Model Checking Guarded Protocols

E. Allen Emerson and Vineet Kahlon*
Department of Computer Sciences,
The University of Texas at Austin, U.S.A.

Abstract

The Parameterized Model Checking Problem *(PMCP) is to decide whether a temporal property holds for a uniform family of systems, $C||U^n$, comprised of a control process, C, and finitely, but arbitrarily, many copies of a user process, U, executing concurrently with interleaving semantics. We delineate the decidability/undecidability boundary of the PMCP for all possible systems that arise by letting processes coordinate using different subsets of the following communication primitives: conjunctive boolean guards, disjunctive boolean guards, pairwise rendezvous, asynchronous rendezvous and broadcast actions. Our focus will be on the following linear time properties: (***p1***) LTL$\backslash$X formulae over C, (***p2***) LTL formulae over C, (***p3***)* regular *properties specified as regular automata, and (***p4***)* ω-regular *properties specified as ω-regular automata.*

We also establish a hierarchy based on the relative expressive power of the primitives by showing that disjunctive guards and pairwise rendezvous are equally expressive, in that we can reduce the PMCP for regular and ω-regular properties for systems with disjunctive guards to ones with pairwise rendezvous and vice versa, but that each of asynchronous rendezvous and broadcasts is strictly more expressive than pairwise rendezvous (and disjunctive guards). Moreover, for systems with conjunctive guards, we give a simple characterization of the decidability/undecidability boundary of the PMCP by showing that allowing stuttering sensitive properties bridges the gap between decidability (for **p1***) and undecidability (for* **p2***). A broad framework for modeling snoopy cache protocols is also presented for which the PMCP for* **p3** *is decidable and that can model all snoopy cache protocols given in [13] thereby overcoming the undecidability results.*

1 Introduction

Systems with finite but unbounded number of homogeneous processes or subcomponents arise in many important applications, including cache coherence protocols, multiprocessor systems and data communication applications. This gives rise to the *Parameterized Model Checking Problem* (PMCP) which is to decide whether a temporal property holds for every size instance, n, of the given system, n being the number of homogeneous processes or subcomponents in the system. PMCP is of great practical importance as it settles, in one fell swoop, the scaling and the state explosion problems. However, the problem is, in general, undecidable [2]. Since tractable decidable cases of the PMCP have many applications, it is important to clarify the relationship between decidability and undecidability for its various formulations.

In this paper, we consider the PMCP for systems of the form, $C||U^n$, consisting of a distinguished control process, C, and finitely, but arbitrarily, many copies of a user process, U, executing concurrently with interleaving semantics. Processes can communicate with each other using pairwise rendezvous (a process sends a message only if there is an enabled receiver), asynchronous rendezvous (a process sends a message which is received iff there is an enabled receiver), broadcast primitives (a process sends a message which is received by all the other processes), and global boolean guards labeling the transitions of the individual processes. The boolean guards could either be purely *conjunctive*, viz., of the general form $\bigwedge(a \vee ... \vee b)$, expressing the condition that each process other than the one firing the transition is currently in one of the local states $a, ..., b$, or purely *disjunctive*, viz., of the general form $\bigvee(c \vee ... \vee d)$, expressing the condition that there exists a process other than the one firing the transition in one of the local states $c, ..., d$. The difference between pairwise rendezvous and asynchronous rendezvous, introduced only recently in the literature [6] in the context of verification of multi-threaded Java programs, is that while in the former case for a process to send a message, a matching receiver must be present in the current global state to receive it, in the later case a pro-

*This work was supported in part by NSF grant CCR-009-8141 & ITR-CCR-020-5483, and SRC contract 2002-TJ-1026. Contact: {emerson,kahlon}@cs.utexas.edu

cess can simply send a message irrespective of whether a receiver is present to receive it or not. If a receiver is present then the message will be received; else it is discarded.

Our goal is to delineate the decidability/undecidability boundary of the PMCP for all possible systems that arise by letting processes using different subsets of the above mentioned primitives. We focus on the following properties (**p1**) LTL$\setminus$X formulae over control process C (**p2**) LTL formulae over C (**p3**) *regular* properties specified as regular automata, and (**p4**) ω-regular properties specified as ω-regular automata. We first show the following.

1. The PMCP for LTL formulae over C is undecidable for families of the form $C||U^n$ where processes are allowed to communicate via conjunctive guards. This result is somewhat surprising because it was shown in [7] that for such systems, the PMCP is not only decidable for LTL$\setminus$X formulae but efficiently so in the size of both C and U. Thus we have the interesting fact that allowing stuttering sensitive properties bridges the gap between decidability and undecidability for systems with conjunctive guards. Moreover, this result greatly sharpens the result in [9] wherein it was shown that the PMCP for **p2** is undecidable for systems with *both* conjunctive and disjunctive guards. Since ω-regular automata are strictly more expressive than LTL, therefore the undecidability of the PMCP for ω-regular properties follows as a corollary. We go on to show that the PMCP for regular properties is also undecidable for systems with conjunctive guards.

2. The PMCP for LTL$\setminus$X formulae over C is undecidable for systems where processes are allowed to communicate via asynchronous rendezvous. As a corollary, we have that the PMCPs for both LTL and ω-regular properties are also undecidable for such systems.

As part of previous work it is known that the PMCP for **p3** is decidable for systems with broadcasts [10] and asynchronous rendezvous [6] but that the PMCP for **p4** is undecidable for systems with broadcasts [10]. Furthermore the PMCP for **p1** is decidable for systems with conjunctive and for systems with disjunctive guards [7] and pairwise rendezvous [12].

We next establish a hierarchy for the communication primitives based on their relative expressive power. We show that disjunctive guards and pairwise rendezvous are equally expressive in that we can reduce reasoning about the PMCP for regular and ω-regular properties for systems with disjunctive guards to systems with pairwise rendezvous and vice versa. However, each of asynchronous rendezvous and broadcasts is strictly more expressive than pairwise rendezvous (and disjunctive guards). Though interesting in its own right, we use this hierarchy, and the decidability and undecidability results to completely pin down the decidability/undecidability boundary for the PMCPs for all four properties **p1**, **p2**, **p3** and **p4** and for all possible combinations of the five primitives (see figure 6).

The undecidability of the PMCP for regular properties for systems with conjunctive guards is discouraging from the point of view of model checking parameterized snoopy cache coherence protocols which require the use of broadcast primitives labeled with conjunctive guards as well as ones labeled with disjunctive guards. However, in practice, templates C and U describing transition diagrams of snoopy cache protocols have the following properties:

1. Templates C and U are *initializable*, viz., for each template there exists an unguarded internal transition from each of its states to its initial state. Such transitions, not usually drawn for simplicity reasons [3], are required to model block replacement for caches which may non-deterministically push a block into its $invalid$ ($initial$) state irrespective of the current state of the block.

2. Each global boolean guard labeling a transition of a template is either disjunctive or is of the specialized form $\bigwedge(\mathsf{i})$, where i is the initial state. Such guards suffice as each cache only needs to test whether there exists another cache possessing the memory block that it requires, a disjunctive guard, or whether no other cache has the required memory block, the specialized conjunctive guard.

Under these assumptions, we show that the PMCP for regular properties is decidable. This gives us a broad and useful framework for modeling snoopy cache protocols for which the PMCP for regular properties is decidable and which is able to handle all of the snoop-based cache coherence protocols in [13], thereby overcoming the undecidability results.

The rest of the paper is organized as follows. The system model along with some other preliminaries is introduced in section 2. The decidability and undecidability results are presented in sections 3 and 4, while the expressiveness hierarchy and decidability/undecidability boundaries are established in section 5. In section 6, we present the framework for modeling cache protocols for which the PMCP for regular properties is decidable, and we conclude with some remarks in section 7.

2 Preliminaries

The System Model We focus on systems comprised of a distinguished control process, C, and finitely, but arbitrarily, many copies of a user process *template*, U, communicating with each other using broadcast primitives, pairwise rendezvous, asynchronous rendezvous actions and guards labeling the transitions of individual processes. Each guard is a boolean expression over the local states of other processes in the system. Templates C and U are given, respectively, by the 4-tuples $(S_C, L, R_C, \mathsf{i}_C)$ and $(S_U, L, R_U, \mathsf{i}_U)$, where

- S_C and S_U are finite, non-empty sets of states.

- L is the set of labels comprised of the set Σ_{in} of internal transition labels; the sets $\Sigma_b \times \{??\}$ and $\Sigma_b \times \{!!\}$ of *input* and *output* broadcast labels, respectively; the sets $\Sigma_{ar} \times \{\downarrow\}$ and $\Sigma_{ar} \times \{\uparrow\}$ of *input* and *output* asynchronous rendezvous labels, respectively; and the sets $\Sigma_{pr} \times \{?\}$ and $\Sigma_{pr} \times \{!\}$ of *input* and *output* pairwise rendezvous labels, respectively, where Σ_{ar}, Σ_b, Σ_{pr} and Σ_l are disjoint finite sets. The elements of $\Sigma = \Sigma_{in} \cup \Sigma_{ar} \cup \Sigma_b \cup \Sigma_l \cup \Sigma_{pr}$ are called *actions*. The labels $(l, !!)$, $(l, ??)$, $(m, !)$, $(m, ?)$, $(m, \uparrow)$ and $(m, \downarrow)$ are abbreviated as $l!!$, $l??$, $m!$, $m?$ $m \uparrow$ and $m \downarrow$, respectively.

- i_C and i_U are the initial states of C and U, respectively.

- $R_C \subseteq S_C \times L \times S_C$ and $R_U \subseteq S_U \times L \times S_U$, are the transition relations of C and U, respectively, satisfying the condition that for each state $a \in S = S_C \cup S_U$, and each $l \in \Sigma_b$, there exists a state $c \in S$ such that $a \xrightarrow{l??} c \in R = R_C \cup R_U$. Furthermore, each transition of R is either guarded by *true* or by a *conjunctive* guard of the general form $\bigwedge(d \vee ... \vee e)$, or by a *disjunctive* guard, of the general form $\bigvee(d \vee ... \vee e)$, where $d, ..., e$ are propositions identified with the states in S.

We assume that for transitions labeled with input broadcast or input rendezvous labels (both pairwise and asynchronous), the guard is true. Thus only the "initiator" of a transition is guarded.

Given templates C and U and $n \in \mathsf{IN}$[1], $C||U^n = (S^n, \Sigma, R^n, \mathsf{i}^n)$, the system comprised of (a unique copy of) C and n copies of the template U running in parallel asynchronously (with interleaving semantics) is defined in the standard way. We use U_0 to represent the unique concrete copy of the control process in $C||U^n$, while for $i \in [1 : n]$, the ith concrete copy of U is denoted by $U_i = (S_i, L, R_i, \mathsf{i}_i)$. A *computation* $x = x_0x_1...$ of $C||U^n$ is a sequence of states such that x_0 is the initial state of $C||U^n$ and for every $i \geq 0, (x_i, a, x_{i+1}) \in R^n$ for some $a \in \Sigma$.

Let $S_U = \{a_1, ..., a_k\}$. Then each state $u \in S^n$ can be represented as a tuple of the form $\mathsf{c} = (c, p_1, ..., p_k)$, where c is the local state of C in u and for each i, p_i indicates the number of copies of a_i in u. Tuple c is then called a *configuration* corresponding to global state u of $\mathsf{GP}(C, U)$, the guarded protocol comprised of all systems $C||U^n$ for $n \geq 1$. We denote c by *config*(u) and p_i by $\mathsf{c}(a_i)$. For configurations c_1 and c_2 and $l \in \Sigma$, we write $\mathsf{c}_1 \xrightarrow{l} \mathsf{c}_2$ to mean that there exists $n \in \mathsf{IN}$ and global states u and v of $C||U^n$ such that $u \xrightarrow{l} v \in R^n$, *config*$(u) = \mathsf{c}_1$ and *config*$(v) = \mathsf{c}_2$. The language of $\mathsf{GP}(C, U)$, from an initial configuration c_0, denoted by $L(\mathsf{GP}(C, U), \mathsf{c}_0)$, is the set of sequences $\alpha \in \Sigma^*$ such that $\mathsf{c}_0 \xrightarrow{\alpha} \mathsf{c}$. The ω-language of $\mathsf{GP}(C, U)$ from c_0 is defined accordingly.

A *parameterized configuration*, p, is a tuple in $S_C \times (\mathsf{IN} \cup \{\perp\})^k$. Intuitively, if for state $a \in S_U$, $\mathsf{p}(a) =\perp$, it means that the number of processes in local state a are arbitrary. Thus parameterized configuration p represents the set of all configurations that result by replacing each $\perp$ in p by a concrete natural number. The *language* of $\mathsf{GP}(C, U)$ from the initial parameterized configuration p_0 (corresponding to having arbitrarily many copies of U in their initial states), denoted by $L(\mathsf{GP}(C, U), \mathsf{p}_0)$ (in short by $L(\mathsf{GP}(C, U))$), is defined as $L(\mathsf{GP}(C, U), \mathsf{p}_0) = \bigcup_{\mathsf{c} \in \mathsf{p}_0} L(\mathsf{GP}(C, U), \mathsf{c})$. The ω-language $L_\omega(\mathsf{GP}(C, U), \mathsf{p}_0)$ is defined accordingly.

Parameterized Model Checking Problems The Parameterized Model Checking Problem is formally defined as follows: given a temporal logic formula f, and templates C and U, to decide whether for all n: $C||U^n \models f$. In this paper, we focus on the following linear time properties:

(**p1**) LTL$\backslash$X formulae with atomic propositions over the local states of control process C,

(**p2**) LTL formulae over C.

Note that the absence of the next-time operator X in LTL$\backslash$X makes it stuttering insensitive. More generally, we also consider the following automata-theoretic formulations of model checking (see, for instance, [14]): Note that in [10], regular and ω-regular properties were termed as *safety* and *liveness* properties, respectively:

(**p3**) **Regular properties**: Given templates C and U, a parameterized configuration p_0 and a regular language L, to decide if $L(\mathsf{GP}(C, U), \mathsf{p}_0) \cap \mathsf{L} = \emptyset$.

(**p4**) **ω-regular properties**: Given a templates C and U, a parameterized configuration p_0 and an ω-regular language L_ω, to decide if $L_\omega(\mathsf{GP}(C, U), \mathsf{p}_0) \cap \mathsf{L}_\omega = \emptyset$.

Counter Machines A *counter machine* is a tuple $M = (Q, \Sigma, C, \Delta, q_0, H)$, where Q is the set of states, Σ the set of labels, C the set of counters, Δ the set of transitions, q_0 the initial state and H the set of halting states. There are three type of transitions in Δ (i) $q \xrightarrow{l:\mathbf{c}:=\mathbf{c}+1} q'$, viz., increment a counter (ii) $q \xrightarrow{l:\mathbf{c}:=\mathbf{c}-1} q'$, viz., decrement a counter if it has a positive value and (iii) $q \xrightarrow{l:\mathbf{c}=0} q'$, test if a counter has value 0. Let $C = \{\mathbf{c}_1, ..., \mathbf{c}_m\}$. Then a *configuration* of M is a tuple $(q, j_1, ..., j_m)$, where $q \in Q$ and j_i is a non-negative integer, indicating the current value of counter $\mathbf{c}_i$. A *run* of M is either an infinite sequence $c_1 \to c_2 \to ...$ or a finite sequence $c_1 \to ... \to c_n$, where c_n is halting. A configuration $(q, j_1, ..., j_m)$ is *initial* if $q = q_0$ and *n-bounded* if $\Sigma_{1 \leq i \leq m} j_i \leq n$. A run is *initial* if its first configuration is initial, *n-bounded* if all its configurations contain only n-bounded configurations, and *bounded* if it is n-bounded for

[1] IN denotes the set of natural numbers.

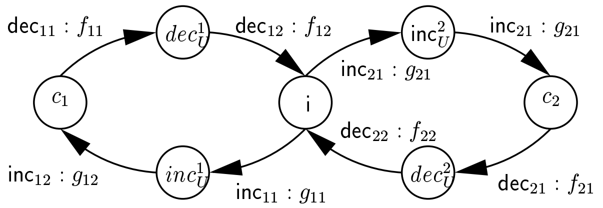

Figure 1. The template U

some number n. For configuration s, we use $c(s)$ to denote the state of M in s and for counter $\mathbf{c}$, we use $\mathbf{c}(s)$ to denote the value of counter $\mathbf{c}$ in s.

The Halting Problem for counter machines is to decide whether a given counter machine M has a halting run, viz., a run leading to configuration in with M in a halting state, beginning at the configuration $(q_0, 0, ..., 0)$. In this paper, we make use of the fact that the Halting Problem is undecidable, in general, for machines with two or more counters.

3 Systems with Conjunctive Guards

In this section, we show that for guarded protocols of the form $\mathsf{GP}(C, U)$, where processes are allowed to communicate via conjunctive guards, the PMCP for LTL formulae over the control process C is undecidable. Furthermore, we show that the undecidability of the PMCP carries over to regular and ω-regular properties.

Undecidability of the PMCP for LTL The proof is by reduction from the halting problem for 2-counter machines. Let $2CM = (Q, \Sigma, \{\mathbf{c}_1, \mathbf{c}_2\}, \Delta, q_0, H)$ be a given 2-counter machine. We construct templates C and U and a LTL formula, h, over C, such that there is a halting run of $2CM$ iff there exists n such that $C||U^n \models \mathsf{E}h$, viz., there exists a computation of $C||U^n$ satisfying h. This reduces the decidability of the halting problem for a given 2-counter machine to an instance of the PMCP for the LTL formula, $\neg h$, over C. Since the halting problem for 2-counter machines is undecidable, so is the PMCP for LTL formulae over C.

Given $2CM$, we now show how to construct templates C and U. The control process C is used to simulate the control part of $2CM$, while the two counters are implemented using the many copies of template process U. The counters are represented in unary, with each copy of U storing one bit of either $\mathbf{c}_1$ or $\mathbf{c}_2$. The transition diagram for template $U = (S_U, \Sigma_U, R_U, \mathsf{i})$ is shown in figure 1. If a copy of U is in local c_i, then that corresponds to one bit of $\mathbf{c}_i$. For $i \in [1:2]$, the states inc_U^i and dec_U^i are called the *intermediate* states of U and are used for book-keeping purposes while simulating incrementing or decrementing of counter $\mathbf{c}_i$.

Definition (Template C) Template C is defined as the tuple $(S_C, \Sigma_C, R_C, q_0)$, where S_C is the set of states; Σ_C the set of labels; R_C the transition relation and q_0 the initial state of C. State set S_C and transition relation R_C is gotten from Δ, the transition relation for $2CM$, as follows. Let $tr \in \Delta$ be a transition of $2CM$. We consider three cases based on whether tr is a zero test, increments a counter or decrements a counter.

1. If tr is of the form $a \xrightarrow{l:\mathbf{c}_i=0} b$, then we add the transition $a \xrightarrow{l:g_l} b$, where guard $g_l = \bigwedge(\mathsf{i} \vee c_{i'})$, with $i' \in [1:2]$ and $i' \neq i$, to R_C. Guard g_l expresses the condition that every copy of U is either in state i or state $c_{i'}$, viz., no copy of U is in state c_i and so the value of counter $\mathbf{c}_i$ is zero.

2. If tr is of the form $a \xrightarrow{l:\mathbf{c}_i:=\mathbf{c}_i+1} b$, then we add the transitions $a \xrightarrow{l_1:g_l} inc_C^i$ and $inc_C^i \xrightarrow{l_2} b$ to R_C. Here guard $g_l = \bigwedge(\mathsf{i} \vee c_1 \vee c_2)$, expresses the condition that all the user processes are in their non-intermediate local states indicating completion of simulation of all previous operations of the 2-counter machine.

3. If tr is of the form $a \xrightarrow{l:\mathbf{c}_i:=\mathbf{c}_i-1} b$, then we add the transitions $a \xrightarrow{l_1:g_l} dec_C^i$ and $dec_C^i \xrightarrow{l_2} b$ to R_C, where, as above, $g_l = \bigwedge(\mathsf{i} \vee c_1 \vee c_2)$.

The states inc_C^i and dec_C^i are called the *intermediate* states of C. Thus, $S_C = Q \bigcup (\cup_{i \in [1:2]} \{inc_C^i, dec_C^i\})$. We now show how the operation $a \xrightarrow{l:\mathbf{c}_i:=\mathbf{c}_i+1} b$ of incrementing counter $\mathbf{c}_i$ is simulated, the decrement operation being handled similarly. First, the control process C, after making sure that the simulations of all previous operations have finished via the guard $\bigwedge(\mathsf{i} \vee c_1 \vee c_2)$, fires the transition $tr_1 = a \xrightarrow{l_1:g_l} inc_C^i$ thus issuing the command to begin simulating the increment operation. Then one of the user processes fires transition $tr_2 = \mathsf{i} \xrightarrow{\mathsf{inc}_{i1}:g_{i1}} inc_U^i$, where $g_{i1} = \bigwedge(inc_C^i \vee \mathsf{i} \vee c_1 \vee c_2)$, expressing the conditions that (a) the control process C is in local state inc_C^i, viz., a command to increment counter $\mathbf{c}_i$ has been issued; and (b) that every other user process is in local state i, c_1 or c_2, viz., none is in an intermediate state of U implying that all previous simulations of the operations of the 2-counter machine have finished. Then the control process transits back to a non-intermediate state by firing transition $tr_3 = inc_C^i \xrightarrow{l_2} b$ and completes its part of the simulation. Having noticed that via the guard $g_{i2} = \bigwedge(\vee_{d \in Q} d)$ (control process is in a non-intermediate state), the (unique) user process in local state inc_U^i fires the transition $tr_4 = inc_U^i \xrightarrow{\mathsf{inc}_{i2}:g_{i2}} c_i$ completing the simulation.

The operation of decrementing counter $\mathbf{c}_i$ is simulated in a similar fashion with a user process now firing the transitions $c_i \xrightarrow{\mathsf{dec}_{i1}:f_{i1}} dec_U^i$ and $dec_U^i \xrightarrow{\mathsf{dec}_{i2}:f_{i2}} \mathsf{i}$, where $f_{i1} = \bigwedge(dec_C^i \vee \mathsf{i} \vee c_1 \vee c_2)$ and $f_{i2} = g_{i2}$.

However the above simulation is not faithful. This is because, for instance, while trying to simulate incrementing counter $\mathbf{c}_i$, the control process could fire transitions tr_1 and

tr_3 back to back without a user process firing any transitions at all. This can happen because tr_3 is guarded by *true*. Thus the control process would indicate execution of an increment of $\mathbf{c}_i$, without any additional user process transiting to local state c_i. From our definitions of C and U, it follows that in any computation of $\mathsf{GP}(C, U)$, process C firing tr_1 is followed either by (a) the firing of tr_3, or (b) the firing of tr_2, tr_3 and tr_4 back to back in that order. A computation of $\mathsf{GP}(C, U)$ faithfully executes the increment operation, iff in the computation, option (a) is never exercised. We now define a LTL formula h_{inc} that captures precisely such computations. Towards that end, we observe that after firing tr_1 in any computation, the current local state inc^i_U of C changes in the very next step of the computation iff tr_3 is fired immediately after firing tr_1. Thus $h_{inc} = \mathsf{G}(\bigwedge_i((\neg\, inc^i_C \wedge \mathsf{X}\ inc^i_C) \Rightarrow \mathsf{XX}\ inc^i_C))$. The case of faithfully simulating the decrement operation is handled similarly. Now, $2CM$ has a halting computation iff there is computation of $\mathsf{GP}(C, U)$ that faithfully executes both increment and decrement operations and leads to a state in which the control process is in a local state of H. By the above discussion, this can be decided by checking whether for some n, $U^n \models \mathsf{E}h$, where h is the LTL formula $h_{halt} \wedge h_{inc} \wedge h_{dec}$. Here, $h_{halt} = \mathsf{F}(\bigvee_{halt \in H} halt)$, $h_{inc} = \mathsf{G}(\bigwedge_i((\neg\, inc^i_C \wedge \mathsf{X}\ \mathrm{inc}^i_C) \Rightarrow \mathsf{XX}\ inc^i_C))$ and $h_{dec} = \mathsf{G}(\bigwedge_i((\neg\, dec^i_C \wedge \mathsf{X}\ dec^i_C) \Rightarrow \mathsf{XX}\ dec^i_C))$. Thus we have the following.

Proposition 3.1 The 2-counter machine $2CM$ has a halting run starting at the initial configuration iff for some n, $C||U^n \models \mathsf{E}h$.

Since the Halting Problem for 2-counter machines is undecidable, therefore as a corollary, we have that

Proposition 3.2 For guarded protocols $\mathsf{GP}(C, U)$, with conjunctive guards, the PMCP for LTL formulae over control process C, is undecidable.

Similarly we can show the following.

Proposition 3.3 The PMCPs for regular and ω-regular properties are undecidable for guarded protocols $\mathsf{GP}(C, U)$, with conjunctive guards.

4 Systems without Conjunctive Guards

We now focus on systems without conjunctive guards, viz., wherein processes communicate using global disjunctive guards, pairwise rendezvous, asynchronous rendezvous and broadcasts. The main results shown are that the PMCP for **p1** (and hence for **p2** and **p4**) is undecidable for guarded protocols with asynchronous rendezvous but that the PMCP for **p3** for guarded protocols without conjunctive guards is decidable.

4.1 Decidability of Regular Properties We show using the backward reachability procedure of [1] that the PMCP for regular properties is decidable for guarded protocols with disjunctive guards, pairwise rendezvous, asynchronous rendezvous and broadcasts. Model checking for regular properties via backward reachability was considered for broadcast protocols in [10] and for asynchronous rendezvous in [6]. In addition to these we also assume global disjunctive guards for our model. The procedure is exactly the same as in [1] therefore we only recapitulate the basic idea below.

Let p_0 be a parameterized initial configuration and let $A = (Q, \Sigma, \delta, q_0, F)$ be an automaton. Then the PMCP for regular properties translates to the following: Does there exists a combined state $\mathsf{n} \in \mathsf{IN}^k \times F$ reachable from a combined state (c_0, q_0), where $\mathsf{c}_0 \in \mathsf{p}_0$?

A set C of combined states is *upwards-closed* if $(\mathsf{c}, q) \in \mathsf{C}$ implies $(\mathsf{c}', q') \in \mathsf{C}$ for every $(\mathsf{c}, q) \trianglelefteq (\mathsf{c}', q')$, where $(\mathsf{c}, q) \trianglelefteq (\mathsf{c}', q')$ iff $\mathsf{c} \preceq \mathsf{c}'$ ($\preceq$ denotes component-wise $\leq$) and $q = q'$. Let $pred(\mathsf{C})$ be the set of combined states from which C is reachable in one step. The algorithm for verifying regular properties then proceeds as follows. Starting at $\mathsf{IN}^k \times F$, an upward-closed set, it iteratively computes the minimal elements of $\mathsf{C}_0 = \mathsf{IN}^k \times F$, $\mathsf{C}_1 = \mathsf{C}_0 \cup pred(\mathsf{IN}^k \times F)$, $\mathsf{C}_2 = pred^2(\mathsf{IN}^k \times F)$, etc. Each iteration relies on the facts that for an upward-closed set C, (i) the set of minimal elements of C is finite (ii) the set $pred(\mathsf{C})$ is upwards-closed and (iii) the minimal elements of $pred(\mathsf{C})$ are effectively computable from the minimal elements of C. Termination is guaranteed because in any infinite set of combined states there exists two elements n and n' such that $\mathsf{n} \trianglelefteq \mathsf{n}'$. Therefore there exists an n such that C_n and $pred^*(\mathsf{IN}^k \times F) = \bigcup_{i \geq 0} \mathsf{C}_i$ coincide.

4.2 Undecidability of LTL\X properties over the control process for Guarded Protocols with Asynchronous Rendezvous The proof is in two parts. First we show that we can simulate pairwise rendezvous using asynchrony rendezvous. Then we show the much stronger result that for systems with pairwise rendezvous and asynchronous rendezvous, even the following LTL\X property is undecidable: Does there exist an n such that there is a computation of $C||U^n$ in which C fires infinitely often ? Then combining the above two facts we have our result.

Simulation of Pairwise Rendezvous by Asynchronous Rendezvous. Let $V = (S, L, R, \mathsf{i})$ be a given template with pairwise rendezvous. Without loss of generality, we may assume that for each label $l \in L$, there exists a unique transition of V labeled by l. We perform the following transformations (figure 2):

- for each matching pair $a \xrightarrow{l!} b$ and $c \xrightarrow{l?} d$ replace $a \xrightarrow{l!} b$ by the pair of transitions $a \xrightarrow{l_1\uparrow} \mathsf{in}_{l!}$ and $\mathsf{in}_{l!} \xrightarrow{l_2\downarrow} b$. and transition $a \xrightarrow{l?} b$ by the pair $a \xrightarrow{l_1\downarrow} \mathsf{in}_{l?}$ and $\mathsf{in}_{l?} \xrightarrow{l_2\uparrow} b$.

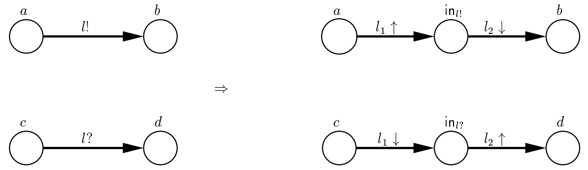

Figure 2. Simulating Pairwise Rendezvous by Asynchronous Rendezvous

- replace each internal transition $a \xrightarrow{l} b$, by the pair of internal transitions $a \xrightarrow{l_1} \mathsf{in}_l$ and $\mathsf{in}_l \xrightarrow{l_2} b$.

Let $V_{\mathsf{pa}} = (S_{\mathsf{pa}}, L_{\mathsf{pa}}, R_{\mathsf{pa}}, \mathsf{i}_{\mathsf{pa}})$ be the resulting template. Define a morphism $\phi_{\mathsf{pa}} : L \to L^*_{\mathsf{pa}}$ such that for every action l, $\phi_{\mathsf{pa}}(l) = l_1 l_2$. Then we have the following result shows that we can reduce reasoning about PMCP for a property in each of the classes $\mathbf{p1}, \mathbf{p3}$ and $\mathbf{p4}$ for $\mathsf{GP}(C, U)$ to reasoning about a property in the same respective classes for $\mathsf{GP}(C_{\mathsf{pa}}, U_{\mathsf{pa}})$.

Proposition 4.1
1. For LTL$\backslash$X formula h over C, $C||U^n \models \mathsf{E}h$ iff $C_{\mathsf{pa}}||U^n_{\mathsf{pa}} \models \mathsf{E}h$, where for each intermediate state in_{label}, such that $a \stackrel{label}{\to} \mathsf{in}_{label}$ is the (unique) transition leading to in_{label} in C_{pa} or U_{pa}, the atomic proposition a is interpreted to be true over in_{label}.
2. $L(\mathsf{GP}(C, U), \mathbf{p}_0) \cap \mathsf{L} = \emptyset$ if and only if $L(\mathsf{GP}(C_{\mathsf{pa}}, U_{\mathsf{pa}}), \mathbf{p}_0) \cap \phi_{\mathsf{pa}}(\mathsf{L}) = \emptyset$, where L is a regular language and $L_\omega(\mathsf{GP}(C, U), \mathbf{p}_0) \cap \mathsf{L}_\omega = \emptyset$ if and only if $L_\omega(\mathsf{GP}(C_{\mathsf{pa}}, U_{\mathsf{pa}}), \mathbf{p}_0) \cap \phi_{\mathsf{pa}}(\mathsf{L}_\omega) = \emptyset$ where L_ω is an ω-regular language.

Undecidability of LTL$\backslash$X for systems with Pairwise Rendezvous and Asynchronous Rendezvous. The proof is by reduction from the halting problem for 2-counter machines and is inspired by the proof for undecidability of ω-regular properties for broadcast protocols [10]. We show that even the LTL$\backslash$X property that a given guarded protocol with pairwise and asynchronous rendezvous has an computation in which C fires infinitely often, is undecidable.

Let $2CM = (Q, \Sigma, \{\mathbf{c}_1, \mathbf{c}_2)\}, \Delta, q_0)$ be a given 2-counter machine. We can assume, without loss of generality, that $2CM$ has no transitions that are self loops. For if $a \xrightarrow{l} a$ is a self looping transitions, then we can add an additional state $a' \neq a$ and replace the self loop by the following pair of transitions: $a \xrightarrow{l} a'$ and the internal transition $a' \xrightarrow{\tau} a$ that executes no operation[2]. We start by defining a 3-counter machine $3CM$ with the following behaviour: Given a run of $2CM$, the machine $3CM$ mimics each step of $2CM$ on counters $\mathbf{c}_1$ and $\mathbf{c}_2$ but, in addition, it uses counter $\mathbf{c}_3$ to keeps track of the number of steps of $2CM$ mimicked by incrementing $\mathbf{c}_3$ each time it does so. If M reaches a configuration with a halting state, then $3CM$ re-initializes itself by draining all the counters and transiting back to the initial control state q_0. Thus every bounded infinite run of $3CM$ has infinitely many initial configurations. Then we have

[2]Strictly speaking, internal transitions are not allowed by our definition of counter machines, but can be implemented by first incrementing a counter, say $\mathbf{c}_1$, and then decrementing it, thus leaving its value unchanged.

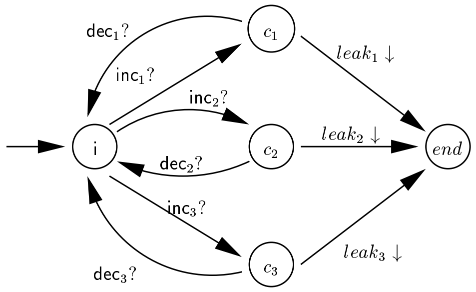

Figure 3. The template U

Proposition 4.2 The 2-counter machine $2CM$ has a halting run starting at $(q_0, 0, 0)$ iff the 3-counter machine $3CM$ has a bounded infinite run starting at $(q_0, 0, 0, 0)$.

We now show how to weakly simulate $3CM$ with a system of the form $\mathsf{GP}(C, U)$, where C and U are templates that communicate with each other using solely pairwise rendezvous and weak asynchrony. The idea is the same as in the previous section. The template process C is used to simulate the control part of $2CM$, whereas the three counters are implemented using multiple copies of template U. The template U (figure 3) has five local states: i, the initial state and c_1, c_2 and c_3 corresponding, respectively, to each of the three counters $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$ and the terminal state *end*.

Given $2CM$, we define template C as the tuple (Q, Σ, R, q_0), where Q is the state set of C, Σ is the set of labels; R is the transition relation and q_0 the initial state: Transition relation R is gotten from Δ as follows:

1. for each transition of the form $a \stackrel{\mathbf{c}_i := \mathbf{c}_i + 1}{\to} b$ in Δ add $a \xrightarrow{\mathsf{inc}_i!} b$ to R.

2. for each transition of the form $a \stackrel{\mathbf{c}_i := \mathbf{c}_i - 1}{\to} b$ in Δ add $a \xrightarrow{\mathsf{dec}_i!} b$ to R.

3. for each transition of the form $a \stackrel{\mathbf{c}_i = 0}{\to} b$ in Δ add $a \xrightarrow{leak_i\uparrow} b$ to R.

Note that since no transition of $2CM$ was a self-loop, neither is any transition of C. Furthermore, all of the simulations except the one for the zero-test are faithful. In simulating the zero test $a \stackrel{\mathbf{c}_i = 0}{\to} b$, we let the control process C

fire $tr = a \xrightarrow{leak_i\uparrow} b$ even if we have copy of U in local state c_i. But in that case, as is evident from the definition of U, firing tr "clobbers" one such copy of U by making it transit to the terminal state *end*. Henceforth that copy of U can no longer execute any transitions. Such a firing of $a \xrightarrow{leak_i\uparrow} b$ is therefore called a "leak".

Proposition 4.3 There exists a bounded infinite run of $3CM$ starting at $(q_0, 0, 0, 0)$ iff there is a run of $C||U^m$, for some m, from its initial state in which C fires infinitely often.

Combining, the previous two results, we have that $2CM$ has a halting run starting at $(q_0, 0, 0)$ iff there is a run of $C||U^m$, for some m, in which C fires infinitely often. Recalling that C has no self loops, we have that the property that there is a computation in which C fires infinitely often can be expressed by the formula $\mathsf{E}h$, where h is the LTL\X formula $\mathsf{G}(\bigvee_{d \in Q}(d \Rightarrow \mathsf{F}\neg d))$. This gives us the following.

Theorem 4.4 The PMCP for LTL\X formulae over the control process C is undecidable for guarded protocols of the form $\mathsf{GP}(C, U)$, with pairwise rendezvous and asynchronous rendezvous.

Then from the first part of proposition 4.1 and theorem 4.4, we have the desired undecidability result. A similar proof can be used to show the undecidability of PMCP for **p1** for systems with broadcast actions.

5 The Decidability/Undecidability Boundaries

We first establish a hierarchy based on the relative expressive power of the various primitives which proves helpful in establishing the decidability/undecidability boundary.

5.1 Comparing expressive power of Communication Primitives. Given two labeled transition systems $S_1 = (Q_1, \Sigma_1, R_1, \mathsf{i}_1)$ and $S_2 = (Q_2, \Sigma_2, R_2, \mathsf{i}_2)$, possibly of infinite size, let (homo)morphism $\phi : \Sigma_1 \rightarrow \Sigma_2^*$ be such that for any regular language L, $L(S_1) \cap \mathsf{L} = \emptyset$ iff $L(S_2) \cap \phi(\mathsf{L}) = \emptyset$ and for any ω-regular language L_ω, $L_\omega(S_1) \cap \mathsf{L}_\omega = \emptyset$ iff $L_\omega(S_2) \cap \phi(\mathsf{L}_\omega) = \emptyset$. Then we say that we can *reduce*, respectively, the model checking problems for regular and ω-regular properties for S_1 to S_2 via ϕ. This is denoted by $S_1 \leq_\phi S_2$. Note that this notion is broader than trace equivalence, to which it reduces when ϕ is the identity map. We say that primitive p_1 is *at least as expressive* as primitive p_2 with respect to the PMCP for regular and ω-regular properties, denoted by $p_1 \leq p_2$, iff for every guarded protocol $\mathsf{GP}(C_1, U_1)$, wherein processes communicate solely using primitives p_1, we can construct a guarded protocol $\mathsf{GP}(C_2, U_2)$ wherein processes communicate solely using primitives p_2, and a morphism ϕ such that

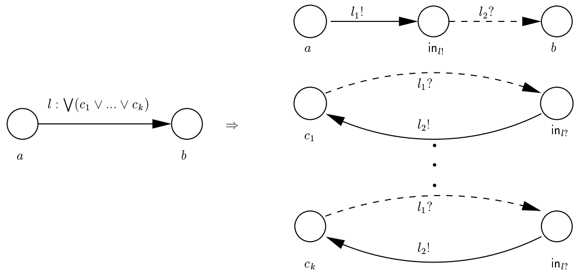

Figure 4. Simulation of Disjunctive Guards by Pairwise Rendezvous

$\mathsf{GP}(C_1, U_1) \leq_\phi \mathsf{GP}(C_2, U_2)$. We use $p_1 < p_2$ to denote $p_1 \leq p_2$ and $\neg(p_2 \leq p_1)$; and $p_1 \sim p_2$ to denote $p_1 \leq p_2$ and $p_2 \leq p_1$.

Disjunctive Guards $\leq$ Pairwise Rendezvous. Let $V = (S, L, R, \mathsf{i})$ be a given template with disjunctive guards. Without loss of generality, we assume that each transition of V is labeled with a unique label. We perform the following operations (figure 4).

- replace each transition $a \xrightarrow{l} b$ guarded by the disjunctive guard $g = (c_1 \vee ... \vee c_k)$ by the two transitions $a \xrightarrow{l_1!} \mathsf{in}_{l!}$ and $\mathsf{in}_{l!} \xrightarrow{l_2?} b$ and for each state c_i introduce the two new transition $c_i \xrightarrow{l_1?} \mathsf{in}_{l?}$ and $\mathsf{in}_{l?} \xrightarrow{l_2!} c_i$.
- replace each internal transition $a \xrightarrow{l} b$ by the two internal transitions $a \xrightarrow{l_1} \mathsf{in}_l$ and $\mathsf{in}_l \xrightarrow{l_2} b$.

Let $V_{\mathsf{dr}} = (S_{\mathsf{dr}}, L_{\mathsf{dr}}, R_{\mathsf{dr}}, \mathsf{i})$ be the resulting template. Define a morphism $\phi_{\mathsf{dr}} : L \rightarrow L_{\mathsf{dr}}^*$ such that for each $l \in \Sigma$, $\phi_{\mathsf{dr}}(l) = l_1 l_2$. Then, analogous to proposition 4.1(2), we can show that given templates C and U, $\mathsf{GP}(C, U) \leq_{\phi_{\mathsf{dr}}} \mathsf{GP}(C_{\mathsf{dr}}, U_{\mathsf{dr}})$.

Pairwise Rendezvous $\leq$ Disjunctive Guards. Let $V = (S, L, R, \mathsf{i})$ be a template with pairwise rendezvous. We perform the following operations.

- for each matching pair $a \xrightarrow{l!} b$ and $c \xrightarrow{l?} d$, replace $a \xrightarrow{l!} b$ by the pair of transitions $a \xrightarrow{l_1 : \bigvee c} \mathsf{in}_{l!}$ and $\mathsf{in}_{l!} \xrightarrow{l_3} b$, and $c \xrightarrow{l?} d$ by the pair of transitions $c \xrightarrow{l_2 : \bigvee \mathsf{in}_{l!}} \mathsf{in}_{l?}$ and $\mathsf{in}_{l?} \xrightarrow{l_4} d$.
- replace each internal transition $a \xrightarrow{l} b$ by the four internal transitions $a \xrightarrow{l_1} \mathsf{in}_{l1}$, $\mathsf{in}_{l1} \xrightarrow{l_2} \mathsf{in}_{l2}$, $\mathsf{in}_{l2} \xrightarrow{l_3} \mathsf{in}_{l3}$, $\mathsf{in}_{l3} \xrightarrow{l_4} b$.

Let $V_{\mathsf{rd}} = (S_{\mathsf{rd}}, R_{\mathsf{rd}}, L_{\mathsf{rd}}, \mathsf{i}_{\mathsf{rd}})$ be the resulting template. Define a morphism $\phi_{\mathsf{rd}} : L \rightarrow L_{\mathsf{rd}}^*$ such that for every label $l \in \Sigma$, $\phi_{\mathsf{rd}}(l) = l_1 l_2 l_3 l_4$. Again, we can show that $\mathsf{GP}(C, U) \leq_{\phi_{\mathsf{rd}}} \mathsf{GP}(C_{\mathsf{rd}}, U_{\mathsf{rd}})$.

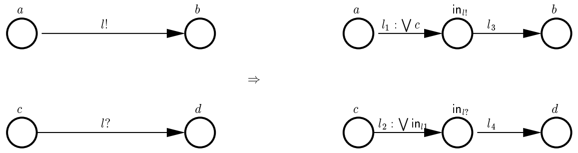

Figure 5. Simulation of Pairwise Rendezvous by Disjunctive Guards

Pairwise Rendezvous $<$ Asynchronous Rendezvous. From proposition 4.1, we have that Pairwise Rendezvous $\leq$ Asynchronous Rendezvous. To prove that $\neg$ (Asynchronous Rendezvous $\leq$ Pairwise Rendezvous), it suffices to show the following.

Proposition 5.3 There exists a guarded protocol $\mathsf{GP}(C, U)$, with asynchronous rendezvous, such that there is no guarded protocol $\mathsf{GP}(C', U')$ with pairwise rendezvous, and a morphism ϕ such that $\mathsf{GP}(C, U) \leq_\phi \mathsf{GP}(C', U')$.

Proof Sketch Given $\mathsf{GP}(C, U)$ with asynchronous rendezvous, let $\mathsf{GP}(C', U')$ with pairwise rendezvous be such that we can construct morphism ϕ with the property $\mathsf{GP}(C, U) \leq_\phi \mathsf{GP}(C', U')$. Then for any ω-regular language L_ω, $L_\omega(\mathsf{GP}(C, U)) \cap \mathsf{L}_\omega = \emptyset$ iff $L_\omega(\mathsf{GP}(C', U')) \cap \phi(\mathsf{L}_\omega) = \emptyset$. Let $\Sigma = \{l_0, ..., l_p\}$ be the set of actions of C and U. Define $\mathsf{L}_u = (l_0 + ... + l_p)^\omega$. Then using the fact that ϕ is a homomorphism, we have that $\mathsf{GP}(C, U)$ has a computation in which C fires infinitely often iff $\mathsf{GP}(C', U')$ has a computation in which C' fires infinitely often and that accepts a string in $\phi(\mathsf{L}_u)$. We then construct templates C'' and U'' that use only pairwise rendezvous as communication primitives and a morphism ϕ' such that $L_\omega(\mathsf{GP}(C'', U'')) = \phi'(L_\omega(\mathsf{GP}(C', U')) \cap \phi(\mathsf{L}_u))$. Then since ϕ' is a morphism, it follows that $\mathsf{GP}(C, U)$ has a computation in which C fires infinitely often iff $\mathsf{GP}(C'', U'')$ has a computation in which C'' fires infinitely often. Thus we have reduced the problem of checking whether a guarded protocol has a computation in which the control process fires infinitely often for protocols with asynchronous rendezvous to ones using solely pairwise rendezvous. But from [12], we know that model checking $\mathsf{GP}(C, U)$ with pairwise rendezvous, for LTL$\backslash$X properties with atomic propositions over the local states of C is decidable. So, in particular, it is decidable whether $\mathsf{GP}(C'', U'')$ has a computation in which C'' executes infinitely often. But this implies that checking whether a guarded protocol $\mathsf{GP}(C, U)$ with asynchronous rendezvous has an infinite computation in which C fires infinitely often is decidable, a contradiction to the result proved in section 4.2. ■

Pairwise Rendezvous $<$ Broadcasts. The fact that pairwise rendezvous $\leq$ broadcasts was shown in [10] Furthermore it was shown that checking whether a guarded protocol with broadcasts has an infinite computation is undecidable. Then using this fact, we can by an argument similar to the one used in proving the above result show the following.

Proposition 5.4 There exist guarded protocols $\mathsf{GP}(C, U)$ with broadcasts such that there is no guarded protocol $\mathsf{GP}(C', U')$ with pairwise rendezvous, and a morphism ϕ such that $\mathsf{GP}(C, U) \leq_\phi \mathsf{GP}(C', U')$.

5.2 Decidability/Undecidability Boundaries First considering regular properties, we have that the PMCP for **p3** is undecidable for systems with conjunctive guards (prop. 3.3), but is decidable for systems with any combination of the remaining primitives (section 4.1). This settles the boundary issue for property **p3**. Next, we have that the PMCP for property **p1**, is undecidable for systems with asynchronous guards and systems with broadcasts (section 4), while it was shown in [7] to be decidable for systems with conjunctive guards and systems with disjunctive guards (and hence for pairwise rendezvous). Moreover, it is undecidable for systems that have both conjunctive and disjunctive guards ([9]). Then using the expressiveness hierarchy, we have that it is undecidable for systems that have conjunctive guards combined with any one (or more) of the four other types of primitives. This pins down the boundary for property **p1**. Since both **p2** and **p4** are more expressive than **p1**, all undecidability results for **p1** carry over for **p2** and **p4**. From proposition 3.2 and 3.3, we have that the PMCPs for **p2** and **p4** are undecidable for systems with conjunctive guards. Furthermore, since guarded protocols with pairwise rendezvous can be modeled as petri nets, therefore using results from petri net theory (see for instance [11]) we have that the PMCP for **p2** and **p4** is decidable for systems with pairwise rendezvous (and therefore for disjunctive guards). The expressiveness hierarchy and the boundaries are pictorially shown in figure 6.

6 Special case for modeling Cache Coherence Protocols

In general, modeling parameterized cache protocols (e.g., Illinois-MESI) requires broadcast transitions labeled with conjunctive and disjunctive guards. But the PMCP for regular properties for such systems is undecidable (prop 3.3). However, as was noted in the introduction, in practice, template $U = (S, \Sigma, R, \mathsf{i})$ for a snoopy cache coherence protocol typically has the following features:

1. Every conjunctive guard labeling the transitions of U is of the specialized form $g_\mathsf{i} = \bigwedge(\mathsf{i})$.

2. U is *initializable*, viz., for each state $a \in S$, there is a local transition of the form $a \to \mathsf{i}$.

We now show that if we consider guarded protocols of the form $\mathsf{GP}(C, U)$, with broadcasts, pairwise and asyn-

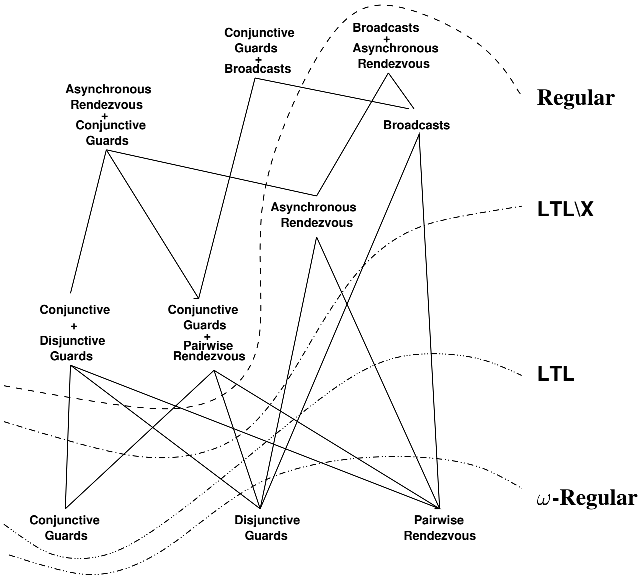

Figure 6. The Decidability-Undecidability boundaries

chronous rendezvous, disjunctive guards and the specialized conjunctive guard with initializable templates C and U, then the PMCP for **p3** is decidable. This framework can model all snoopy protocols in [13].

For simplicity, we assume that $C = U$, though our technique works in general. Let x be a finite computation of U^n that results by executing the sequence of actions $\lambda = l_0...l_k$. Then λ can be parsed as $\lambda = \lambda^0_{nc}\lambda^0_c...\lambda^{m-1}_c\lambda^m_{nc}$, where for each j, λ^j_{nc} is a (possibly empty) subsequence of actions labeling transitions of U not guarded by $g_{\mathsf{i}} = \bigwedge(\mathsf{i})$, and λ^j_c is a subsequence of actions labeling transitions guarded by g_{i}. Consider the non-empty sequence $\lambda^j_c = l_{j_0}...l_{j_0+p}$. Then since each of the transitions corresponding to action l_{j_0+i}, $i \in [0 : p-1]$ is guarded by g_{i}, therefore for each i, in state x_{j_0+i+1}, resulting by firing the transition labeled with l_{j_0+i}, there is exactly one process in its non-initial state and so it is of the form $(\mathsf{i}, ..., \mathsf{i}, a, \mathsf{i}, ..., \mathsf{i})$, for some $a \in S$. The state x_{j_0+p+1}, on the other hand, could be of one of the following three forms.

1. $(\mathsf{i}, ..., \mathsf{i}, a, \mathsf{i}, ..., \mathsf{i})$: in case the transition labeled by l_{j_0+p} is a local transition leading to state a or an asynchronous rendezvous labeled by $l_{j_0+p} \uparrow$ leading to a with no matching receive from i.

2. $(\mathsf{i}, ..., \mathsf{i}, a, \mathsf{i}, ..., \mathsf{i}, b, \mathsf{i}, ..., \mathsf{i})$: in case the transition labeled by l_{j_0+p} was a pairwise rendezvous or an asynchronous rendezvous with a matching receive from i, with one of the two processes to fire ending up in state a and the other one in b.

3. $(a, ..., a, b, a, ..., a)$: in case the transition labeled by l_{j_0+p} was a broadcast transition with the broadcast send leading to b and $\mathsf{i} \rightarrow a$ being the matching receive.

Keeping these three type of states in mind, we define the parameterized configurations $\mathsf{p}_{(a,b)}$ and $\mathsf{p}_{(a,b,c)}$ where $a, b, c \in S$, as follows: $\mathsf{p}_{(a,b)}(d) = 1$ if $d = a$, $\perp$ if $d = b$ and 0 otherwise; $\mathsf{p}_{(a,b,c)}(d) = 1$ if $d = a$ or $d = b$, $\perp$ if $d = c$ and 0 otherwise. Now let p_0 be a parameterized initial configuration and let $A = (Q, \Sigma, \delta, q_0, F)$ be an automaton. Recall that the model checking problem for regular properties is the following: Does there exist a combined state $\mathsf{n} \in \mathsf{IN}^k \times F$ reachable from a combined state (c_0, q_0), where $\mathsf{c}_0 \in \mathsf{p}_0$? Consider the set of combined states $\mathsf{S} = \{(\mathsf{p}_{(a,\mathsf{i})}, q) | a \in S \setminus \{\mathsf{i}\}, q \in Q\} \cup \{(\mathsf{p}_{(a,b,\mathsf{i})}, q) | a, b \in S \setminus \{\mathsf{i}\}, q \in Q\}$. Let $Reach(\mathsf{S}) \subseteq \mathsf{S}$ be the set of parameterized configurations of S reachable from the combined initial configuration (p_0, q_0). Then we can see that a combined state $\mathsf{n} \in \mathsf{IN}^k \times F$ can be reached from a combined state (c_0, q_0), where $\mathsf{c}_0 \in \mathsf{p}_0$, iff it is reachable from a state in $Reach(\mathsf{S})$ by firing a sequence of only those transitions that are not labeled with the specialized conjunctive guard g_{i}, viz., that are labeled either with *true* or a disjunctive guard. Thus if $Reach(\mathsf{S})$ can be determined then using the backward reachability procedure presented in section 6.1, we can decide whether n is reachable from (c_0, q_0) or not. Therefore the problem boils down to determining the set $Reach(\mathsf{S})$. This can be done by a fixed point computation starting from the set $\{(\mathsf{p}_0, q_0)\}$ that in each iteration, given the current set of nodes S' determines what set of combined states of $\mathsf{S} \setminus \mathsf{S}'$ are reachable from S'. This is done as follows. Add $\mathsf{s} \in \mathsf{S} \setminus \mathsf{S}'$ to S' iff :

- $\mathsf{s} = (\mathsf{p}_{(b,\mathsf{i})}, q)$, there exists $(\mathsf{p}_{(a,\mathsf{i})}, q') \in \mathsf{S}'$ and $l \in \Sigma_{in}$ such that $a \xrightarrow{l} b \in R$ and $q' \xrightarrow{l} q \in \delta$ with $a \xrightarrow{l} b$ labeled with guard g_{i}.

- $\mathsf{s} = (\mathsf{p}_{(b,c,\mathsf{i})}, q)$, there exists $(\mathsf{p}_{(a,\mathsf{i})}, q') \in \mathsf{S}'$ and $l \in \Sigma_{pr}$ such that $a \xrightarrow{l!} b, \mathsf{i} \xrightarrow{l?} c \in R$ and $q' \xrightarrow{l} q \in \delta$ with $a \xrightarrow{l!} b$ being labeled with guard g_{i}.

- $\mathsf{s} = (\mathsf{p}_{(b,c)}, q)$, there exists $(\mathsf{p}_{(a,\mathsf{i})}, q') \in \mathsf{S}'$ and $l \in \Sigma_b$ such that $a \xrightarrow{l!!} b, \mathsf{i} \xrightarrow{l??} c \in R$ and $q' \xrightarrow{l} q \in \delta$ with $a \xrightarrow{l!!} b$ being labeled with guard g_{i}.

- $\mathsf{s} = (\mathsf{p}_{(b,\mathsf{i})}, q)$, there exists $(\mathsf{p}_{(a,\mathsf{i})}, q') \in \mathsf{S}'$ and $l \in \Sigma_{ar}$ such that $a \xrightarrow{l\uparrow} b \in R$, there does not exist a transition of the form $\mathsf{i} \xrightarrow{l\downarrow} c$, where $c \neq \mathsf{i}$ in R, and $q' \xrightarrow{l} q \in \delta$ with $a \xrightarrow{l\uparrow} b$ being labeled with g_{i}.

- $\mathsf{s} = (\mathsf{p}_{(b,c,\mathsf{i})}, q)$, there exists $(\mathsf{p}_{(a,\mathsf{i})}, q') \in \mathsf{S}'$ and $l \in \Sigma_{ar}$ such that $a \xrightarrow{l\uparrow} b, \mathsf{i} \xrightarrow{l\downarrow} c \in R$ and $q' \xrightarrow{l} q \in \delta$

with $a \stackrel{l\uparrow}{\rightarrow} b$ being labeled with guard g_i.

- $\mathsf{s} = (\mathsf{p}_{(a,i)}, q)$ is reachable from a state in S' by firing only those transitions that are not labeled with g_i. Note that since U is initializable, therefore we can, starting in any global state of U^n force any set of process not in their initial state into their initial states one by one by firing the initializing transition. Thus to check whether s is reachable from S' by firing only those transitions that are not labeled with g_i, it suffices to check whether the upward closed set $\mathsf{C} = \{\mathsf{t} | \mathsf{s} \trianglelefteq \mathsf{t}\}$ is reachable from a combined state in S' by firing only those transitions that are not labeled with g_i. This is done using the backward reachability procedure (section 4.1).

The above step is repeated until no more state are added. Clearly at most $|\mathsf{S}|$ iterations are needed. We thus have:

Theorem 6.1 The PMCP for regular properties is decidable for guarded protocols of the form $\mathsf{GP}(C, U)$, where templates C and U are initializable and the conjunctive guard is of the specialized form $g_i = \bigwedge(i)$.

7 Concluding Remarks

We have studied the decidability of the PMCP for guarded protocols and have delineated the decidability/undecidability boundary for various linear time properties. For protocols with conjunctive guards, we first showed undecidability of the PMCP for LTL, regular and ω-regular properties, allowing us to exhibit a decidability gap between stuttering insensitive (LTL\X) and stuttering sensitive properties (LTL). Roughly speaking, this is because the presence of the next-time operator, X, allows us to "count" the number of transitions fired starting at a given state, thereby giving us the ability to simulate incrementing and decrementing the counters of a 2-counter machine by exactly one as opposed to stuttering insensitive logics, where there is no control on the amount by which counters can be incremented or decremented. We next settled the decidability issue for PMCP for guarded protocols with asynchronous rendezvous, by showing undecidability of the PMCP for **p1**, **p2** and **p4**. This was followed by the establishment of a hierarchy based on the relative expressive power of the various communication primitives which is not only useful in its own right but also allowed us to pin down the decidability/undecidability of PMCP for its various formulations. The results are represented in a compact form in figure 6 from where one can deduce the decidability of the PMCP for any of the properties **p1**, **p2**, **p3** and **p4** for a guarded protocol with any combination of the five communication primitives considered in this paper.

In the context of sound and complete verification of cache coherence protocols, Delzanno [5] has identified a broad framework for reasoning about safety properties, but this decision procedure is guaranteed to terminate only for the special case of broadcast protocols which cannot model protocols like the Illinois-MESI that require conjunctive guards. On the other hand, by exploiting features common to most cache coherence protocols, we have proposed a broad framework than can be used to model parameterized versions of all cache coherence protocols in [13] and for which the PMCP for safety properties is decidable. Thus by appropriately restricting the framework, we could identify a useful framework for which the PMCP for safety is decidable thereby beating the undecidability results.

References

[1] P. Abdulla, K. Cerans, B. Johnson, Y. K. Tsay. General Decidability Theorems for Infinite State Systems. *LICS*. 1996.

[2] K. Apt and D. Kozen. Limits for automatic verification of finite-state concurrent systems. *IPL*, 15, 307-309, 1986.

[3] D. E. Culler and J. P. Singh. Parallel Computer Architecture: A Hardware/Software Approach. Morgan Kaufmann Publishers, 1998.

[4] E. M. Clarke and E. A. Emerson. Design and Synthesis of Synchronization Skeletons using Branching Time Temporal Logic. Logics of Programs Workshop, LNCS 131, pp 52-71, 1981.

[5] G. Delzanno. Automatic Verification of Parameterized Cache Coherence Protocols. CAV 2000, 51-68.

[6] G. Delzanno, J. Raskin, L. V. Begin. Towards the Automated Verification of Multithreaded Java Programs. TACAS 2002. 173-187.

[7] E.A. Emerson and V. Kahlon. Reducing Model Checking of the Many to the Few. CADE. 2000.

[8] E.A. Emerson and K.S. Namjoshi. On Model Checking for Non-Deterministic Infinite-State Systems. LICS 1998.

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

[10] J. Esparza, A Finkel and R. Mayr, On the Verification of Broadcast Protocols. LICS 1999, 352-359.

[11] J. Esparza and M. Nielsen: Decidability Issues for Petri Nets - a survey Journal of Information Processing and Cybernetics 30(3), 1994, pp. 143-160.

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

[13] J. Handy. The Cache Memory Book. Academic Press, 1993.

[14] M. Vardi and P. Wolper. An Automata-Theoretic Approach to Automatic Program Verification. LICS, 1986.