Model and Program Repair via Group Actions and Structure Unwinding

PAUL C. ATTIE, School of Computer and Cyber Sciences, Augusta University, Augusta, Georgia, USA
WILLIAM L. COCKE, School of Computer and Cyber Sciences, Augusta University, Augusta, Georgia, USA and Language Technologies Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA

Given a program P, one can construct a Kripke structure $\mathcal{M}$. Model checking verifies that P satisfies a behavioral property given by a temporal logic formula φ by checking that $\mathcal{M}$ models φ. However, $\mathcal{M}$ can be exponentially large in P. The action of a symmetry group G on $\mathcal{M}$ and φ can produce a smaller structure $\overline{\mathcal{M}}$. When $\mathcal{M}$ does not satisfy φ, one can look for a substructure that satisfies φ. We call this *substructure repair*.

We show that repairs of $\overline{\mathcal{M}}$ lift to repairs of $\mathcal{M}$, i.e., we can repair a concurrent program by repairing the smaller structure $\overline{\mathcal{M}}$ and symmetrizing the resulting program. The substructures of $\overline{\mathcal{M}}$ map to substructures of $\mathcal{M}$ preserved by G. We present relative completeness results, which give conditions under which the existence of a repair of $\mathcal{M}$ implies the existence of a repair of $\overline{\mathcal{M}}$.

In cases where there is no repair of a Kripke structure $\mathcal{M}$ w.r.t. a formula, we show that there are instances where it is possible to "unwind" $\mathcal{M}$ to generate a structure $\mathcal{M}'$ that is strongly bisimilar to $\mathcal{M}$ and for which a repair exists. This leads to a natural semantic notion, *repairability*, which is not preserved by strong bisimulation. We illustrate the combined use of symmetry reduction and unwinding to effect a repair.

Finally, we provide closed-form results for the reductions in number of states in the Kripke structure that can be achieved by symmetry reduction.

CCS Concepts: • **Theory of computation → Verification by model checking**; *Logic and verification*;

Additional Key Words and Phrases: Model checking, symmetry reduction, model repair

ACM Reference format:
Paul C. Attie and William L. Cocke. 2025. Model and Program Repair via Group Actions and Structure Unwinding. *ACM Trans. Comput. Logic* 26, 2, Article 11 (April 2025), 44 pages.
https://doi.org/10.1145/3719008

1 Introduction

To model check a concurrent program $P = P_1 \parallel \cdots \parallel P_n$ one first constructs the Kripke structure $\mathcal{M}$ that is generated by all the executions of P. The model checking problem for P with respect to a temporal logic formula φ is to verify that $\mathcal{M}$ satisfies φ, i.e., $M \models \varphi$, where the satisfaction relation $\models$

W. Cocke was supported by a GEN Omar N. Bradley Officer Research Fellowship in Mathematics.
Authors' Contact Information: Paul C. Attie, School of Computer and Cyber Sciences, Augusta University, Augusta, Georgia, USA; e-mail: pattie@augusta.edu; William L. Cocke (corresponding author), School of Computer and Cyber Sciences, Augusta University, Augusta, Georgia, USA and Language Technologies Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA; e-mail: wcocke@augusta.edu

ACM 1557-945X/2025/4-ART11
https://doi.org/10.1145/3719008

is given by the semantics of temporal logic [13, 18]. A major obstacle to model checking a concurrent program via its Kripke structure is *state explosion*: in general, the size of $\mathcal{M}$ is exponential in the number of processes n.

The use of *symmetry reduction* [15, 20, 24, 27] to ameliorate state explosion can yield a significant reduction in the complexity of model checking $\mathcal{M} \models \varphi$ when both $\mathcal{M}$ and φ have a high degree of symmetry in the process index set $\{1, \ldots, n\}$. We capture the available symmetry via the group G of automorphisms of both $\mathcal{M}$ and φ. The quotient structure $\overline{\mathcal{M}} = \mathcal{M}/G$ of $\mathcal{M}$ by G often has significantly fewer states than $\mathcal{M}$. Since $\overline{\mathcal{M}}$ can be computed directly from G and the concurrent program P, we avoid the expensive computation of the large structure $\mathcal{M}$. Model checking $\overline{\mathcal{M}} \models \varphi$ is linear in the size of $\overline{\mathcal{M}}$ [13], so this provides significant savings if $\overline{\mathcal{M}}$ is small, i.e., if G is large.

If $\mathcal{M} \not\models \varphi$, then we can *repair* $\mathcal{M}$, i.e., we search for a model $\mathcal{N}$ related to $\mathcal{M}$ such that $\mathcal{N} \models \varphi$. In this article we focus on *substructure repair*: we require $\mathcal{N}$ to be a substructure of $\mathcal{M}$. The key idea behind substructure repair is to remove execution paths which violate required properties, e.g., paths that lead to a violation of φ.

Sometimes a subtractive repair does not exist as there is no substructure $\mathcal{N}$ of $\mathcal{M}$ such that $\mathcal{N} \models \varphi$. However, it is possible, in some cases, to find a structure $\mathcal{M}'$ that is bisimilar to $\mathcal{M}$, and then to perform a subtractive repair on $\mathcal{M}'$. In practice, we want $\mathcal{M}'$ to be related to $\mathcal{M}$ and to contain a larger number of states. Since $\mathcal{M}'$ has more substructures than $\mathcal{M}$, this can produce a repair where none existed before. We investigate one method for enlarging $\mathcal{M}$ which we call vertical unwinding. In single unwinding, a copy of $\mathcal{M}$ is created, and transitions in $\mathcal{M}$ that would cause a repeated state along a path (i.e., a cycle) are redirected to the copy instead. Likewise, transitions in the copy are redirected to the original $\mathcal{M}$. The generalization to n-unwinding is straightforward.

1.1 Our Contributions

This article is an extension of the paper "Model and Program Repair via Group Action" [2] and contains full proofs of all theorems in that paper. In addition, this article contains results investigating repairability (i.e., the existence of a substructure satisfying φ) as it relates to bisimilarity. We note that bisimilar structures differ in their repairability with respect to different formulae. Using so-called vertical unwindings, we show that structures for which the propositional labeling function is injective always have a bisimilar structure with which they are not equi-repairable. Thus, unwinding can be used to generate repairs that do not exist in the original (not unwound) structure. We also show how vertical unwindings and symmetry reduction can be used in tandem to repair structures. The article concludes with asymptotic bounds regarding the reduction in state space via symmetry reduction.

We first present a theory of substructures of Kripke structures (Section 4). Then, after briefly recalling group actions, we show how one can use a group to obtain a quotient $\overline{\mathcal{M}}$ of a Kripke structure $\mathcal{M}$ (Section 5). Using the theory of substructures, we establish an evaluation-preserving correspondence between certain substructures of the original Kripke structure $\mathcal{M}$ and the substructures of the quotient structure $\overline{\mathcal{M}} = \mathcal{M}/G$: Theorem 5.16 below. This correspondence is a functorial form of bisimilarity between a certain lattice of substructures of $\mathcal{M}$ and the lattice of substructures of $\overline{\mathcal{M}}$. Hence for a given temporal logic formula φ, any substructure repair of $\overline{\mathcal{M}}$ with respect to φ can be *lifted* to a substructure repair of $\mathcal{M}$ with respect to φ: Theorem 5.22 (Repair Correspondence) below. This correspondence of Kripke substructure lattices is of independent mathematical interest as an example of a monotone Galois connection. We use the term *symmetry-reduced substructure repair* to refer to the repair of $\mathcal{M}$ by repairing $\overline{\mathcal{M}}$ and then lifting the repair to $\mathcal{M}$.

We build on our theory to extend group-theoretic model checking to *concurrent program repair*: given a concurrent program P that may not satisfy φ, modify P to produce a program that does

satisfy φ (Section 6). Given P, φ, and a group G that acts on both P and φ, our method directly computes the quotient $\overline{M} = M/G$ (following [24]), then repairs $\overline{M}$, using the algorithm of [3], and finally extracts a correct program from the repaired structure. The key difficulty is dealing with *irregular transitions* in $\overline{M}$: those transitions in which more than one process changes state.

The Repair Correspondence Theorem (Theorem 5.22) shows that symmetry-reduced substructure repair is *sound*: a repair $\overline{N}$ of $\overline{M}$ can always be lifted to a repair N of M. However, symmetry-reduced repair is not *complete*: a repair N of M may exist while a repair of $\overline{M}$ does not. We give an example structure M that has a repair while its quotient $\overline{M}$ does not. That is, we show that the repair of a symmetry-reduced quotient $\overline{M}$ is not complete with respect to the repair of the original structure M. We identify a fragment of the temporal logic **Computation Tree Logic (CTL)** for which symmetry-reduced substructure repair is complete (Section 8).

A disadvantage of substructure repair is that sometimes a structure M does not have a substructure repair, but a repair can be produced by "adding" states and transitions. Toward this end, we introduce the notion of *vertical unwinding* and show that it can be used to generate repairs even if the original structure is not repairable. That is, we show how a structure and formula that are not repairable can be repaired if we first unwind the original structure. We show via an example how symmetry reduction and unwinding can be combined to produce a repair without state explosion, when the original exponentially large structure M has no subtractive repair (Section 9).

We introduce the notions of *repairability* and *equi-reparable*: M is repairable w.r.t. φ iff there exists a substructure of M that satisfies φ. Two structures are equi-reparable iff they are repairable w.r.t. the same set of formulae. We show that equi-reparability and strong bisimulation are distinct relations over Kripke structures: neither implies the other. Historically, strongly bisimilar structures were taken to be essentially equivalent. Our result is therefore interesting as equi-reparability is the first (to our knowledge) semantic relation over Kripke structures which is not implied by strong bisimulation. We also show that, subject to very weak assumptions, the unwinding of a Kripke structure is not equi-reparable to the original (Section 10).

Finally, we give some closed-form analytic results for the reduction in size from M to its quotient $\overline{M} = M/(G, \vartheta_G)$. We consider a structure M that is generated by a concurrent program $P = P_1 \| \cdots \| P_n$. We first deal with the case where G is the full permutation group S_n over $\{1...n\}$. First, we consider the case where the processes P_i have a constant number ℓ of local states, while the number n of processes varies. This is the setting of PMCP: the parametrized model checking problem [6, 7]. In this case, the number of states of $\overline{M}$ is in $O(n^{\ell-1})$. Compare with the number of states in M which is ℓ^n. Second, we consider the case where each process P_i has a number ℓ of local states that is logarithmic in the number n of processes overall, i.e., $\ell = \Theta(\log n)$. This holds for algorithms where each process must store a constant number of indices of other processes, e.g., leader election in a ring [36, Chapter 3], distributed graph search and spanning tree construction [36, Chapter 15], and diffusing computations [16]. In this case, the number of states of $\overline{M}$ is bounded from both below and above by $n^{d \log n}$, with different constants d in the lower and upper bounds. Third, we consider the case where each process P_i has a number ℓ of local states that is linear in the number n of processes overall, i.e., $\ell = \Theta(n)$. This applies, for example, in a broadcast model where each process receives (and records locally) information from every other process [29, 38]. In this case, the number of states of $\overline{M}$ is bounded from both below and above by d^n, with different constants d in the lower and upper bounds. Hence, the size of $\overline{M}$ remains exponential in the number of processes n. Compare with the number of states in M which is ℓ^n, i.e., $O((cn)^n)$ for some constant c. When G is a proper subgroup of S_n, we partition the processes into subsets with full symmetry between the processes in each subset. The above results then apply to each

subset in isolation, and the size of $\overline{\mathcal{M}}$ is then the product of the "size contributions" of each subset (Section 11).

The rest of the article proceeds as follows. Section 2 presents related work. Section 3 gives some necessary technical preliminaries. Section 4 presents the central notion of a substructure of a Kripke structure. In Section 5 we define the quotient $\overline{\mathcal{M}}$ of $\mathcal{M}$ and demonstrate the repair correspondence between $\mathcal{M}$ and $\overline{\mathcal{M}}$. We extend our results to the repair of concurrent programs in Section 6. Section 7 presents some examples of the repair of a Kripke structure via the repair of its quotient. We address the issue of completeness in Section 8. Section 9 introduces the notion of *vertical unwinding* to effect the repair of Kripke structures, and given examples of its use. Section 10 introduces the notion of repairability and the relation over Kripke structures of equi-reparability. Section 11 gives our closed-form quantitative results concerning the reduction in size from $\mathcal{M}$ to $\overline{\mathcal{M}}$. Section 12 presents our conclusions.

2 Related Work

Our work combines model/program repair [10, 31, 35, 40] and symmetry reductions via group actions [12, 15, 22, 24–28]. Le Goues et al. [31] provide a modern introduction to program repair; although their results generally relate to program repair based on the textual representation of the program. Our approach repairs a Kripke structure with respect to a CTL formula and uses that to repair the corresponding program.

2.1 CTL Repair

Buccafuri et al. [10] posed the repair problem for CTL and solved it using abductive reasoning to generate repair suggestions that are verified by model checking. Jobstmann et al. [35] and Staber et al. [40] used game-based repair methods for programs and circuits, although their method is complete for invariants only.

Chatzieleftheriou et al. [11] repair abstract structures, using Kripke modal transition systems and three-valued CTL semantics. Von Essen and Jobstmann [42] present a game-based repair method which attempts to keep the repaired program close to the original faulty program, by also specifying a set of traces that the repair must leave intact.

The work of Attie et al. [3] establishes that repair by abstraction can avoid state explosion. However, repairs of abstracted structures do not always lift to repairs of the original structure. Within networks, Namjoshi and Trefler [37] have shown that a combination of abstraction and group actions can be used to produce smaller structures.

2.2 Group-Theoretic Model Checking

Group-theoretic approaches to symmetry reduction in model checking began in 1995 [12, 15, 20, 22, 24–28]. The general approach is to compute the quotient $\mathcal{M}/G$ and model check $\mathcal{M}/G$, instead of the original (much larger) structure $\mathcal{M}$. The group-theoretic approach to model checking works because $\mathcal{M}$ and $\mathcal{M}/G$ are bisimilar with respect to certain formulae.

A requirement for group-theoretic model checking or repair is calculating the group of symmetries in question. We will see that larger groups of symmetries result in smaller quotient models. Clarke et al. [12] showed that calculating the orbit of a group action, a part of model checking via symmetry, is at least as difficult as graph isomorphism. However, in many practical cases, concurrent programs have a natural symmetry by swapping certain process indices. Hence many concurrent programs have a small known symmetry group in advance. Donaldson and Miller [17] showed that there is a process to build a larger symmetry group for a program from a smaller symmetry group.

A related approach is the use of structural methods to express symmetric designs, e.g., parameterized systems, where processes are all instances of a common template (possibly with a distinguished controller process) [1, 14, 30], and rings of processes, where all communication is between a process and its neighbors in the ring [14, 21, 23].

3 CTL, Kripke Structures, and Concurrent Programs

3.1 Kripke Structures

We use Kripke structures to model concurrency.

Definition 3.1 (Kripke Structure). A *Kripke structure* $\mathcal{M}$ is a tuple $(S, S_0, T, L, AP, V, \mathcal{SH})$ where:

(1) S is a finite set of states,
(2) $S_0 \subseteq S$ is a set of initial states
(3) $T \subseteq (S \times S)$ is a transition relation
(4) AP is a finite set of atomic propositions
(5) $L : S \to 2^{AP}$ is a labeling function that maps each state $s \in S$ to the subset of atomic propositions that hold in state s
(6) $\mathcal{SH}$ is a finite set of shared variables
(7) V maps each state to a shared variable valuation, i.e., an assignment to each $x \in \mathcal{SH}$ of a value in the domain of x

We require that $\mathcal{M}$ be total: $\forall s \in S, \exists t \in S : (s, t) \in T$, and that $S \neq \emptyset$ implies $S_0 \neq \emptyset$. We admit the empty Kripke structure, i.e., $S = \emptyset$, due to mathematical necessity.

When referring to the constituents of $\mathcal{M} = (S, S_0, T, L, AP, V, \mathcal{SH})$, we write $\mathcal{M}_S$, $\mathcal{M}_{S_0}$, $\mathcal{M}_T$, $\mathcal{M}_L$, $\mathcal{M}_{AP}$, $\mathcal{M}_V$, and $\mathcal{M}_{\mathcal{SH}}$, respectively. State t is a *successor* of state s in M iff $(s, t) \in T$. We write $s \to t$ in this case. A path π in $\mathcal{M}$ is a (finite or infinite) sequence of states, $\pi = s_0, s_1, \ldots$, such that $\forall i \geq 0 : (s_i, s_{i+1}) \in T$. A *fullpath* is an infinite path.

3.1.1 Multiprocess Kripke Structures. To model the behavior of a concurrent program $P = P_1 \parallel \cdots \parallel P_n$, we define a special type of Kripke structure: a *multiprocess Kripke structure* is a Kripke structure with additional structure to reflect the semantics of concurrent program execution in a shared variables model. In particular, the set of atomic propositions AP is partitioned into disjoint subsets $AP_1, \ldots, AP_n$, where AP_i is the set of atomic propositions "owned" by process P_i: propositions in AP_i can only be changed by P_i, but can be read by other processes. Likewise, the set of transitions T is partitioned into disjoint subsets $T_1, \ldots, T_n$, where T_i is the set of transitions generated by the execution of process P_i. We write $s \xrightarrow{i} t$ for a transition in T_i, i.e., transitions are labeled by the index of the executing process. A state s has the form $(s_1, \ldots, s_n, v_1, \ldots, v_m)$ where s_i is the local state of P_i, and $v_1, \ldots, v_m$ are the values, respectively, of the shared variables $\mathcal{SH} = \{x_1, \ldots, x_m\}$, in state s. Each local state s_i is labeled by the subset of AP_i whose propositions are true in s_i, so the truth value of $p \in AP_i$ in global state $(s_1, \ldots, s_n, v_1, \ldots, v_m)$ is given by its value in local state s_i. We therefore redefine the Kripke structure labeling function L so that its domain is $\bigcup_i S_i$, where S_i is the set of local states of P_i, and we require that $L(s_i) \subseteq AP_i$ for all $i \in \{1...n\}$, i.e., L labels local states of P_i with atomic propositions from AP_i only. We also require, without loss of generality, that different processes have the same sets of atomic propositions, modulo indices, i.e., $AP_j = \{Q_j \mid Q_i \in AP_i\}$.[1] We also impose on multiprocess Kripke structures a *unique labeling condition* as follows.

[1]Any "extra" atomic propositions that arise and which a process does not use can simply be initialized to an arbitrary truth value and not referenced in the processes' code.

Definition 3.2 (Unique Labeling Condition). Let $\mathcal{M}$ be a multiprocess Kripke structure and let s, t be distinct states of $\mathcal{M}$, i.e., $s, t \in \mathcal{M}_S$, $s \neq t$. Then $\mathcal{M}_L(s) \neq \mathcal{M}_L(t) \vee \mathcal{M}_V(s) \neq \mathcal{M}_V(t)$.

That is, different states in a multiprocess Kripke structure are distinguished by either an atomic proposition or a shared variable. This condition is needed so that the operation of extracting a concurrent program P from a Kripke structure $\mathcal{M}$ is faithfull: execution of the program generates the same Kripke structure $\mathcal{M}$. See in particular Definition 6.3 in Section 6 below. Unique labeling is not required for any of our symmetry reduction, unwinding, and structure repair results to hold, only for correct extraction of concurrent programs from Kripke structures, i.e., it is only needed for the results of Section 6.

Definition 3.3 (State Projection Operators). For state $s = (s_1, \ldots, s_n, v_1, \ldots, v_m)$, define $s\restriction i = s_i$, $s\restriction \mathcal{S}H = (v_1, \ldots, v_m)$, and $s\downarrow i = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n)$.

For every transition $s \xrightarrow{i} t$, a multiprocess Kripke structure must satisfy $s\downarrow i = t\downarrow i$, i.e., transitions by P_i do not change the atomic propositions of the other processes P_j, $j \neq i$.

3.2 CTL

CTL is a propositional branching time temporal logic used to model the possible computational branches taken by a system [18, 19]. Given a set AP of atomic propositions we define CTL by the following grammar:

$$\varphi ::= \mathsf{true} \mid \mathsf{false} \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{AX}\varphi \mid \mathsf{EX}\varphi \mid \mathsf{A}[\varphi\,\mathsf{R}\,\varphi] \mid \mathsf{E}[\varphi\,\mathsf{R}\,\varphi]$$

where $p \in AP$, and true, false are constant propositions whose interpretation is the semantic truth values tt, ff respectively. A propositional formula is a CTL formula not containing the AX, EX, AR, ER temporal modalities.

The notation $[\varphi\,\mathsf{R}\,\psi]$ (φ "releases" ψ) means that ψ must hold up to and including the first state where φ holds, and forever if φ never holds. We introduce the abbreviations $\mathsf{A}[\varphi\,\mathsf{U}\,\psi]$ for $\neg\mathsf{E}[\neg\varphi\,\mathsf{R}\,\neg\psi]$, $\mathsf{E}[\varphi\,\mathsf{U}\,\psi]$ for $\neg\mathsf{A}[\neg\varphi\,\mathsf{R}\,\neg\psi]$, $\mathsf{AF}\varphi$ for $\mathsf{A}[\mathsf{true}\,\mathsf{U}\,\varphi]$, $\mathsf{EF}\varphi$ for $\mathsf{E}[\mathsf{true}\,\mathsf{U}\,\varphi]$, $\mathsf{AG}\varphi$ for $\mathsf{A}[\mathsf{false}\,\mathsf{R}\,\varphi]$, $\mathsf{EG}\varphi$ for $\mathsf{E}[\mathsf{false}\,\mathsf{R}\,\varphi]$. We use tt, ff for the (semantic) truth values of true, false, respectively; true, false are atomic propositions whose interpretation is always tt, ff, respectively.

The semantics of CTL is defined with respect to a Kripke structure as follows. A CTL formula φ is evaluated (i.e., is true or false) in a state s of a Kripke structure $\mathcal{M}$ [19]. We write $\mathcal{M}, s \models \varphi$ when φ is true in state s of structure $\mathcal{M}$, and write $\mathcal{M} \models \varphi$ to abbreviate $\forall s_0 \in S_0 : \mathcal{M}, s_0 \models \varphi$, i.e., φ holds in all initial states of $\mathcal{M}$. The formal definition of $\models$ proceeds by induction on the structure of CTL formulae:

Definition 3.4 (CTL Semantics). We write $\mathcal{M}, s \models \varphi$ to mean φ is true in state s of structure M, and $\mathcal{M}, s \not\models \varphi$ to mean that φ is false in s. The standard inductive definition follows:

(1) $\mathcal{M}, s \models \mathsf{true}$ and $\mathcal{M}, s \not\models \mathsf{false}$
(2) $\mathcal{M}, s \models p$ iff $p \in L(s)$ where atomic proposition $p \in AP$
(3) $\mathcal{M}, s \models \neg\varphi$ iff $\mathcal{M}, s \not\models \varphi$
(4) $M, s \models \varphi \wedge \psi$ iff $\mathcal{M}, s \models \varphi$ and $M, s \models \psi$
(5) $M, s \models \varphi \vee \psi$ iff $\mathcal{M}, s \models \varphi$ or $M, s \models \psi$ or both
(6) $\mathcal{M}, s \models \mathsf{AX}\varphi$ iff for all t such that $(s, t) \in T : (\mathcal{M}, t) \models \varphi$
(7) $M, s \models \mathsf{EX}\varphi$ iff there exists t such that $(s, t) \in T$ and $(\mathcal{M}, t) \models \varphi$
(8) $\mathcal{M}, s \models \mathsf{A}[\varphi\,\mathsf{R}\,\psi]$ iff for all fullpaths $\pi = s_0, s_1, \ldots$ starting from $s = s_0$:
$\forall k \geq 0 : (\forall j < k : \mathcal{M}, s_j \not\models \varphi)$ implies $\mathcal{M}, s_k \models \psi$
(9) $\mathcal{M}, s \models \mathsf{E}[\varphi\,\mathsf{R}\,\psi]$ iff for some fullpaths $\pi = s_0, s_1, \ldots$ starting from $s = s_0$:
$\forall k \geq 0 : (\forall j < k : \mathcal{M}, s_j \not\models \varphi)$ implies $\mathcal{M}, s_k \models \psi$

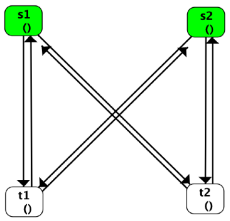

Fig. 1. The box Kripke structure.

Example 1 (Example Box) The "Box" Kripke structure in Figure 1 has four states and transitions as shown. Its set of atomic propositions is empty, and so all states have empty labels, as indicated by "()." There is a natural group acting on this Kripke structure, i.e., the group generated by the action which exchanges the state **s1** with **s2**, and the state **t1** with **t2**.

3.3 Concurrent Programs

A concurrent program $P = P_1 \| \cdots \| P_n$ consists of n sequential processes executing in parallel. Each process P_i is a set of *i-actions* $(s_i, B \longrightarrow A, t_i)$, where s_i, t_i are local states of P_i, B is a guard (a predicate on the global state), and A is a multiple assignment statement that updates (some of) the shared variables $\mathcal{SH}$. We say *action* when we ignore the process id. We assume a given set S_0 of initial states.

Let $s(B)$ denote the value of guard B in global state s, and, for $s = (s_1, \ldots, s_n, v_1, \ldots, v_m)$, let $\{|s\restriction\mathcal{SH}|\} \triangleq x_1 = v_1 \wedge \cdots \wedge x_m = v_m$, i.e., $\{|s\restriction\mathcal{SH}|\}$ converts the shared variable values in state s into an assertion.

Definition 3.5 (Semantics of Concurrent Programs). The program $P_1 \| \cdots \| P_n$ generates a transition $s \xrightarrow{i} t$ iff (1) $s{\downarrow}i = t{\downarrow}i$, i.e., the transition changes only atomic propositions in AP_i, and (2) P_i contains an i-action $(s_i, B \longrightarrow A, t_i)$ such that:

(1) $s\restriction i = s_i$, i.e., s_i is the local state of P_i in global state s,
(2) $t\restriction i = t_i$, i.e., t_i is the local state of P_i in global state t,
(3) $s(B) = true$, i.e., the guard B holds in global state s, and
(4) $\{|s\restriction\mathcal{SH}|\}\, A\, \{|t\restriction\mathcal{SH}|\}$, which is the usual Hoare triple notation indicating that execution of the multiple assignment A changes the shared variable values from those in s to those in t.

Given a set S_0 of initial states for P, the global state-transition graph of P is the multiprocess Kripke structure that is the closure of this "transition generation" operation, starting in the initial state set S_0. The transitions are $T = T_1 \cup \ldots \cup T_n$, where T_i is the set of transitions generated by process P_i as described above. We write $GSTD(P)$ for the global state-transition graph of P.

We consider that a concurrent program P satisfies a CTL formula φ iff its global state-transition graph does, i.e., $P \models \varphi \triangleq GSTD(P) \models \varphi$.

4 Substructures

The theory of substructures presented below is motivated by the concept of a substructure repair of a Kripke structure $\mathcal{M}$ with respect to a formula φ, i.e., a substructure $\mathcal{N}$ of $\mathcal{M}$ such that $\mathcal{N} \models \varphi$.

Definition 4.1 (Substructure, $\leq$). Given Kripke structures $\mathcal{M}$ and $\mathcal{N}$, we say that $\mathcal{N}$ is a *substructure* of $\mathcal{M}$, denoted $\mathcal{N} \leq \mathcal{M}$, iff the following all hold:

(1) $\mathcal{N}_S \subseteq \mathcal{M}_S$.
(2) $\mathcal{N}_{S_0} = \mathcal{M}_{S_0} \cap \mathcal{N}_S$.
(3) $\mathcal{N}_T \subseteq \mathcal{M}_T$.
(4) $\mathcal{N}_{AP} = \mathcal{M}_{AP}$.
(5) $\mathcal{N}_L = \mathcal{M}_L \restriction \mathcal{N}_S$ (where $\restriction$ denotes domain restriction).
(6) $\mathcal{N}_{SH} = \mathcal{M}_{SH}$
(7) $\mathcal{N}_V = \mathcal{M}_V \restriction \mathcal{N}_S$ (where $\restriction$ denotes domain restriction).
(8) For all $s \in \mathcal{N}_S$ there is a $t \in \mathcal{N}_S$ such that $(s,t) \in \mathcal{N}_T$, i.e., $\mathcal{N}$ is total.

For mathematical necessity in what follows, we allow for the "empty" substructure. We do not, however, accept an empty substructure as a valid repair. It is immediate that $\leq$ is a reflexive partial order. Lemmas 4.2 and 4.3 below imply that the substructures of $\mathcal{M}$ can be regarded as a lattice, with join and meet operations as follows.

Lemma 4.2. *Let $\mathcal{M}$ be a Kripke structure and suppose that $\mathcal{N}$ and $\mathcal{N}'$ are substructures of $\mathcal{M}$. Then*

$$\mathcal{N} \vee \mathcal{N}' = (\mathcal{N}_S \cup \mathcal{N}'_S,\ \mathcal{N}_{S_0} \cup \mathcal{N}'_{S_0},\ \mathcal{N}_T \cup \mathcal{N}'_T,\ \mathcal{M}_{AP},\ \mathcal{M}_L \restriction (\mathcal{N}_S \cup \mathcal{N}'_S),\ \mathcal{M}_{SH},\ \mathcal{M}_V \restriction (\mathcal{N}_S \cup \mathcal{N}'_S))$$

is the smallest substructure of $\mathcal{M}$ containing both $\mathcal{N}$ and $\mathcal{N}'$.

Given a non-empty finite set $X = \{X_0, X_1, \ldots, X_n\}$ of substructures of $\mathcal{M}$, we define the structure $\bigvee X = X_0 \vee X_1 \vee \cdots \vee X_n$.

Lemma 4.3. *Let $\mathcal{M}$ be a Kripke structure and suppose that $\mathcal{N}$ and $\mathcal{N}'$ are substructures of $\mathcal{M}$. Then there exists a largest substructure of $\mathcal{M}$ contained in both $\mathcal{N}$ and $\mathcal{N}'$.*

Definition 4.4 (Join, Meet of Substructures). Let $\mathcal{N}$ and $\mathcal{N}'$ be two substructures of $\mathcal{M}$. The *join* of $\mathcal{N}$ and $\mathcal{N}'$, written $\mathcal{N} \vee \mathcal{N}'$, is the smallest substructure of $\mathcal{M}$ containing both $\mathcal{N}$ and $\mathcal{N}'$. The *meet* of $\mathcal{N}$ and $\mathcal{N}'$, written $\mathcal{N} \wedge \mathcal{N}'$, is the largest substructure of $\mathcal{M}$ contained in both $\mathcal{N}$ and $\mathcal{N}'$.

The join $\mathcal{N} \vee \mathcal{N}'$ has a simple description as given in Lemma 4.2. However, the meet $\mathcal{N} \wedge \mathcal{N}'$, while well-defined, does not have such a simple description. It is possible that for two substructures $\mathcal{N}$ and $\mathcal{N}'$ of a Kripke structure $\mathcal{M}$, there are no non-empty substructures contained in both $\mathcal{N}$ and $\mathcal{N}'$. Hence the largest substructure contained in both $\mathcal{N}$ and $\mathcal{N}'$ could be empty.

We can now define a lattice of substructures $\Lambda_{\mathcal{M}}$ for a given structure $\mathcal{M}$.

Definition 4.5 (Lattice of Substructures). Given a Kripke structure $\mathcal{M}$ the *lattice of substructures of* $\mathcal{M}$ is $\Lambda_{\mathcal{M}} = (\{\mathcal{N} : \mathcal{N} \text{ is a substructure of } \mathcal{M}\}, \leq)$ where the meet and join in $\Lambda_{\mathcal{M}}$ are as given in Definition 4.4, and $\leq$ is as given in Definition 4.1.

5 Quotient Structures

We capture the symmetry in a Kripke structure $\mathcal{M}$ with the notion of *state mapping*: a graph isomorphism on $\mathcal{M}$ which preserves initial states. State mappings also preserve paths since they are isomorphisms. We ignore for now the labeling function $\mathcal{M}_L$, i.e., which atomic propositions hold in which states, and concern ourselves only with the graph structure of $\mathcal{M}$. Since the atomic proposition labeling obviously affects the truth of CTL formulae in states of $\mathcal{M}$, it must be accounted for. We do this below using the notion of G-invariant CTL formula. Thus, we decompose the symmetry characterization of $\mathcal{M}$ into two separate concerns: the graph structure of $\mathcal{M}$, handled using state mapping, and the atomic proposition labeling of states of $\mathcal{M}$, handled using G-invariant CTL formulae.

A type of symmetry of particular interest is the symmetry of a multiprocess Kripke structure w.r.t. the process indices $1, \ldots, n$ of the corresponding concurrent program $P_1 \parallel \cdots \parallel P_n$, as we illustrate below. Our theory, however, applies to Kripke structures in general.

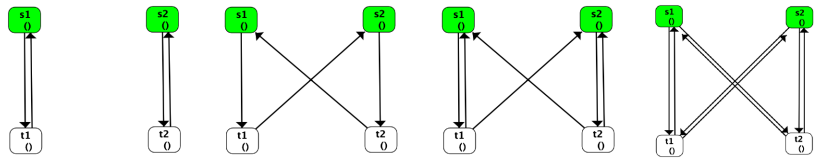

Fig. 2. Four G-closed substructures of Example Box. Where G is the group generated by the simultaneous swapping of indexes of both the $\mathbf{s_i}$ and the $\mathbf{t_i}$ ($i = 1, 2$). Note that each of the structures is a substructure of the substructure to the right. Looking ahead to Definition 5.13, only the entire structure (d) is G-maximal.

5.1 Groups Acting on Kripke Structures

Definition 5.1 (State Mapping). A state mapping of $\mathcal{M}$ is a graph isomorphism of the state-space of $\mathcal{M}$ such that its restriction to the initial states is also an isomorphism, i.e., takes initial states to initial states. Formally, for a Kripke structure $\mathcal{M}$, a *state mapping* of $\mathcal{M}$ is a bijection $f : \mathcal{M}_S \to \mathcal{M}_S$ such that:

$-f(\mathcal{M}_{S_0}) = \mathcal{M}_{S_0}$;
$-$For states $s, t \in \mathcal{M}_S$ we have that $(s, t) \in \mathcal{M}_T \iff (f(s), f(t)) \in \mathcal{M}_T$.

The set of all state mappings of $\mathcal{M}$ forms a group. This means that the composition of any two state mappings is another state mapping and for any state mapping f on $\mathcal{M}$ there is another state mapping g on $\mathcal{M}$ such that $f(g(s)) = s$ and $g(f(s)) = s$. We refer to the manuscripts by Issacs [33, 34], and Serre [39] for a more in-depth introduction to group theory.

Definition 5.2 (G-Closed). For a group G of state mappings of a Kripke structure $\mathcal{M}$, a substructure $\mathcal{N}$ of $\mathcal{M}$ is called *G-closed* if G is a group of state mappings of $\mathcal{N}$, i.e., for every $g \in G$ and $s \in \mathcal{N}_S$ we have $g(s) \in \mathcal{N}_S$.

Lemma 5.3. *Let $\mathcal{M}$ be a Kripke structure and let G be a group of state mappings of $\mathcal{M}$. Let $\mathcal{N}$, $\mathcal{N}'$ be two G-closed substructures of $\mathcal{M}$. Then $\mathcal{N} \vee \mathcal{N}'$ and $\mathcal{N} \wedge \mathcal{N}'$ are both G-closed.*

By Lemma 5.3, we see that the G-closed substructures of $\mathcal{M}$ form a sublattice of $\Lambda_{\mathcal{M}}$. This is a proper sublattice in that the meet and join operations are the same as those of $\Lambda_{\mathcal{M}}$.

Definition 5.4 (Lattice of G-Closed Substructures). Given a Kripke structure $\mathcal{M}$ and a group G of state mappings of $\mathcal{M}$, the poset of G-closed substructures of $\mathcal{M}$ forms a lattice. We call this the *lattice of G-closed substructures of* $\mathcal{M}$ and write it as $\Lambda_{\mathcal{M},G}$.

Example 5.5 (Example Box). Let $\mathcal{M}$ be Example Box, i.e., the Kripke structure presented in Figure 1. Let g be the map that simultaneously switches $\mathbf{s_1}$ and $\mathbf{s_2}$, and switches $\mathbf{t_1}$ and $\mathbf{t_2}$, i.e., $g(\mathbf{s_1}) = \mathbf{s_2}$, $g(\mathbf{s_2}) = \mathbf{s_1}$, $g(\mathbf{t_1}) = \mathbf{t_2}$, $g(\mathbf{t_2}) = \mathbf{t_1}$. Let G be the group consisting of g and the identity map on $\mathcal{M}_S$. We note that G is not the entire group of state mappings of $\mathcal{M}$. The structure $\mathcal{M}$ has 10 G-closed substructures, including the empty structure. We present some of these structures in Figure 2.

5.2 Constructing the Quotient Structures

Given a group G of state mappings of a structure $\mathcal{M}$, we want to construct a quotient structure $\mathcal{M}/G$. However, as noted, state mappings do not contain any information about $\mathcal{M}_L$ and $\mathcal{M}_V$. To remedy this situation, we need a function that assigns a representative to each orbit of G, where for $s \in \mathcal{M}_S$ the orbit of s is $\{g(s) : g \in G\}$.

Definition 5.6 (Representative Map). Let $\mathcal{M}$ be a Kripke structure and suppose that G is a group of state mappings of $\mathcal{M}$. A *representative map* of $\mathcal{M}$ with respect to G is a function $\vartheta_G : \mathcal{M}_S \to \mathcal{M}_S$ satisfying the following:

- For all $s, s' \in \mathcal{M}_S$, if there is some $g \in G$ such that $g(s) = s'$ then $\vartheta_G(s) = \vartheta_G(s')$. (respects orbits)
- For all $s, s' \in \mathcal{M}_S$, if there is no $g \in G$ such that $g(s) = s'$ then $\vartheta_G(s) \neq \vartheta_G(s')$. (separates orbits)
- For all $s \in \mathcal{M}_S$, we have that $\vartheta_G(\vartheta_G(s)) = \vartheta_G(s)$, i.e., each orbit has a stable representative. (idempotent)

We define $\vartheta_G(S) = \{\vartheta_G(s) \mid s \in S\}$.

Definition 5.7 (Quotient Structure). Given a Kripke structure $\mathcal{M}$, a group G of state mappings of $\mathcal{M}$, and a representative map ϑ_G of $\mathcal{M}$ with respect to G, we define the *quotient structure* $\overline{\mathcal{M}} = \mathcal{M}/(G, \vartheta_G)$ of $\mathcal{M}$ with respect to G and ϑ_G as follows, where we write $\overline{s}$, $\overline{t}$ for $\vartheta_G(s)$, $\vartheta_G(t)$, respectively:

- $\overline{\mathcal{M}}_S = \vartheta_G(\mathcal{M}_S)$, i.e., the states of $\overline{\mathcal{M}}$ are the image under ϑ_G of the states of $\mathcal{M}$.
- $\overline{\mathcal{M}}_T$ consists of all $(\overline{s}, \overline{t})$ such that there exist $s \in \mathcal{M}_S$ with $\vartheta_G(s) = \overline{s}$ and $t \in \mathcal{M}_S$ with $\vartheta_G(t) = \overline{t}$ such that $(s, t) \in \mathcal{M}_T$.
- $\overline{\mathcal{M}}_{S_0} = \vartheta_G(\mathcal{M}_{S_0})$, i.e., the initial states of $\overline{\mathcal{M}}$ are the image under ϑ_G of the initial states of $\mathcal{M}$.
- $\overline{\mathcal{M}}_{AP} = \mathcal{M}_{AP}$, i.e., $\overline{\mathcal{M}}$ has the same atomic propositions as $\mathcal{M}$.
- $\overline{\mathcal{M}}_L(\overline{s}) = \mathcal{M}_L(\overline{s})$, i.e., the label of a state in $\overline{\mathcal{M}}$ is the same as its label in $\mathcal{M}$.
- $\overline{\mathcal{M}}_{SH} = \mathcal{M}_{SH}$, i.e., $\overline{\mathcal{M}}$ has the same shared variables as $\mathcal{M}$.
- $\overline{\mathcal{M}}_V(\overline{s}) = \mathcal{M}_V(\overline{s})$, i.e., the shared variables valuation of a state in $\overline{\mathcal{M}}$ is the same as its shared variables valuation in $\mathcal{M}$.

Thus *the states of a quotient structure correspond exactly to the orbits of states of the original structure* under the group of state mappings. For transitions, we have a slightly more subtle correspondence. Consider the following examples:

Example 5.8. In Figure 3 we demonstrate the correspondence between Kripke structures, G-closed substructures, and their quotients. In the figure, we present a multiprocess Kripke structure $\mathcal{M}$ corresponding to two concurrent processes P_1 (atomic propositions and transitions in blue) and P_2 (atomic propositions and transitions in red). The group G of state mappings swaps the indices of the processes. This structure has a G-closed substructure $\mathcal{N}$ constructed by removing the "center" state $\mathbf{u_0}$. Define ϑ_G to take the "left-most" state in the orbit, i.e., $\vartheta_G(\mathbf{t_1}) = \mathbf{t_0}$, $\vartheta_G(\mathbf{t_5}) = \mathbf{t_2}$, $\vartheta_G(\mathbf{u_0}) = \mathbf{u_0}$, $\vartheta_G(\mathbf{t_6}) = \mathbf{t_3}$, $\vartheta_G(\mathbf{t_4}) = \mathbf{t_4}$. The quotient structure $\mathcal{M}/(G, \vartheta_G)$ appears in the top right. While the quotient structure is isomorphic to a substructure of $\mathcal{M}$, this is not always the case. (See Figure 7 for an example where the quotient gains a new transition.) The quotient structure $\mathcal{N}/(G, \vartheta_G|\mathcal{N}_S)$ appears in the bottom right.

Example 5.9 (Example Box). Let $\mathcal{M}$ and G be as in Example 5.5. Let ϑ_G be defined by $\vartheta_G(\mathbf{s_1}) = \vartheta_G(\mathbf{s_2}) = \mathbf{s_1}$ and $\vartheta_G(\mathbf{t_1}) = \vartheta_G(\mathbf{t_2}) = \mathbf{t_1}$. Then the quotient structure $\mathcal{M}/(G, \vartheta_G)$ has exactly two states, $\mathbf{s_1}$ and $\mathbf{t_1}$ with transitions $(\mathbf{s_1}, \mathbf{t_1})$, $(\mathbf{t_1}, \mathbf{s_1})$. Also, the G-closed substructures of $\mathcal{M}$ given in Figure 2(a)–(c) also map to this quotient structure via $\mathcal{N} \to \mathcal{N}/(G, \vartheta_G)$. Note that the transition $(\mathbf{t_1}, \mathbf{s_1})$ is present in the quotient, but is not present, for example, in the structure of Figure 2(b). However, the "corresponding" transition $(\mathbf{t_2}, \mathbf{s_1})$ is present in Figure 2(b).

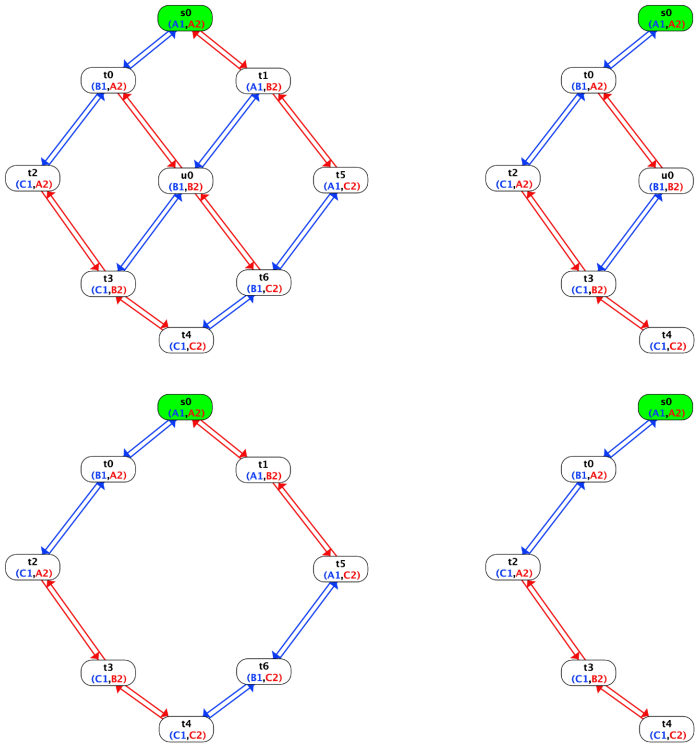

Fig. 3. As discussed in Example 5.8, we have a Kripke structure in the top left and a G-closed substructure in the bottom left. On the right, we have the quotients of the whole structure (top) and the G-closed substructure (bottom).

Example 5.9 shows that *many G-closed substructures can have the same quotient structure*, and also that, in general, *a transition in the quotient may not itself be present in the original structure*. We show, however, in Theorem 5.10 below that *a "corresponding" transition is guaranteed to be present in the original structure*. These corresponding transitions can be joined into a path which corresponds state-by-state to the path in the quotient. This "path correspondence" is what allows for model checking of $\mathcal{M}$ via model checking of $\overline{\mathcal{M}}$ and is formalized in the following theorem, which is Lemma 3.1 in Emerson and Sistla [24].

In the sequel, we fix a Kripke structure $\mathcal{M}$, a group G of state mappings of $\mathcal{M}$, and a representative map ϑ_G of $\mathcal{M}$ with respect to G.

Theorem 5.10 (Path Correspondence Theorem [24]). *There is a bidirectional correspondence between paths of $\mathcal{M}$ and paths of $\overline{\mathcal{M}}$. Formally we have the following:*

(1) *If $x = s_0, s_1, s_2, \ldots$ is a path in $\mathcal{M}$, then $\overline{x} = \overline{s_0}, \overline{s_1}, \overline{s_2}, \ldots$ is a path in $\overline{\mathcal{M}}$ where $\overline{s_i} = \vartheta_G(s_i)$.*
(2) *If $\overline{x} = \overline{s_0}, \overline{s_1}, \overline{s_2}, \ldots$ is a path in $\overline{\mathcal{M}}$, then for every state $s'_0 \in \mathcal{M}_S$ such that $\vartheta_G(s'_0) = \overline{s_0}$ there is a path $s'_0, s'_1, s'_2, \ldots$ in $\mathcal{M}$ such that $\vartheta_G(s'_i) = \overline{s_i}$.*

We now extend the path correspondence between $\mathcal{M}$ and $\overline{\mathcal{M}}$ to a correspondence between G-closed substructures of $\mathcal{M}$ and substructures of $\overline{\mathcal{M}}$. Define $\Psi : \Lambda_{\mathcal{M},G} \to \Lambda_{\overline{\mathcal{M}}}$, by $\Psi(\mathcal{N}) = \mathcal{N}/(G, \vartheta_G)$, so that Ψ maps a G-closed substructure $\mathcal{N}$ of $\mathcal{M}$ to a corresponding substructure of $\overline{\mathcal{M}}$,

by taking the quotient w.r.t. (G, ϑ_G). We call Ψ the *quotient map*. Ψ establishes a join-semilattice homomorphism between $\Lambda_{\mathcal{M},G}$ and $\Lambda_{\overline{\mathcal{M}}}$ as we now show in the following series of lemmas.

Lemma 5.11. *For every substructure $\overline{\mathcal{N}}$ of $\overline{\mathcal{M}}$, there is a G-closed substructure $\mathcal{N}$ of $\mathcal{M}$ such that $\mathcal{N}/(G, \vartheta_G) = \overline{\mathcal{N}}$.*

Lemma 5.11 establishes that Ψ is surjective. We note that every substructure $\overline{\mathcal{N}}$ of $\overline{\mathcal{M}}$ defines a set of states of $\overline{\mathcal{M}}$, i.e., the orbits of the states in $\overline{\mathcal{N}}$. However, in general, the transitions of $\overline{\mathcal{N}}$ do not uniquely define transitions in $\overline{\mathcal{M}}$.

The next lemma demonstrates that Ψ is a homomorphism of the join-semilattices $\Lambda_{M,G}$ and $\Lambda_{\overline{\mathcal{M}}}$. We note that it is not a homomorphism of the lattices themselves because the meet of two G-closed structures might be empty.

Lemma 5.12 (Quotient Map Respects Join). *Let $\mathcal{N}, \mathcal{N}' \in \Lambda_{\mathcal{M},G}$. Then*

$$\Psi(\mathcal{N} \vee \mathcal{N}') = \Psi(\mathcal{N}) \vee \Psi(\mathcal{N}').$$

As seen in Example 5.9, it is possible for multiple G-closed substructures of $\mathcal{M}$ to map to the same substructure of the quotient structure $\overline{\mathcal{M}}$. To obtain a single well-defined preimage for each substructure of the quotient structure, we introduce the concept of G-maximal. Recall that the join of G-closed substructures of $\mathcal{M}$ is G-closed.

Definition 5.13 (G-Maximal). A G-closed substructure $\mathcal{N}$ of $\mathcal{M}$ is *G-maximal* if

$$\mathcal{N} = \bigvee_{\substack{\mathcal{N}' \in \Lambda_{\mathcal{M},G} \\ \mathcal{N}'/(G,\vartheta_G) \leq \mathcal{N}/(G,\vartheta_G)}} \mathcal{N}'.$$

That is, $\mathcal{N}$ is the join of all G-closed substructures of $\mathcal{M}$ whose quotient is a substructure of the quotient of $\mathcal{N}$ itself, namely of $\mathcal{N}/(G, \vartheta_G)$. A G-closed substructure $\mathcal{N}$ fails to be G-maximal exactly when there are states $s, t \in \mathcal{N}$, such that $(s, t) \in \mathcal{N}/(G, \vartheta_G)$, but (s, t) is not in $\mathcal{N}$.

Among all of the G-closed substructures in Figure 2 only the entire structure itself is G-maximal

Lemma 5.14. *Let $\mathcal{M}', \mathcal{M}''$ be two G-maximal substructures of $\mathcal{M}$. Then $\mathcal{M}' \vee \mathcal{M}''$ is G-maximal and $\mathcal{M}' \wedge \mathcal{M}''$ is G-maximal.*

Lemma 5.14 allows us to make the following definition.

Definition 5.15 (G-Maximal Lattice of Substructures). The set of G-maximal substructures of $\mathcal{M}$ forms a sublattice $\Lambda_{\mathcal{M},G-\max}$ of $\Lambda_{\mathcal{M}}$.

While in general the quotient map from $\Lambda_{\mathcal{M},G}$ to $\Lambda_{\overline{\mathcal{M}}}$ is always surjective, when restricted to $\Lambda_{\mathcal{M},G-\max}$, the map is injective and is a lattice isomorphism.

Theorem 5.16 (G-Maximal Lattice Correspondence). *The restriction of the quotient map Ψ to $\Lambda_{\mathcal{M},G-max}$ is an isomorphism from $\Lambda_{\mathcal{M},G-max}$ to $\Lambda_{\overline{\mathcal{M}}}$, i.e., between the lattice of G-maximal substructures of $\mathcal{M}$ and the lattice of structures of $\overline{\mathcal{M}}$.*

At this point, we would like to remind the reader of the various lattices that we have defined and how they relate to each other:

$$\underbrace{G\text{-maximal substructures}}_{\Lambda_{\mathcal{M},G-\max}} \subseteq \underbrace{G\text{-closed substructures}}_{\Lambda_{\mathcal{M},G}} \subseteq \underbrace{\text{All substructures}}_{\Lambda_{\mathcal{M}}}.$$

5.3 Semantic Relationships between Structures and Quotient Structures

A subformulae φ' of a CTL formula φ is a propositional subformulae iff φ' contains no temporal modalities. A subformulae φ' of a CTL formula φ is a maximal propositional subformulae iff φ' is a propositional subformula of φ, and there is no propositional subformula φ'' of φ which properly contains φ'.

Definition 5.17. Let G be a group of state mappings of $\mathcal{M}$. A CTL formula φ is *G-invariant* over $\mathcal{M}$, if for every state s, every $g \in G$, and all maximal propositional subformulae φ' of φ, we have

$$\mathcal{M}, s \models \varphi' \iff \mathcal{M}, g(s) \models \varphi'.$$

Lemma 5.18. *If φ is G-invariant, then the valuation of φ in $\overline{\mathcal{M}}$ does not depend on the choice of representative map ϑ_G.*

This allows us to connect semantic statements about $\mathcal{M}$ with semantic statements about $\overline{\mathcal{M}}$ for formulae that are G-invariant. The path correspondence theorem establishes a strong bisimulation between $\mathcal{M}$ and $\overline{\mathcal{M}}$, in which state s of $\mathcal{M}$ and state $\bar{s}$ of $\overline{\mathcal{M}}$ are bisimilar iff s is in the orbit of $\bar{s}$, i.e., $s = g(\bar{s})$ for some $g \in G$. We call such a bisimulation a G-bisimulation, and use the usual symbol $\sim$ to denote it. Hence, G-bisimilar states satisfy the same propositional subformulae of any G-invariant CTL formula φ. A straightforward induction over path length then shows that s and $\bar{s}$ satisfy the same G-invariant CTL formulae:

Corollary 5.19. *$\mathcal{M} \models \varphi$ iff $\overline{\mathcal{M}} \models \varphi$ for all G-invariant CTL formulae φ.*

Lemma 5.20. *Let $s \in \mathcal{M}_S$, $t \in \overline{\mathcal{M}}_S$. Let φ be a G-invariant CTL formula. If $t = \vartheta_G(s)$, then $\mathcal{M}, s \models \varphi \iff \overline{\mathcal{M}}, t \models \varphi$.*

Section 3.2 developed the theory of substructures of a Kripke structure. This development was motivated by the following definition and theorem.

Definition 5.21 (Substructure Repair). Given a structure $\mathcal{M}$ and a CTL formula φ, we call a non-empty substructure $\mathcal{N}$ of $\mathcal{M}$ a *substructure repair* of $\mathcal{M}$ with respect to φ if $\mathcal{N} \models \varphi$.

If a CTL formula φ is G-invariant, then the lattice correspondence will respect the valuation of φ. This leads to one of our main results, the repair-correspondence theorem, which provides the basis for repairing a substructure of $\overline{\mathcal{M}}$ and then lifting the repair to $\mathcal{M}$.

Theorem 5.22 (Repair Correspondence). *Let φ be a G-invariant CTL formula. Let $\mathcal{N}$ be a non-empty G-closed substructure of $\mathcal{M}$, $s \in \mathcal{N}_S$, and $\overline{\mathcal{N}} = \mathcal{N}/(G, \vartheta_G)$. Then $\mathcal{N}, s \models \varphi \iff \overline{\mathcal{N}}, \vartheta_G(s) \models \varphi$.*

6 Repair of Concurrent Programs

We now apply the Repair Correspondence Theorem to realize our goal: the repair of concurrent programs. Given a concurrent program P and a CTL formula φ, we wish to modify P to produce a repaired program P^r such that $\mathcal{M}^r \models \varphi$, where $\mathcal{M}^r = GSTD(P^r)$, i.e., $\mathcal{M}^r$ is the global state-transition graph of P^r. That is, we obtain $P^r \models \varphi$. The modification is "subtractive," that is, it only removes behaviors and does not add them.

Our concurrent program repair method is, roughly, as follows. Appendix B presents the algorithm of Emerson and Sistla [24, Figure 1] that we use to generate the symmetry-reduced state-transition graph $\overline{\mathcal{M}}$ of P. We then apply the model repair algorithm of Attie et al. [3] to repair $\overline{\mathcal{M}}$ w.r.t. φ,

which either produces $\overline{N} \leq \overline{M}$ such that $\overline{N} \models \varphi$ or reports that no such $\overline{N}$ exists.[2] We then extract a repaired concurrent program P^r from $\overline{N}$. There are, however, some issues to be dealt with, arising from the fact that symmetry reduction can create "irregular" transitions, in which the state of more than one process is changed. The following technical development deals with these issues.

We assume henceforth that when M is a multiprocess Kripke structure over process indices $1, \ldots, n$, that the symmetry group G is a subgroup of S_n, the group of permutations on $\{1, \ldots, n\}$. To be able to properly define the effect of $g \in S_n$ on the values of the shared variables, we restrict the possible domains of shared variables to those that can be built up starting with the process indices $\{1, \ldots, n\}$.

Definition 6.1 (Process-Index Based Domain). A process index based domain is one of the following:

- The process indices $\{1, \ldots, n\}$.
- k-tuples of process indices, i.e., the domain is $\{1, \ldots, n\}^k$.
- Sets of process indices, i.e., the domain is the powerset of $\{1, \ldots, n\}$.

Definition 6.2 (Application of G to Process-Index Based Domain). Let $g \in G$. Then $g(\langle i_1, \ldots, i_k \rangle = \langle g(i_1), \ldots, g(i_k) \rangle$ and $g(\{i_1, \ldots, i_k\}) = \{g(i_1), \ldots, g(i_k)\}$. That is, index permutations are applied element-wise.

We proceed to extract a repaired concurrent program from $\overline{N}$ using the projection method of [5, 19]: each transition $s \xrightarrow{i} t$ is turned into an i-action, as given by Definition 6.3:

Definition 6.3 (Extraction of an i-Action from a Transition). For transition $s \xrightarrow{i} t$ define $action(s \xrightarrow{i} t) \triangleq (s\restriction i, B \rightarrow A, t\restriction i)$ where

- $B \triangleq \{|s|\}$ where $\{|s|\} \triangleq$ "$(\bigwedge_{Q \in \mathcal{N}_L(s)} Q) \wedge (\bigwedge_{Q \notin \mathcal{N}_L(s)} \neg Q)$" $\wedge \{|s\restriction \mathcal{S}H|\}$, and Q ranges over AP.
- The assignment A is the (unique) smallest assignment statement that satisfies $\{s\restriction \mathcal{S}H\}A\{t\restriction \mathcal{S}H\}$.

The guard B must check that the current global state is actually s, i.e., B must be true in s and false in all other global states. The definition of B above ensures that B evaluates to true in state s, and, in a state t which differs from s in some atomic proposition or some shared variable, B evaluates to false. We assumed above (see Section 3) that any two different global states do differ in some atomic proposition or some shared variable, and so B does indeed hold only in state s. The assignment A effects the changes in shared variable values needed to change the shared variable valuation from that in s (i.e., $M_V(s)$) to that in t (i.e., $M_V(t)$). In effect, for each shared variable x such that $M_V(s)(x) \neq M_V(t)(x)$, we add the assignment $x := M_V(t)(x)$ to A. Definition 6.3 ensures that the inclusion of $action(s \xrightarrow{i} t)$ in process P_i of a concurrent program will generate exactly the transition $s \xrightarrow{i} t$, and no other transitions.

A key problem concerning the extraction of actions from transitions is that the definition of the quotient $\overline{M}$ allows transitions in which the atomic propositions of more than one process are changed, since any representative of an orbit can be chosen. Hence the repaired $\overline{N} \leq \overline{M}$ can also contain such transitions, e.g., the transition from *S6* to *S1* in Figure 8, which we write as $[C_1\, T_2] \rightarrow [T_1\, N_2]$. Note that the propositions of both processes 1 and 2 are changed. To be able to

[2]This algorithm is sound and complete, so that if $\overline{M}$ has some substructure that satisfies φ, then the algorithm will return such a substructure $\overline{N}$. If not, the algorithm will report that no repair exists.

extract an i-action, such transitions must be converted so that only the atomic propositions of a single process P_i are modified.

Definition 6.4 (Regular and Irregular Transitions). Define a transition from s to t to be *regular* iff it modifies atomic propositions in at most one AP_i, so that $s{\downarrow}i = t{\downarrow}i$ for some process index i, and write the transition as $s \xrightarrow{i} t$.

Define a transition from s to t to be *irregular* iff it is not regular, i.e., it modifies atomic propositions in more than one AP_i, and write the transition as $s \rightarrow t$, with no process index labeling the arrow.

Lemma 6.5. *For each irregular transition $s \rightarrow t \in \overline{N}_T$, there is $g' \in G$ such that $s \rightarrow g'(t)$ is regular.*

For example, by applying the permutation of process indices 1, 2 to $[T_1\, N_2]$, from the irregular transition $[C_1\, T_2] \rightarrow [T_1\, N_2]$ we extract the regular transition $[C_1\, T_2] \xrightarrow{1} [N_1\, T_2]$.

Definition 6.6 (Regular Transition Extraction, $Reg_i(\overline{N}_T)$). Define $Reg_i(\overline{N}_T) \triangleq \{s \xrightarrow{i} g(t) \mid g \in G \text{ and } s \rightarrow t \in \overline{N}_T\}$, i.e., the set consisting of the regular transitions $s \xrightarrow{i} g(t)$ that are in $\overline{N}_T$ (g is the identity mapping) and the regular transitions $s \xrightarrow{i} g(t)$ such that $s \rightarrow t$ is an irregular transition in $\overline{N}_T$ (g is not the identity mapping).

Since g can be the identity element of G, it follows that $Reg_i(\overline{N}_T)$ accounts for both regular and irregular transitions in $\overline{N}_T$. Our method proceeds by extracting all the regular transitions from $\overline{N}$, turning these transitions into actions, and then "completing" the actions by symmetrizing them. We extract all possible regular transitions from $\overline{N}$ by converting irregular transitions to regular ones, using Definition 6.6. To extract an i-action from each regular transition, we apply Definition 6.7:

Definition 6.7 (Extraction of i-Actions, $Act_i(\overline{N}_T)$). Define $Act_i(\overline{N}_T) = \{action(s \xrightarrow{i} t) \mid s \xrightarrow{i} t \in Reg_i(\overline{N}_T)\}$, i.e., the set of i-actions obtained from $Reg_i(\overline{N}_T)$.

This produces an initial (incomplete) set of actions for each process P_i in the repaired program. We complete the actions of P_i by "symmetrizing" them: we apply every $g \in G$ to all extracted actions as formalized in Definitions 6.8–Definitions 6.10 below. The resulting set of actions are then allocated to their respective processes, and the repaired concurrent program is complete. Our concurrent program repair algorithm is summarized in Figure 4.

Definition 6.8 (Application of G to Local Process States). Define the action of $g \in G$ on a local state s_i of process P_i to be the local state s_j of process P_j where $j = g(i)$ and $L(s_j) = \{Q_j \mid Q_i \in L(s_i)\}$, i.e., the "same" atomic propositions, modulo process indices, hold in s_j as in s_i.

Definition 6.9 (Application of G to Concurrent Program Syntax). Define the action of $g \in G$ on to the various elements of the syntax of a process P_i as follows.

- For atomic proposition Q_i, $g(Q_i) = Q_{g(i)}$.
- For guard B, by induction: $g(\neg B) = \neg g(B)$ and $g(B1 \wedge B2) = g(B1) \wedge g(B2)$, with the base case given by $g(Q_i)$ above.
- For multiple-assignment statement A, let A be $x_1, \ldots, x_k := v_1, \ldots, v_k$. Then $g(x_1, \ldots, x_k := v_1, \ldots, v_k) \triangleq$ "$x_1, \ldots, x_k := g(v_1), \ldots, g(v_k)$", i.e., g is applied to the assigned values, according to Definition 6.2. Note that the shared variables remain unchanged.
- For i-action $(s_i, B \rightarrow A, t_i)$: $g(s_i, B \rightarrow A, t_i) = (g(s_i), g(B) \rightarrow g(A), g(t_i))$. That is, we apply g to all elements of the i-action, with $g(s_i)$, $g(t_i)$ given by Definition 6.8 above.

1. The input is a concurrent program P and a CTL formula φ
2. Generate the symmetry-reduced state transition graph $\overline{\mathcal{M}}$ of P using the algorithm of Emerson and Sistla [24, Figure 1].
3. Apply the model repair algorithm of Attie et al. [3] to repair $\overline{\mathcal{M}}$ w.r.t. φ. Let $\overline{\mathcal{N}}$ be the substructure of $\overline{\mathcal{M}}$ resulting from the repair.
4. Apply Definition 6.6 to extract all regular transitions from $\overline{\mathcal{N}}$:
   $Reg_i(\overline{\mathcal{N}}_T) \triangleq \{s \xrightarrow{i} g(t) \mid g \in G \text{ and } s \to t \in \overline{\mathcal{N}}_T\}$.
5. Apply Definition 6.7 to extract an action from each regular transition.
   $Act_i(\overline{\mathcal{N}}_T) = \{action(s \xrightarrow{i} t) \mid s \xrightarrow{i} t \in Reg_i(\overline{\mathcal{N}}_T)\}$.
   This produces an initial (incomplete) set of actions for each process P_i in the repaired program.
6. Apply Definition 6.10 to complete the actions of P_i by "symmetrizing" them:
   $Act_i^G(\overline{\mathcal{N}}_T) = \{g(a) \mid g \in G, a \in Act_j(\overline{\mathcal{N}}_T), g(j) = i\}$.
7. The repaired concurrent program is then $\overline{P}^G = \overline{P}_1^G \parallel \cdots \parallel \overline{P}_n^G$, where $\overline{P}_i^G$ consists of the i-actions in $Act_i^G(\overline{\mathcal{N}}_T)$.

Fig. 4. Concurrent program repair algorithm.

Definition 6.10 (Symmetrized Actions, $Act_i^G(\overline{\mathcal{N}}_T)$). Define $Act_i^G(\overline{\mathcal{N}}_T)$, the symmetrization of $Act_i(\overline{\mathcal{N}}_T)$, by $Act_i^G(\overline{\mathcal{N}}_T) = \{g(a) \mid g \in G, a \in Act_j(\overline{\mathcal{N}}_T), g(j) = i\}$.

Theorem 6.11. *Let $\overline{P}^G$ be the concurrent program extracted from $\overline{\mathcal{N}}$ as above, let $\mathcal{N}^p$ be the state-transition graph generated by the execution of $\overline{P}^G$, and let $\overline{\mathcal{N}^p} = \mathcal{N}^p/(G, \vartheta_G)$. Then $\mathcal{N}^p$ is G-closed and $\overline{\mathcal{N}^p} = \overline{\mathcal{N}}$.*

Corollary 6.12. *Let $\overline{P}^G$ be the concurrent program extracted from $\overline{\mathcal{N}}$ as above, and φ the CTL specification that was used to repair $\overline{\mathcal{M}}$, resulting in $\overline{\mathcal{N}}$. Then $\overline{P}^G \models \varphi$.*

As noted, applying this algorithm to the symmetry-reduced state-transition graph is only complete with respect to symmetric repairs, see Example 7.3 below.

7 Examples

7.1 Two Process Mutual Exclusion

We illustrate the concurrent program repair algorithm of Figure 4 using mutual exclusion for two processes P_1, P_2. Each P_i has three local states: N_i (neutral, computing locally), T_i (trying, has requested critical section entry), and C_i (in the critical section). For Step 1, we start with the "trivial" program P shown in Figure 5 in which all action guards are "true," and the specification $\varphi = \mathsf{AG}\neg(C_1 \wedge C_2) \wedge \mathsf{AG}((T_1 \vee T_2) \Rightarrow \mathsf{AF}(C_1 \vee C_2))$. The first conjunct specifies mutual exclusion of the critical sections (safety) and the second specifies progress: if some process requests the critical section then some process will obtain it (liveness).

Figure 6 shows the global state-transition diagram $\mathcal{M}$ of the concurrent program of Figure 5. The transitions of P_1, P_2 are shown in blue, red, respectively. Clearly, $\mathcal{M} \not\models \varphi$. Actually both conjuncts are violated: $\mathsf{AG}\neg(C_1 \wedge C_2)$ due to the reachability of state *S8* from the initial state, and $\mathsf{AG}((T_1 \vee T_2) \Rightarrow \mathsf{AF}(C_1 \vee C_2))$ due to the self-loop on state *S4*.

$\mathcal{M}$ has exactly two symmetries: the identity map and the map that swaps the process indices 1 and 2. So G is the group $\{e, (1, 2)\}$ where e is the identity map and $(1, 2)$ denotes the permutation of

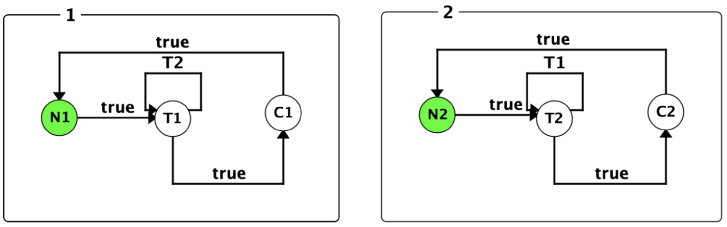

Fig. 5. Initial incorrect mutual exclusion program from Section 7.1.

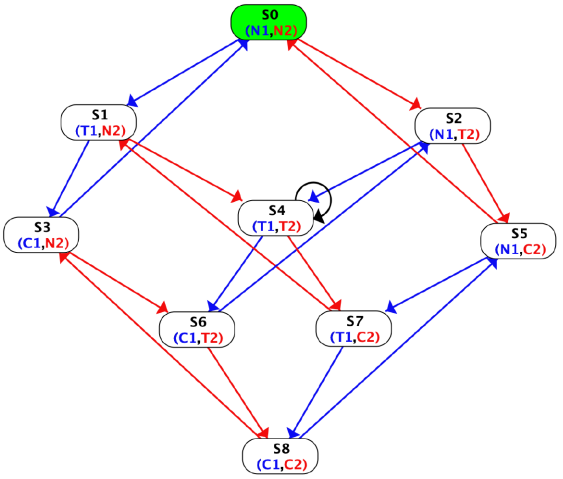

Fig. 6. The Kripke structure $\mathcal{M}$ that is the global state-transition graph of the concurrent program of Figure 5.

process indices 1 and 2. It is easy to see that φ above is G-invariant, since its maximal propositional subformulae are unchanged by the permutation $(1, 2)$.

Our concurrent program repair algorithm does not generate $\mathcal{M}$ since, in general, $\mathcal{M}$ may be large, and we show $\mathcal{M}$ only for exposition. Instead, step 2 applies the Emerson-Sistla algorithm [24, Figure 1] to generate the symmetry-reduced Kripke structure $\overline{\mathcal{M}} = \mathcal{M}/(G, \vartheta_G)$ directly from the concurrent program of Figure 5. $\overline{\mathcal{M}}$ is given in Figure 7. $\overline{\mathcal{M}}$ has a transition (shown in black) from state *S6* to *S1*, which is the quotient of the transition from *S6* to *S2* in $\mathcal{M}$, i.e., $\vartheta_G(\mathbf{S6}) = \mathbf{S6}$ and $\vartheta_G(\mathbf{S2}) = \mathbf{S1}$ so the edge $(\vartheta_G(\mathbf{S6}), \vartheta_G(\mathbf{S2}))$ occurs in $\overline{\mathcal{M}}$.

Step 3 applies the model repair algorithm of Attie et al. [3] to repair $\overline{\mathcal{M}}$ w.r.t. φ, resulting in $\overline{\mathcal{N}}$, which is given in Figure 8.

Figure 9 shows the lifting of the repair to $\mathcal{M}$. The deleted transitions and states are shown dashed. This is not part of the repair algorithm, and is shown for exposition only.

Step 4 applies Definition 6.6 to extract all regular transitions from $\overline{\mathcal{N}}$ (Figure 8). The red and blue transitions in Figure 8 are already regular. The black transition $\mathbf{S6} \rightarrow \mathbf{S1}$ is made regular by applying $(1, 2) \in G$ which permutes indices 1 and 2 to the target state **S1**. This results in the regular transition $\mathbf{S6} \rightarrow \mathbf{S2}$, where **S2** is the state labeled with $\{N_1, T_2\}$. Thus the state of P_1 changes from C_1 to N_1, while P_2 remains in T_2.

Step 5 applies the Definition 6.7 to extract an action from each regular transition. This results in the "incomplete" concurrent program shown in Figure 10.

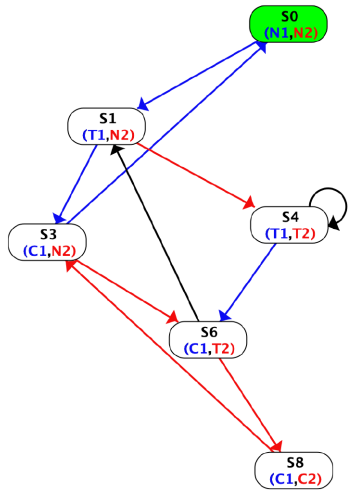

Fig. 7. The quotient $\overline{\mathcal{M}} = \mathcal{M}/(G, \vartheta_G)$ of the Kripke structure $\mathcal{M}$ of Figure 6.

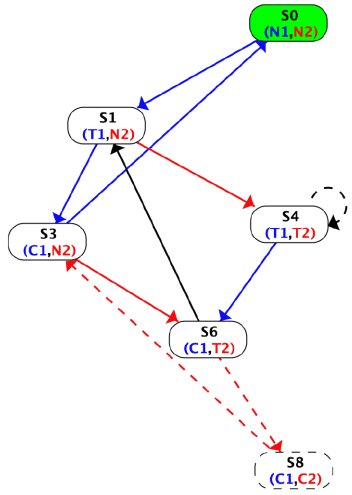

Fig. 8. The repair $\overline{\mathcal{N}}$ of $\overline{\mathcal{M}}$ w.r.t. specification φ.

Step 6 applies Definition 6.10 to complete the actions of P_i by "symmetrizing" them. Step 7 then gives the final repaired concurrent program $\overline{P}^G$, shown in Figure 11. Note that $\oplus$ means "disjunction" of guarded commands: any command may be executed if its guard is true in the current global state [4]. An action $(s_i, B_1 \oplus B_2, t_i)$ really represents two i-actions: (s_i, B_1, t_i) and (s_i, B_2, t_i).

By Corollary 6.12, $\overline{P}^G \models \varphi$, i.e., the repaired program satisfies the specification $\varphi = \mathsf{AG}\neg(C_1 \wedge C_2) \wedge \mathsf{AG}((T_1 \vee T_2) \Rightarrow \mathsf{AF}(C_1 \vee C_2))$.

7.2 *n*-Process Mutual Exclusion

We now consider mutual exclusion for n-processes. We give a concrete example for three processes (Figures 12 and 13) while our discussion deals with the general case of n processes. To reduce

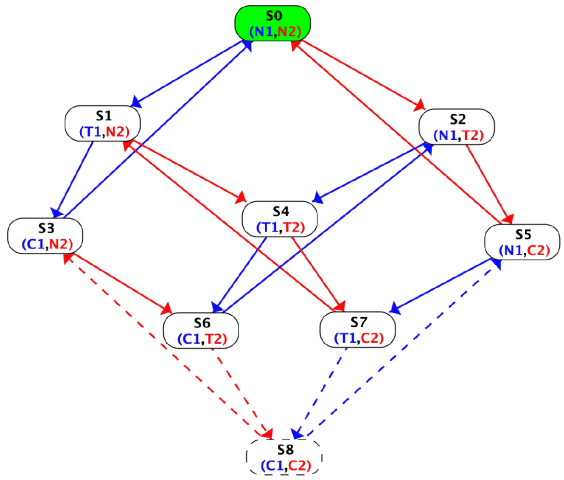

Fig. 9. The lifting of the repair $\overline{N}$ to $\mathcal{M}$.

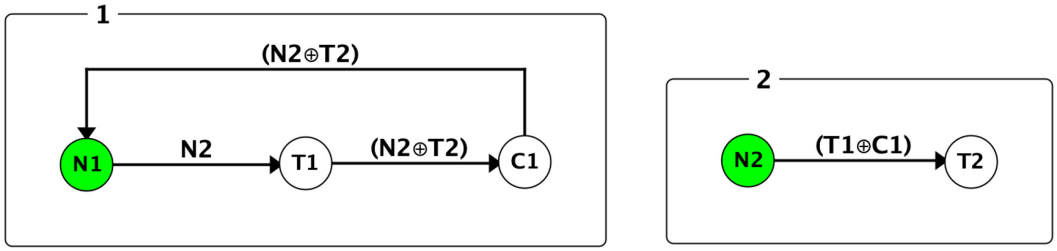

Fig. 10. The incomplete repaired concurrent program extracted from Figure 9.

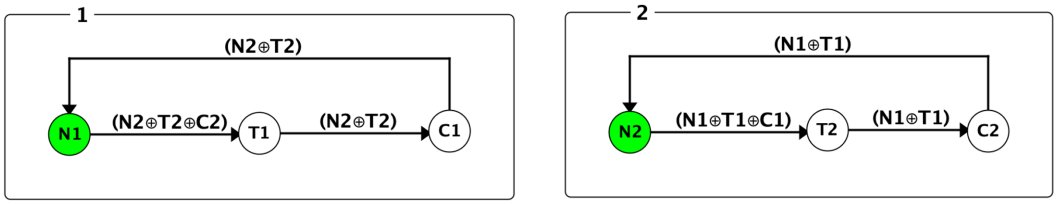

Fig. 11. The repaired concurrent program $\overline{P}^G$ obtained by symmetrizing Figure 10.

clutter in the example, we remove the trying (T_i) state, thus each process P_i can move directly from N_i to C_i, i.e., the guards on all actions are initially "true," just like in Figure 5.

We consider the mutual exclusion specification

$$\bigwedge_{i \neq j} \mathrm{AG} \neg (C_i \wedge C_j)$$

which requires that no two processes are in their critical section at the same time. This implies mutual exclusion amongst all n processes. The above is equivalent to

$$\mathrm{AG} \left(\bigwedge_{i \neq j} \neg (C_i \wedge C_j) \right).$$

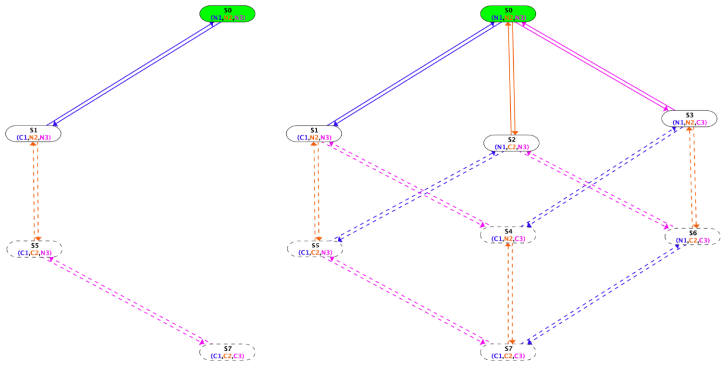

Fig. 12. The Kripke structure defined in Section 7.2. On the left is the repair of $\overline{\mathcal{M}}$ and the lifting of the repair to $\mathcal{M}$ appears on the right.

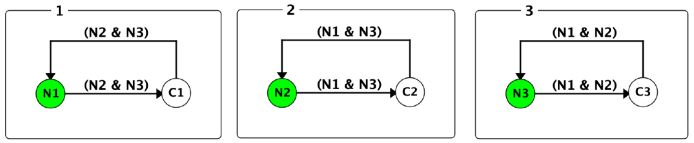

Fig. 13. The repaired program $\overline{P}^G$ for the program in Section 7.2.

The group of state mappings G for both structure and specification is the full permutation group on the indices $\{1, \ldots, n\}$. We see by inspection that the maximal propositional subformula $\bigwedge_{i \neq j} \neg(C_i \wedge C_j)$ is unchanged by any permutation of $\{1, \ldots, n\}$, and so $\mathrm{AG}(\bigwedge_{i \neq j} \neg(C_i \wedge C_j))$ is G-invariant.

For n processes, the quotient model by the full group of symmetries has $n + 1$ states, while the original model would have 2^n states. Figure 12 shows the repair of the quotient $\overline{\mathcal{M}}$ and then the lifting of the repair to the original structure $\mathcal{M}$. Figure 13 shows the correct (repaired) program $\overline{P}^G$ that is extracted from the repaired quotient in Figure 12. For n processes, the guard on actions of $\overline{P}_i^G$ is $\bigwedge_{j \neq i} N_j$.

7.3 No G-Closed Repairs

Consider the structure in Figure 14 and the formula $\varphi = \mathrm{AXAXAX}p$. The structure $\mathcal{M}$ has a single initial state. Let G be the group consisting of the identity and the map swapping *S1* and *S2*. In Figure 14 we see that the quotient structure $\mathcal{M}/(G, \vartheta_G)$ does not have any non-empty repairs with respect to φ. But, $\mathcal{M}$ does contain a substructure $\mathcal{N}$ that satisfies φ.

8 Relative Completeness of Group-Theoretic Repair

By the Repair Correspondence (Theorem 5.22), the existence of a repair $\overline{\mathcal{N}}$ of $\overline{\mathcal{M}}$ implies the existence of a repair $\mathcal{N}$ of $\mathcal{M}$. In Example 7.3, we gave an example in which a repair $\mathcal{N}$ of $\mathcal{M}$ exists but no G-closed repair does, i.e., $\overline{\mathcal{M}}$ has no repairs. This leads us to ask: is there a fragment of CTL, and/or a class of Kripke structures, for which group-theoretic repair is complete? That is, the existence of a repair (substructure $\mathcal{N}$ of $\mathcal{M}$ that satisfies φ) implies the existence of a G-closed repair (substructure $\overline{\mathcal{N}}$ of $\overline{\mathcal{M}}$ that satisfies φ).

An attempt to answer this question is to examine formulae and structures where substructures are equivalent to the smallest G-closed substructure containing them. Assume that there exists $\mathcal{N} \leq \mathcal{M}$ such that $\mathcal{N} \models \varphi$. Write $\mathcal{N}^G$ for the smallest G-closed substructure that contains $\mathcal{N}$. We

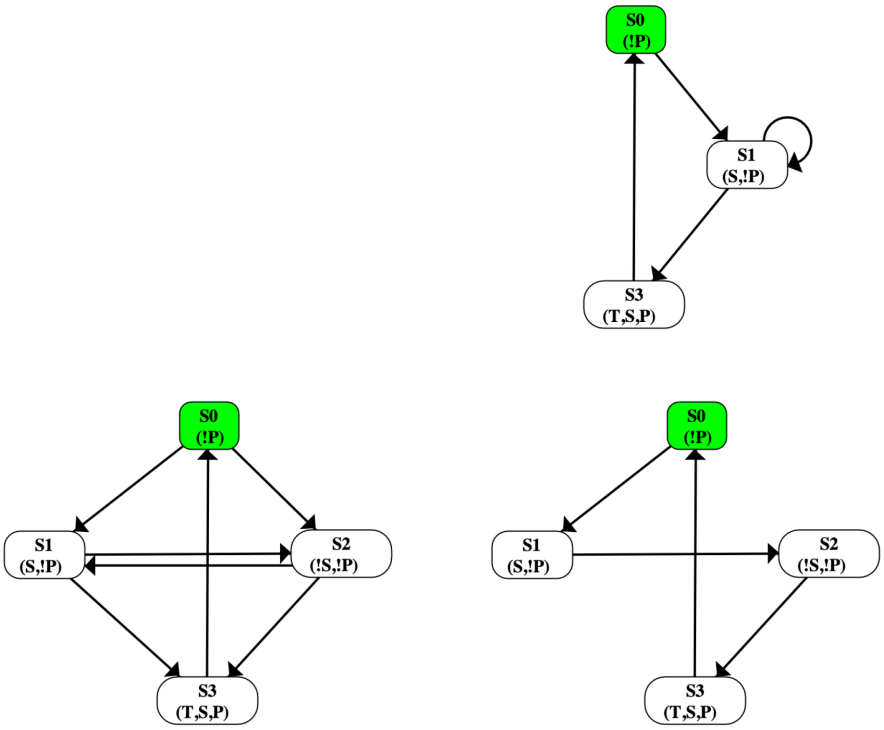

Fig. 14. The models from Section 7.3 from left to right: the model $\mathcal{M}$, the quotient of $\mathcal{M}$, a repair of $\mathcal{M}$ with respect to $f = \mathsf{AXAXAX}p$ that is not G-closed.

call $\mathcal{N}^G$ the G-closure of $\mathcal{N}$ in $\mathcal{M}$. If $\mathcal{N}^G$ is bisimilar to $\mathcal{N}$, then $\mathcal{N}^G \models \varphi$ and $\overline{\mathcal{N}^G} \models \varphi$ which is a substructure of $\overline{\mathcal{M}}$.

In [20], Emerson et al. give a criteria for a structure $\mathcal{M}$ to be bisimilar to the symmetrized structure $\mathcal{M}^G$, their criteria is: for any transition $(s,t) \in (\mathcal{M}^G)_T$, there must be a $g \in G$ such that $(s, gt) \in \mathcal{M}_T$. When asking about substructures, it is not clear what criteria on $\mathcal{M}$ is needed to ensure that each substructure $\mathcal{N}$ of $\mathcal{M}$ is bisimilar to $\mathcal{N}^G$.

Definition 8.1 (G-Repair Complete). Let $\mathcal{M}$ be a Kripke structure with a group of state mappings G and φ a G-invariant CTL formula. Let $\mathcal{N} \leq \mathcal{M}$ be any repair of $\mathcal{M}$ with respect to φ, and let s be any state in $\mathcal{N}_S$. Then the pair $(\mathcal{M}, \varphi)$ is *G-repair complete* if: $\mathcal{N}, s \models \varphi$ implies for all $g \in G$, we have $\mathcal{N}^G, g(s) \models \varphi$.

It is clear that propositional formulae are always G-repair complete. In addition we note the following:

Theorem 8.2. *If φ and ψ are purely propositional formulae then for any Kripke structure $\mathcal{M}$, the pair $(\mathcal{M}, A[\varphi\, R\, \psi])$ is G-repair complete.*

There exist structures $\mathcal{M}$ and formulae φ, ψ such that $(\mathcal{M}, \varphi)$ is G-repair complete, and $(\mathcal{M}, \psi)$ is G-repair complete, but $(\mathcal{M}, \varphi \wedge \psi)$ is not G-repair complete.

Example 8.3. Let $\mathcal{M}$ be the Kripke structure described by Figure 15. Let G be the group of state mappings generated by swapping *S1* and *S2*. Let $\varphi = \mathsf{A}[p\, \mathsf{R}\, q]$ and $\psi = \mathsf{AF}\neg q$. The structure $\mathcal{M}$ has

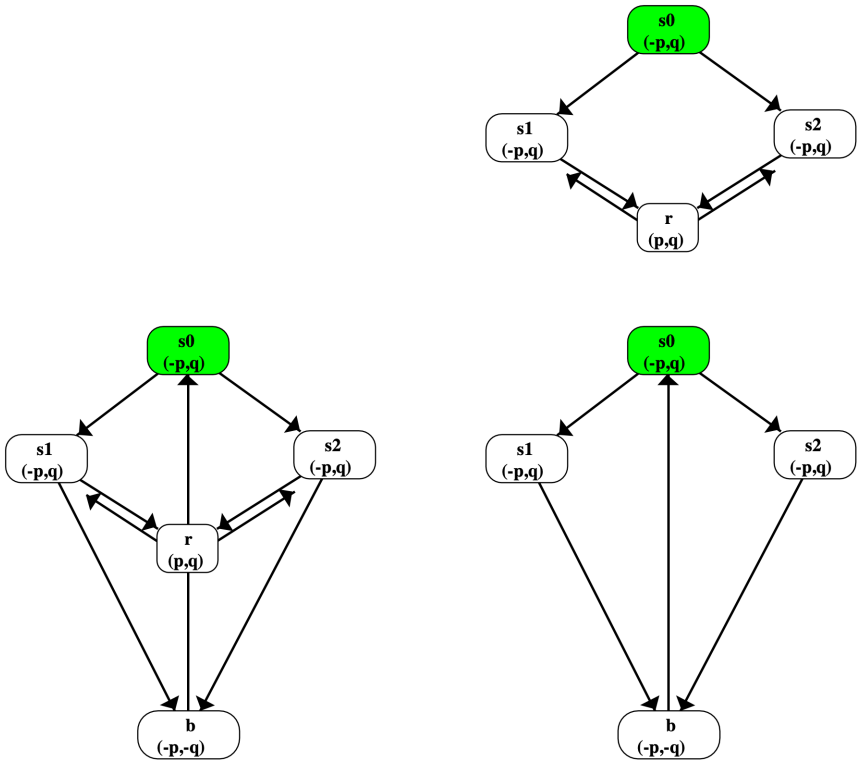

Fig. 15. The Kripke structure from Example 8.3 (on the left). Note that (b, s_0) is a transition, while (b, r) is not. Structures in the center and on the right show G-closed repairs of $\mathcal{M}$ with respect to the formulae $\mathsf{A}[p\,\mathsf{R}\,q]$ and $\mathsf{AF}\neg q$ respectively.

a non-empty G-closed repair for φ. Similarly there is a single non-empty G-closed repair for ψ. But $\mathcal{M}$ has no G-closed repairs of $\varphi \wedge \psi$.

9 Repair via Unwinding

A disadvantage of substructure repair is that sometimes a structure $\mathcal{M}$ has no substructure repair, but a repair can be produced by "adding" states and transitions. We now investigate one way of adding states and transitions which we call "unwinding." We show that "unwinding" a Kripke structure, by replicating strongly bisimilar states, can result in a structure which is repairable while the original structure is not.

We take the usual notion of strong bisimulation: bisimilar states must agree on all atomic propositions and must have bisimilar successor states. We allow bisimilar states to have different shared variable values, since shared variables are not referenced in CTL formulae, i.e., they are a mechanism to implement synchronization between processes.

Definition 9.1 (Vertical Unwinding). Given a Kripke structure $\mathcal{M}$ with a designated ordering of states $[s_0, \ldots, s_n]$, the ith *vertical unwinding* of $\mathcal{M}$ is a Kripke structure $\mathcal{M}'$ constructed as follows:

- $\mathcal{M}'_S = \bigsqcup_i \mathcal{M}_S$ (the disjoint union of i copies of $\mathcal{M}_S$).
- $\mathcal{M}'_{S_0} = \mathbf{1} \cdot \mathcal{M}_{S_0}$, the injection of S_0 into the first of the copies of $\mathcal{M}_S$.
- $\mathcal{M}'_{AP} = \mathcal{M}_{AP}$.

— $\mathcal{M}'_L = \mathbf{i} \cdot s \mapsto \mathcal{M}_L(s)$.
— $\mathcal{M}'_{SH} = \mathcal{M}_{SH}$.
— $\mathcal{M}'_V = \mathbf{i} \cdot s \mapsto \mathcal{M}_V(s)$.
— $\mathcal{M}'_T$ is defined as follows: starting with s_0 use a depth first search traversal of the state-transition space to identify back edges, i.e., edges that would result in a cycle. Where at each node, the depth first search traversal follows the designated ordering of the states. Let T_{back} be the set of back edges. Then

$$\mathcal{M}'_T = \bigcup_{\mathbf{j}=\mathbf{1}}^{\mathbf{i}} \left[\left(\bigcup_{(s,t) \in \mathcal{M}'_T \setminus T_{back}} (\mathbf{j} \cdot s, \mathbf{j} \cdot t) \right) \cup \left(\bigcup_{(s,t) \in T_{back}} (\mathbf{j} \cdot s, \mathbf{j+1} \cdot t) \right) \right]$$

where we use the convention that $\mathbf{i+1} = \mathbf{1}$.

For multiprocess Kripke structures, which represent concurrent program computations, we add a shared variable that tracks which copy of the model the state belongs to. This is unnecessary for the general construction, but is needed to satisfy Definition 3.2 for multiprocess Kripke structures.

By the construction of vertical unwindings, we immediately have the following theorem.

Theorem 9.2. *For a Kripke structure $\mathcal{M}$ with a designated ordering of states $[s_0, \ldots, s_n]$ the ith vertical unwinding of $\mathcal{M}$ is bisimilar to $\mathcal{M}$.*

In the future, we will assume that, in the context of unwinding, all Kripke structures have an ordering of states.

Consider the repaired Kripke structure in Figure 9, i.e., the dashed transitions are deleted. We wish to "further repair" this structure so that it satisfies absence of individual starvation: $\mathsf{AG}(T_1 \Rightarrow \mathsf{AF}_1 C_1) \wedge \mathsf{AG}(T_2 \Rightarrow \mathsf{AF}_2 C_2)$, i.e., whenever a process enters its trying state, it subsequently enters its critical state. If we repair w.r.t. this formula (by itself), we can obtain unsatisfactory repairs that, for example, prevent entry into the trying state (and therefore satisfy absence of starvation vacuously), or that enforce alternate entry into the critical region, by preventing a process form remaining in neutral forever. To prevent such pathological repairs, we use a more complete specification as follows:

$$\mathsf{AG}(\neg(C_1 \wedge C_2)) \wedge \mathsf{AG}(T_1 \Rightarrow \mathsf{AF}_1 C_1) \wedge \mathsf{AG}(T_2 \Rightarrow \mathsf{AF}_2 C_2) \wedge$$
$$\mathsf{AG}(N_1 \Rightarrow \mathsf{EX}_1 T_1) \wedge \mathsf{AG}(N_2 \Rightarrow \mathsf{EX}_2 T_2) \ \wedge \mathsf{EG}(N_1) \wedge \mathsf{EG}(N_2).$$

In addition to mutual exclusion and absence of individual starvation, this formula also requires that: (1) a process in neutral is always able to move to trying, i.e., is always able to request the critical section: $\mathsf{AG}(N_1 \Rightarrow \mathsf{EX}_1 T_1) \wedge \mathsf{AG}(N_2 \Rightarrow \mathsf{EX}_2 T_2)$, and (2) a process can always remain in neutral forever $\mathsf{EG}(N_1) \wedge \mathsf{EG}(N_2)$. (1) presents vacuous satisfaction of absence of starvation, and (2) prevents forced alternation of the critical section, even if a process does not require it. The Kripke structure of Figure 9 cannot be repaired w.r.t. this formula, i.e., it cannot be repaired to ensure absence of individual starvation.

Now consider Figure 16, which is obtained from Figure 9 (with dashed transitions deleted) by *vertical unwinding*, which, roughly speaking, means that, along a path, when a transition would lead to a repeated state, instead a copy of the state is created and the transition goes to that copy. This results in a "single" unwinding, which creates a single copy of the original structure. Transitions in the copy that would go to a repeated state will now go back to the original structure. The unwound structure thus consists of two copies of the original, which we call copy 0 and copy 1. Transitions in each copy that would cause a repeated state along a path are now redirected to go to the duplicate of that state in the other copy. The two copies are distinguished by introducing a shared variable

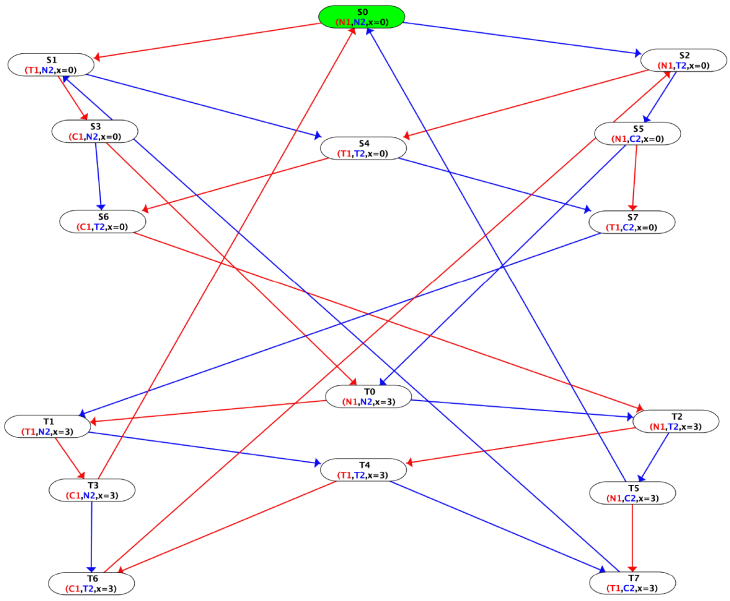

Fig. 16. Vertically unwound two-process mutual exclusion.

x which roughly corresponds to the injections from the disjoint union. However, due to system constraints from the software we used to generate our models, we use $x = 0$ in the "original" copy and $x = 3$ in the "unwound" copy. States in the original copy are named **S0**, **S1**, **S2**, ..., while states in the unwound copy are named **T0**, **T1**, **T2**, ...,

This unwound structure can be repaired w.r.t. the above CTL specification, as shown in Figure 17. The reason for this is that P_1 has priority in the original structure (copy 0) and P_2 has priority in the copied structure (copy 1).

We give the algorithm for computing the single vertical unwinding of a Kripke structure $\mathcal{M}$ in Figure 18. It is straightforward to generalize this notion to that of n-unwinding, in which transitions go from states in copy $i-1$ to states in copy i (instead of repeated states in copy i), for $0 \leq i < n$, and transitions from states in copy $n-1$ go to states in copy 0 (instead of repeated states in copy i).

9.1 Example Combining Symmetry Reduction and Unwinding

We now illustrate how symmetry reduction and unwinding can be combined. Consider the following example. The problem is to read in data from two different input sources into a single shared buffer. The input sources can be network connections, files on disk, etc. To improve throughput and fault-tolerance, each source is assigned two processes: processes P_1, P_2 read from the first source, and processes P_3, P_4 read from the second source. The resulting multiprocess Kripke structure and concurrent program will be symmetric in processes 1 and 2, and also in processes 3 and 4. So the group H generated by the permutations $\{(1,2),(3,4)\}$ is a symmetry group of $\mathcal{M}$.

We use the usual mutual exclusion terminology, where a process P_i $(1 \leq i \leq 4)$ is in N_i when it is performing local computation prior to reading data from its source, is in T_i when it is ready to read, and is in C_i when it is reading. We start with the following specification $\varphi_{progress}$, where i, j ranges over all pairs of processes, i.e., $i \neq j$ and $1 \leq i, j \leq 4$:

$$\mathsf{AG}(\wedge_{i \neq j} \neg (C_i \wedge C_j)) \wedge$$
$$\mathsf{AG}((T_1 \vee T_2 \vee T_3 \vee T_4) \Rightarrow \mathsf{AF}(C_1 \vee C_2 \vee C_3 \vee C_4)).$$

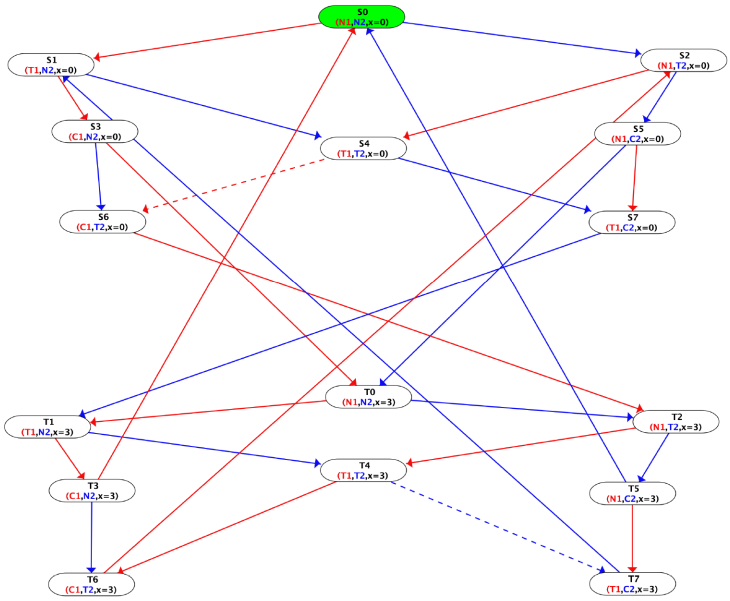

Fig. 17. Repair of vertically unwound two-process mutual exclusion with respect to no-individual-starvation.

SingleVerticalUnwinding($\mathcal{M}$, $[s_0, \ldots, s_n]$)

Make two copies of $\mathcal{M}_S$, labeled as $\mathbf{1} \cdot S$ and $\mathbf{2} \cdot S$
`BackEdges` = []
`Transitions` = []
Execute a depth first traversal of the state space of $\mathcal{M}$ starting from the state s_0.
(use the ordering of states as necessary)
    During the traversal, whenever a back edge $s \longrightarrow t$ is found add (s, t) to `BackEdges`:
For edges in $\mathcal{M}_T$:
    If edge (s, t) in `BackEdges`
        Add $(\mathbf{1} \cdot s, \mathbf{2} \cdot t)$ and $(\mathbf{2} \cdot s, \mathbf{1} \cdot t)$ to `Transitions`
        else: Add $(\mathbf{1} \cdot s, \mathbf{1} \cdot t)$ to `Transitions`.

Fig. 18. Algorithm for single vertical unwinding.

The first conjunction $\mathsf{AG}(\wedge_{i \neq j} \neg(C_i \wedge C_j))$ requires mutual exclusion among all four processes. The second conjunct $\mathsf{AG}((T_1 \vee T_2 \vee T_3 \vee T_4) \Rightarrow \mathsf{AF}(C_1 \vee C_2 \vee C_3 \vee C_4))$ requires *progress*, that is, if some process requests the critical section, then some process is granted the critical section. We observe that $\varphi_{progress}$ is H-invariant, since the maximal propositional subformulae $\wedge_{i \neq j} \neg(C_i \wedge C_j)$, $T_1 \vee T_2 \vee T_3 \vee T_4$, and $C_1 \vee C_2 \vee C_3 \vee C_4$ are unchanged by any permutation of the process indexes $1, \ldots, 4$.

Using symmetry reduction we can find a structure that satisfies $\varphi_{progress}$. This structure is a substructure of the quotient of the 81-state unrestricted model containing four processes with three states ($81 = 3^4$). We present this model in Figure 20. Figure 19 shows the same structure with back edges omitted, for clarity. Note that we have chosen orbit representatives in such a way that most forward edges are regular transitions, the only exceptions being the forward transitions into states **S14** and **S18**. Some of the back edges also represent irregular transitions.

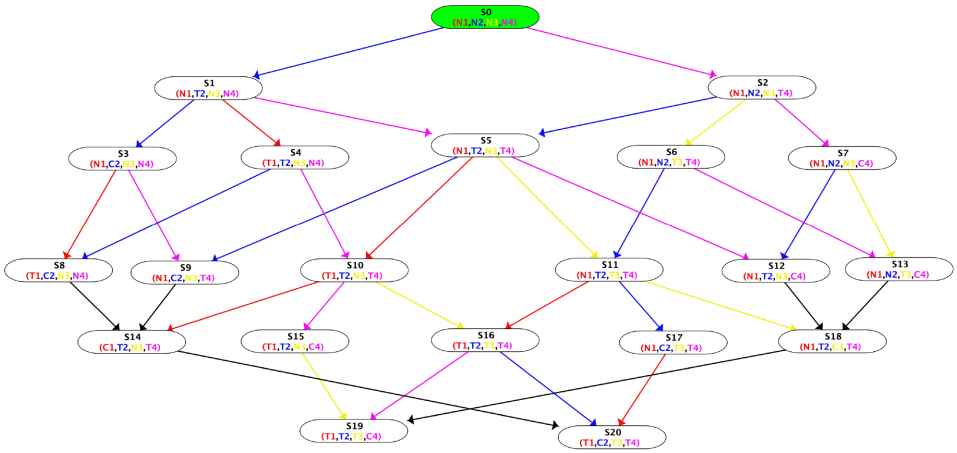

Fig. 19. Kripke structure for four process example with no back edges.

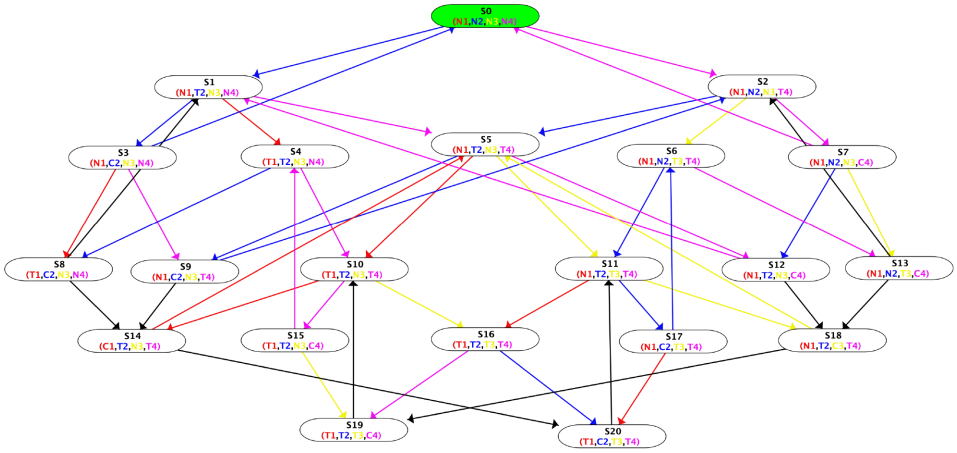

Fig. 20. Kripke structure for four process example.

We will now use unwinding to produce a Kripke structure that satisfies, in addition to $\varphi_{progress}$, the properties of absence of starvation (every process that requests the critical section is eventually granted the critical section) and fair alternation (successive entries into the contention state $[T_1, T_2, T_3, T_4]$ are resolved in alternating manner w.r.t. entry to the critical section by P_1, P_2 versus P_3, P_4).

The absence of starvation requirement $\varphi_{starvation}$ is as follows:

$$\begin{gathered}\mathsf{AG}((T_1 \vee T_2) \Rightarrow \mathsf{AF}(C_1 \vee C_2)) \wedge \mathsf{AG}((T_3 \vee T_4) \Rightarrow \mathsf{AF}(C_3 \vee C_4)) \wedge \\ \mathsf{AG}((N_1 \wedge N_2) \Rightarrow \mathsf{EX}(T_1 \vee T_2)) \wedge \mathsf{AG}((N_3 \wedge N_4) \Rightarrow \mathsf{EX}(T_3 \vee T_4)) \wedge \\ \mathsf{EG}(N_1 \wedge N_2) \wedge \mathsf{EG}(N_3 \wedge N_4).\end{gathered}$$

The first conjunct $\mathsf{AG}((T_1 \vee T_2) \Rightarrow \mathsf{AF}(C_1 \vee C_2)) \wedge \mathsf{AG}((T_3 \vee T_4) \Rightarrow \mathsf{AF}(C_3 \vee C_4))$ requires that if either of P_1, P_2 is trying to enter critical (i.e., to write into the buffer) then eventually one of them will do so. Likewise for P_3, P_4. The second conjunct $\mathsf{AG}((N_1 \wedge N_2) \Rightarrow \mathsf{EX}(T_1 \vee T_2)) \wedge \mathsf{AG}((N_3 \wedge N_4) \Rightarrow \mathsf{EX}(T_3 \vee T_4))$ requires that the pair P_1, P_2 are never prevented from requesting the critical section,

and neither is the pair P_3, P_4. The third conjunct $\mathsf{EG}(N_1 \wedge N_2) \wedge \mathsf{EG}(N_3 \wedge N_4)$ allows the pair P_1, P_2 to remain in neutral forever, i.e., to not request the critical section, and likewise for the pair P_3, P_4.

We also impose a *fair alternation* requirement. Consider the "full contention" state $[T_1, T_2, T_3, T_4]$. The transitions leaving this state must lead to one process being in the critical section. We require that successive entries into $[T_1, T_2, T_3, T_4]$ grant the critical section in alternating manner: if P_1 or P_2 last obtained critical, then the next time into $[T_1, T_2, T_3, T_4]$, either P_3 or P_4 will obtain critical. Also vice-versa, if P_3, P_4 last obtained critical, then next time P_1, P_2 obtain critical. A CTL formula $\varphi_{alternation}$ that specifies this alternation is as follows:

$$\begin{aligned}&\mathsf{EF}((T_1 \wedge T_2 \wedge T_3 \wedge T_4) \wedge \mathsf{EX}(T_1 \wedge T_2 \wedge T_3 \wedge C_4)) \wedge \\ &\mathsf{EF}((T_1 \wedge T_2 \wedge T_3 \wedge T_4) \wedge \mathsf{EX}(T_1 \wedge C_2 \wedge T_3 \wedge T_4)) \wedge \\ &\mathsf{AG}((T_1 \wedge T_2 \wedge T_3 \wedge T_4) \Rightarrow \mathsf{EX}(T_1 \wedge T_2 \wedge T_3 \wedge C_4) \oplus \mathsf{EX}(T_1 \wedge C_2 \wedge T_3 \wedge T_4)).\end{aligned}$$

This is constructed using the method given in the proof of Lemma 10.6 below. It states that (1) a transition from $[T_1, T_2, T_3, T_4]$ to $[T_1, T_2, T_3, C_4]$ is present, and (2) a transition from $[T_1, T_2, T_3, T_4]$ to $[T_1, C_2, T_3, T_4]$ is also present, and (3) any occurrence of the state $[T_1, T_2, T_3, T_4]$ is a source for exactly one of these transitions ($\oplus$ denotes exclusive or).

We proceed by analogy with the example given in Figures 16 and 17. We will construct a Kripke structure that satisfies $\varphi_{progress} \wedge \varphi_{starvation} \wedge \varphi_{alternation}$ by starting with the Kripke structure of Figure 20 (which satisfies φ) and unwinding it once, resulting in Figure 21. The shared variable x distinguishes between the two copies: $x = 0$ in the "original" copy and $x = 5$ in the unwound copy. We use 5 and not 1 since shared variable values that are equal to a process index have special meaning in the Eshmun model repair tool, which we used to produce the figures [3]. Eshmun enables the creation, editing, and repair of Kripke structures. For given Kripke structure $\mathcal{M}$ and CTL formula φ, Eshmun generates a Boolean formula $repair(\mathcal{M}, \varphi)$ and passes it to a SAT solver. A satisfying truth-assignment for $repair(\mathcal{M}, \varphi)$ defines a repair of $\mathcal{M}$ with respect to φ, i.e., a substructure $\mathcal{N}$ of $\mathcal{M}$ such that $\mathcal{N} \models \varphi$. As in the previous example, states in the original copy are named **S0**, **S1**, **S2**, ..., while states in the unwound copy are named **T0**, **T1**, **T2**, ..., Figure 22 gives a repair with respect to $\varphi_{progress} \wedge \varphi_{starvation} \wedge \varphi_{alternation}$. The repair was obtained using Eshmun. Note that the "alternation" required by $\varphi_{alternation}$ cannot be achieved in Figure 20. That is, Figure 20 is not repairable with respect to $\varphi_{progress} \wedge \varphi_{starvation} \wedge \varphi_{alternation}$.

10 Repairability, Unwinding, and Bisimulation

The above discussion naturally leads to the notions of repairability and equi-repairability.

A Kripke structure $\mathcal{M}$ is *repairable* w.r.t. a CTL formula φ iff there exists a substructure $\mathcal{N}$ of $\mathcal{M}$ such that $\mathcal{N} \models \varphi$. We use $\lhd$ for the repairability relation.

Definition 10.1 (Repairable). $\mathcal{M} \lhd \varphi$ iff there exists $\mathcal{N}$ such that $\mathcal{N} \leq \mathcal{M}$ and $\mathcal{N} \models \varphi$.

Two Kripke structures are *equi-repairable* iff they have the same repairability w.r.t. all CTL formulae.

Definition 10.2 (Equi-Repairable). $\mathcal{M} \equiv_{\lhd} \mathcal{M}'$ iff for all CTL formulae φ, $\mathcal{M} \lhd \varphi$ iff $\mathcal{M}' \lhd \varphi$.

It is clear that unwinding preserves bisimilarity: $\mathcal{M}$ is bisimilar to the n-unwinding of $\mathcal{M}$ for all n: all copies of a state in the n-unwinding are bisimilar to the original state in $\mathcal{M}$. Thus, unwinding provides a natural example of why bisimulation and repairability are distinct concepts.

Historically, (strong) bisimulation has been taken as being essentially equivalent: if $\mathcal{M}$ and $\mathcal{M}'$ are strongly bisimilar, then they are essentially equivalent, since they differ only by duplication of bisimilar states and by unwinding, as given above. We have shown that, with respect to repairability,

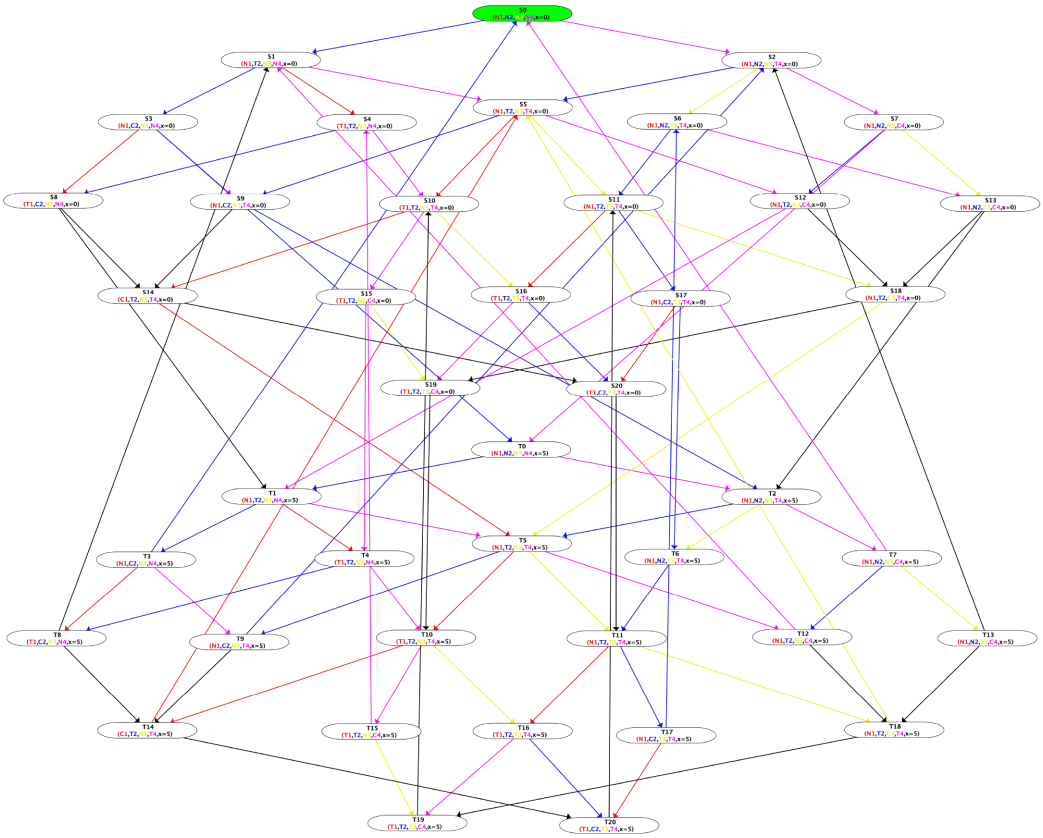

Fig. 21. One-unwinding of the Kripke structure for 4-2 Example.

i.e., the presence of substructures satisfying a particular CTL formula, that bisimilarity is not equivalence-preserving. This is, to our knowledge, a novel natural property of Kripke structures (apart from such "syntactic" properties as the number of states, transitions, etc.) that is not preserved by strong bisimulation. The question of repairability as a relation between structures arose from Figure 7 of van Glabbeeks paper in FoSSaCs [41] immediately preceding the conference version of the Correspondence Theorem (Theorem 5.16 [2]).

We now show that $\equiv_{\triangleleft}$ and $\sim$ are incomparable relations on Kripke structures. Lemma 10.4 shows that equi-repairable structures need not be bisimilar and Lemma 10.3 shows that bisimilar structures need not be equi-repairable.

Lemma 10.3. *Bisimilarity does not imply equi-repairability, that is, there exist $\mathcal{M}$ and $\mathcal{M}'$ such that $\mathcal{M} \sim \mathcal{M}'$ and $\mathcal{M} \not\equiv_{\triangleleft} \mathcal{M}'$.*

Lemma 10.4. *Equi-repairability does not imply bisimilarity of structures, that is, there exist $\mathcal{M}$ and $\mathcal{M}'$ such that $\mathcal{M} \equiv_{\triangleleft} \mathcal{M}'$ and $\mathcal{M} \not\sim \mathcal{M}'$.*

Figure 14 gives an example for Lemma 10.3 and Figure 23 gives an example for Lemma 10.4. The following is then immediate.

Theorem 10.5. *Equi-repairability and bisimilarity are incomparable relations over Kripke structures.*

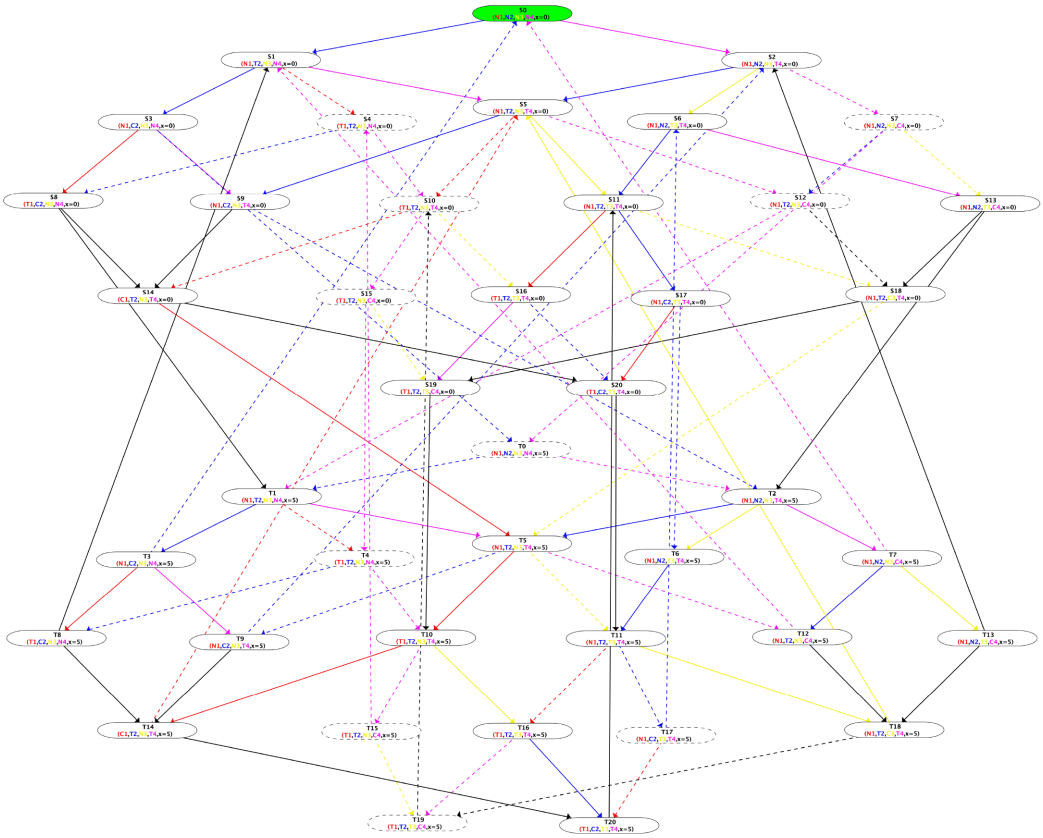

Fig. 22. Repair of the one-unwinding of the Kripke structure for 4-2 Example.

Fig. 23. Two Kripke structures that are equi-reparable but not bisimilar.

10.1 Unwinding Implies Non-Equi-Reparability

We used unwinding to establish Lemma 10.3 above. The proof required only the exhibition of a single example. We now generalize the proof to show that, under very mild restrictions, any one-unwound structure $\mathcal{M}'$ is not equi-reparable to the original structure $\mathcal{M}$: there always exists a formula φ such that $\mathcal{M}' \lhd \varphi$ and not $\mathcal{M} \lhd \varphi$.

Lemma 10.6. *Let $\mathcal{M}$ be a Kripke structure such that*

(1) *$\mathcal{M}$ contains a state s with at least two successors, each of which lies on a cycle containing s, and*
(2) *all states in $\mathcal{M}$ have different propositional labelings. Let $\mathcal{M}'$ be the one-unwinding of $\mathcal{M}$.*

Then $\mathcal{M}$ and $\mathcal{M}'$ are not equi-repairable.

We generalize this result as follows.

Theorem 10.7. *Let $\mathcal{M}$ be a Kripke structure such that all states in $\mathcal{M}$ have different propositional labelings. Let $\mathcal{M}^i$ be the i-unwinding of $\mathcal{M}$, and $\mathcal{M}^j$ the j-unwinding of $\mathcal{M}$, for $i > j \geq 1$. If $\mathcal{M}$ contains a state s with at least i successors, each of which lies on a cycle containing s, then $\mathcal{M}^i$ and $\mathcal{M}^j$ are not equi-repairable.*

11 Complexity Considerations

We now present some results regarding the reduction in the size of the Kripke structure that is achievable by taking the quotient w.r.t. a group G. We use the number of states as our measure of size. This is adequate since the number of transitions is at most quadratic in the number of states.

We first observe that the number of states in $\overline{\mathcal{M}} = \mathcal{M}/(G, \vartheta_G)$ is just the number of orbits of $\mathcal{M}$ under G. With general Kripke structures $\mathcal{M}$ and groups G, one cannot say much more than this. Fortunately, better results are available for multiprocess Kripke structures that are generated by the execution of symmetric concurrent programs. Hence, let $\mathcal{M} = GSTD(P)$, i.e., $\mathcal{M}$ be the global state-transition diagram of concurrent program $P = P_1 \| \cdots \| P_n$. In addition, G is a subgroup of S_n, the permutation group on process indices $\{1...n\}$.

The number of global states of P, i.e., states of $\mathcal{M}$, is bounded by the product of the state spaces of the processes P_i and the product of the size of the domains of all the shared variables. In specific examples, the number of reachable states may be much smaller than this. However, general results can be stated only in terms of the total number of states, as reachability is a function of the code of P.

Throughout this section, we use ℓ for the number of local states of a process P_i of concurrent program P, and n for the total number of processes in P. We assume without further repetition that $n \geq 2$ and $\ell \geq 2$.

Now consider the effect of shared variables. A shared variable x with finite domain D_x increases the number of global states by a factor of $|D_x|$, i.e., by the number of possible values of x. However, the number of states of $\overline{\mathcal{M}}$ remains the same, since each orbit is determined by the atomic propositions only. The values of x are drawn from a process-index based domain (Definition 6.1), and so any value of x can be taken to any other value by some element of S_n, the full permutation group over $\{1...n\}$.

11.1 Maximum Symmetry: G Is the Full Permutation Group

We consider the case $G = \mathsf{S}_n$, so all processes P_i in P are symmetric. Let P_i have ℓ local states. Then $\mathcal{M}$ has ℓ^n (global) states.

Theorem 11.1. *Let $G = \mathsf{S}_n$. Then the number of states in $\overline{\mathcal{M}} = \mathcal{M}/(G, \vartheta_G)$ is $\binom{n+\ell-1}{n}$.*

When the number ℓ of local states of a process is an increasing function of the total number of processes n, we have the following asymptotic result.

Corollary 11.2. *The number of states in $\overline{\mathcal{M}}$ is asymptotically equal to*

$$\sqrt{\frac{n+\ell-1}{2\pi n(\ell-1)}}\left(1+\frac{\ell-1}{n}\right)^n\left(1+\frac{n}{\ell-1}\right)^{\ell-1}.$$

We now show that when the number ℓ of local states of a process is constant, then symmetry reduction gives an exponential savings. This is the setting of parametrized model checking [6, 7], in which all processes P_i are instances of a fixed template (parametrized by the process index i), while

the number n of processes varies. In this case, we show below that the number of states in $\overline{M}$ is $\Theta(n^{\ell-1})$, and so gives an exponential savings over model checking/repairing M directly, which has ℓ^n states.

Theorem 11.3. *If ℓ is constant, then the number of states in $\overline{M}$ is $\Theta(n^{\ell-1})$.*

We also deal with the case when the number of local states ℓ is logarithmic in the number of processes n. This happens, e.g., in algorithms that require a process to store a constant number of id's of other processes, e.g., leader election in a ring [36, Chapter 3], distributed graph search and spanning tree construction [36, Chapter 15], and diffusing computations [16]. In this case, the number of states in $\overline{M}$ is super-polynomial in n, i.e., is not bounded from above by any polynomial in n. Let $\log n$ denote $\log_2(n)$, i.e., log with a base of 2.

Theorem 11.4. *If $\ell = \Theta(\log n)$, then the number of states in $\overline{M}$ is $O(n^{d \log n})$ for some positive constant d.*

Theorem 11.5. *If $\ell = \Theta(\log n)$, then the number of states in $\overline{M}$ is $\Omega(n^{b \log n})$ for some positive constant b.*

We finally deal with the case when the number of local states ℓ is linear in the number of processes n. This happens, e.g., in broadcast based algorithms, when a process receives messages from all other processes and records some information as a result [29, 38]. In this case, we show that the number of states in $\overline{M}$ is exponential in n.

Theorem 11.6. *If $\ell = \Theta(n)$, then the number of states in $\overline{M}$ is $O(d^n)$ for some positive constant $d > 1$.*

Theorem 11.7. *If $\ell = \Theta(n)$, then the number of states in $\overline{M}$ is $\Omega(b^n)$ for some positive constant $b > 1$.*

11.2 Non-Maximum Symmetry: G Is a Proper Subgroup of the Full Permutation Group

In this case, the processes $P_1, \ldots, P_n$ are partitioned into m sets $C_1, C_2, \ldots, C_m$ with full symmetry between the processes in each class, including shared variables. Let S_j^c be the full symmetry group over processes in C_j, for $j = 1, \ldots, m$. Then $G = S_1^c \times \cdots \times S_m^c$. We consider two cases for each C_j. Let n_j be the number of processes in C_j, and let ℓ_j be the number of local states of each process. Define $\mathcal{S}(j)$ as follows. If ℓ_j is constant, then $\mathcal{S}(j) \triangleq n_j^{\ell_j - 1}$, i.e., the number of states for n_j symmetric processes with constant number ℓ_j of local states each, as given by Theorem 11.3. If ℓ_j increases with n_j, then $\mathcal{S}(j) \triangleq \sqrt{\frac{n_j + \ell_j - 1}{2\pi n_j (\ell_j - 1)}} \left(1 + \frac{\ell_j - 1}{n_j}\right)^{n_j} \left(1 + \frac{n_j}{\ell_j - 1}\right)^{\ell_j - 1}$, i.e., the asymptotic number of states in $\overline{M}$ for n_j symmetric processes with ℓ_j local states each, as given by Corollary 11.2. Then, the subset C_j contributes $\mathcal{S}(j)$ states. Since the different C_j have no symmetry relations, the number of states in $\overline{M}$ is then the product of all these $\mathcal{S}(j)$, and we immediately obtain:

Lemma 11.8. *The number of states in $\overline{M}$ is $\Pi_{j=1}^m \mathcal{S}(j)$.*

Finally, we remark that, if in addition to the symmetric processes $P_1, \ldots, P_n$, there exist some processes $P_{n+1}, \ldots, P_{n'}$ which have no symmetry relation whatsoever, then, the number of states in $\overline{M}$ as given above for the $P_1, \ldots, P_n$ is multiplied by the product of the numbers of local states of each non-symmetric process $P_{n+1}, \ldots, P_{n'}$.

12 Conclusions

We presented a theory of how group actions can be used to reduce the complexity of repair of a Kripke structure. We also presented a theory for the substructures of a given Kripke structure $\mathcal{M}$, their organization into lattices, and how these substructures interact with a group of state mappings of $\mathcal{M}$. We show a lattice isomorphism between substructure repairs of $\overline{\mathcal{M}}$ and G-maximal repairs of $\mathcal{M}$ (Theorem 5.22: Repair Correspondence). This monotone Galois correspondence guarantees that a repair of $\overline{\mathcal{M}}$ lifts to a repair of $\mathcal{M}$: that is to say that substructure repairs of $\overline{\mathcal{M}}$ with respect to a G-invariant CTL formula φ lift to substructure repairs of $\mathcal{M}$ with respect to φ. Using this theory we devised a method for repairing concurrent programs which exploits this correspondence, thus avoiding state explosion. We construct the quotient structure $\overline{\mathcal{M}}$ directly from concurrent program $P = P_1 \parallel \cdots \parallel P_n$ without the need to construct the large structure $\mathcal{M}$. By our correspondence, repairing $\overline{\mathcal{M}}$ lifts to a repair of the structure $\mathcal{M}$, which in turn corresponds to a repair of P, since $\mathcal{M}$ is the structure that is generated by all the executions of P. Thus we repair P while circumventing the creation of its global state-transition diagram $\mathcal{M}$, which is in general exponentially large in the number of processes n. In case a repair of structure $\mathcal{M}$ does not exist, we showed how to obtain a structure $\mathcal{M}'$ by unwinding $\mathcal{M}$ such that $\mathcal{M}'$ is strongly bisimilar to $\mathcal{M}$ and a repair of $\mathcal{M}'$ does exist, in some cases.

We introduced the notion of equi-reparable and showed that it does not coincide with strong bisimulation, thereby providing an example of a natural semantic notion that is not implied by strong bisimulation. We also showed that an unwound structure is not equi-reparable to the original. Strongly bisimilar structures have historically been regarded as essentially equivalent, e.g., one can replace the other in any context, and they satisfy the same CTL* formulae, i.e., they agree on both linear and branching time formulae. It is therefore notable to identify a meaningful semantic relation over Kripke structures which is not implied by strong bisimulation.

Finally, we presented closed-form results for the reduction in size from $\mathcal{M}$ to $\overline{\mathcal{M}} = \mathcal{M}/(G, \vartheta_G)$, where $\mathcal{M}$ is the global state-transition diagram of a concurrent program $P = P_1 \parallel \cdots \parallel P_n$. These are, to our knowledge, the first closed-form results for the size reduction effected by group-based symmetry. Our closed-form results show a reduction in size from exponential ($\mathcal{M}$) to polynomial ($\overline{\mathcal{M}}$) when G is the full permutation group S_n on $\{1...n\}$, and each process P_i has a constant number of states, i.e., the setting of parametrized model checking [6, 7]. When each process has a number of local states that is logarithmic in n, the total number of processes, then $\overline{\mathcal{M}}$ has super-polynomial size: $n^{d \log n}$ for a constant d. When each process has a number of local states that is linear in n, then $\overline{\mathcal{M}}$ is still smaller than $\mathcal{M}$ but remains exponentially large in n. This shows that, in most cases (in practice) in which the number of local states of a process increases with n, then even full symmetry does not guarantee a reduction in size to polynomial in n. When G is a proper subgroup of S_n, the above results then apply to each subset of processes that are fully symmetric, in isolation, and the size of $\overline{\mathcal{M}}$ is then the product of the "size contributions" of each subset.

Acknowledgment

We thank the reviewers for their suggestions and comments. The views expressed are those of the authors and do not reflect the official policy or position of the Department of the Army, the Department of Defense, or the US Government.

References

[1] Benjamin Aminof, Swen Jacobs, Ayrat Khalimov, and Sasha Rubin. 2014. Parameterized model checking of token-passing systems. In *Proceedings of the 15th International Conference on Verification, Model Checking, and Abstract*

Interpretation (VMCAI '14). Kenneth L. McMillan and Xavier Rival (Eds.), *Lecture Notes in Computer Science*, *Vol.* 8318, Springer, 262–281. DOI: https://doi.org/10.1007/978-3-642-54013-4_15

[2] Paul C. Attie and William L. Cocke. 2023. Model and program repair via group actions. In *Foundations of Software Science and Computation Structures*. Orna Kupferman and Pawel Sobocinski (Eds.), Springer Nature, Cham, 520–540.

[3] Paul C. Attie, Kinan Dak-Al-Bab, and Mouhammad Sakr. 2018. Model and program repair via SAT solving. *ACM Trans. Embed. Comput. Syst*. 17, 2 (2018), 32:1–32:25. DOI: https://doi.org/10.1145/3147426

[4] Paul C. Attie and E. Allen Emerson. 1998. Synthesis of concurrent systems with many similar processes. *ACM Trans. Program. Lang. Syst*. 20, 1 (Jan. 1998), 51–115. DOI: https://doi.org/10.1145/271510.271519

[5] Paul C. Attie and E. Allen Emerson. 2001. Synthesis of concurrent programs for an atomic read/write model of computation. *ACM Trans. Program. Lang. Syst*. 23, 2 (Mar. 2001), 187–242. DOI: https://doi.org/10.1145/383043.383044

[6] Roderick Bloem, Swen Jacobs, Ayrat Khalimov, Igor Konnov, Sasha Rubin, Helmut Veith, and Josef Widder. 2015. *Decidability of Parameterized Verification*. Morgan and Claypool Publishers, San Rafael, CA.

[7] Roderick Bloem, Swen Jacobs, Ayrat Khalimov, Igor Konnov, Sasha Rubin, Helmut Veith, and Josef Widder. 2016. Decidability in parameterized verification. *ACM SIGACT News* 47, 2 (2016), 53–64.

[8] Miklós Bóna. 2002. *A Walk Through Combinatorics: An Introduction to Enumeration and Graph Theory*. World Scientific.

[9] Michael C. Browne, Edmund M. Clarke, and Orna Grümberg. 1988. Characterizing finite Kripke structures in propositional temporal logic. *Theor. Comput. Sci*. 59, 1–2 (1988), 115–131.

[10] Francesco Buccafurri, Thomas Eiter, Georg Gottlob, and Nicola Leone. 1999. Enhancing model checking in verification by AI techniques. *Artif. Intell*. 112 (1999), 57–104.

[11] George Chatzieleftheriou, Borzoo Bonakdarpour, Scott A. Smolka, and Panagiotis Katsaros. 2012. Abstract model repair. In *NASA Formal Methods*. Alwyn E.Goodloe and SuzettePerson (Eds.), Lecture Notes in Computer Science, Vol. 7226. Springer, Berlin, 341–355.

[12] Edmund M. Clarke, E. Allen Emerson, Somesh Jha, and A. Prasad Sistla. 1998. Symmetry reductions in model checking. In *Proceedings of the Computer Aided Verification (CAV)*. Lecture Notes in Computer Science, Vol. 1427, Springer, 147–158.

[13] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst*. 8, 2 (1986), 244–263.

[14] Edmund M. Clarke, Orna Grumberg, and Somesh Jha. 1997. Verifying parameterized networks. *ACM Trans. Program. Lang. Syst*. 19, 5 (1997), 726–750. DOI: https://doi.org/10.1145/265943.265960

[15] Edmund M. Clarke, Somesh Jha, Reinhard Enders, and Thomas Filkorn. 1996. Exploiting symmetry in temporal logic model checking. *Formal Methods Syst. Des*. 9, 1/2 (1996), 77–104. DOI: https://doi.org/10.1007/BF00625969

[16] Edsger W. Dijkstra and C. S. Scholten. 1980. Termination detection for diffusing computations. *Inform. Process. Lett*. 11, 1 (1980), 1–4. DOI: https://doi.org/10.1016/0020-0190(80)90021-6

[17] Alastair F. Donaldson and Alice Miller. 2005. Automatic symmetry detection for model checking using computational group theory. In *Proceedings of the International Symposium on Formal Methods*. Springer, 481–496.

[18] E. Allen Emerson. 1990. Temporal and Modal Logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*. Elsevier and MIT Press, Cambridge, MA, 995–1072.

[19] E. Allen Emerson and Edmund M. Clarke. 1982. Using branching time temporal logic to synthesize synchronization skeletons. *Sci. Comput. Program*. 2, 3 (1982), 241–266.

[20] E. Allen Emerson, John Havlicek, and Richard J. Trefler. 2000. Virtual symmetry reduction. In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society, 121–131.

[21] E. Allen Emerson and Vineet Kahlon. 2004. Parameterized model checking of ring-based message passing systems. In *Proceedings of the International Workshop on Computer Science Logic*. Lecture Notes in Computer Science, Vol. 3210, Springer, 325–339.

[22] E. Allen Emerson and Kedar S. Namjoshi. 1995. Reasoning about rings. In *Proceedings of the on Principles of Programming Languages (POPL)*. ACM, 85–94.

[23] E. Allen Emerson and Kedar S. Namjoshi. 2003. On reasoning about rings. *Int. J. Found. Comput. Sci*. 14, 4 (2003), 527–550.

[24] E. Allen Emerson and A. Prasad Sistla. 1996. Symmetry and model checking. *Formal Methods Syst. Des*. 9, 1/2 (1996), 105–131.

[25] E. Allen Emerson and A. Prasad Sistla. 1997. Utilizing symmetry when model-checking under fairness assumptions: An automata-theoretic approach. *ACM Trans. Program. Lang. Syst*. 19, 4 (1997), 617–638.

[26] E. Allen Emerson and Richard J. Trefler. 1998. Model checking real-time properties of symmetric systems. In *Proceedings of the Mathematical Foundations of Computer Science (MFCS)*. Lecture Notes in Computer Science, Vol. 1450, Springer, 427–436.

[27] E. Allen Emerson and Richard J. Trefler. 1999. From asymmetry to full symmetry: New techniques for symmetry reduction in model checking. In *Proceedings of the Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME)*. Lecture Notes in Computer Science, Vol. 1703, Springer, 142–156.
[28] E. Allen Emerson and Thomas Wahl. 2005. Dynamic symmetry reduction. In *Proceedings of the Tools and Algorithms for the Construction and Analysis (TACAS)*. Lecture Notes in Computer Science, Vol. 3440, Springer, 382–396.
[29] Alan Fekete, David Gupta, Victor Luchangco, Nancy Lynch, and Alex Shvartsman. 1996. Eventually-serializable data services. In *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing (PODC '96)*. ACM, New York, NY, 300–309. DOI: https://doi.org/10.1145/248052.248113
[30] Steven M. German and A. Prasad Sistla. 1992. Reasoning about systems with many processes. *J. ACM* 39, 3 (1992), 675–735. DOI: https://doi.org/10.1145/146637.146681
[31] Claire Le Goues, Michael Pradel, and Abhik Roychoudhury. 2019. Automated program repair. *Commun. ACM* 62, 12 (nov 2019), 56–65. DOI: https://doi.org/10.1145/3318162
[32] G. H. Hardy. 1992. *A Course of Pure Mathematics*. Cambridge Mathematical Library, Cambridge.
[33] I Martin Isaacs. 2008. *Finite Group Theory*, Vol. 92. American Mathematical Society, Providence, RI.
[34] I Martin Isaacs. 2009. *Algebra: A Graduate Course*, Vol. 100. American Mathematical Society , Providence, RI.
[35] B. Jobstmann, A. Griesmayer, and R. Bloem. 2005. Program repair as a game. In *Proceedings of the Computer Aided Verification (CAV)*. Springer-Verlag, Berlin, 226–238.
[36] Nancy Lynch. 1996. *Distributed Algorithms*. Morgan Kaufmann, San Francisco, CA.
[37] Kedar S. Namjoshi and Richard J. Trefler. 2013. Uncovering symmetries in irregular process networks. In *Proceedings of the International Workshop on Verification, Model Checking, and Abstract Interpretation*. Springer, 496–514.
[38] Glenn Ricart and Ashok K. Agrawala. 1981. An optimal algorithm for mutual exclusion in computer networks. *Commun. ACM* 24, 1 (Jan. 1981), 9–17. DOI: https://doi.org/10.1145/358527.358537
[39] Jean-Pierre Serre. 2016. *Finite Groups: An Introduction*. International Press, Somerville, MA.
[40] S. Staber, B. Jobstmann, and R. Bloem. 2005. Finding and fixing faults. In *Proceedings of the Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME '05)*. LNCS, Vol. 3725, Springer-Verlag, Berlin, 35–49.
[41] Rob van Glabbeek. 2023. Just testing. In *Proceedings of the Foundations of Software Science and Computation Structures (FoSSaCS)*, 498–519.
[42] Christian von Essen and Barbara Jobstmann. 2015. Program repair without regret. *Formal Methods Syst. Des.* 47, 1 (2015), 26–50. DOI: https://doi.org/10.1007/s10703-015-0223-6

Appendices

A Proofs of Lemmata and Theorems

Lemma 4.2. *Let $\mathcal{M}$ be a Kripke structure and suppose that $\mathcal{N}$ and $\mathcal{N}'$ are substructures of $\mathcal{M}$. Then*

$$\mathcal{N} \vee \mathcal{N}' = (\mathcal{N}_S \cup \mathcal{N}'_S,\ \mathcal{N}_{S_0} \cup \mathcal{N}'_{S_0},\ \mathcal{N}_T \cup \mathcal{N}'_T,\ \mathcal{M}_{AP},\ \mathcal{M}_L\restriction(\mathcal{N}_S \cup \mathcal{N}'_S),\ \mathcal{M}_{SH},\ \mathcal{M}_V\restriction(\mathcal{N}_S \cup \mathcal{N}'_S))$$

is the smallest substructure of $\mathcal{M}$ containing both $\mathcal{N}$ and $\mathcal{N}'$.

Proof. We will write $\mathcal{N}_\vee$ for $\mathcal{N} \vee \mathcal{N}'$. By construction $\mathcal{N}_\vee$ contains both $\mathcal{N}$ and $\mathcal{N}'$. Let $\mathcal{T}$ be any substructure of $\mathcal{M}$ containing both $\mathcal{N}$ and $\mathcal{N}'$. Clearly $(\mathcal{N}_\vee)_S \subseteq \mathcal{T}_S$, $(\mathcal{N}_\vee)_{S_0} \subseteq \mathcal{T}_{S_0}$, $(\mathcal{N}_\vee)_R \subseteq \mathcal{T}_R$, $\mathcal{T}_L$ extends $(\mathcal{N}_\vee)_L$, and $\mathcal{T}_V$ extends $(\mathcal{N}_\vee)_V$.

It remains to show that $\mathcal{N}_\vee$ is a substructure, i.e., that $\mathcal{N}_\vee$ is total. Given a state s_1 in $(\mathcal{N}_\vee)_S$, we know that either $s \in \mathcal{N}_S$ or $s \in \mathcal{N}'_S$. If $s \in \mathcal{N}_S$, then there is a $t \in \mathcal{N}_S$ such that $(s,t) \in \mathcal{N}_T \subseteq (\mathcal{N}_\vee)_T$. By symmetry if $s \in \mathcal{N}'_S$ then there is a $t \in \mathcal{N}'_S$ with $(s,t) \in \mathcal{N}'_T \subseteq (\mathcal{N}_\vee)_T$. □

Lemma 4.3. *Let $\mathcal{M}$ be a Kripke structure and suppose that $\mathcal{N}$ and $\mathcal{N}'$ are substructures of $\mathcal{M}$. Then there is a largest substructure of $\mathcal{M}$ contained in both $\mathcal{N}$ and $\mathcal{N}'$. [This structure may be empty.]*

Proof. The empty structure is contained in both $\mathcal{N}$ and $\mathcal{N}'$. Let X be the collection of structures contained in both $\mathcal{N}$ and $\mathcal{N}'$. Since $\mathcal{M}_S$ is finite, the set X is finite. Then $\bigvee X$ is contained in both $\mathcal{N}$ and $\mathcal{N}'$ and is the largest substructure of $\mathcal{M}$ contained in both $\mathcal{N}$ and $\mathcal{N}'$. □

Lemma 5.3. *Let $\mathcal{M}$ be a Kripke structure and let G be a group of state mappings of $\mathcal{M}$. Let $\mathcal{N}, \mathcal{N}'$ be two G-closed substructures of $\mathcal{M}$. Then $\mathcal{N} \vee \mathcal{N}'$ and $\mathcal{N} \wedge \mathcal{N}'$ are both G-closed.*

Proof. We first show that $\mathcal{N} \vee \mathcal{N}'$ is G-closed. Since $(\mathcal{N} \vee \mathcal{N}')_S = \mathcal{N}_S \cup \mathcal{N}'_S$, the group G acts on the states of $\mathcal{N} \vee \mathcal{N}'$. We will show that every element of G is a state mapping on $\mathcal{N} \vee \mathcal{N}'$.

Let $s, t \in (\mathcal{N} \vee \mathcal{N}')_S$. If $g(s) = g(t)$ then $s = t$ since g is a bijection on $\mathcal{M}_S$. Similarly, for any $s \in (\mathcal{N} \vee \mathcal{N}')_S$ we have that $g^{-1}(s) \in (\mathcal{N} \vee \mathcal{N}')_S$.

If $(s, t) \in (\mathcal{N} \vee \mathcal{N}')_T$, then (s, t) is in one of $\mathcal{N}_T$ or $\mathcal{N}'_T$. In either case, $(g(s), g(t))$ is contained in the same set of transitions, and thus contained in $(\mathcal{N} \vee \mathcal{N}')_T$.

We now consider $\mathcal{N} \wedge \mathcal{N}'$. Recall that $\mathcal{N} \wedge \mathcal{N}'$ is the largest substructure contained in both $\mathcal{N}$ and $\mathcal{N}'$. If $s \in (\mathcal{N} \wedge \mathcal{N}')_S$, and $g \in G$, then there is a substructure of $\mathcal{N}$ and $\mathcal{N}'$ containing $g(s)$. Moreover as the largest substructure of $\mathcal{N}$ and $\mathcal{N}'$, $\mathcal{N} \wedge \mathcal{N}'$ must contain this substructure, hence $g(s) \in \mathcal{N} \wedge \mathcal{N}'$. Since G is a group of state mappings of $\mathcal{M}$, we conclude that every $g \in G$ acts as a bijection on $\mathcal{N} \wedge \mathcal{N}'$ and thus G is a group of state mappings of $\mathcal{N} \wedge \mathcal{N}'$.

Consider $(s, t) \in (\mathcal{N} \wedge \mathcal{N}')_T$. Since G is a group of state mappings of $\mathcal{N} \wedge \mathcal{N}'$ there is a substructure of $\mathcal{N}$ and $\mathcal{N}'$ containing $(g(s), g(t))$. Since $\mathcal{N} \wedge \mathcal{N}'$ is the join of all substructures contained in $\mathcal{N}$ and $\mathcal{N}'$, we conclude that it must contain $(g(s), g(t))$ as well. □

Theorem 5.10. *There is a bidirectional correspondence between paths of $\mathcal{M}$ and paths of $\overline{\mathcal{M}}$. Formally we have the following:*

(1) *If $x = s_0, s_1, s_2, \ldots$ is a path in $\mathcal{M}$, then $\overline{x} = \overline{s_0}, \overline{s_1}, \overline{s_2}, \ldots$ is a path in $\overline{\mathcal{M}}$ where $\overline{s_i} = \vartheta_G(s_i)$.*
(2) *If $\overline{x} = \overline{s_0}, \overline{s_1}, \overline{s_2}, \ldots$ is a path in $\overline{\mathcal{M}}$, then for every state $s'_0 \in \mathcal{M}_S$ such that $\vartheta_G(s'_0) = \overline{s_0}$ there is a path $s'_0, s'_1, s'_2, \ldots$ in $\mathcal{M}$ such that $\vartheta_G(s'_i) = \overline{s_i}$.*

Proof.

1. Given $(s_i, s_{i+1}) \in \mathcal{M}_T$, then $(\overline{s_i}, \overline{s_{i+1}}) \in \overline{\mathcal{M}}_T$.
2. Suppose that $(\overline{s_i}, \overline{s_{i+1}}) \in \overline{\mathcal{M}}_T$. Then there is some $(t, t') \in \mathcal{M}_T$ with $t \in \overline{s_i}$ and $t' \in \overline{s_{i+1}}$. Suppose that we have a path $s'_0, \ldots, s'_i$ such that $s'_0 \in \overline{s_0}$, $s'_1 \in \overline{s_1}, \ldots, s'_i \in \overline{s_i}$ and $(s'_0, s'_1), (s'_1, s'_2), \ldots, (s'_{i-1}, s'_i) \in \mathcal{M}_T$. Since both $t, s'_i \in \overline{s_i}$, there is a group element $g \in G$ such that $g(t) = s'_i$. Then $g(s'_0), g(s'_1), \ldots, g(s'_i) = t, s_{i+1}$ is a partial path in $\mathcal{M}$. Continuing in this manner we see the existence of the desired path in $\mathcal{M}$. □

Lemma 5.11. *For every substructure $\overline{\mathcal{N}}$ of $\overline{\mathcal{M}}$, there is a G-closed substructure $\mathcal{N}$ of $\mathcal{M}$ such that $\mathcal{N}/(G, \vartheta_G) = \overline{\mathcal{N}}$.*

Proof. We will construct a G-closed substructure $\mathcal{N}'$ of $\mathcal{M}$ such that $\mathcal{N}'/(G, \vartheta_G) = \overline{\mathcal{N}}$. For the states of $\mathcal{N}'$, we select exactly the G-orbits of the representative states of $\overline{\mathcal{N}}$, i.e., $\mathcal{N}'_S = \{g(s) : \forall g \in G, s \in \overline{\mathcal{N}}_S\}$. For transitions, we note that for every transition $(s, t) \in \overline{\mathcal{N}}_T$ there is at least one transition $(s', t') \in \mathcal{M}_T$ such that $\vartheta(s') = s$ and $\vartheta(t') = t$. We add all such transitions and their orbits under G to $\mathcal{N}'$. Hence, for any state $s' \in \mathcal{N}'_S$ there is a transition (s', t') for some $t' \in \mathcal{N}'_S$. We conclude that $\mathcal{N}'$ is total. Since $\mathcal{N}'$ is determined by its states and transitions as a substructure, the lemma is proved. □

Lemma 5.12. *Let $\mathcal{N}, \mathcal{N}' \in \Lambda_{\mathcal{M},G}$. Then*

$$(\mathcal{N} \vee \mathcal{N}')/(G, \vartheta_G) = \mathcal{N}/(G, \vartheta_G) \vee \mathcal{N}'/(G, \vartheta_G).$$

Proof. Clearly, by definition a state $\vartheta_G(s) \in (\mathcal{N}/(G,\vartheta_G) \vee \mathcal{N}'/(G,\vartheta_G))_S$ if and only if it is in at least one of $(\mathcal{N}/(G,\vartheta_G))_S$ or $(\mathcal{N}'/(G,\vartheta_G))_S$ which occurs if and only if $\vartheta_G(s)$ is in one of the G-closed substructures $\mathcal{N}$ or $\mathcal{N}'$.

For transitions, we note that a transition in $(\mathcal{N}/(G,\vartheta_G) \vee \mathcal{N}'/(G,\vartheta_G))_T$ is entirely contained in one of $(\mathcal{N}/(G,\vartheta_G))_T$ or $(\mathcal{N}'/(G,\vartheta_G))_T$. Hence the transition must come from either $\mathcal{N}$ or $\mathcal{N}'$. □

Lemma 5.14. *Let $\mathcal{M}'$, $\mathcal{M}''$ be two G-maximal substructures of $\mathcal{M}$. Then $\mathcal{M}' \vee \mathcal{M}''$ is G-maximal and $\mathcal{M}' \wedge \mathcal{M}''$ is G-maximal.*

Proof. Consider a G-closed substructure $\mathcal{N}'$ such that $\mathcal{N}'/(G,\vartheta_G) \leq (\mathcal{M}' \vee \mathcal{M}'')/(G,\vartheta_G)$. Let $(s,t) \in \mathcal{N}'_T$. Since $(\vartheta_G(s),\vartheta_G(t)) \in (\mathcal{M}' \vee \mathcal{M}'')/(G,\vartheta_G)_T$ we conclude that $(\vartheta_G(s),\vartheta_G(t))$ is in either $\mathcal{M}'/(G,\vartheta_G)$ or $\mathcal{M}''/(G,\vartheta_G)$. In either case, since $\mathcal{M}'$ and $\mathcal{M}''$ are both G-maximal the transitions $(g(s),g(t))$ are already contained in $\mathcal{M}'$ or $\mathcal{M}''$.

Suppose that $\mathcal{N}$ is the G-maximal substructure of $\mathcal{M}$ corresponding to $(\mathcal{M}' \wedge \mathcal{M}'')/(G,\vartheta_G)$. We note that $\mathcal{N}/(G,\vartheta_G) \leq \mathcal{M}'/(G,\vartheta_G)$. Hence $\mathcal{N} \leq \mathcal{M}'$. By symmetry $\mathcal{N} \leq \mathcal{M}''$ and thus $\mathcal{N} \leq \mathcal{M}' \wedge \mathcal{M}''$. □

Theorem 5.16. *The restriction of the quotient map Ψ to $\Lambda_{\mathcal{M},G-max}$ is an isomorphism from $\Lambda_{\mathcal{M},G-max}$ to $\Lambda_{\overline{\mathcal{M}}}$, i.e., between the lattice of G-maximal substructures of $\mathcal{M}$ and the lattice of structures of $\overline{\mathcal{M}}$.*

Proof. As noted, the G-maximal lattice of substructures is in bijection with the lattice of substructures of $\mathcal{M}/(G,\vartheta_G)$. Since the meet and join of two G-maximal structures are determined entirely by the meet and join of their image in $M/(G,\vartheta_G)$, the correspondence is a lattice isomorphism. □

Lemma 5.18. *If φ is G-invariant, then the valuation of φ in $\overline{\mathcal{M}}$ does not depend on the choice of representative map ϑ_G.*

Proof. Let ϑ_G, φ_G be two representative maps with respect to G. The quotients $\mathcal{M}/(G,\vartheta_G)$ and $\mathcal{M}/(G,\varphi_G)$ have corresponding transition maps. Moreover, for any maximal propositional subformula, P of φ we have that

$$\mathcal{M}/(G,\vartheta_G), \vartheta_G(s) \models P \iff \mathcal{M}/(G,\varphi_G), \varphi_G(s) \models P.$$

The valuation of φ depends only on the valuation of its maximal propositional subformulae at all other states of the model. Since ϑ and φ have the same evaluation of the maximal propositional subformulae at all states, the evaluations of ϑ and φ at all states is the same. □

Lemma 5.20. *Let $s \in \mathcal{M}_S$, $t \in \overline{\mathcal{M}}_S$. Let φ be a G-invariant CTL formula. If $t = \vartheta_G(s)$, then $\mathcal{M}, s \models \varphi \iff \overline{\mathcal{M}}, t \models \varphi$.*

Proof. We observe that $s \sim t$ iff $t = \vartheta_G(s)$ is a G-bisimulation, since the required transition matching follows immediately from Theorem 5.10 (Path Correspondence), applied to single transitions.

Since φ is G-invariant over $\mathcal{M}$, then for all $s \in \mathcal{M}_S$, $t \in \overline{\mathcal{M}}_S$, $s \sim t$ we have that $\mathcal{M}, s \models P$ iff $\overline{\mathcal{M}}, t \models P$, where P is any maximal propositional subformula of φ. That is, s and t agree on all maximal propositional subformulae of φ. This provides the base case for the induction, where φ is an atomic proposition.

The rest of the proof is then by induction on the structure of CTL formulae, as given by the grammar in Section 3.2.

The induction steps when φ has main operator one of $\neg, \wedge, \vee$ are straightforward, and omitted. The induction steps when φ is one of $\mathsf{AX}\psi$, $\mathsf{EX}\psi$ and $\mathsf{A}[\psi\,\mathsf{R}\,\chi]$ use the bisimulation $\sim$ to establish the existence of the necessary transitions/paths. The details are standard, see for example Theorem 3.2 in [9]. □

Theorem 5.22. *Let φ be a G-invariant CTL formula. Let $\mathcal{N}$ be a non-empty G-closed substructure of $\mathcal{M}$, $s \in \mathcal{N}_S$, and $\overline{\mathcal{N}} = \mathcal{N}/(G, \vartheta_G)$. Then $\mathcal{N}, s \models \varphi \iff \overline{\mathcal{N}}, \vartheta_G(s) \models \varphi$.*

Proof. We observe that $\mathcal{N}$ and $\overline{\mathcal{N}}$ are G-bisimilar under the usual relation of each state s in $\mathcal{N}_S$ with $\vartheta_G(s)$ in $\overline{\mathcal{N}}_S$. The required transition matching follows immediately from Theorem 5.10 (Path Correspondence), applied to single transitions.

The conclusion follows by induction on the structure of CTL formula, in an identical manner to the proof of Lemma 5.20 above. □

Lemma 6.5. *For each irregular transition $s \to t \in \overline{\mathcal{N}}_T$, there is $g' \in G$ such that $s \to g'(t)$ is regular.*

Proof. Such an element g' always exists. Let $\bar{s} \to \bar{t} \in \overline{\mathcal{M}}_T$ for arbitrary $\overline{\mathcal{M}}_T$. By Definition 5.7, there exists $s \to t \in \mathcal{M}_T$ such that $\bar{s} = \vartheta_G(s)$ and $\bar{t} = \vartheta_G(t)$. Hence there is some $g \in G$ such that $g(s) = \bar{s}$ since s and $\bar{s}$ are in the same orbit. Since g is a symmetry of $\mathcal{M}$, we have $g(s) \to g(t) \in \mathcal{M}_T$. Hence $\bar{s} \to g(t) \in \mathcal{M}_T$. Now $t = h(\bar{t})$ for some $h \in G$ since t and $\bar{t}$ are in the same orbit. Hence $\bar{s} \to g(h(\bar{t})) \in \mathcal{M}_T$, and so the needed g' is the product of g and h. □

Theorem 6.11. *Let $\overline{P}^G$ be a concurrent program extracted from $\overline{\mathcal{N}}$ as above, and let $\mathcal{N}^p$ be the state-transition graph generated by the execution of $\overline{P}^G$. Then $\mathcal{N}^p$ is G-closed and $\overline{\mathcal{N}} = \overline{\mathcal{N}^p}$.*

Proof. In the proof, g, g' denote arbitrary elements of G, i, j denote arbitrary process indices in $\{1...n\}$, s, t denote primed variants global states, and subscripted (and possibly primed) variants of s, t denote local states. We use, without further justification, the identities $g(s\restriction i) = g(s)\restriction g(i)$ and $g(s{\downarrow}i) = g(s){\downarrow}g(i)$.

We first observe that $\mathcal{N}^p$ is G-closed by construction, since each process $\overline{P}^G_i$ consists of the i-actions in $Act^G_i(\overline{\mathcal{N}}_T)$, which by construction is the closure under G of the set of actions $Act_j(\overline{\mathcal{N}}_T)$, as j varies over $\{1...n\}$. We define the quotient $\overline{\mathcal{N}^p}$ to be the quotient of $\mathcal{N}^p$ according to Definition 5.7 and by *using the same representation map ϑ_G as the substructure $\overline{\mathcal{N}}$ of $\overline{\mathcal{M}}$ does* i.e., the map that $\overline{\mathcal{N}}$ "inherits" from the quotienting of $\mathcal{M}$ to produce $\overline{\mathcal{M}}$. This map is implicitly generated by applying the Emerson-Sistla algorithm [24, Figure 1] to the original concurrent program P that is to be repaired.

In the rest of the proof, we implicitly use Definitions 6.4–6.9.

Let $\bar{s} \to \bar{t}$ be an arbitrary transition in $\overline{\mathcal{N}^p_T}$. By Definition 5.7:

there exists $s \xrightarrow{i} t \in \mathcal{N}^p_T$ such that $\vartheta_G(s) = \bar{s}, \vartheta_G(t) = \bar{t}$.

By operational semantics of concurrent programs,

$s \xrightarrow{i} t \in \mathcal{N}^p_T$ iff $(s_i, B_i \to A_i, t_i) \in \overline{P}^G_i$
with $s\restriction i = s_i$, $t\restriction i = t_i$, $B_i = \{|s|\}$, and $\{|s\restriction \mathcal{S}H|\}\ A_i\ \{|t\restriction \mathcal{S}H|\}$

By definition, $\overline{P}^G_i$ consists of the i-actions in $Act^G_i(\overline{\mathcal{N}}_T)$, so we have

$$s \xrightarrow{i} t \in \mathcal{N}^p_T \text{ iff } (s_i, B_i \to A_i, t_i) \in Act^G_i(\overline{\mathcal{N}}_T)$$
$$\text{with } s\restriction i = s_i, t\restriction i = t_i, B_i = \{|s|\} \text{ and } \{|s\restriction \mathcal{S}H|\}\ A_i\ \{|t\restriction \mathcal{S}H|\} \tag{A1}$$

By definition of $Act^G_i(\overline{\mathcal{N}}_T)$,

$$(s_i, B_i \to A_i, t_i) \in Act_i^G(\overline{N}_T) \text{ iff}$$
$$\exists g, j : (s_j, B_j \to A_j, t_j) \in Act_j(\overline{N}_T), g(s_j, B_j \to A_j, t_j) = (s_i, B_i \to A_i, t_i), g(j) = i \tag{A2}$$

By definition of $Act_j(\overline{N}_T)$,

$$\begin{aligned}&(s_j, B_j \to A_j, t_j) \in Act_j(\overline{N}_T) \text{ iff}\\ &(s_j, B_j \to A_j, t_j) = action(s' \xrightarrow{j} t') \text{ where } s' \xrightarrow{j} t' \in Reg_j(\overline{N}_T)\\ &\text{with } s_j = s'\restriction j, t_j = t'\restriction j, B_j = \{|s'|\}, \text{and} \{|s'\restriction \mathcal{S}H|\}\ A_j\ \{|t'\restriction \mathcal{S}H|\}.\end{aligned} \tag{A3}$$

By definition of $Reg_j(\overline{N}_T)$,

$$s' \xrightarrow{j} t' \in Reg_j(\overline{N}_T) \text{ iff } \exists g', t'' : t' = g'(t''), s' \to t'' \in \overline{N}_T \tag{A4}$$

By stringing together all the above equivalences, we obtain

$$s \xrightarrow{i} t \in \mathcal{N}_T^p \text{ iff } s' \to t'' \in \overline{N}_T \tag{A5}$$

From (A1), $s\restriction i = s_i$, $t\restriction i = t_i$, $B_i = \{|s|\}$, and $\{|s\restriction \mathcal{S}H|\}\ A_i\ \{|t\restriction \mathcal{S}H|\}$.
From (A2), $g(j) = i$, and $g(s_j, B_j \to A_j, t_j) = (s_i, B_i \to A_i, t_i)$, so that $g(s_j) = s_i$, $g(t_j) = t_i$, $g(B_j) = B_i$, and $g(A_j) = A_i$.
From (A3), $s_j = s'\restriction j$, $t_j = t'\restriction j$, $B_j = \{|s'|\}$, and $\{|s'\restriction \mathcal{S}H|\}\ A_j\ \{|t'\restriction \mathcal{S}H|\}$.
From $B_i = \{|s|\}$, $B_j = \{|s'|\}$, $g(B_j) = B_i$, and $g(A_j) = A_i$, we obtain $g(s') = s$, since B_i characterizes s uniquely, B_j characterizes s' uniquely (by the unique labeling condition on multiprocess Kripke structures, Definition 3.2) and A_i, A_j are assignments of constant values to shared values, with the constant values of the form given in Definition 6.1. Hence s, s' are in the same orbit.
Since $s \xrightarrow{i} t \in \mathcal{N}_T^p$, we have $s\downarrow i = t\downarrow i$ by operational semantics. Also, $s' \xrightarrow{j} t' \in Reg_j(\overline{N}_T)$ and so $s'\downarrow j = t'\downarrow j$ by operational semantics.
Now $g(t'\downarrow j) = g(s'\downarrow j) = g(s')\downarrow g(j) = s\downarrow i = t\downarrow i$. Also, $g(t'\restriction j) = g(t_j) = t_i = t\restriction i$. Putting these two equalities together, we obtain $g(t') = t$. From (A4), $t' = g'(t'')$, and so $t = g(g'(t''))$. Hence, t, t'' are in the same orbit. From above, $\vartheta_G(s) = \bar{s}$, $\vartheta_G(t) = \bar{t}$. Hence also $\vartheta_G(s') = \bar{s}$, $\vartheta_G(t'') = \bar{t}$. Now s', t'' are states in the quotient structure $\overline{\mathcal{N}}$ Since $\overline{\mathcal{N}}$ and $\overline{\mathcal{N}_T^p}$ have the same representation map, we conclude that $\vartheta_G(s') = \bar{s}$ implies $s' = \bar{s}$ and $\vartheta_G(t'') = \bar{t}$ implies $t'' = \bar{t}$, since quotients contain only a single element of each orbit under G.
From (A5), and Definition 5.7, we have:

$$\bar{s} \to \bar{t} \in \overline{\mathcal{N}_T^p} \text{ iff } s' \to t'' \in \overline{N}_T.$$

And hence

$$s' \to t'' \in \overline{\mathcal{N}_T^p} \text{ iff } s' \to t'' \in \overline{N}_T.$$

Thus $\overline{\mathcal{N}^p}$ and $\overline{\mathcal{N}}$ have the same transitions. Since initial states are related by (G, ϑ_G), by definition, we conclude that $\overline{\mathcal{N}^p} = \overline{\mathcal{N}}$. □

Corollary 6.12. *Let $\overline{P}^G$ be the repaired program and φ the CTL specification from* Theorem 6.11. *Then $\overline{P}^G \models \varphi$.*

Proof. Since $\overline{N}$ results from repairing $\overline{M}$ with respect to φ, we have $\overline{N} \models \varphi$. By Theorem 6.11, $\overline{N^p} = \overline{N}$, and so $\overline{N^p} \models \varphi$. By Theorem 5.22 (Repair Correspondence), $\overline{N^p} \models \varphi$ iff $N^p \models \varphi$. Hence $N^p \models \varphi$. Since N^p is the state-transition graph of $\overline{P}^G$, we conclude $\overline{P}^G \models \varphi$. □

Theorem 8.2. *If φ and ψ are purely propositional formulae then for any Kripke structure M, the pair $(M, A[\varphi\, R\, \psi])$ is G-repair complete.*

Proof. Suppose that $N \leq M$ and $N, s \models A[\varphi\, R\, \psi])$. Then $N, s \models \psi$ and either $N, s \models \varphi$ or $N, s \models AXA[\varphi\, R\, \psi]$. Since φ is purely propositional, we conclude that $N^G, g(s) \models \varphi \iff N, s \models \varphi$. The rest of the proof follows from the path correspondence theorem, together with the requirement that Kripke structures do not contain unreachable states. □

Lemma 10.3. *Bisimilarity does not imply equi-repairability, that is, there exist M and M' such that $M \sim M'$ and $M \not\equiv_{\triangleleft} M'$.*

Proof. Figure 9 (with dashed transitions removed) and Figure 16 are bisimilar, since the latter is an "unwinding" of the former, as discussed below. However, they are not equi-reparable. Figure 17 gives a repair of Figure 16 which ensures absence of starvation for both processes. No repair is possible for the structure in Figure 9. □

Lemma 10.4. *Equi-repairability does not imply bisimilarity of structures, that is, there exist M and M' such that $M \equiv_{\triangleleft} M'$ and $M \not\sim M'$.*

Proof. Figure 23 provides a pair of Kripke structures that are equi-reparable (every substructure of one is bisimilar to a substructure of the other) but not bisimilar (state *s1* in the left structure has no bisimilar state in the right structure). □

Lemma 10.6. *Let M be a Kripke structure such that*

(1) *M contains a state s with at least two successors, and*
(2) *all states in M have different propositional labelings. Let M' be the one-unwinding of M.*

Then M and M' are not equi-repairable.

Proof. Let u and t be two successors of s. Let $[|s|] \triangleq$ "$(\bigwedge_{Q \in L(s)} Q) \wedge (\bigwedge_{Q \notin L(s)} \neg Q)$". That is, $[|s|]$ converts the atomic proposition labeling of s into a propositional formula, while ignoring the shared variables, if any (compare $[|s|]$ with $\{|s|\}$ defined above). Let φ be the CTL formula

$$\mathsf{EF}([|s|] \wedge \mathsf{EX}([|t|])) \wedge \mathsf{EF}([|s|] \wedge \mathsf{EX}([|u|])) \wedge \mathsf{AG}([|s|] \Rightarrow \mathsf{EX}([|t|]) \oplus \mathsf{EX}([|u|])).$$

where $\oplus$ denotes exclusive or. So φ requires that any state bisimilar to s (i.e., s itself and any copy of s in an unwound structure) has exactly one successor from $\{t, u\}$, but not both. This cannot be achieved in M, since there is only one copy of s, namely s itself. M' contains two copies of s; the original and the one-unwinding. Hence M' can be repaired w.r.t. φ: delete the transition from the original s to the original t and from the one-unwound copy of s to the one-unwound copy of u. Hence $M \not\triangleleft \varphi$ and $M' \triangleleft \varphi$. Hence $M \not\equiv_{\triangleleft} M'$. □

Theorem 10.7. *Let M be a Kripke structure such that all states in M have different propositional labelings. Let M^i be the i-unwinding of M, and M^j the j-unwinding of M, for $i > j \geq 1$. If M contains a state s with at least i successors, then M^i and M^j are not equi-repairable.*

Proof. We generalize the proof of Lemma 10.6. Let $t_1, \ldots, t_i$ be the i successors of s. Let φ be the CTL formula

$$\left(\bigwedge_{1 \le k \le i} \mathsf{EF}([|s|] \wedge \mathsf{EX}([|t_k|])) \right) \wedge \mathsf{AG}([|s|] \Rightarrow \mathsf{ONE}(\mathsf{EX}([|t_1|]), \ldots, \mathsf{EX}([|t_i|])))$$

where $\mathsf{ONE}(q_1, \ldots, q_i)$ is a propositional formula that is true iff exactly one of its arguments $q_1, \ldots, q_i$ is true. The construction of ONE is a standard exercise in propositional logic and is omitted.

Following the proof idea of Lemma 10.6, we repair $\mathcal{M}'$ as follows. For k from 1 to i, we delete the transitions from the k'th copy of state s to the k copies of $t_1, \ldots, t_{k-1}, t_{k+1}, \ldots, t_i$. That is, for the k'th vcopy of s, we retain only its transition to t_k. These deletions cause the repaired structure to satisfy $\mathsf{AG}([|s|] \Rightarrow \mathsf{ONE}(\mathsf{EX}([|t_1|]), \ldots, \mathsf{EX}([|t_i|]))$ while they preserve the satisfaction of $\left(\bigwedge_{1 \le k \le i} \mathsf{EF}([|s|] \wedge \mathsf{EX}([|t_k|]))\right)$. This repair is not possible in $\mathcal{M}^j$ for $j < i$, since there are not enough copies of s. □

Theorem 11.1. *Let $G = \mathsf{S}_n$. Then the number of states in $\overline{\mathcal{M}} = \mathcal{M}/(G, \vartheta_G)$ is $\binom{n+\ell-1}{n}$.*

Proof. We construct a state of $\mathcal{M}$ by choosing a local state, i.e., one out of ℓ, for each of n processes. If the process indices are irrelevant, i.e., states $[N_1\, T_2]$ and $[T_1\, N_2]$ are equivalent, then this is a situation of choosing n objects (local states for n processes) from out of ℓ different types (the ℓ different local states for each process). Since repetitions are allowed (different processes can have the same local state), we obtain the well-known result $\binom{n+\ell-1}{n}$ for the number of n element multi-sets with elements taken from a set of size ℓ [8, Theorem 3.1]. □

Corollary 11.2. *The number of states in $\overline{\mathcal{M}}$ is asymptotically equal to*

$$\sqrt{\frac{n+\ell-1}{2\pi n(\ell-1)}} \left(1 + \frac{\ell-1}{n}\right)^n \left(1 + \frac{n}{\ell-1}\right)^{\ell-1}.$$

Proof. Recall Stirling's approximation: $n! \approx \sqrt{2\pi n}(n/e)^n$, where $\approx$ means that the ratio of $n!$ and $\sqrt{2\pi n}(n/e)^n$ tends to 1 as n goes to infinity.[3] Now $\binom{n+\ell-1}{n} = \frac{(n+\ell-1)!}{n!(\ell-1)!}$. Applying Stirlings approximation, we obtain that

$$\binom{n+\ell-1}{n} \approx \frac{\sqrt{n+\ell-1}\,((n+\ell-1)/e)^{n+\ell-1}}{\sqrt{2\pi}\,\sqrt{n}\,(n/e)^n\,\sqrt{\ell-1}\,(\ell-1/e)^{\ell-1}}.$$

By cancelling and rearranging terms, we simplify the right hand side to

$$\frac{\sqrt{n+\ell-1}\,(n+\ell-1)^{n+\ell-1}}{\sqrt{2\pi}\sqrt{n(\ell-1)}\,n^n\,(\ell-1)^{\ell-1}}.$$

We now group, to obtain

$$\sqrt{\frac{n+\ell-1}{2\pi n(\ell-1)}} \left(\frac{n+\ell-1}{n}\right)^n \left(\frac{n+\ell-1}{\ell-1}\right)^{\ell-1}.$$

And simplify to obtain the desired result

$$\sqrt{\frac{n+\ell-1}{2\pi n(\ell-1)}} \left(1 + \frac{\ell-1}{n}\right)^n \left(1 + \frac{n}{\ell-1}\right)^{\ell-1}.$$

□

[3] The usual symbol for this is $\sim$, but we are using $\sim$ for G-bisimulation.

Theorem 11.3. *If ℓ is constant, then the number of states in $\overline{\mathcal{M}}$ is $\Theta(n^{\ell-1})$.*

Proof. We start with the term for the number of states in $\overline{\mathcal{M}}$ given by Theorem 11.1: $\binom{n+\ell-1}{n}$. Expanding, we obtain

$$\frac{(n+\ell-1)!}{(\ell-1)!\,n!}$$

Factoring out the $n!$ term, we obtain

$$\frac{1}{(\ell-1)!}\,\Pi_{i=1}^{\ell-1}(n+i)$$

The binomial expansion of $\Pi_{i=1}^{\ell-1}(n+i)$ consists of $\ell-1$ terms, with $n^{\ell-1}$ being the largest of these asymptotically. Hence $\Pi_{i=1}^{\ell-1}(n+i) = \Theta(n^{\ell-1})$.

Since $\frac{1}{(\ell-1)!}$ is constant, we obtain that the number of states in $\overline{\mathcal{M}}$ is $\Theta(n^{\ell-1})$. □

Theorem 11.4. *If $\ell = \Theta(\log n)$, then the number of states in $\overline{\mathcal{M}}$ is $O(n^{d \log n})$ for some positive constant d.*

Proof. Since n and ℓ both increase to infinity, the asymptotic result of Corollary 11.2 is applicable. To keep the algebra simple, let $\ell = 1 + c \ln n$, where c is some constant, $c > 0$ and ln denote the natural logarithm, which we use for technical reasons. Then $\ell - 1 = c \ln n$. By Corollary 11.2 the number of states in $\overline{\mathcal{M}}$ is asymptotically equal to

$$\sqrt{\frac{n+\ell-1}{2\pi n(\ell-1)}}\left(1+\frac{\ell-1}{n}\right)^{n}\left(1+\frac{n}{\ell-1}\right)^{\ell-1}$$

and replacing $\ell - 1$ by $c \ln n$, we obtain

$$\sqrt{\frac{n+c\ln n}{2\pi n(c\ln n)}}\left(1+\frac{c\ln n}{n}\right)^{n}\left(1+\frac{n}{c\ln n}\right)^{c\ln n}.$$

We shall use $\approx$ for "asymptotically equal," i.e., the ratio approaches 1 in the limit. The usual symbol for this is $\sim$, but we are using $\sim$ for G-bisimulation. We first establish two preliminary claims.

Claim 1: $\left(1+\frac{c\ln n}{n}\right)^{n} \approx n^{c}$

Proof of Claim 1: Let $a_n = (1+\frac{c\ln n}{n})^n$. Then $\ln a_n = n \ln (1+\frac{c\ln n}{n})$. For $-1 < x \leq 1$ we have the following power-series expansion [32, Chapter 9]:

$$\ln(1+x) = x - x^2/2 + x^3/3 - \cdots$$

Now $(c \ln n)/n > 0$ since $n \geq 2$, as we assume a concurrent program contains at least two processes. For large n, $(c \ln n)/n \lll 1$. Hence the above power series applies, and we have

$$\ln\left(1+\frac{c\ln n}{n}\right) = \frac{c\ln n}{n} - \frac{1}{2}\left(\frac{c\ln n}{n}\right)^2 + \cdots$$

Since $(c \ln n)/n$ tends to 0 as n goes to infinity, we have

$$\ln\left(1+\frac{c\ln n}{n}\right) \approx \frac{c\ln n}{n}$$

and so

$$\ln a_n \approx n\frac{c\ln n}{n}.$$

Hence $\ln a_n \approx c \ln n$. Hence $a_n \approx e^{c \ln n} = n^c$.
End proof of claim 1.

Claim 2: $\left(1 + \frac{n}{c \ln n}\right)^{c \ln n} \approx n^{c \ln n}$
Proof of Claim 2. Let $a_n = (1 + \frac{n}{c \ln n})^{c \ln n}$. Then

$$\ln a_n = (c \ln n) \ln \left(1 + \frac{n}{c \ln n}\right).$$

For large n, $\frac{n}{c \ln n} \ggg 1$, and so

$$\ln a_n \approx (c \ln n) \ln \left(\frac{n}{c \ln n}\right).$$

Now $\ln \left(\frac{n}{c \ln n}\right) = \ln n - \ln (c \ln n)$. For large n, $\ln n \ggg \ln (c \ln n)$, and so $\ln n - \ln (c \ln n) \approx \ln n$. And hence

$$\ln \left(\frac{n}{c \ln n}\right) \approx \ln n$$

Hence $\ln a_n \approx (c \ln n) \ln n = c \ln^2(n)$. So $a_n \approx e^{c \ln^2 n} = (e^{\ln n})^{c \ln n} = n^{c \ln n}$.
End proof of claim 2.

Now, substituting Claims 1 and 2 into our expression above for the number of states in $\overline{\mathcal{M}}$, we obtain that the number of states in $\overline{\mathcal{M}}$ is asymptotically equal to

$$n^c n^{c \ln n} \sqrt{\frac{n + c \ln n}{2\pi n (c \ln n)}}.$$

The term $\sqrt{\frac{n + c \ln n}{2\pi n (c \ln n)}}$ is asymptotically less than $\sqrt{\frac{2n}{2\pi n}}$ which is constant. Hence the number of states in $\overline{\mathcal{M}}$ is asymptotically less than $n^c n^{c \ln n}$. This is asymptotically less than $n^{d \log n}$ for some constant $d > c$. □

Theorem 11.5. *If $\ell = \Theta(\log n)$, then the number of states in $\overline{\mathcal{M}}$ is $\Omega(n^{b \log n})$ for some positive constant b.*

Proof. Repeating the first part of the previous proof, we have that the number of states in $\overline{\mathcal{M}}$ is asymptotically equal to

$$n^c n^{c \ln n} \sqrt{\frac{n + c \ln n}{2\pi n (c \ln n)}}.$$

Now $\sqrt{\frac{n + c \ln n}{2\pi n (c \ln n)}}$ is greater than $\sqrt{\frac{n}{2\pi n (c \ln n)}}$ is equal to $\sqrt{\frac{1}{2\pi c \ln n}}$.
Hence the number of states in $\overline{\mathcal{M}}$ is asymptotically greater than

$$n^c n^{c \ln n} \sqrt{\frac{1}{2\pi c \ln n}}.$$

which is asymptotically greater than

$$n^c n^{c \ln n} \sqrt{\frac{1}{2\pi c n^\epsilon}},$$

where $0 < \epsilon < c$, since n^ϵ is asymptotically larger than $\ln n$. This is $\Omega(n^{c - \epsilon/2} n^{c \ln n})$, which in turn is $\Omega(n^{c \ln n})$, since $\epsilon < c$. This is $\Omega(n^{b \log n})$ for $b = c / \log e$. □

Theorem 11.6. *If $\ell = \Theta(n)$, then the number of states in $\overline{\mathcal{M}}$ is $O(d^n)$ for some positive constant $d > 1$.*

Proof. Since n and ℓ both increase to infinity, the asymptotic result of Corollary 11.2 is applicable. To keep the algebra simple, let $\ell = cn + 1$, where c is some constant, $c > 0$. Then $\ell - 1 = cn$. Then, by Corollary 11.2 the number of states in $\overline{\mathcal{M}}$ is asymptotically equal to

$$\sqrt{\frac{n+\ell-1}{2\pi n(\ell-1)}}\left(1+\frac{\ell-1}{n}\right)^n\left(1+\frac{n}{\ell-1}\right)^{\ell-1}.$$

equals

$$\sqrt{\frac{n+cn}{2\pi n(cn)}}\left(1+\frac{cn}{n}\right)^n\left(1+\frac{n}{cn}\right)^{cn}$$

equals

$$\sqrt{\frac{1+c}{2\pi cn}}\,(1+c)^n\left(1+\frac{1}{c}\right)^{cn}$$

The term $\sqrt{\frac{1+c}{2\pi cn}}$ goes to zero as n goes to infinity, and so can be removed from an upper bound calculation. The term $\left(1+\frac{1}{c}\right)^c$ has the limit e as c goes to infinity and is always smaller than e. Hence $\left(1+\frac{1}{c}\right)^{cn}$ is always smaller than its limit e^n. So an upper bound for the number of states in $\overline{\mathcal{M}}$ is

$$e^n\,(1+c)^n$$

That is

$$(e(1+c))^n$$

And so the number of states in $\overline{\mathcal{M}}$ is $O(d^n)$, where the constant $d > 1$ is $e(1+c)$. □

Theorem 11.7. *If $\ell = \Theta(n)$, then the number of states in $\overline{\mathcal{M}}$ is $\Omega(b^n)$ for some positive constant $b > 1$.*

Proof. Repeating the first few steps of the proof of Theorem 11.6, we have that the number of states in $\overline{\mathcal{M}}$ is asymptotically equal to

$$\sqrt{\frac{1+c}{2\pi cn}}\,(1+c)^n\left(1+\frac{1}{c}\right)^{cn}$$

where $c > 0$ is the constant such that $\ell = cn + 1$. Hence the term $\sqrt{\frac{1+c}{2\pi cn}}$ can be rewritten as $c'\sqrt{\frac{1}{n}}$ where c' is a constant. For $n \geq 2$ and $c > 0$, we have $\left(1+\frac{1}{c}\right)^{cn} \geq 1$. Hence the number of states in $\overline{\mathcal{M}}$ is asymptotically at least

$$c'\sqrt{\frac{1}{n}}\,(1+c)^n$$

Since $c > 0$, we have $1 + c > 1$. Write $1 + c = ab$, where a, b are constants such that $a > 1$ and $b > 1$. So $(1+c)^n = (ab)^n = a^n b^n$. Now $c'a^n\sqrt{\frac{1}{n}}$ is asymptotically larger than 1, so the number of states in $\overline{\mathcal{M}}$ is asymptotically at least b^n. □

B Generating the Symmetry-Reduced Kripke Structure

The following is verbatim Figure 1 in Emerson and Sistla [24] and gives an algorithm to construct the reduced structure $\overline{\mathcal{M}} = (\overline{\mathcal{S}}, \overline{s}_0, \overline{\mathcal{R}}, L, AP)$ from the concurrent program P.

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

Received 28 October 2024; revised 16 February 2025; accepted 17 February 2025