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, U.S.A.

January 31, 2000

Abstract

Systems with an arbitrary number of homogeneous processes occur in many applications. The *Parametrized Model Checking Problem* (PMCP) is to determine whether a temporal property is true for every size instance of the system. 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 n 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) a small *cutoff* size c. This establishes decidability of PMCP as it is only necessary 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 homogeneous copies of the class template (e.g., m readers and n writers).

*This work was supported in part by NSF grant CCR-980-4737 and SRC contract 99-TJ-685. The authors can be reached at {emerson,kahlon}@cs.utexas.edu and at http://www.cs.utexas.edu/users/{emerson,kahlon}

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 the system. 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 arbitrary size instances. The general problem, known in the literature as the *Parametrized Model Checking Problem (PMCP)* is the following: to decide whether a temporal property is true of every size instance of a given system. This problem is known to be undecidable in general [1]. However, by imposing certain stipulations on the organization of the processes 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 thru 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 (indexes 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 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 for n-tuples $(m_1, ..., m_k)$ and $(c_1, ..., c_k)$, 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 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 $c_i = |U_i| + 2$, the size (number of local states of template U_i) plus 2. Then, for both conjunctive and disjunctive

guards a cutoff of $(c_1, ..., c_k)$ suffices for all three types of formulas described above. These results give decision procedure for PMCP for conjunctive or disjunctive guards. Since this is 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 $(1, ..., 1)$. In fact, depending on the type of property, we can show that it is possible to simplify the guards to ensure that only one or two classes need be retained. On the other hand for conjunctive guards if we restrict ourselves to model checking over infinite paths, then sharper cutoffs of the form (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 modeled as *synchronization skeletons* (cf. [6]). Here, an individual concrete process has a transition of the form $l \xrightarrow{g} m$ indicating that the process can transit form 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 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 = (a copy of) an interval $\{1, \ldots, c\}$ of natural numbers, denoted $\{1_j, \ldots, c_j\}$ 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 $\{1, \ldots n_1\}$ thru 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 $(U_1, ..., U_k)^{(n_1, ..., n_k)}$ 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 edges R_l, and an initial (local) state i_l. Each transition in R_l is labelled with a guard, a boolean expressions over atomic propositions corresponding to the 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, the guard of the 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 $(a_1 + ... + b_1) \bigvee ... \bigvee (a_k + ... + b_k)$, where the various $a_l, ...b_l$ are (propositions identified with the) local states of template U_l, label each transition

[1] E.g., if I_1 is a copy of $\{1, 2, 3\}$, the 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.

$(s_l, t_l) \in R_l$. In concrete process i of class l, U_l^i, in 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

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

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

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

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

Note that the initial local states of processes must be present in these guards. Thus, the initial state of a process has a "neutral" character so that when another process, say, r is in its initial state, it does not prevent a move by given process i. This natural condition permits modeling 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, $(U_1, ..., U_k)^{(n_1,...,n_k)} = (S^{(n_1,...,n_k)}, R^{(n_1,...,n_k)}, \mathrm{i}^{(n_1,...,n_k)})$ be the global state graph of the system instance $(n_1, n_2, ..., n_k)$. A state $s \in S^{(n_1,...,n_k)}$ is an $(n_1 + ... + n_k)$-tuple $(u_1^1, ..., u_1^{n_1}, ..., u_k^{n_k})$ 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 $\mathrm{i}^{(n_1,...,n_k)} = (\mathrm{i}_1^1, ..., \mathrm{i}_k^{n_k})$. A global transition $(s, t) \in R^{(n_1,...,n_k)}$ iff t results from s by firing some 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, $s(i, l) = u_l^i$, $t(i, l) = v_l^i$, 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 $\mathrm{i}^{(n_1,...,n_k)}$.

Finally, for global state s, define $Set(s) = \{t \mid s \text{ contains an indexed local copy of t}\}$. 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 $P_j = (x_j)^r$. 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 Appeals to Symmetry

We can exploit symmetry inherent in the system model and the properties in the same spirit characterized as "state symmetry" [9] (cf. [18], [13]) to simplify our proof obligation. To establish formulas 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 formulas 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 satisfies the property, for all i in $[1..n]$. Consult the appendix for details.

4 Systems with Disjunctive Guards

In this section we show how, for systems with disjunctive guards, to reduce the PMCP to model checking systems of size 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 it is not efficient decidability. We go on to show that in the case of universal-path-quantified specification formulas ($\mathsf{A}h$), efficient decidability can be obtained.

4.1 Properties ranging over all processes in a single class: $\bigwedge_{i_l} g(i_l)$

We will first establish the

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

$\forall(n_1, \ldots, n_k) \succeq (1, \ldots, 1)$: $(U_1, \ldots, U_k)^{(n_1,\ldots,n_k)} \models f$ 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 $c_l = |U_l| + 1$, and for $i \neq l : c_i = |U_i|$.

As a corollary we will have the:

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

Proof idea By the 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.

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 We formalize the intuition that, in a system comprised of processes with disjunctive guards, if there is a certain computation, then in the system resulting from adding an aditional process, there is an analogous computation. See the appendix for details. QED

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

Bounding Lemma

(i) $\forall n \geq |S_2| + 1, (V_1, V_2)^{(1,n)} \models \mathsf{E}h(1_2)$ iff $(V_1, V_2)^{(1,c_2)} \models \mathsf{E}h(1_2)$, where $c_2 = |S_2| + 1$.

(ii) $\forall n \geq |S_2|, (V_1, V_2)^{(1,n)} \models \mathsf{E}h(1_1)$ iff $(V_1, V_2)^{(1,|S_2|)} \models \mathsf{E}h(1_1)$.

Proof (i): ($\Rightarrow$) Let $x = x_0 x_1 ...$, be a valid computation sequence of $(V_1, V_2)^{(1,n)}$. Define $Reach \subseteq S_2$, to be the set of all local states(ignoring indices) of V_2 occuring in the local computation sequences of process $V_2^j, j \in [2..n]$, in x. It is clear that for each state $t \in Reach$, there exists a finite local computation $t_1, t_2, ..., t_m$, say, of minimal length ending in t. Then let $MinLength(t)$ denote m and $MinComputation(t)$ denote $t_1, t_2, ..., t_{m-1}, (t_m)^\omega$. For definiteness, let $Reach = \{s_1, ..., s_t\}$.

Define $y = y_0 y_1 ...$, where $y(1,1) = x(1,1), y(2,1) = x(2,1)$ and $\forall j \in [1..m] : x(2, j+1) = MinComputation(s_j)$. Note that in assigning $MinComputation(s_j)$ to $x(2, j+1)$, we have to index all states in it by $j+1$. Then, we claim that y is a valid "stuttering" computation sequence of $(V_1, V_2)^{(1,|S_2|+1)}$.

To prove this, it suffices to show that for any any i such that $y_i \neq y_{i+1}$, the guard g_i labelling the fired transition is satisfied by y_i i.e. $y_i \models g_i$. From our construction of y, it follows that there exists a corresponding transition from x_i to x_{i+1} labelled by the guard $g_i^{'}$, say. Then, it is plain that $x_i \models g_i^{'}$. Now, $t \in Set(x_i)$ implies that $MinLength(t) \leq i$. Also, $t \in Set(x_i)$ implies that $t \in Reach$ i.e. $t = s_q$ for some $q \in [1..m]$. Then, $y_i(2, q+1)$ is an indexed copy of s_q, i.e. $t \in Set(y_i)$. Thus, $t \in Set(x_i)$ implies that $t \in Set(y_i)$ i.e. $Set(x_i) \subseteq Set(y_i)$ and hence, by the existential nature of guards, it follows that $y_i \models g_i$.

But, by our construction, $y(2,1) = x(2,1)$. Thus we have proved that for every computation sequence x of $(V_1, V_2)^{(1,n)}$, there exists a computation sequence y of $(V_1, V_2)^{(1,c_2)}$, such that the local computation sequences of process V_2^1 are same in both x and y. ¿From this path correspondence, the result follows easily.

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

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

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

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 $n_2^{'} = min(n_2, |S_2| + 1)$ and $n_1^{'} = min(n_1, |S_1|)$.

Proof If $n_2 > |S_2|+1$, 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 Bounding Lemma) iff $(U_1, U_2)^{(n_1,n_2^{'})} \models \mathsf{E}h(1_2)$.

If $n_1 \leq |S_1|$, 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 $(V_1, V_2)^{(1,|S_1|)} \models \mathsf{E}h(1_1)$ (by the 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 Truncation Lemma is the following

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

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

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

Proof By appeal to symmetry and the fact that A and E are duals, it suffices to prove the result for formulas 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 $n_1^{'} = min(n_1, |S_1|)$, $n_2^{'} = min(n_2, |S_2|+1)$. 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 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, $k \geq 1$, 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: $\bigwedge \mathsf{A}h$

It can be shown that "for some future" properties, corresponding to formulas of the type $\bigwedge \mathsf{E}h$, the reduction entailed in the previous result is, in general, the best possible. (See the appendix in the full paper.)

However, for universal-path-quantified single index properties, $\bigwedge_{i_l} \mathsf{A}h(l, i_l)$, it is possible to be much more efficient. We will establish the

Reduction Theorem $(U_1, ..., U_k)^{(c_1,...,c_k)} \models \bigwedge_{i_l} \mathsf{A}h(i_l)$ iff $U_l^{'(1)} \models \mathsf{A}h(i_1)$, where $c_l = |S_l| + 1$ and $c_i = |S_i|$ 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 formulas 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 one or two template processes.

To this end, we first prove the following lemma which states that the PMCP problem for the above reduces to model checking them for just the *single* system instance of size equal 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 $c_1 = |S_1|, c_2 = |S_2| + 1$.

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 $k_1^{'} = min(k_1, c_1), k_2^{'} = min(k_2, c_2)$. 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 Truncation Lemma) implies $(U_1, U_2)^{(c_1,c_2)} \models \mathsf{E}h(1_2)$ (by 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 it 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.

$P_l^0 = \{\mathsf{i}_l\}$.

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

Soundness Lemma Let $a_l = |P_l^j|$. 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 $\forall s_l \in P_l^j : (\exists p \in [1..a_l] : x_m(l, p) = s_l^p)$.

Proof See appendix. QED

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

Proof See appendix. QED

We now modify the k-tuple of template processes $(U_1, ..., U_k)$ to get the k-tuple $(U_1^{'}, ..., U_k^{'})$, where $U_i^{'} = (S_i^{'}, R_i^{'}, \mathsf{i}_i)$, with $(s_i, t_i) \in R_i^{'}$ iff the 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 global state the system is in. The motivation behind these definitions is that since for any $n_1, n_2, ..., n_k \geq 1$, no indexed copy of states in $S_i \setminus R_i$ 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 $S_i \setminus R_i$, will 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 $(U_1, ..., U_k)^{(c_1,...,c_k)} \models \mathsf{A}h(1_p)$ iff $U_p^{'(1)} \models \mathsf{A}h(1_1)$, where $c_p = |S_p| + 1$ and $c_i = |S_i|$ for $i \neq p$.

Proof We show that $(U_1, ..., U_k)^{(c_1,...,c_k)} \models \mathsf{E}h(1_p)$ iff $U_p^{'(1)} \models \mathsf{E}h(1_1)$.

($\Rightarrow$) Let (s_p^1, t_p^1) be any transition of U_p^1 in $(U_1, ..., U_k)^{(c_1,...,c_k)}$ labelled by guard g, say. Then, since by definition of $S_i^{'}$, $Set(s) \subseteq \bigcup_i S_i^{'}$ for any global state s of $(U_1, ..., U_k)^{(c_1,...,c_k)}$, therefore g must have an indexed copy of a state in $\bigcup_i S_i^{'}$, which implies that (s_p, t_p) is a transition of $U_p^{'(1)}$. Also, $\mathsf{i}_p \in S_p^{'}$. It follows that for any computation sequence x of $(U_1, ..., U_k)^{(c_1,...,c_k)}$, there exists a computation sequence y of $U_p^{'}$, such that $x(p, 1)$ and y are same up to re-indexing of the states. From this, the result follows easily.

($\Leftarrow$) We define a relation "$\sqsubseteq$" from $U_p^{'}(1)$ to $(U_1, ..., U_k)^{(c_1,...,c_k)}$ as follows: for $s \in S_p^{'}, t \in S^{(c_1,...,c_k)}$, $s \sqsubseteq t$ iff $s = t(p, c_p)$, modulo indices, and $\forall j \in [1..k] : \forall q_j \in S_j^{'} : \exists u \in [1..|S_j|] : t(j, u) = q_j$. Thus, $s \sqsubseteq t$, implies that $\bigcup_i S_i^{'} \subseteq Set(t)$.

Now, let (s, u) be a transition of $U_p^{'(1)}$ labelled with guard $g^{'}$, say. By our construction, the expression for $g^{'}$ must contain some state in $\bigcup_i S_i^{'}$. But since $\bigcup_i S_i^{'} \subseteq Set(t)$, it follows that the transition $(s(p, c_p), u(p, c_p))$ can be fired, to reach state v of $(U_1, ..., U_k)^{(c_1,...,c_k)}$, with the property that $u = v(p, c_p)$, modulo indices, and $\forall j \in [1..k] : \forall q_j \in S_j^{'} : \exists u \in [1..|S_j|] : t(j, u) = q_j$. Hence $u \sqsubseteq v$.

Also, it follows from the Soundness and Completeness Lemmas, that there exists a finite computation path $x = x_0, x_1, ..., x_m$ of $(U_1, ..., U_k)^{(c_1,...,c_k)}$ starting at $\mathsf{i}^{(c_1,...,c_k)}$, such that $x(p, c_p) = (\mathsf{i}_p^{c_p})^{m+1}$ and $\forall j \in [1..k] : \forall q_j \in S_j^{'} : \exists u \in [1..|S|_j] : x_m(j, u) = q_j$.

So, for each computation path y of $(U_p^{'})^{(1)}$, there exists a computation path z of $(U_1, ..., U_k)^{(c_1,...,c_k)}$,

such that, modulo stuttering, $z(p, c_p)$ is the same as y, up to re-indexing of states. This completes the proof of the result. QED

Finally, we get the

Efficient Decidability Theorem For systems with disjunctive 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 See the appendix. QED

4.3 Properties ranging over pairs of processes from two classes l, m: $\bigwedge_{i_l, j_m} g(i_l, j_m)$

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 $\bigwedge_{i_l,j_m} \mathsf{E}h(i_l, j_m)$, for LTL\X formula h and $l \in [1..k]$. Then we have the following equivalence:

$\forall(n_1, \ldots, n_k) \succeq (1, \ldots, 1)$: $(U_1, \ldots, U_k)^{(n_1,\ldots,n_k)} \models f$ 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 $c_l = |U_l| + 1$, $c_m = |U_m| + 1$ and for $i \neq l, m : c_i = |U_i|$.

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 $c_l = |S_l| + 1, c_m = |S_m| + 1$ and $\forall i \neq l, m : c_i = |U_i|$.

Again we get the analogous Decidability Theorem and Efficient Decidability Theorem. Moreover, we can specialized these results to apply when l=m. This permits reasoning about formulas f being $\bigwedge_{i_l \neq j_l} \mathsf{A}h(i_l, j_l)$ 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 x' 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) $\forall n \geq |S_2| + 1, (V_1, V_2)^{(1,n)} \models \mathsf{E}h(1_2)$ iff $(V_1, V_2)^{(1,c_2)} \models \mathsf{E}h(1_2)$, where $c_2 = |S_2| + 1$.

(ii) $\forall n \geq |S_2|, (V_1, V_2)^{(1,n)} \models \mathsf{E}h(1_1)$ iff $(V_1, V_2)^{(1,|S_2|)} \models \mathsf{E}h(1_1)$.

Proof Sketch For an "infinite" computation x of $(V_1, V_2)^{(1,n)}$ we can construct an infinite computation X' of $(V_1, V_2)^{(1,c_2)}$ by letting process V_2^1 behave as in x and in case there exists another process that performs an infinite local computation by letting that process behave as before. We let the rest of the processes stutter in their initial states. Then it can be proved that x' is a stuttering of a valid infinite computation of $(V_1, V_2)^{(1,c_2)}$.

In case $x = x_0x_1...x_d$ is a deadlocked computation sequence of $(V_1, V_2)^{(1,n)}$, we construct a deadlocked computation x' of $(V_1, V_2)^{(1,c_2)}$ as follows. For each state s in $Set(x_d) \setminus \{x_d(2, 1)\}$ pick a process P_s of $(V_1, V_2)^{(1,n)}$ in local state s in x_d and make P_s behave as in x. Also let process V_2^1 behave as in x. Then one can show that x' is a stuttering of a deadlocked computation of $(V_1, V_2)^{(1,c_2)}$.

Note that in both cases, when constructing x' from x, we preserved the local computation sequence of process V_2^1. This easily gives us the result. QED

Again as before, the following Truncation 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 $n_2' = min(n_2, |S_2| + 1)$ and $n_1' = min(n_1, |S_1|)$.
Proof Idea Use the Reduction Lemma and associativity of the $||$ operator. QED
Conjunctive Cutoff Result Let f be $\bigwedge_{i_l} \mathsf{A}h(i_l)$ or $\bigwedge_{i_l} \mathsf{E}h(i_l)$, for any LTL\X formula h and $l \in [1..2]$. Then we have the following equivalence:

$\forall(n_1, n_2) \succeq (1, 1)$: $(U_1, U_2)^{(n_1,n_2)} \models f$ 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 $c_l = |U_l| + 1$, and for $i \neq l : c_i = |U_i|$.
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 $\bigwedge_{i_l} \mathsf{E}h(i_l)$, for any LTL\X formula h and $l \in [1..k]$. Then we have the following equivalence:

$\forall(n_1, ..., n_k) \succeq (1, ..., 1)$: $(U_1, ..., U_k)^{(n_1,...,n_k)} \models f$ 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 $c_l = |U_l| + 1$, and for $i \neq l : c_i = |U_i|$.

Although the above results yield decidability for PMCP in the Conjunctive guards case, it is not efficient decidability.

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 $\mathsf{A_{inf}}$ for "for all infinite paths", $\mathsf{E_{inf}}$ for "for some infinite path", $\mathsf{A_{fin}}$ for "for all finite paths", and $\mathsf{E_{fin}}$ for "for some finite path".
Infinite Conjunctive Reduction Theorem For any LTL\X formula h, and $l \in [1..k]$ we have:

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

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

Proof Sketch To obtain (a), by appeal to symmetry, it suffices to establish that for each $(n_1, \ldots, n_k) \succeq (1, \ldots, 1)$, $(U_1, ..., U_k)^{(n_1,...,n_k)} \models \mathsf{E}h(1_l)$ iff $(U_1, ..., U_k)^{(1,...,1)} \models \mathsf{E}h(1_l)$. Using the duality between $\mathsf{A_{inf}}$ and $\mathsf{E_{inf}}$ 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 $x = x_0 \xrightarrow{b_0,g_0} x_1 \xrightarrow{b_1,g_1} \ldots$ denote an "infinite" computation of $(U_1, ..., U_k)^{(n_1,...,n_k)}$, 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 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 $(U_1, ..., U_k)^{(n_1,...,n_k)}$ has the above property then that copy is in fact the concrete process U_p^1.

Define a (formal) sequence $y = y_0 \xrightarrow{b_0',g_0'} y_1 \xrightarrow{b_1',g_1'} \ldots$ by projecting each global state x_i onto the process 1 coordinate for each class to get a state y_i and by letting $b_i' = 1_l$ if $b_i = 1_l$ else ϵ, 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 $(U_1, ..., U_k)^{(1,...,1)}$. To see this, note that for any i such that $y_i \neq y_{i+1}$, the associated (formal) transitions labelled with $b_i' = 1_l$ 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 $(U_1, ..., U_k)^{(1,...,1)}$. For any stuttering i where $y_i = y_{i+1}$, the (formal) transition is labelled by $b_i' = \epsilon$. Thus, we have shown that for every infinite computation path of $(U_1, ..., U_k)^{(n_1,...,n_k)}$, there exists a stuttering of an infinite computation path of $(U_1, ..., U_k)^{(1,...,1)}$, 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 a valid infinite computation path of $(U_1, ..., U_k)^{(1,...,1)}$. Then consider

the sequence of states $= x_0, x_1, ...,$, where $x(l,1) = y(l,1)$, and $\forall (k,j) \neq (l,1) : x(k,j) = (i_k^j)^\omega$. Let g_i be the guard labelling the transition $s_l^1 \rightarrow t_l^1$ in state σ_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 process as "nonblocking" states in that there being present the global state does not falsify any guards, we have $x_i \models g_i$. Thus, we have shown that for every infinite computation path y of $(U_1, ..., U_k)^{(1,...,1)}$, there exists an infinite computation path x of $(U_1, ..., U_k)^{(n_1,...,n_k)}$, 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) $\forall (n_1, \ldots, n_k) \succeq (1, \ldots, 1)$ $(U_1, ..., U_k)^{(n_1,...,n_k)} \models \bigwedge_{i_l} \mathsf{E_{fin}} h(i_l)$, iff $(U_1, ..., U_k)^{(1,...,1)} \models \mathsf{E_{fin}} h(1_l)$;

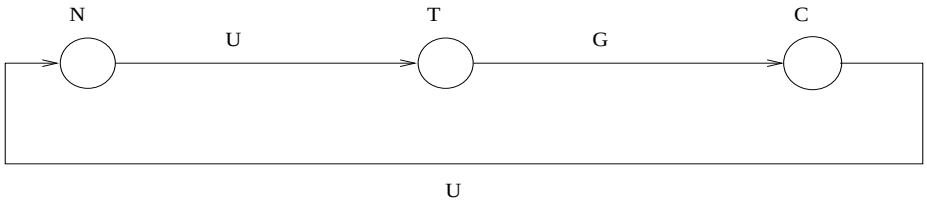
(b) $\forall (n_1, \ldots, n_k) \succeq (1, \ldots, 1)$ $(U_1, ..., U_k)^{(n_1,...,n_k)} \models \bigwedge_{i_l} \mathsf{A_{fin}} h(i_l)$, iff $(U_1, ..., U_k)^{(1,...,1)} \models \mathsf{A_{fin}} h(1_l)$.

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 $(U_1, ..., U_k)^{(n_1,...,n_k)}$, then there exists a finite path leading to a bad state in $(U_1, ..., U_k)^{(1,...,1)}$. Thus checking that there is no finite path leading to bad state in $(U_1, ..., U_k)^{(n_1,...,n_k)}$ reduces to checking it for $(U_1, ..., U_k)^{(1,...,1)}$.

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

6 Applications

In the full paper, we will apply our method to the readers-writers problem, and a cache coherence protocol. 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. $U = T + N + C$ denotes the "universal" guard, which is 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 $G = N + T$, instantiated for process i of n processes, takes the conjunctive form $\wedge_{j \neq i}(N_j + T_j)$. 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 the guards are conjunctive with neutral (i.e., non-blocking) initial state N. Thus, by the the Finite Conjunctive Reduction Theorem for multi-indexed properties, PMCP for all sizes n with mutual exclusion property $\bigwedge_{i,j,i \neq j} \mathsf{A_{fin}G}\neg(C_i \wedge C_j)$ 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 reduced to checking a 4 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 various restrictions, a variety of positive results have been obtained. Early work includes [16] which uses an exponential-size abstract graph "downstairs" to capture then behavior 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 [14] introduce the related notion of a *process invariant* (cf. [24]). Ip and Dill [13] describe another approach to dealing with many processes using an abstract graph; it is sound but not guaranteed to be complete; [20] 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], [23], [21]).

For systems with CCS processes German and Sistla [11] combine 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 double exponential decision procedure results, which likely limits its practical use. Emerson and Namjoshi [8] 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. bounds in considered in [10].

In some sense, the closest results might be those of Emerson and Namjoshi [7], who for the token ring model, reduce reasoning, for multi-indexed temporal logic formulas, 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 correpondence is not an "iff", but is established in only one direction for conjuntive-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 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) correpondences (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 looks just like a system of size n, but for its size. This constrasts 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 Parametrized 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. Temporal and Modal Logic. In *Handbook of Theoretical Computer Science*, Vol. B,(J. van Leeuwen, ed.), Elsevier/North Holland, pages 997-1072, 1991.

[7] 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.

[8] abstract E.A. Emerson and K.S. Namjoshi. Automatic Verification of Parameterized Synchronous Systems. CAV, 1996.XSXS

[9] E.A. Emerson and A.P. Sistla. Symmetry and Model Checking. *Formal Methods in Systems Design*, 1996.

[10] E. Emerson and R. Trefler, Parametric Quantitative Temporal Reasoning. LICS 1999: 336-343.

[11] S.M. German and A.P. Sistla. Reasoning about Systems with Many Processes. *J. ACM*,39(3), July 1992.

[12] 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.

[13] C. Ip, D. Dill, Verifying Systems with Replicated Components in Murphi, pp. 147-158 CAV 1996.

[14] 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.

[15] 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.

[16] B. Lubachevsky. An Approach to Automating the Verification of Compact Parallel Coordination Programs I.*Acta Informatica 21*, 1984.

[17] Z. Manna and A. Pnueli.*Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, 1992.

[18] K. McMillan, Verification of Infinite State Systems by Compositional Model Checking, CHARME'99.

[19] A. Pnueli. The Temporal Logic of Programs. In *Proceedings of the eighteenth Symposium on Foundations of Computer Science*. 1977.

[20] F. Pong and M. Dubois. A New Approach for the Verification of Cache Coherence Protocols. *IEEE Transactions on Parallel and Distributed Systems*, August 1995.

[21] A. P. Sistla, Parametrized Verification of Linear Networks Using Automata as Invariants, CAV, 1997, 412-423.

[22] 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.

[23] I. Vernier. Specification and Verification of Parametrized Parallel Programs. In *Proceedings of the 8th International Symposium on Computer and Information Sciences*, Istanbul, Turkey, pages 622-625,1993.

[24] 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.

Optional Appendix

A3 Appeals to Symmetry

The aim of the following lemmas is to exploit symmetry inherent in the system model and the properties in the spirit of "state symmetry" codified by Emerson-Sistla [9]. Let $Sym\ n$ denote the set of all permutations over the set $[1..n]$.

Lemma $\forall n_l \geq 1,\ (U_1, ..., U_k)^{(n_1,...,n_k)} \models \bigwedge_{j_l} f(j_l)$ iff $(U_1, ..., U_k)^{(n_1,...,n_k)} \models f(1_l)$.
Proof ($\Rightarrow$) Follows directly from the definition of $\bigwedge_{j_l}$.

($\Leftarrow$) $(U_1, ..., U_k)^{(n_1,...,n_k)} \models f(1_l)$ implies that $\pi_l((U_1, ..., U_k)^{(n_1,...,n_k)}) \models f(\pi_l(1_l))$, where $\pi_l \in Sym\ n_l$ just permutes copies of the template process U_l, leaving the other processes unchanged. Since the initial state $\mathrm{i}^{(n_1,...,n_k)}$ is completely symmetric, $\pi_l(\mathrm{i}^{(n_1,...,n_k)}) = \mathrm{i}^{(n_1,...,n_k)}$. Also, because of the symmetric nature of guards, for both the disjunctive and the conjunctive cases, it is not hard to see that $\pi((U_1, ..., U_k)^{(n_1,...,n_k)}) = (U_1, ..., U_k)^{(n_1,...,n_k)}$. Thus, for any $\pi_l \in Sym\ n_l$, we have that $(U_1, ..., U_k)^{(n_1,...,n_k)}, \models f(\pi_l(1_l))$. So, given i_l, by choosing π_l such that $\pi_l(1_l) = i_l$, we see that $(U_1, ..., U_k)^{(n_1,...,n_k)} \models f(i_l)$. This implies that $\forall i_l,\ (U_1, ..., U_k)^{(n_1,...,n_k)} \models f(i_l)$ i.e. $(U_1, ..., U_k)^{(n_1,...,n_k)} \models \bigwedge_{j_l} f(j_l)$. QED

Similarly, one can show that

Lemma $\forall n_l \geq 2,\ (U_1, ..., U_k)^{(n_1,...,n_k)} \models \bigwedge_{i_l \neq j_l} f(i_l, j_l)$ iff $(U_1, ..., U_k)^{(n_1,...,n_k)} \models f(1_l, 2_l)$.

Lemma $\forall n_l, n_m \geq 1,\ (U_1, ..., U_k)^{(n_1,...,n_k)} \models \bigwedge_{i_l, j_m} f(i_l, j_m)$ iff $(U_1, ..., U_k)^{(n_1,...,n_k)} \models f(1_l, 1_m)$.

A direct consequence of the above lemmas is that to prove our results about properties involving formulas 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 formulas replaced by $f(1_l)$, $f(1_l, 2_l)$ and $f(1_l, 1_m)$ respectively.

A4 Proofs for Disjunctive Guard Section

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

Proof (i) We define define a relation "$\sqsubseteq$" from $(V_1, V_2)^{(1,n)}$ to $(V_1, V_2)^{(1,n+1)}$ as follows: for $s \in S^{(1,n)}, t \in S^{(1,n+1)}$, $s \sqsubseteq t$ iff $s(1,1) = t(1,1)$, $t(2,1) = t(2,1)$ and $\forall j \in [2..n] : s(2,j) = t(2,j)$.

Clearly, $\mathrm{i}^{(1,n)} \sqsubseteq \mathrm{i}^{(1,n+1)}$. Suppose that $\sqsubseteq\ t$ and let u be such that (s, u) is a transition of $(V_1, V_2)^{(1,n)}$. Then, there exist $l \in [1..2]$ and $i \in [1..n]$, such that we transit to u from s by firing $(s(l,i), u(l,i))$. The definition of "$\sqsubseteq$", implies that $Set(s) \subseteq Set(t)$. Consider the corresponding transition from $t(l,k)$ in $(V_1, V_2)^{(1,n+1)}$ and let it be labelled by guard g. ¿From from the existential nature of guard g, and the fact that $Set(s) \subseteq Set(t)$, it follows that $t \models g$. Then, if by firing the transition from $t(l,i)$, we reach state $v \in S^{(1,n+1)}$, it is easy to check that $u \sqsubseteq v$.

Thus, "$\sqsubseteq$" is a simulation relation from $(V_1, V_2)^{(1,n)}$ to $(V_1, V_2)^{(1,n+1)}$ with the property that, $s \sqsubseteq t$ implies $s(2,1) = t(2,1)$. This enables us to exhibit, for every compuatation sequence x of $(V_1, V_2)^{(1,n)}$, a computation sequence y of $(V_1, V_2)^{(1,n+1)}$, such that the local computation sequences of process V_2^1 are same in both x and y. ¿From this path correspondence, the result follows easily.

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

Soundness Lemma Let $a_l = |P_l^j|$. 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 $\forall s_l \in P_l^j : (\exists p \in [1..a_l] : x_m(l,p) = s_l^p)$.

Proof The proof is by induction on j. The base case, $j = 0$, is vacuously true. Assume that the result holds for $j \sqsubseteq u$ 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 $\forall s_l \in P_l^u : (\exists p \in [1..r_l] : x_m(l,p) = s_l^p)$.

Now, assume that $P_l^{u+1} \neq P_l^u$, and let $s_l \in P_l^{u+1} \setminus P_l^u$. 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, $\exists q : y_t(l,q)$ 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 $y^{'}(l, r_l+1) = (_l^{r_l+1})^t z$, 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 $\forall s_l \in P_l^u, \exists p : y_{2t+1}(l,p) = s_l^p$ and $y_{2t+1}^{'}(l, r_l+1) = s_l^{r_l+1}$. Repeat the above procedure for all states in $P_l^{u+1} \setminus P_l^u$ to 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, $\forall i, i \in [1..k] : P_i \subseteq S_i^{'}$. If possible, suppose that $(S_1^{'}, ..., S_k^{'}) \neq (P_1, ..., P_k)$. Then, the set $D = \bigcup_i (S_i{'} - P_i) \neq \emptyset$. Let $s_l \in D \bigcap S_l$. Then by definition, 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 $\sigma_{j-1} \models g$. But this implies that s_l would be included in P_l^t for some 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

Efficient Decidability Theorem For systems with disjuctive guards and properties of the type $\bigwedge_{i_l} Ah(i_l)$, 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 $i \geq j : P^i = P^j$. 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 $j \sqsubseteq \sum_l |S_l|$. Furthermore, given P_l^j to evaluate P_l^{j+1}, it suffices to make a pass through all the transitions leading to states in $S_l \setminus P_l^{j+1}$ to check if a guard leading to any of these states conatains 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 $O((\sum_j (|S_j| + |R_j|))^2)$.

The Reduction Theorem reduces the PMCP problem to model checking for a system containing just one copy of the modified template process $U_l^{'}$. Now, $U_l^{'} \models Ah(1_1)$ iff $U_l^{'} \models \neg E \neg h(1_1)$. Thus it suffices to check whether $U_l^{'} \models E \neg h(1_1)$, for which we use the automata-theoretic approach of [22]. 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 $(U_l^{'})^{(1)}$ and $\mathcal{B}_{\neg h}$ is non-empty(cf [15]). Since the nonemptiness check for $\mathcal{P}$ can be done in time linear in the size of $\mathcal{P}$, we are done. QED