000.
19. Y. Liu, J. Pang, J. Sun, and J. H. Zhao. Verification of Population Ring Protocols in PAT. In *TASE*, pages 81–89. IEEE, 2009.
20. U. Nitsche and P. Wolper. Relative Liveness and Behavior Abstraction (Extended Abstract). In *PODC*, pages 45–52. ACM, 1997.
21. D. Peled. Combining Partial Order Reductions with On-the-fly Model-Checking. In *CAV*, pages 377–390, 1994.
22. Doron Peled. All from one, one for all: on model checking using representatives. In *CAV*, pages 409–423, 1993.
23. A. Pnueli. On the Extremely Fair Treatment of Probabilistic Algorithms. In *STOC*, pages 278–290, New York, NY, USA, 1983. ACM.
24. A. Pnueli and Y. Sa'ar. All You Need Is Compassion. In *VMCAI*, pages 233–247, 2008.
25. A. Pnueli, J. Xu, and L. D. Zuck. Liveness with (0, 1, infinity)-Counter Abstraction. In *CAV*, LNCS, pages 107–122. Springer, 2002.
26. F. Pong and M. Dubois. A New Approach for the Verification of Cache Coherence Protocols. *IEEE Transactions on Parallel and Distributed Systems*, 6(8):773–787, 1995.
27. J. Sun, Y. Liu, J. S. Dong, and J. Pang. PAT: Towards Flexible Verification under Fairness. In *CAV*, LNCS. Springer, 2009.
28. J. Sun, Y. Liu, A. Roychoudhury, S. S. Liu, and J. S. Dong. Fair Model Checking with Process Counter Abstraction. In *FM*, LNCS, pages 123–139. Springer, 2009.
29. R. Tarjan. Depth-first Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 2:146–160, 1972.
30. U. Ultes-Nitsche and S. St. James. Improved Verification of Linear-time Properties within Fairness: Weakly Continuation-closed Behaviour Abstractions Computed from Trace Reductions. *Software Testing, Verification & Reliability*, 13(4):241–255, 2003.