Reducing Model Checking of the Many to the Few *

E. Allen Emerson and Vineet Kahlon

Department of Computer Sciences
The University of Texas at Austin, Austin TX-78712, USA
{emerson,kahlon}@cs.utexas.edu
http://www.cs.utexas.edu/users/{emerson,kahlon}

Abstract. The *Parameterized Model Checking Problem* (PMCP) is to determine whether a temporal property is true for every size instance of a system comprised of many homogenous processes. Unfortunately, it is undecidable in general. We are able to establish, nonetheless, decidability of the PMCP in quite a broad framework. We consider asynchronous systems comprised of an arbitrary number of homogeneous copies of a generic process template. The process template is represented as a synchronization skeleton while correctness properties are expressed using Indexed CTL*\X. We reduce model checking for systems of arbitrary size n to model checking for systems of size up to (of) a small *cutoff* size c. This establishes decidability of PMCP as it is only necessary to model check a finite number of relatively small systems. Efficient decidability can be obtained in some cases. The results generalize to systems comprised of multiple heterogeneous classes of processes, where each class is instantiated by many homogenous copies of the class template (e.g., m readers and n writers).

1 Introduction

Systems with an arbitrary number of homogeneous processes can be used to model many important applications. These include classical problems such as mutual exclusion, readers and writers, as well as protocols for cache coherence and data communication among others. It is often the case that correctness properties are expected to hold irrespective of the size of the system, as measured by the number of processes in it. However, time and space constraints permit us to verify correctness only for instances with a small number of processes. This makes it impossible to guarantee correctness in general and thus motivates consideration of automated methods to permit verification for system instances of arbitrary sizes. The general problem, known in the literature as the *Parameterized Model Checking Problem (PMCP)* is the following: to decide whether a temporal property is true for every size instance of a given system. This problem

* This work was supported in part by NSF grant CCR-980-4737, SRC contract 99-TJ-685 and TARP project 003658-0650-1999.

is known to be undecidable in general [1]. However, by imposing certain stipulations on the organization of the processess we can get a useful framework with a decidable PMCP.

We establish our results in the *synchronization skeleton* framework. Our results apply to systems comprised of multiple heterogeneous *classes* of processes with many homogeneous process *instances* in each class. Thus, given family $(U_1, ..., U_k)$ of k process classes, and tuple $(n_1, ..., n_k)$ of natural numbers, we let $(U_1, ..., U_k)^{(n_1, ..., n_k)}$ denote the concrete system composed of n_1 copies or instances of U_1 through n_k copies or instances of U_k running in parallel asynchronously (i.e., with interleaving semantics). By abuse of notation, we also write $(U_1, ..., U_k)^{(n_1, ..., n_k)}$ for the associated state graph, where each process starts in its designated initial state.

Correctness properties are expressed using a fragment of Indexed CTL*\X. The basic assertions are of the form "for all processes $\mathsf{A}h$", or "for all processes $\mathsf{E}h$", where h is an LTL\X formula (built using F "sometimes", G "always", U, "until", but without X "next-time") over propositions indexed just by the processes being quantified over, and A "for all futures" and E "for some future" are the usual path quantifiers. Use of such an indexed, stuttering-insensitive logic is natural for parameterized systems.

We consider correctness properties of the following types:

1. Over all individual processes of single class U_l:
   $\bigwedge_{i_l} \mathsf{A}h(i_l)$ and $\bigwedge_{i_l} \mathsf{E}h(i_l)$, where i_l ranges over (indices of) individual processes in U_l.
2. Over pairs of different processes of a single class U_l:
   $\bigwedge_{i_l \neq j_l} \mathsf{A}h(i_l, j_l)$ and $\bigwedge_{i_l \neq j_l} \mathsf{E}h(i_l, j_l)$, where i_l, j_l range over pairs of distinct processes in U_l.
3. Over one process from each of two different classes U_l, U_m:
   $\bigwedge_{i_l, j_m} \mathsf{A}h(i_l, j_m)$ and $\bigwedge_{i_l, j_m} \mathsf{E}h(i_l, j_m)$, where i_l ranges over U_l and j_m ranges over U_m.

We say that the k-tuple $(c_1, ..., c_k)$ of natural numbers is a *cutoff* of $(U_1, ..., U_k)$ for formula f iff : $\forall(n_1, ..., n_k), (U_1, ..., U_k)^{(n_1, ..., n_k)} \models f$ iff $\forall(m_1, ..., m_k) \preceq (c_1, ..., c_k) : (U_1, ..., U_k)^{(m_1, ..., m_k)} \models f$, where we write $(m_1, ..., m_k) \preceq (c_1, ..., c_k)$ to mean $(m_1, ..., m_k)$ is *component-wise* less than or equal to $(c_1, ..., c_k)$ and $(m_1, ..., m_k) \succeq (c_1, ..., c_k)$ to mean $(c_1, ..., c_k) \preceq (m_1, ..., m_k)$.

In this paper, we show that for systems in the synchronization skeleton framework with transition guards of a particular *disjunctive* or *conjunctive* form, there is a small cutoff. This, in effect, reduces PMCP to ordinary model checking over a relatively few small, finite sized systems. In some cases, depending on the kind of property and guards, we can get an efficient (quadratic in the size of the template processes) solution to PMCP.

Each process class is described by a generic process, a process *template* for the class. A system with k classes is given by templates $(U_1, ..., U_k)$. For such a system, define $c_i = |U_i| + 3$ and $d_i = 2|U_i| + 1$, where $|U_i|$ is the size i.e. the number of local states of template U_i. Then, for both conjunctive and disjunctive guards, cutoffs of $(d_1, ..., d_k)$ and $(c_1, ..., c_k)$ respectively suffice for all three types of formulae described above. These results give decision procedures for PMCP for conjunctive or disjunctive guards. Since these are a broad framework and PMCP is undecidable in general, we view this as quite a positive result. However, the decision procedures are not necessarily efficient ones, although they may certainly be usable on small examples. Because the cutoff is proportional to the sizes of the template processes, the global state graph of the cutoff system is of size exponential in the template sizes, and the decision procedures are also exponential. In the case of disjunctive guards, if we restrict ourselves to the A path quantifiers, but still permit all three type of properties, then the cutoff can be reduced, in quadratic time in the size of the template processes, to something of the form $(1, ..., 2, ..., 1)$ or $(1, ..., 3, ..., 1)$. In fact, depending on the type of property, we can show that it is possible to simplify the guards to ensure that only two or three classes need be retained. On the other hand, for conjunctive guards, if we restrict ourselves to model checking over infinite paths or over finite paths, then sharper cutoffs of the form (1,...,3,...,1), (1,...,2,...,1) or even (1,...,1) can, in some cases, be obtained.

The rest of the paper is organized as follows. Section 2 defines the system model. Section 3 describes how to exploit the symmetry inherent in the model and correctness properties. Sections 4 and 5 prove the results pertaining to disjunctive and conjunctive guards respectively. We show some applications of our results in Section 6. In the concluding Section 7, we discuss related work.

2 The System Model

We focus on systems comprised of multiple heterogeneous *classes* of processes modelled as *synchronization skeletons*(cf. [2]) . Here, an individual concrete process has a transition of the form $l \xrightarrow{g} m$ indicating that the process can transit from local state l to local state m, provided the guard g is true. Each class is specified by giving a generic process *template*. If I is (an) index set $\{1, \ldots, n\}$, then we use U^I, or $(U)^{(n)}$ for short, to denote the concurrent system $U^1 \| \ldots \| U^n$ comprised of the n isomorphic (up to re-indexing) processes U^i running in parallel asynchronously. For a system with k classes associated with the given templates $U_1, U_2, ..., U_k$, we have corresponding (disjoint) index sets $I_1, I_2, \ldots I_k$. Each index set I_j is (a copy of) an interval $\{1, \ldots, c\}$ of natural numbers, denoted $\{1_j, \ldots, n_j\}$ for emphasis[1]. In practice, we assume the k index sets are specified by giving a k-tuple $(n_1, ..., n_k)$ of natural numbers, corresponding to I_1 being (a copy of) interval $\{1, \ldots n_1\}$ through I_k being (a copy of) interval $\{1, \ldots, n_k\}$.

Given family $(U_1, ..., U_k)$ of k template processes, and a k-tuple $(n_1, ..., n_k)$ of natural numbers, we let $(U_1, ..., U_k)^{(n_1, \ldots, n_k)}$ denote the concrete system composed on n_1 copies of U_1 through n_k copies of U_k running in parallel asynchronously (i.e., with interleaving semantics). A template process $U_l = (S_l, R_l, \mathsf{i}_l)$ for class l, is comprised of a finite set S_l of (local) states, a set of transition

[1] e.g., if I_1 is a copy of $\{1, 2, 3\}$, the said copy is denoted $\{1_1, 2_1, 3_1\}$. Informally, subscripted index 3_1 means process 3 of class 1; formally, it is the ordered pair $(3, 1)$ as is usual with indexed logics

edges R_l, and an initial(local) state i_l. Each transition R_l is labelled with a guard, a boolean expression over atomic propositions corresponding to local states of other template processes. Then given index i and template process U_l, $U_l^i = (S_l^i, R_l^i, \mathsf{i}_l^i)$ is used to denote the ith copy of the template process U_l. Here S_l^i, the state set of U_l^i, R_l^i its transition relation and i_l^i its initial state are obtained from S_l, R_l and i_l respectively by uniformly superscripting the states of U_l with i. Thus, for local states s_l, t_l of S_l, s_l^i, t_l^i denote local states of U_l^i and $(s_l, t_l) \in R_l$ iff $(s_l^i, t_l^i) \in R_l^i$.

Given guards of transitions in the template process, we now describe how to get the corresponding guards for the concrete process U_l^i of $(U_1, ..., U_k)^{(n_1,...,n_k)}$. In this paper, we consider the following two types of guards.

i) Disjunctive guards - of the general form $(a_1 \vee ... \vee b_1) \bigvee ... \bigvee (a_k \vee ... \vee b_k)$, where the various $a_l, ..., b_l$ are (propositions identified with the) local states of template U_l, label each transition $(s_l, t_l) \in R_l$. In concrete process U_l^i of the system $(U_1, ..., U_k)^{(n_1,...,n_k)}$, the corresponding transition $(s_l^i, t_l^i) \in R_l^i$ is then labelled by the guard

$$\bigvee_{r \neq i}(a_l^r \vee ... \vee b_l^r) \vee \bigvee_{j \neq l}(\bigvee_{k \in [1..n_j]} (a_j^k \vee ... \vee b_j^k)),$$

where proposition a_j^k is understood to be true when process k in class U_j i.e. U_j^k is in local state a_j for template process U_j.

ii) Conjunctive guards with initial state - of the general form $(\mathsf{i}_1 \vee a_1 \vee ... \vee b_1) \bigwedge ... \bigwedge (\mathsf{i}_k \vee a_k \vee ... \vee b_k)$. In concrete process i of class l, U_l^i, in the system $(U_1, ..., U_k)^{(n_1,...,n_k)}$, the corresponding transition is then labelled by the guard

$$\bigwedge_{r \neq i}(\mathsf{i}_l^r \vee a_l^r \vee ... \vee b_l^r) \wedge \bigwedge_{j \neq l}(\bigwedge_{k \in [1..n_j]} (\mathsf{i}_j^k \vee a_j^k \vee ... \vee b_j^k)).$$

Note that the initial local states of processes must be present in these guards. Thus, the inital state of a process has a "neutral" character so that when process j is in its initial state, it does not prevent progress by another process i. This natural condition permits modelling a broad range of applications (and is helpful technically).

We now formalize the asynchronous concurrent (interleaving) semantics. A process transition with guard g is *enabled* in global state s iff $s \models g$ i.e., g is true over the local states in s. A transition can be fired in global state s iff its guard g is enabled. Let $(U_1, ..., U_k)^{(n_1,...,n_k)} = (S^{(n_1,...,n_k)}, R^{(n_1,...,n_k)}, \mathsf{i}^{(n_1,...,n_k)})$ be the global state transition graph of the system instance $(n_1, n_2, ..., n_k)$. A state $s \in S^{(n_1,...,n_k)}$ is written as a $(n_1 + ... + n_k)$-tuple $(u_1^1, ..., u_1^{n_1}, u_2^1, ..., u_k^{n_k})$ where the projection of s onto process i of class l, denoted $s(l, i)$, equals u_l^i, the local state of the ith copy of the template process U_l. The initial state $\mathsf{i}^{(n_1,...,n_k)} = (\mathsf{i}_1^1, ..., \mathsf{i}_k^{n_k})$. A global transition $(s, t) \in R^{(n_1,...,n_k)}$ iff t results from s by firing an enabled transition of some process i.e., there exist i, l such that the guard labelling $(u_l^i, v_l^i) \in R_l^i$ is enabled at s, $s(l, i) = u_l^i, t(l, i) = v_l^i$, and for all

$(j,k) \neq (i,l)$, $s(k,j) = t(k,j)$. We write $(U_1, ..., U_k)^{(n_1,...,n_k)} \models f$ to indicate that the global state graph of $(U_1, ..., U_k)^{(n_1,...,n_k)}$ satisfies f at initial state $\mathsf{i}^{(n_1,...,n_k)}$.

Finally, for global state s, define $Set(s) = \{t \mid s$ *contains an indexed local copy of* $t\}$. For computation path $x = x_0, x_1, ...$ we define $PathSet(x) = \bigcup_i Set(x_i)$. We say that the sequence of global states $y = y_0, y_1, ...$ is a *stuttering* of computation path x iff there exists a parsing $P_0 P_1 ...$ of y such that for all $j \geq 0$ there is some $r > 0$ with $P_j = (x_j)^r$(cf. [3]). Also, we extend the definition of projection to include computation sequences as follows: for $i \in [1..n_l]$, the sequence of local states $x_0(l,i), x_1(l,i), ...$ is denoted by $x(l,i)$.

3 Appeal to Symmetry

We can exploit symmetry inherent in the system model and the properties in the spirit of "state symmetry" codified by [8](cf. [16],[12]) to simplify our proof obligation. To establish formulae of types $\bigwedge_{i_l} f(i_l)$, $\bigwedge_{i_l \neq j_l} f(i_l, j_l)$ and $\bigwedge_{i_l, j_m} f(i_l, j_m)$, it suffices to show the results with the formulae replaced by $f(1_l)$, $f(1_l, 2_l)$ and $f(1_l, 1_m)$, respectively. The basic idea is that in a system comprised of fully interchangeable processes 1 through n of a given class, symmetry considerations dictate that process 1 satisfies a property iff each process $i \in [1..n]$ satisfies the property. Proofs are omitted for the sake of brevity.

4 Systems with Disjunctive Guards

In this section, we show how to reduce the PMCP for systems with disjunctive guards, to model checking systems of sizes bounded by a small cutoff, where the size of the cutoff for each process class is essentially the number of local states of individual process template for the class. This yields decidability for this formulation of PMCP, a pleasant result since PMCP is undecidable in full generality. But this result, by itself, does not give us an efficient decision procedure for the PMCP at hand. We go on to show that in the case of universal-path-quantified specification formulae ($\mathsf{A}h$), efficient decidability can be obtained.

4.1 Properties ranging over all processes in a single class

We will first establish the

Disjunctive Cutoff Theorem
Let f be $\bigwedge_{i_l} \mathsf{A}h(i_l)$ or $\bigwedge_{i_l} \mathsf{E}h(i_l)$, where h is an LTL\X formula and $l \in [1..k]$. Then we have the following

$$\forall (n_1, \ldots, n_k) \succeq (1, \ldots, 1) : (U_1, \ldots, U_k)^{(n_1,\ldots,n_k)} \models f \quad \text{iff}$$
$$\forall (d_1, \ldots, d_k) \preceq (c_1, \ldots, c_k) : (U_1, \ldots, U_k)^{(d_1,\ldots,d_k)} \models f,$$

where the cutoff $(c_1, \ldots, c_k)$ is given by $c_l = |U_l| + 2$, and for $i \neq l : c_i = |U_i| + 1$.

As a corollary, we will have the

Disjunctive Decidability Theorem PMCP for systems with disjunctive guards and single-index assertions as above is decidable in exponential time.

Proof idea
By the Disjunctive Cutoff Theorem, it is enough to model check each of the exponentially many exponential size state graphs corresponding to systems $(U_1, \ldots, U_k)^{(d_1,\ldots,d_k)}$ for all $(d_1, \ldots, d_k) \preceq (c_1, \ldots, c_k)$. QED

For notational brevity, we establish the above results for systems with just two process classes. We begin by proving the following lemmas.

Disjunctive Monotonicity Lemma
(i) $\forall n \geq 1 : (V_1, V_2)^{(1,n)} \models \mathsf{E}h(1_2)$ implies $(V_1, V_2)^{(1,n+1)} \models \mathsf{E}h(1_2)$.
(ii) $\forall n \geq 1 : (V_1, V_2)^{(1,n)} \models \mathsf{E}h(1_1)$ implies $(V_1, V_2)^{(1,n+1)} \models \mathsf{E}h(1_1)$.

Proof idea
(i) The idea is that for any computation x of $(V_1, V_2)^{(1,n)}$, there exists an analogous computation y of $(V_1, V_2)^{(1,n+1)}$ wherein the $(n+1)$st copy of template process V_2 stutters in its initial state and the rest of the processes behave as in x.
(ii) This part follows by using a similar argument. QED

The following lemma allows reduction in system size, one coordinate at a time.

Disjunctive Bounding Lemma
(i) $\forall n \geq |V_2| + 2 : (V_1, V_2)^{(1,n)} \models \mathsf{E}h(1_2)$ iff $(V_1, V_2)^{(1,c_2)} \models \mathsf{E}h(1_2)$, where $c_2 = |V_2| + 2$.
(ii) $\forall n \geq |V_2| + 1 : (V_1, V_2)^{(1,n)} \models \mathsf{E}h(1_1)$ iff $(V_1, V_2)^{(1,|V_2|+1)} \models \mathsf{E}h(1_1)$.

Proof
(i) ($\Rightarrow$) Let $x = x_0, x_1, \ldots$ denote a computation sequence of $(V_1, V_2)^{(1,n)}$. Define $Reach = \{s_1, \ldots, s_r\}$ to be the set of all local states of template process V_2 occuring in x. For $s_t \in Reach$, let $t_1, t_2, \ldots, t_m$ be a finite local computation of minimal length in x ending in s_t. Then we use $MinLength(s_t)$ to denote m and $MinComputation(s_t)$ to denote the sequence $t_1, t_2, \ldots, t_{m-1}, (t_m)^\omega$. Let $v = (\mathsf{i}_2)^\omega$. If x is an infinite computation sequence and $x(1,1)$ and $x(2,1)$ are finite local computations, then there exists an infinite local computation sequence u, say. In that case, reset $v = u$.

Construct a formal sequence $y = y_0, y_1, \ldots$ of global states of $(V_1, V_2)^{(1,|V_2|+1)}$ from x as follows

1. $y(1,1) = x(1,1)$ and $y(2,1) = x(2,1)$ i.e. the local computation paths in x of process index 1 of classes V_1, V_2 are preserved, and

2. For each state $s_j \in Reach$, we set $y(2, j+1) = MinComputation(s_j)$ i.e. we let the $(j+1)$st copy of V_2 perform a local computation of minimum length in x leading to s_j and then let it stutter in s_j forever. The above condition has the implication that for all $i \geq 1, Set(x_i) \subseteq Set(y_i)$. To see this, let $t \in Set(x_i)$. Then, $MinLength(t) \leq i$. Also, $t \in Set(x_i)$ implies that $t \in Reach$ i.e. $t = s_q$ for some $q \in [1..r]$. Then $y(2, q+1)$ stutters in s_q for all $k \geq MinLength(s_t)$ and therefore for all $k \geq i$, also. Hence $y_i(2, q+1)$ is an indexed copy of s_q, i.e. $t \in Set(y_i)$. Thus for all $i \geq 1, Set(x_i) \subseteq Set(y_i)$.

3. $y(2, c_2) = v$. This ensures that if x is an infinite computation sequence then in y infinitely many local transitions are fired.

However, it might be the case that sequence y violates the interleaving semantics requirement. Clearly, this happens iff the following scenario occurs. Let states $s_p, s_q \in Reach$, be such that $MinComputation(s_p)$ and $MinComputation(s_q)$ are realized by the same local computation of x and suppose that $MinLength(s_p) \leq MinLength(s_q)$. Then, if for $i < MinLength(s_p)$, (t_i, t_{i+1}) is a transition in $MinComputation(s_p)$, $(y_i(2,p+1), y_{i+1}(2,p+1))$ and $(y_i(2,q+1), y_{i+1}(2,q+1))$ are both local transitions driving y_i to y_{i+1}. This violates the interleaving semantics condition requiring that there be atmost one local transition driving each global transition. There are two things to note here. First, for a transition (y_i, y_{i+1}), the violation occurs only for values of $i \leq max_{j \in [1..r]}\ MinLength(s_j)$ and secondly, for a fixed i, all violations are caused by a unique template transition (s, t) of V_2, namely one which was involved in the transition (x_i, x_{i+1}).

To solve this problem, we construct a sequence of states $w = w_0, w_1, ...$ from y by "staggering" copies of the same local transition as described below. Let (y_i, y_{i+1}) be a transition where the interleaving semantics requirement is violated by process indices $in_1, ..., in_d$ of V_2 executing indexed copies $(s_2^{in_1}, t_2^{in_1}), ..., (s_2^{in_d}, t_2^{in_d})$ respectively of the template transition (s_2, t_2) of V_2. Replace (y_i, y_{i+1}) with a sequence $u_1, u_2, ..., u_f$ such that $u_1 = y_i$, $u_f = y_{i+1}$ and for all j, transition (u_j, u_{j+1}) results by executing local transition $(s_2^{in_j}, t_2^{in_j})$. Clearly the interleaving semantics requirement is met as atmost one local transition is executed for each global transition. Also, it is not hard to see that for all j, $Set(y_i) \subseteq Set(u_j)$ and hence for all k, transition (u_k, u_{k+1}) is valid. Finally, note that states with indices other than $in_1, ..., in_d$ are made to stutter finitely often in $u_1, ..., u_f$ which is allowed since we are considering only formulae without the next-time operator X.

Thus, given a computation path x of $(V_1, V_2)^{(1,n)}$, we have constructed a stuttering computation path w of $(V_1, V_2)^{(1,c_2)}$, such that the local computation sequence $w(2, 1)$ is a stuttering of the local computation sequence $x(2, 1)$. From this path correspondence, we easily have the result.

($\Leftarrow$) The proof follows by repeated application of the Disjunctive Monotonicity Lemma.

(ii) This part follows by using a similar argument. QED

The following lemma allows reduction in system size over multiple coordinates simultaneously (2 coordinates for notational brevity).

Disjunctive Truncation Lemma
$\forall n_1, n_2 \geq 1 : (U_1, U_2)^{(n_1,n_2)} \models \mathsf{E}h(1_2)$ iff $(U_1, U_2)^{(n_1',n_2')} \models \mathsf{E}h(1_2)$, where $n_2' = min(n_2, |U_2| + 2)$ and $n_1' = min(n_1, |U_1| + 1)$.

Proof
If $n_2 > |U_2| + 2$, set $V_1 = U_1^{n_1}$ and $V_2 = U_2$. Then, $(U_1, U_2)^{(n_1,n_2)} \models \mathsf{E}h(1_2)$ iff $(V_1, V_2)^{(1,n_2)} \models \mathsf{E}h(1_2)$ iff $(V_1, V_2)^{(1,n_2')} \models \mathsf{E}h(1_2)$ (by the Disjunctive Bounding Lemma) iff $(U_1, U_2)^{(n_1,n_2')} \models \mathsf{E}h(1_2)$.

If $n_1 \leq |U_1| + 1$, then $n_1 = n_1'$ and we are done, else set $V_1 = U_2^{n_2'}$ and $V_2 = U_1$. Then, $(U_1, U_2)^{(n_1,n_2')} \models \mathsf{E}h(1_2)$ iff $(U_2, U_1)^{(n_2',n_1)} \models \mathsf{E}h(1_1)$ iff $(V_1, V_2)^{(1,n_1)} \models \mathsf{E}h(1_1)$ iff $(V_1, V_2)^{(1,|U_1|+1)} \models \mathsf{E}h(1_1)$ (by the Disjunctive Bounding Lemma) iff $(U_1, U_2)^{(n_1',n_2')} \models \mathsf{E}h(1_2)$. QED

An easy but important consequence of the Disjunctive Truncation Lemma is the following

Disjunctive Cutoff Result
Let f be $\bigwedge_{i_l} \mathsf{A}h(i_l)$ or $\bigwedge_{i_l} \mathsf{E}h(i_l)$, where h is a LTL\X formula and $l \in [1..2]$. Then we have the following

$\forall (n_1, n_2) \succeq (1, 1) : (U_1, U_2)^{(n_1,n_2)} \models f$ iff
$\forall (d_1, d_2) \preceq (c_1, c_2) : (U_1, U_2)^{(d_1,d_2)} \models f$

where the cutoff (c_1, c_2) is given by $c_l = |U_l| + 2$, and for $i \neq l : c_i = |U_i| + 1$.

Proof
By appeal to symmetry and the fact that A and E are duals, it suffices to prove the result for formulae of the type $\mathsf{E}h(1_2)$. The $(\Rightarrow)$ direction is trivial. For the $(\Leftarrow)$ direction, let $n_1, n_2 \geq 1$. Define $n_1' = min(n_1, |U_1| + 1)$, $n_2' = min(n_2, |U_2| + 2)$. Then, $(U_1, U_2)^{(n_1,n_2)} \models f(1_2)$ iff $(U_1, U_2)^{(n_1',n_2')} \models f(1_2)$ by the Disjunctive Truncation Lemma. The latter is true since $(n_1', n_2') \preceq (c_1, c_2)$. This proves the cutoff result. QED

The earlier-stated Cutoff *Theorem* re-articulates the above Cutoff *Result* more generally for systems with $k \geq 1$, different classes of processes; since its proof is along similar lines but is notationally more complex, we omit it for the sake of brevity.

4.2 Efficient decidability for "for all future" properties

It can be shown that for "for some future" properties, corresponding to formulae of the type $\bigwedge \mathsf{E}h$, the reduction entailed in the previous result is, in general, the best possible. We omit the proof for the sake of brevity.

However, for universal-path-quantified properties, it is possible to be much more efficient. We will establish the

Reduction Theorem Define $V = U_l^{'}$ if for some $l \in [1..k]$, the transition graph for $U_l^{'}$ has a nontrivial strongly connected component else set $V = U_1^{'}$. Then, $(U_1, ..., U_k)^{(c_1,...,c_k)} \models \bigwedge_{i_l} \mathsf{A}h(i_l)$ iff $(U_l^{'}, V)^{(1,1)} \models \mathsf{A}h(1_1)$, where $c_l = |U_l| + 2$, $c_i = |U_i| + 1$ for $i \neq l$ and $U_l^{'}$ is the simplified process that we get from U_l by the reduction technique described below.

This makes precise our claim that for formulae of the type $\bigwedge_{i_l} \mathsf{A}h(i_l)$, it is possible to give efficient decision procedures for the PMCP at hand, by reducing it to model checking systems consisting of two or three template processes.

To this end, we first prove the following lemma which states that the PMCP problem for the above mentioned properties reduces to model checking just the *single* system instance of size equal to the (small) cutoff (as opposed to all systems of size less than or equal to the cutoff).

Single-Cutoff Lemma
$\forall n_1, n_2 \geq 1 : (U_1, U_2)^{(n_1,n_2)} \models \mathsf{A}h(1_2)$ iff $(U_1, U_2)^{(c_1,c_2)} \models \mathsf{A}h(1_2)$, where $c_1 = |U_1| + 1$ and $c_2 = |U_2| + 2$.

Proof

($\Rightarrow$) This direction follows easily by instantiating $n_1 = c_1$ and $n_2 = c_2$ on the left hand side.

($\Leftarrow$) Choose arbitrary $k_1, k_2 \geq 1$. Set $k_1' = min(k_1, c_1)$ and $k_2' = min(k_2, c_2)$. Then, $(U_1, U_2)^{(k_1,k_2)} \models \mathsf{E}h(1_2)$ iff $(U_1, U_2)^{(k_1',k_2')} \models \mathsf{E}h(1_2)$ (by the Disjunctive Truncation Lemma) which implies $(U_1, U_2)^{(c_1,c_2)} \models \mathsf{E}h(1_2)$ (by repeated application of the Disjunctive Monotonicity Lemma). Now, by contraposition, $(U_1, U_2)^{(c_1,c_2)} \models \mathsf{A}h(1_2)$ implies $(U_1, U_2)^{(k_1,k_2)} \models \mathsf{A}h(1_2)$. Since k_1, k_2 were arbitrarily chosen, the proof is complete. QED

Next, we transform the given template processes and follow that up with lemmas giving the soundness and completeness proofs for the transformation. Given template processes $U_1, ..., U_k$, define $ReachableStates(U_1, ..., U_k) = (S_1^{'}, ..., S_k^{'})$, where $S_i^{'} = \{ t \,|\, t \in S_i$, *such that for some* $n_1, n_2, ..., n_k \geq 1$, *there exists a computation path of* $(U_1, ..., U_k)^{(n_1,...,n_k)}$, *leading to a global state that contains a local indexed copy of* $t\}$. $\forall j \geq 0$, $\forall l \in [1..k]$, we define P_l^j as follows:

$$P_l^0 = \{\mathsf{i}_l\}.$$
$$P_l^{j+1} = P_l^j \bigcup \{p' : \exists p \in P_l^j : \exists p \xrightarrow{g} p' \in R_l \textit{ and expression } g \textit{ contains a state in } \bigcup_t P_t^j\}.$$

For $l \in [1..k]$, define $P_l = \bigcup_j P_l^j$. Then we have the

Soundness Lemma Given j, for all $l \in [1..k]$, define $a_l = |P_l^j|$. Then, there exists a finite computation sequence $x = x_0, x_1, ..., x_m$ of $(U_1, ..., U_k)^{(a_1,...,a_k)}$, such that $\forall l \in [1..k] : \forall s_l \in P_l^j : (\exists p \in [1..a_l] : x_m(l,p) = s_l^p)$.

Proof
The proof is by induction on j. The base case, $j = 0$, is vacuously true. Assume that the result holds for $j \leq u$ and let $y = y_0, y_1, ..., y_t$ be a computation sequence of $(U_1, ..., U_k)^{(r_1,...,r_k)}$, where $r_l = |P_l^u|$, with the property that $\forall l \in [1..k]$: $\forall s_l \in P_l^u : (\exists p \in [1..r_l] : x_m(l,p) = s_l^p)$.

Now, assume that $P_l^{u+1} \neq P_l^u$, and let $s_l \in P_l^{u+1} \backslash P_l^u$. Furthermore, let (s_l', s_l) be the transition that led to the inclusion of s_l into P_l^{u+1}. Clearly, $s_l' \in P_l^j$. Then, by the induction hypothesis, $\exists q \in [1..r_l] : y_t(l,q)$ is an indexed copy of s_l'. Consider the sequence $y' = y_0', y_1', ..., y_{2t+1}'$ of states of $(U_1, ..., U_k)^{(r_1,...,r_l+1,...,r_k)}$, where for $i \in [1..k] : c \in [1..r_i] : y'(i,c) = y(i,c)(y_t(i,c))^{t+1}$ and $y'(l, r_l+1) = (\mathrm{i}_l^{r_l+1})^t z$, where z is $y(l,q)s_l^{r_l+1}$ with the index q replaced by r_l+1. It can be seen that y' is a valid stuttering computation path of $(U_1, ..., U_k)^{(r_1,...,r_l+1,...,r_k)}$, where y_{2t+1}' has the property that $\forall l \in [1..k] : \forall s_l \in P_l^u : \exists p \in [1..r_l] : y_{2t+1}(l,p) = s_l^p$ and $y_{2t+1}'(l, r_l+1) = s_l^{r_l+1}$. Repeating the above procedure for all states in $P_l^{u+1} \setminus P_l^u$, we get a computation path with the desired property. This completes the induction step and proves the lemma. QED

Completeness Lemma $(S_1^{'}, ..., S_k^{'}) = (P_1, ..., P_k)$.

Proof
By the above lemma, $\forall i \in [1..k] : P_i \subseteq S_i^{'}$. If possible, suppose that $(S_1^{'}, ..., S_k^{'}) \neq (P_1, ..., P_k)$. Then, the set $D = \bigcup_i (S_i^{'} - P_i) \neq \emptyset$. For definiteness, let $s_l \in D \bigcap S_l$. Then by definition of $S_l^{'}$, there exists a finite computation sequence $x = x_0, x_1, ..., x_m$ such that for some i, $x_m(l,i) = s_l^i$. Let $j \in [0..m]$ be the smallest index such that $Set(x_j) \bigcap D \neq \emptyset$. Then, $PathSet(x_0, ..., x_{j-1}) \subseteq \bigcup_i P_i$ which implies that there exists a transition (s_l', s_l) in R_l, with guard g such that $x_{j-1} \models g$. But this implies that for some t, s_l would be included in P_l^t i.e. $s_l \in P_l$, a contradiction to our assumption that $s_l \in D$. Thus $D = \emptyset$ and we are done. QED

We now modify the k-tuple of template processes $(U_1, ..., U_k)$ to get the k-tuple $(U_1^{'}, ..., U_k^{'})$, where $U_i^{'} = (S_i^{'}, R_i^{'}, \mathrm{i}_i)$, with $(s_i, t_i) \in R_i^{'}$ iff guard g_i labelling (s_i, t_i) in U_i contains an indexed copy of a state in $\bigcup_{i \in [1..k]} S_i^{'}$. Furthermore, any transition in the new system is labelled with g_U, a universal guard that evaluates to true irrespective of the current global state of the system. The motivation behind these definitions is that since for any $n_1, n_2, ..., n_k \geq 1$, no indexed copy of states in $S_i \setminus S_i^{'}$ is reachable in any computation of $(U_1, ..., U_k)^{(n_1,...,n_k)}$, we can safely delete these states from their respective template process. Also, any guard of a template process involving only states in $S_i \setminus S_i^{'}$, will then always evaluate to false and hence the transition labelled by this guard will never be fired. This justifies deleting such transitions from the transition graph of respective template processes. This brings us to the following Reduction Result, which by appeal to symmetry yields the Reduction Theorem stated before.

Reduction Result Define $V = U_l^{'}$ if for some $l \in [1..k]$, the transition graph for $U_l^{'}$ has a nontrivial strongly connected component else set $V = U_1^{'}$. Then, $(U_1, ..., U_k)^{(c_1,...,c_k)} \models \mathsf{A}h(1_p)$ iff $(U_p^{'}, V)^{(1,1)} \models \mathsf{A}h(1_1)$, where $c_p = |U_p| + 2$ and $c_i = |U_i| + 1$ for $i \neq p$.

Proof

We show that $(U_1, ..., U_k)^{(c_1,...,c_k)} \models \mathsf{E}h(1_p)$ iff $(U_p^{'}, V)^{(1,1)} \models \mathsf{E}h(1_1)$. For definiteness, let $V = U_r^{'}$.

($\Rightarrow$) Define sequence $u = (\mathsf{i}_r)^\omega$. If $U_r^{'}$ has a nontrivial strongly connected component, then there exists an infinite path v, say, in its transition graph. In that case, reset $u = v$.

Let $x = x_1, x_2, ...$ be a computation sequence of $(U_1, ..., U_k)^{(c_1,...,c_k)}$. Define a formal sequence $y = y_1, y_2, ...$ as follows. Set $y(1,1) = x(p,1)$ and in case $x(p,1)$ is a finite computation sequence of length f, say, set $y(2,1) = (\mathsf{i}_r)^f u$ else set $y(2,1) = (\mathsf{i}_r)^\omega$. To prove that y is a valid computation sequence of $(U_p^{'}, V)^{(1,1)}$, it suffices to show that all transitions of local path $y(1,1)$ are valid. This follows from the definition of $U_1^{'}$ and by noting that all states occuring in $x(p,1)$ are reachable and all transitions in $x(p,1)$ are labelled by guards whose expressions involve a state in $\bigcup_j S_j^{'}$ and hence they occur in $R_p^{'}$.

($\Leftarrow$) By the Soundness and Completeness lemmas, it follows that there exists a finite computation path $u = u_0, u_1, ..., u_m$ of $(U_1, ..., U_k)^{(|U_1|,...,|U_k|)}$ starting at $\mathsf{i}^{(|U_1|,...,|U_k|)}$, such that $\forall j \in [1..k] : \forall q_j \in S_j^{'} : \exists t \in [1..|U_j|] : u_m(j,t) = q_j^t$. Let $x = x_0, x_1, ...$ be a computation path of $(U_p^{'}, V)^{(1,1)}$. Define a formal sequence $y = y_0, y_1, ...$ of states of $(U_1, ..., U_k)^{(c_1,...,c_k)}$ as follows. Set $y(p,1) = ((\mathsf{i}_p)^m)x(1,1)$, $y(r,1) = ((\mathsf{i}_r)^m)x(2,1)$, $\forall z \in [1..|U_p|] : y(p, z+1) = u(p,z)$, $\forall z \in [1..|U_r|] : y(r, z+1) = u(r,z)$, and $\forall j \in [1..k], j \neq p, r : \forall z \in [1..|U_j|] : y(j,z) = u(j,z)(u_m(j,z))^\omega$. Note that, $\forall l \geq m : Set(y_l) = \bigcup_j S_j^{'}$ and hence for all $l \geq m$, all template transitions in $R_1^{'} \cup R_r^{'}$ are enabled in y_l. Thus for all $i \geq m$, all transitions (y_i, y_{i+1}) are valid and hence it follows that y is a stuttering of a valid computation path of $(U_1, ..., U_k)^{(c_1,...,c_k)}$ with local path $y(1,1)$ being a stuttering of local path $x(1,1)$. The path correspondence gives us the result. QED

Finally, we get the

Efficient Decidability Theorem For systems with disjuctive guards and properties of the type $\bigwedge_{i_l} \mathsf{A}h(i_l)$, the PMCP is decidable in time quadratic in the size of the given family $(U_1, ..., U_k)$, where size is defined as $\sum_j(|S_j| + |R_j|)$, and linear in the size of the Büchi Automaton for $\neg h(1_l)$.

Proof
We first argue that we can construct the simplified system U_l' efficiently. By definition, $\forall j \geq 0 : P_l^j \subseteq P_l^{j+1}$. Let $P^i = \bigcup_l P_l^i$. Then, it is easy to see that, $\forall j \geq 0 : P^j \subseteq P^{j+1}$ and if $P^j = P^{j+1}$, then $\forall i \geq j : P^i = P^j$. Also, $\forall i : P^i \subseteq \bigcup_l S_l^{'}$. Thus to evaluate sets P_l^j, for all j, it suffices to evaluate them for values of $j \leq \sum_l |S_l|$. Furthermore, given P_l^j to evaluate P_l^{j+1}, it suffices to make a pass through all transitions leading to states in $S_l \setminus P_l^j$ to check if a guard leading to any of these states contains a state in $\bigcup_l P_l^j$. This can clearly be accomplished in time $\sum_j (|S_j| + |R_j|)$. The above remarks imply that evaluation of sets P_l^j, can be done in time $O((\sum_j (|S_j| + |R_j|))^2)$. Furthermore, given p, whether $U_p^{'}$ has a nontrivial strongly connected component can be decided in time $O(|S_p^{'}| + |R_p^{'}|)$ by constructing all strongly connected components of $U_p^{'}$. Thus, determining whether such a p exists can be done in time $O(\sum_j (|S_j| + |R_j|))$.

The Reduction Theorem reduces the PMCP problem to model checking for the system $(U_l^{'}, V)^{(1,1)}$, where $V = U_r^{'}$ if for some $r \in [i..k]$, the transition graph for $U_r^{'}$ has a nontrivial strongly connected component else $V = U_1^{'}$. Now, $(U_l^{'}, V)^{(1,1)} \models Ah(1_1)$ iff $(U_l^{'}, V)^{(1,1)} \models \neg E \neg h(1_1)$. Thus it suffices to check whether $(U_l^{'}, V)^{(1,1)} \models E \neg h(1_1)$, for which we use the automata-theoretic approach of [20]. We construct a Büchi Automaton $\mathcal{B}_{\neg h}$ for $\neg h(1_1)$, and check that language of the product Büchi Automaton $\mathcal{P}$, of $(U_l^{'}, V)^{(1,1)}$ and $\mathcal{B}_{\neg h}$ is nonempty(cf [14]). Since the nonemptiness check for $\mathcal{P}$ can be done in time linear in the size of $\mathcal{P}$, and the size of $(U_l^{'}, V)^{(1,1)}$ is $O((\sum_j (|S_j| + |R_j|))^2)$, we are done.

4.3 Properties ranging over pairs of processes from two classes

Using similar kinds of arguments as were used in proving assertions in the sections 4.1 and 4.2, we can prove the following results.

Cutoff Theorem
Let f be $\bigwedge_{i_l, j_m} \mathsf{A}h(i_l, j_m)$ or $\bigwedge_{i_l, j_m} \mathsf{E}h(i_l, j_m)$, where h is an LTL\X formula and $l, m \in [1..k]$. Then we have the following

$\forall (n_1, \ldots, n_k) \succeq (1, \ldots, 1) : (U_1, \ldots, U_k)^{(n_1, \ldots, n_k)} \models f$ iff

$\forall (d_1, \ldots, d_k) \preceq (c_1, \ldots, c_k) : (U_1, \ldots, U_k)^{(d_1, \ldots, d_k)} \models f$,

where the cutoff $(c_1, \ldots, c_k)$ is given by $c_l = |U_l| + 2$, $c_m = |U_m| + 2$ and for $i \neq l, m : c_i = |U_i| + 1$.

Reduction Theorem
$(U_1, ..., U_k)^{(c_1, \ldots, c_k)} \models \bigwedge_{i_l, j_m} \mathsf{A}h(i_l, j_m)$ iff $(U_l^{'}, U_m^{'})^{(1,1)} \models \bigwedge_{i_l, j_m} \mathsf{A}h(i_l, j_m)$, where $c_l = |U_l| + 2, c_m = |U_m| + 2$ and $\forall i \neq l, m : c_i = |U_i| + 1$.

Again, we get the analogous Decidability Theorem and Efficient Decidability Theorem. Moreover, we can specialize these results to apply when $l{=}m$. This permits reasoning about formulae of the type $\bigwedge_{i_l \neq j_l} \mathsf{A}h(i_l, j_l)$ or $\bigwedge_{i_l \neq j_l} \mathsf{E}h(i_l, j_l)$, for properties ranging over all pairs of processes in a single class l.

5 Systems with Conjunctive Guards

The development of results for conjunctive guards closely resembles that for disjunctive guards. Hence, for the sake of brevity, we only provide a proof sketch for each of the results.

Conjunctive Monotonicity Lemma
(i) $\forall n \geq 1 : (V_1, V_2)^{(1,n)} \models \mathsf{E}h(1_2)$ implies $(V_1, V_2)^{(1,n+1)} \models \mathsf{E}h(1_2)$.
(ii) $\forall n \geq 1 : (V_1, V_2)^{(1,n)} \models \mathsf{E}h(1_1)$ implies $(V_1, V_2)^{(1,n+1)} \models \mathsf{E}h(1_1)$.

Proof Sketch The intuition behind this lemma is that for any computation x of $(V_1, V_2)^{(1,n)}$, there exists an analogous computation y of $(V_1, V_2)^{(1,n+1)}$ wherein the $(n+1)$st copy of template process V_2 stutters in its initial state and the rest of the processes behave as in x. QED

Conjunctive Bounding Lemma
(i) $\forall n \geq 2|V_2| + 1 : (V_1, V_2)^{(1,n)} \models \mathsf{E}h(1_2)$ iff $(V_1, V_2)^{(1,c_2)} \models \mathsf{E}h(1_2)$, where $c_2 = 2|V_2| + 1$.
(ii) $\forall n \geq 2|V_2| : (V_1, V_2)^{(1,n)} \models \mathsf{E}h(1_1)$ iff $(V_1, V_2)^{(1,2|V_2|)} \models \mathsf{E}h(1_1)$.

Proof Sketch
Let x be an infinite computation of $(V_1, V_2)^{(1,n)}$. Set $v = (\mathsf{i}_2)^\omega$, where i_2 is the initial state of V_2. If none of $x(1,1)$ or $x(2,1)$ is an infinite local computation then there exists $l \neq 1$ such that $x(2,l)$ is an infinite local computation. In that case, reset $v = x(2,l)$. Construct a formal sequence y of $(V_1, V_2)^{(1,c_2)}$ as follows. Set $y(1,1) = x(1,1)$, $y(2,1) = x(2,1)$, $y(2,2) = v$ and $\forall j \in [3..c_2] : y(2,j) = (\mathsf{i}_2)^\omega$. Then, it can be proved that y is a stuttering of a valid infinite computation of $(V_1, V_2)^{(1,c_2)}$.

Now consider the case when $x = x_0 x_1 ... x_d$ is a deadlocked computation sequence of $(V_1, V_2)^{(1,n)}$. Let $S = Set(x_d) \cap S_2$. For each $s \in S$, define an index set I_s as follows. If there exists a unique indexed copy $x_d(2, in)$ of s in x_d set $I_s = \{in\}$ else set $I_s = \{in_1, in_2\}$, where $x_d(2, in_1)$ and $x_d(2, in_2)$ are indexed copies of s and $in_1 \neq in_2$. Let $I = \bigcup_s I_s$. Also, for index j and global state s define $Set(s, j) = \{t | t \in S_1 \cup S_2 \textit{ and } t \textit{ has a copy with index other than } j \textit{ in } s\}$

Construct a formal sequence $y = y_0, ..., y_d$ of states of $(V_1, V_2)^{(1,2|V_2|+1)}$ by projecting each global state x_i onto process 1 coordinate of V_1 and process $index$ coordinate of V_2, where $index = 1$ or $index \in I$. From our construction, it follows that for all j, $Set(y_j) \subseteq Set(x_j)$. Hence all transitions (y_i, y_{i+1}) are valid. Also for each $i \in [1..(2|V_2| + 1)]$, there exists $j \in [1..n]$ such that $y(2, i)$ is a projection of $x(2, j)$. Then, from our construction, it follows that $Set(x_d, j) = Set(y_d, i)$ and thus process V_2^i is deadlocked in y_d iff V_2^j is deadlocked in x_d. Then from the fact that x_d is deadlocked, we can conclude that y_d is a deadlocked state and hence y is a stuttering of a deadlocked computation of $(V_1, V_2)^{(1,c_2)}$.

In both cases, when constructing y from x, we preserved the local computation sequence of process V_2^1. This path correspondence gives us the result. QED

Again as before, the following lemma allows reduction in system size over multiple coordinates simultaneously (2 coordinates for notational brevity).

Conjunctive Truncation Lemma
$\forall n_1, n_2 \geq 1 : (U_1, U_2)^{(n_1,n_2)} \models \mathsf{E}h(1_2)$ iff $(U_1, U_2)^{(n'_1,n'_2)} \models \mathsf{E}h(1_2)$, where $n'_2 = min(n_2, 2|U_2| + 1)$ and $n'_1 = min(n_1, 2|U_1|)$.

Proof Idea
Use the Conjunctive Bounding Lemma and associativity of the || operator. QED

Conjunctive Cutoff Result
Let f be $\bigwedge_{i_l} \mathsf{A}h(i_l)$ or $\bigwedge_{i_l} \mathsf{E}h(i_l)$, where h is a LTL\X formula and $l \in [1..2]$. Then we have the following

$\forall(n_1, n_2) \succeq (1, 1) : (U_1, U_2)^{(n_1,n_2)} \models f$ iff
$\forall(d_1, d_2) \preceq (c_1, c_2) : (U_1, U_2)^{(d_1,d_2)} \models f$,

where the cutoff (c_1, c_2) is given by $c_l = 2|U_l| + 1$, and for $i \neq l : c_i = 2|U_i|$.

Proof Sketch Follows easily from the Truncation Lemma. QED

More generally, for systems with $k \geq 1$ class of processes we have

Conjunctive Cutoff Theorem
Let f be $\bigwedge_{i_l} \mathsf{A}h(i_l)$ or $\bigwedge_{i_l} \mathsf{E}h(i_l)$, where h is a LTL\X formula and $l \in [1..k]$. Then we have the following

$\forall(n_1, ..., n_k) \succeq (1, ..., 1) : (U_1, ..., U_k)^{(n_1,...,n_k)} \models f$ iff
$\forall(d_1, ..., d_k) \preceq (c_1, ...c_k) : (U_1, ..., U_k)^{(d_1,...,d_k)} \models f$,

where the cutoff $(c_1, ..., c_k)$ is given by $c_l = 2|U_l| + 1$, and for $i \neq l : c_i = 2|U_i|$.

Although the above results yield decidability for PMCP in the Conjunctive guards case, the decision procedures are not efficient.

We now show that if we limit path quantification to range over *infinite* paths only (i.e. ignore deadlocked paths); or *finite* paths only; then we can give an efficient decision procedure for this version of the PMCP. We use $\mathsf{A}_{\mathsf{inf}}$ for "for all infinite paths", $\mathsf{E}_{\mathsf{inf}}$ for "for some infinite path", $\mathsf{A}_{\mathsf{fin}}$ for "for all finite paths", and $\mathsf{E}_{\mathsf{fin}}$ for "for some finite path".

Infinite Conjunctive Reduction Theorem
For any LTL\X formula h and $l \in [1..k]$, we have

(a) $\forall(n_1, \ldots, n_k) \succeq (1, \ldots, 1) : (U_1, ..., U_k)^{(n_1,...,n_k)} \models \bigwedge_{i_l} \mathsf{E}_{\mathsf{inf}}h(i_l)$, iff $(U_1, ..., U_k)^{(c_1,...,c_k)} \models \mathsf{E}_{\mathsf{inf}}h(1_l)$;

(b) $\forall(n_1, \ldots, n_k) \succeq (1, \ldots, 1) : (U_1, ..., U_k)^{(n_1,...,n_k)} \models \bigwedge_{i_l} \mathsf{A}_{\mathsf{inf}}h(i_l)$, iff $(U_1, ..., U_k)^{(c_1,...,c_k)} \models \mathsf{A}_{\mathsf{inf}}h(1_l)$,

where $(c_1, ..., c_k) = (\underbrace{1, ..., 2}_{l}, ..., 1)$.

Proof Sketch

To obtain (a), by appeal to symmetry, it suffices to establish that for each $(n_1, \ldots, n_k) \succeq (1, \ldots, 1) : (U_1, ..., U_k)^{(n_1, \ldots, n_k)} \models \mathsf{E}_{\mathsf{inf}} h(1_l)$ iff $(U_1, ..., U_k)^{(c_1, \ldots, c_k)} \models \mathsf{E}_{\mathsf{inf}} h(1_l)$. Using the duality between $\mathsf{A}_{\mathsf{inf}}$ and $\mathsf{E}_{\mathsf{inf}}$ on both sides of the latter equivalence, we can also appeal to symmetry to obtain (b). We establish the latter equivalence as follows.

($\Rightarrow$) Let $x = x_0 \stackrel{b_0, g_0}{\longrightarrow} x_1 \stackrel{b_1, g_1}{\longrightarrow} \ldots$ denote an infinite computation of $(U_1, ..., U_k)^{(n_1, \ldots, n_k)}$, where b_i indicates which process fired the transition driving the system from global states x_i to x_{i+1} and g_i is the guard enabling the transition. Since x is infinite, it follows that there exists some process such that the result of projecting x onto that process results in a stuttering of an infinite local computation of the process. By appeal to symmetry, we can without loss of generality, assume that for each process class U_p, if a copy of U_p in $(U_1, ..., U_k)^{(n_1, \ldots, n_k)}$ has the above property then that copy is in fact the concrete process U_p^1 in case $p \neq l$ and the concrete process U_p^2 in case $p = l$ and local computation $x(l, 1)$ is finite.

Define a (formal) sequence $y = y_0 \stackrel{b'_0, g'_0}{\longrightarrow} y_1 \stackrel{b'_1, g'_1}{\longrightarrow} \ldots$ by projecting each global state x_i onto process 1 coordinate for each class U_p for $p \neq l$ and onto process coordinates 1 and 2 for process class U_l to get a state y_i. We let $b'_i = 1_l$ if $b_i = 1_l$, $b'_i = 2_l$ if $b_i = 2_l$, else set $b'_i = \epsilon$, while g'_i is the syntactic guard resulting from g_i by deleting all conjuncts corresponding to indices not preserved in the projection. Then, by our construction and the fact that x was an infinite computation, we have that y denotes a stuttering of a genuine infinite computation of $(U_1, ..., U_k)^{(c_1, \ldots, c_k)}$. To see this, note that for any i such that $y_i \neq y_{i+1}$, the associated (formal) transitions have their guard g'_i true, since for conjunctive guards g_i and their projections g'_i we have $x_i \models g_i$ implies $y_i \models g'_i$, and can thus fire in $(U_1, ..., U_k)^{(c_1, \ldots, c_k)}$. For any stuttering i where $y_i = y_{i+1}$, the (formal) transition is labelled by $b'_i = \epsilon$.

Thus, given infinite computation path of $(U_1, ..., U_k)^{(n_1, \ldots, n_k)}$, there exists a stuttering of an infinite computation path of $(U_1, ..., U_k)^{(c_1, \ldots, c_k)}$, such that the local computation path of U_l^1 is the same in both. This path correspondence proves the result.

($\Leftarrow$) Let $y = y_0, y_1, ...$ be an infinite computation path of $(U_1, ..., U_k)^{(c_1, \ldots, c_k)}$. Then, consider the sequence of states $= x_0, x_1, ...,$, where $x(l, 1) = y(l, 1)$, $x(l, 2) = y(l, 2)$ and $\forall (k, j) \neq (l, 1), (l, 2) : x(k, j) = (\mathsf{i}_k^j)^\omega$. Let g_i be the guard labelling the transition $s_l^1 \rightarrow t_l^1$ in state y_i. Then all the other processes are in their initial states in x_i, and since the guards do allow initial states of all template processes as "nonblocking" states in that their being present in the global state does not falsify any guards, we have $x_i \models g_i$.

Thus, given infinite computation path y of $(U_1, ..., U_k)^{(c_1, \ldots, c_k)}$, there exists an infinite computation path x of $(U_1, ..., U_k)^{(n_1, \ldots, n_k)}$, such that the local computation path of U_l^1 is the same in both. This path correspondence easily gives us the desired result. QED

In a similar fashion we may prove the following result.

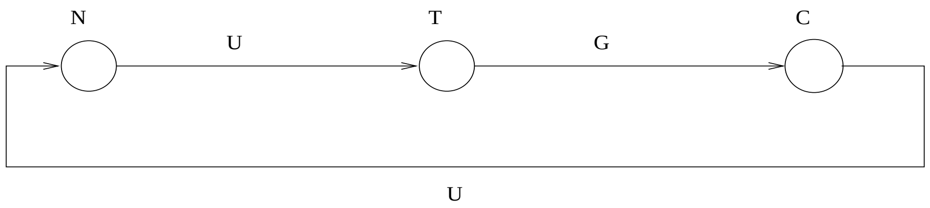
Finite Conjunctive Reduction Theorem
For any LTL\X formula h, and $l \in [1..k]$ we have
(a) $\forall(n_1,\ldots,n_k) \succeq (1,\ldots,1) : (U_1,...,U_k)^{(n_1,\ldots,n_k)} \models \bigwedge_{i_l} \mathsf{E_{fin}} h(i_l)$, iff $(U_1,...,U_k)^{(1,\ldots,1)} \models \mathsf{E_{fin}} h(1_l)$;
(b) $\forall(n_1,\ldots,n_k) \succeq (1,\ldots,1) : (U_1,...,U_k)^{(n_1,\ldots,n_k)} \models \bigwedge_{i_l} \mathsf{A_{fin}} h(i_l)$, iff $(U_1,...,U_k)^{(1,\ldots,1)} \models \mathsf{A_{fin}} h(1_l)$.

Note that the above theorem permits us to verify safety properties efficiently. Informally, this is because if there is a finite path leading to a "bad" state in the system $(U_1,...,U_k)^{(n_1,\ldots,n_k)}$, then there exists a finite path leading to a bad state in $(U_1,...,U_k)^{(1,\ldots,1)}$. Thus, checking that there is no finite path leading to bad state in $(U_1,...,U_k)^{(n_1,\ldots,n_k)}$ reduces to checking it for $(U_1,...,U_k)^{(1,\ldots,1)}$.

We can use this to obtain an Efficient Conjunctive Decidability Theorem. Moreover, the results can be readily extended to formulae with multiple indices as in the disjunctive guards case.

6 Applications

Here, we consider a solution to the *mutual exclusion* problem. The template process is given below. Initially, every process is in local state N, the non-critical



region. $U = T \vee N \vee C$ denotes the universal guard, which is always true independent of the local states of other processes. If a process wants to enter the critical section C, it goes into the trying region T which it can always do since U is always true. Guard $G = N \vee T$, instantiated for process i of n processes, takes the conjunctive form $\bigwedge_{j \neq i}(N_j \vee T_j)$. When G is true, no other process is in the critical section, and the transition from T to C can be taken. Note that all guards are conjunctive with neutral(i.e., non-blocking) initial state N. Thus, by the Finite Conjunctive Reduction Theorem for multi-indexed properties, PMCP for all sizes n with the mutual exclusion property $\bigwedge_{i,j,i\neq j} \mathsf{A_{fin}} \mathsf{G} \neg(C_i \wedge C_j)$ can be reduced to checking a 2-process instance. Using the Conjunctive Cutoff Theorem, the starvation-freedom property $\bigwedge_i \mathsf{A}(\mathsf{G}(T_i \Rightarrow \mathsf{F} C_i))$ can be checked by a 7-process instance. In this simple example, mutual exclusion is maintained but starvation-freedom fails.

7 Concluding Remarks

PMCP is, in general, undecidable[1]. However, under certain restrictions, a variety of positive results have been obtained. Early work includes [15] which uses an abstract graph of exponential size "downstairs" to capture the behaviour of arbitrary sized parameterized asynchronous programs "upstairs" over Fetch-and-Add primitives; however, while it caters for partial automation, the completeness of the method is not established, and it is not clear that it can be made fully automatic. A semi-automated method requiring construction of a *closure process* which represents computations of an arbitrary number of processes is described in [4]; it is shown that, if for some k, $C||U^k$ is appropriately bisimilar to $C||U^{k+1}$, then it suffices to check instances of size at most k to solve the PMCP. But it is not shown that such a cutoff k exists, and the method is not guaranteed to be complete. Kurshan and McMillan [13] introduce the related notion of a *process invariant* (cf. [22]). Ip and Dill [12] describe another approach to dealing with many processes using an abstract graph; it is sound but not guaranteed to be complete; [18] proposes a similar construction for verification of safety properties of cache coherence protocols, which is also sound but not complete. A theme is that most these methods suffer, first, from the drawback of being only partially automated and hence requiring human ingenuity, and, second, from being sound but not guaranteed complete (i.e., a path "upstairs" maps to a path "downstairs", but paths downstairs do not necessarily lift). Other methods can be fully automated but do not appear to have a clearly defined class of protocols on which they are guaranteed to terminate successfully (cf. [5], [21], [19]).

For systems comprised of CCS processes, German and Sistla [10] combine the automata-theoretic method with process closures to permit efficient solution to PMCP for single index properties, modulo deadlock. But efficient solution is only yielded for processes in a single class. Even for systems of the form $C||U^n$ a doubly exponential decision procedure results, which likely limits its practical use. Emerson and Namjoshi [7] show that in a single class (or client-server) synchronous framework PMCP is decidable but with PSPACE-complete complexity. Moreover, this framework is undecidable in the asynchronous case. A different type of parameterized reasoning about time bounds is considered in [9].

In some sense, the closest results might be those of Emerson and Namjoshi [6] who for the token ring model, reduce reasoning, for multi-indexed temporal logic formulae, for rings of arbitrary size to rings up to a small cutoff size. These results are significant in that, like ours, correctness over all sizes holds iff correctness of (or up to) the small cutoff size holds. But these results were formulated only for a single process class and, for a restricted version of the token ring model, namely one where the token cannot be used to pass values. Also, related are the results of Attie and Emerson [2]. In the context of program synthesis, rather than program verification, it is shown how certain 2-process solutions to synchronization problems could be inflated to n-process solutions. However, the correspondence is not an "iff", but is established in only one direction for conjunctive-type guards. Disjunctive guards are not considered, nor are multiple process classes.

We believe that our positive results on PMCP are significant for several reasons. Because PMCP solves (a major aspect of) the state explosion problem and the scalability problem in one fell swoop, many researchers have attempted to make it more tractable, despite its undecidability in general. Of course, PMCP seems to be prone to undecidability in practice as well, as is evidenced by the wide range of solution methods proposed that are only partially automated or incomplete or lack a well-defined domain of applicability. Our methods are fully automated returning a yes/no answer, they are sound and complete as they rely on establishing exact (up to stuttering) correspondences (yes upstairs iff yes downstairs). In many cases, our methods are efficient, making the problem genuinely tractable. An additional advantage, is that downstairs we have a small system of cutoff size that, but for its size, looks like a system of size n. This contrasts with methods that construct an abstract graph downstairs which may have a complex and non-obvious organization.

References

1. K. Apt and D. Kozen. Limits for automatic verification of finite-state concurrent systems. *Information Processing Letters*, 15, pages 307-309, 1986.
2. P.C. Attie and E.A. Emerson. Synthesis of Concurrent Systems with Many Similar Processes. *ACM Transactions on Programming Languages and Systems*, Vol. 20, No. 1, January 1998, pages 51-115.
3. M.C. Browne, E.M. Clarke and O. Grumberg. Reasoning about Networks with Many Identical Finite State Processes. *Information and Control*, 81(1), pages 13-31, April 1989.
4. E.M. Clarke and O. Grumberg. Avoiding the State Explosion Problem in Temporal Logic Model Checking Algorithms. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 294-303, 1987.
5. E.M. Clarke, O. Grumberg and S. Jha. Verifying Parameterized Networks using Abstracion and Regular Languages. In *CONCUR '95: Concurrency Theory, Proceedings of the 6th International Conference*, LNCS 962, pages 395-407, Springer-Verlag, 1995.
6. E.A. Emerson and K.S. Namjoshi. Reasoning about Rings. In *Conference Record of POPL '95: 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 85-94, 1995.
7. E.A. Emerson and K.S. Namjoshi. Automatic Verification of Parameterized Synchronous Systems. In *Computer Aided Verification, Proceedings of the 8th International Conference*. LNCS , Springer-Verlag, 1996.
8. E.A. Emerson and A.P. Sistla. Symmetry and Model Checking. In *Computer Aided Verification, Proceedings of the 5th International Conference*. LNCS 697, Springer-Verlag, 1993.
9. E. Emerson and R. Trefler, Parametric Quantitative Temporal Reasoning. LICS 1999, 336-343.
10. S.M. German and A.P. Sistla. Reasoning about Systems with Many Processes. *J. ACM*, 39(3), July 1992.

11. C. Ip and D. Dill. Better verification through symmetry. In *Proceedings of the 11th International Symposium on Computer Hardware Description Languages and their Applications.*1993.
12. C. Ip and D. Dill, Verifying Systems with Replicated Components in Murphi, pp. 147-158 CAV 1996.
13. R.P. Kurshan and L. McMillan. A Structural Induction Theorem for Processes. In *Proceedings of the Eight Annual ACM Symposium on Principles of Distributed Computing*, pages 239-247, 1989.
14. O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specifications. In *Conference Record of POPL '85: 12nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 97-107, 1985.
15. B. Lubachevsky. An Approach to Automating the Verification of Compact Parallel Coordination Programs I.*Acta Informatica 21*, 1984.
16. K. McMillan, Verification of Infinite State Systems by Compositional Model Checking, CHARME'99.
17. A. Pnueli. The Temporal Logic of Programs. In *Proceedings of the eighteenth Symposium on Foundations of Computer Science.* 1977.
18. F. Pong and M. Dubois. A New Approach for the Verification of Cache Coherence Protocols. *IEEE Transactions on Parallel and Distributed Systems*, August 1995.
19. A. P. Sistla, Parameterized Verification of Linear Networks Using Automata as Invariants, CAV, 1997, 412-423.
20. M. Vardi and P. Wolper. An Automata-theoretic Approach to Automatic Program Verification. In *Proceedings, Symposium on Logic in Computer Science*, pages 332-344, 1986.
21. I. Vernier. Specification and Verification of Parameterized Parallel Programs. In *Proceedings of the 8th International Symposium on Computer and Information Sciences*, Istanbul, Turkey, pages 622-625,1993.
22. P. Wolper and V. Lovinfosse. Verifying Properties of Large Sets of Processes with Network Invariants. In J. Sifakis(ed) *Automatic Verification Metods for Finite State Systems*, Springer-Verlag, LNCS 407, 1989.