Parameterized Model Checking of Rendezvous Systems *

Benjamin Aminof[1], Tomer Kotek[2], Sasha Rubin[2], Francesco Spegni[3], Helmut Veith[2]

[1] IST Austria
[2] TU Wien, Austria
[3] UnivPM Ancona, Italy

Abstract. A standard technique for solving the parameterized model checking problem is to reduce it to the classic model checking problem of finitely many finite-state systems. This work considers some of the theoretical power and limitations of this technique. We focus on concurrent systems in which processes communicate via pairwise rendezvous, as well as the special cases of disjunctive guards and token passing; specifications are expressed in indexed temporal logic without the next operator; and the underlying network topologies are generated by suitable Monadic Second Order Logic formulas and graph operations. First, we settle the exact computational complexity of the parameterized model checking problem for some of our concurrent systems, and establish new decidability results for others. Second, we consider the cases that model checking the parameterized system can be reduced to model checking some fixed number of processes, the number is known as a cutoff. We provide many cases for when such cutoffs can be computed, establish lower bounds on the size of such cutoffs, and identify cases where no cutoff exists. Third, we consider cases for which the parameterized system is equivalent to a single finite-state system (more precisely a Büchi word automaton), and establish tight bounds on the sizes of such automata.

1 Introduction

Many concurrent systems consist of an arbitrary number of identical processes running in parallel, possibly in the presence of an environment or control process. The parameterized model checking problem (PMCP) for concurrent systems is to decide if a given temporal logic specification holds irrespective of the number of participating processes.

Although the PMCP is undecidable in general (see [1,2]) for some combinations of communication primitives, network topologies, and specification languages, it is often proved decidable by a reduction to model checking finitely

* The second, third, fourth and fifth authors were supported by the Austrian National Research Network S11403-N23 (RiSE) of the Austrian Science Fund (FWF) and by the Vienna Science and Technology Fund (WWTF) through grants PROSEED, ICT12-059, and VRG11-005.

many finite-state systems [3,4,2,5,6]. In many of these cases it is even possible to reduce the problem of whether a parameterized system satisfies a temporal specification for any number of processes to the same problem for systems with at most c processes. The number c is known as a *cutoff* for the parameterized system. In other cases the reduction produces a single finite-state system, often in the form of an automaton such as a Büchi automaton, that represents the set of all execution traces of systems of all sizes.

The goal of this paper is to better understand the power and limitations of these techniques, and this is done by addressing three concrete questions.

Question 1: For which combinations of communication primitive, specification language, and network topologies is the PMCP decidable? In the decidable cases, what is the computational complexity of the PMCP?

In case a cutoff c exists, the PMCP is decidable by a reduction to model checking c many finite-state systems. The complexity of this procedure depends on the size of the cutoff. Thus we ask:

Question 2: When do cutoffs exist? In case a cutoff exists, can it be computed? And if so, what is a lower bound on the cutoff?

The set of execution traces of a parameterized system (for a given process type P) is defined as the projection onto the local states of P of all (infinite) runs of systems of all sizes.[4] In case this set is ω-regular, one can reduce the PMCP of certain specifications (including classic ones such as coverability) to the language containment problem for automata (this is the approach taken in [3, Section 4]). Thus we ask:

Question 3: Is the set of executions of the system ω-regular? If so, what is a lower bound on the sizes of the non-deterministic Büchi word automata recognizing the set of executions?

System model. In order to model and verify a concurrent system we should specify three items: (i) the communication primitive, (ii) the specification language, and (iii) the set of topologies.

We focus on concurrent systems in which processes communicate via pairwise rendezvous [3], as well as two other communication primitives (expressible in terms of pairwise rendezvous), namely disjunctive guards [4] and token-passing systems [2,5,6]. Two special cases are systems with one process template U (in other words, all processes run the same code), and systems with two process templates C, U in which there is exactly one copy of C; in other words, all processes run the same code, except for one (which is called the controller).

Specifications of parameterised systems are typically expressed in indexed temporal logic [7] which allows one to quantify over processes (e.g., $\forall i \neq j.$ $\mathsf{AG}(\neg(\text{critical}, i) \vee \neg(\text{critical}, j))$ says that no two processes are in their critical sections at the same time). We focus on a fragment of this logic where the process quantifiers only appear at the front of a temporal logic formula — al-

[4] Actually we consider the destuttering of this set, as explained in Section 2.5.

lowing the process quantifiers to appear in the scope of path quantifiers results in undecidability even with no communication between processes [8].

The sets of topologies we consider all have either bounded tree-width, or more generally bounded clique-width, and are expressible in one of three ways. (1) Using MSO, a powerful and general formalism for describing sets of topologies, which can express e.g. planarity, acyclicity and ℓ-connectivity. (2) As *iteratively constructible* sets of topologies, an intuitive formalism which creates graph sequences by iterating graph operations [9]. Many typical classes of topologies (e.g., all rings, all stars, all cliques) are iteratively constructible. (3) As *clique-like* sets of topologies, which includes the set of cliques and the set of stars, but excludes the set of rings. Iteratively constructible and clique-like sets of topologies are MSO-definable, the former in the presence of certain auxiliary relations.

Prior work and our contributions. For each communication primitive (rendezvous, disjunctive guards, token passing) and each question (decidability and complexity, cutoffs, equivalent automata) we summarise the known answers and our contributions. Obviously, the breadth of questions along these axis is great, and we had to limit our choices as to what to address. Thus, this article is not meant to be a comprehensive taxonomy of PMCP. That is, it is not a mapping of the imaginary hypercube representing all possible choices along these axis. Instead, we started from the points in this hypercube that represent the most prominent known results and, guided by the three main questions mentioned earlier, have explored the unknown areas in each point's neighborhood.

Pairwise Rendezvous.

Decidability and Complexity: The PMCP for systems which communicate by pairwise rendezvous, on clique topologies, with a controller C, for 1-index $\mathsf{LTL}\backslash\mathsf{X}$ specifications is EXPSPACE-complete [3,10] (and PSPACE without a controller [3, Section 4]). We show the PMCP is undecidable if we allow the more general 1-index $\mathsf{CTL}^*\backslash\mathsf{X}$ specifications. Thus, for the results on pairwise rendezvous we fix the specification language to be 1-index $\mathsf{LTL}\backslash\mathsf{X}$. We introduce sets of topologies that naturally generalise cliques and stars, and exclude rings (the PMCP is already undecidable for uni-directional rings and 1-index safety specifications [1,2]), which we call *clique-like* sets of topologies, and show that the PMCP of 1-index $\mathsf{LTL}\backslash\mathsf{X}$ on clique-like topologies is EXPSPACE-complete (PSPACE-complete without a controller). We also prove that the program complexity is EXPSPACE-complete (respectively PTIME).

Cutoffs: We show that even for clique topologies there are not always cutoffs.

Equivalent automata: We prove that the set of (destuttered) executions of systems with a controller are not, in general, ω-regular, already for clique topologies. On the other hand, we extend the known result that the set of (destuttered) executions for systems with only user processes U (i.e., without a controller) is ω-regular for clique topologies [3] to clique-like topologies, and give an effective construction of the corresponding Büchi automaton.

Disjunctive guards.

In this section we focus on clique topologies and 1-index $\mathsf{LTL}\backslash\mathsf{X}$ specifications. Though we sometimes consider more general cases (as in Theorem 10), we postpone these cases for future work.

Decidability and Complexity: We show the PMCP is undecidable if we allow 1-index $\mathsf{CTL}^*\backslash\mathsf{X}$ specifications, already for clique topologies. We prove that for systems with a controller the complexity of the PMCP is PSPACE-complete and the program complexity is coNP-complete, whereas for systems without a controller the complexity is PSPACE-complete and the program complexity is in PTIME. We note that the PTIME and PSPACE upper bounds follow from [3,4], although we improve the time complexity for the case with a controller.

Cutoffs: Cutoffs exist for such systems and are of size $|U| + 2$ [4]. We prove these cutoffs are tight.

Equivalent automaton: We prove that the set of (destuttered) executions is accepted by an effectively constructible Büchi automaton of size $O(|C| \times 2^{|U|})$. It is very interesting to note that this size is smaller than the smallest system size one gets (in the worst-case) from the cutoff result, namely $|C| \times |U|^{|U|+2}$. Hence, the PMCP algorithm obtained from the cutoff is less efficient than the one obtained from going directly to a Büchi automaton. As far as we know, this is the first theoretical proof of the existence of this phenomenon. We also prove that, in general, our construction is optimal, i.e., that in some cases every automaton for the set of (destuttered) executions must be of size $2^{\Omega(|U|+|C|)}$.

Token passing systems.

In this section we focus on MSO-definable set of topologies of bounded tree-width or clique-width, as well as on iteratively-constructible sets of topologies.

Decidability and Complexity: We prove that the PMCP is decidable for indexed $\mathsf{CTL}^*\backslash\mathsf{X}$on such topologies. This considerably generalises the results of [6], where decidability for this logic was shown for a few concrete topologies such as rings and cliques.

Cutoffs: For the considered topologies and indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ we prove that the PMCPs have *computable* cutoffs. From [6] we know that there is a (computable) set of topologies and a system template such that there is no algorithm that given an indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ formula can compute the associated cutoff (even though a cutoff for *the given formula* always exists). This justifies our search of sets of topologies for which the PMCP for $\mathsf{CTL}^*\backslash\mathsf{X}$ has computable cutoffs. We also give a lower bound on cutoffs for iteratively-constructible sets and indexed $\mathsf{LTL}\backslash\mathsf{X}$.

Equivalent automaton: Our ability to compute cutoffs for 1-index $\mathsf{LTL}\backslash\mathsf{X}$ formulas and the considered topologies implies that the (destuttered) sets of execution traces are ω-regular, and the construction of Büchi automata which compute these traces is effective.

Due to space limitations, in many cases proofs/sketches are not given, and only a statement of the basic technique used for the proof is given. The reader is referred to the full version of the article for more details.

2 Definitions and Preliminaries

A *labeled transition system (LTS)* is a tuple $(S, R, I, \Phi, \mathsf{AP}, \Sigma)$, where S is the set of *states*, $R \subseteq S \times \Sigma \times S$ is the *transition relation*, $I \subseteq S$ are the *initial states*, $\Phi : S \to 2^{\mathsf{AP}}$ is the *state-labeling*, AP is a set of *atomic propositions* or *atoms*, and Σ is the *transition-labels alphabet*. When AP and Σ are clear from the context we drop them. A *finite LTS* is an LTS in which S, R, Σ are finite and $\Phi(s)$ is finite for every $s \in S$. Transitions $(s, a, s') \in R$ may be written $s \xrightarrow{\mathsf{a}} s'$. A *transition system (TS)* (S, R, I, Σ) is an LTS without the labeling function and without the set of atomic propositions. A *run* is an infinite path that starts in an initial state. For a formal definition of path, state-labeled path, action-labeled path, refer to the full version of this paper.

2.1 Process Template, Topology, Pairwise Rendezvous System

We define how to (asynchronously) compose processes that communicate via pairwise rendezvous into a single system. We consider time as being discrete (i.e. not continuous). Processes are not necessarily identical, but we assume only a finite number of different process types. Roughly, at every vertex of a topology (a directed graph with vertices labeled by process types) there is a process of the given type running; at every time step either, and the choice is nondeterministic, exactly one process makes an internal transition, or exactly two processes with an edge between them in the topology instantaneously synchronize on a message (sometimes called an action) $\mathsf{m} \in \Sigma_{\mathsf{sync}}$. The sender of the message m performs an $\mathsf{m}!$ transition, and the receiver an $\mathsf{m}?$ transition. Note that the sender can not direct the message to a specific neighbouring process (nor can the receiver choose from where to receive it), but the pair is chosen non-deterministically. [5]

Fix a countable set of atoms (also called atomic propositions) $\mathsf{AP}_{\mathsf{pr}}$. Fix a finite synchronization alphabet Σ_{sync} (that does not include the symbol τ), and define the *communication alphabet* $\Sigma = \{\mathsf{m}!, \mathsf{m}? \mid \mathsf{m} \in \Sigma_{\mathsf{sync}}\}$.

Process Template, System Arity, System Template. A *process template* is a finite LTS $P = (S, R, \{\iota\}, \Phi, \mathsf{AP}_{\mathsf{pr}}, \Sigma \cup \{\tau\})$. Since $\mathsf{AP}_{\mathsf{pr}}$ and the transition-labels alphabet are typically fixed, we will omit them. The *system arity* is a natural number $r \in \mathbb{N}$. It refers to the number of different process types in the system. A *(r-ary) system template* is a tuple of process templates $\overline{P} = (P_1, \cdots, P_r)$ where r is the system arity. The process template $P_i = (S_i, R_i, \{\iota_i\}, \Phi_i)$ is called the *ith process template*.

Topology G. An *r-topology* is a finite structure $G = (V, E, T_1, \cdots, T_r)$ where $E \subseteq V \times V$, and the $T_i \subseteq V$ partition V. The *type* of $v \in V$ denoted $type(v)$ is the unique j such that $v \in T_j$. We might write V_G, E_G and $type_G$ to stress G.

We sometimes assume that $V := \{1, \cdots, n\}$ for some $n \in \mathbb{N}$. For instance, an *r-ary clique topology* with $V = \{1, \cdots, n\}$ has $E = \{(i, j) \in [n]^2 \mid i \neq j\}$ (and

[5] In models in which we allow processes to send in certain directions, e.g., send left and send right in a bi-directional ring, then PMCP is quickly undecidable [6].

some partition of the nodes into sets $T_1, \cdots, T_r$); and the 1-*ary ring topology* with $V = \{1, \cdots, n\}$ has $E = \{(i,j) \in [n]^2 \mid j = i+1 \mod n\}$ and $T_1 = V$.

(Pairwise-Rendezvous) System. Given system arity r, system template $\overline{P} = (P_1, \cdots, P_r)$ with $P_i = (S_i, R_i, \{\iota_i\}, \Phi_i)$, and r-topology $G = (V, E, \overline{T})$, define the *system* $\overline{P}^G$ as the LTS $(Q, \Delta, Q_0, \Lambda, \mathsf{AP}_{\mathsf{pr}} \times V, \Sigma_{\mathsf{sync}} \cup \{\tau\})$ where

- The set Q is the set of functions $f : V \to \cup_{i \leq r} S_i$ such that $f(v) \in S_i$ iff $type(v) = i$ (for all $v \in V, i \leq r$). Such functions (sometimes written as vectors) are called *configurations*.
- The set Q_0 consists of the unique *initial configuration* f_ι defined as $f_\iota(v) = \iota_{type(v)}$ (for all $v \in V$).
- The set of *global transitions* Δ are tuples $(f, \mathsf{m}, g) \in Q \times (\Sigma_{\mathsf{sync}} \cup \{\tau\}) \times Q$ where one of the following two conditions hold:
  - $\mathsf{m} = \tau$ and there exists $v \in V$ such that $f(v) \xrightarrow{\tau} g(v)$ is a transition of the process template $P_{type(v)}$, and for all $w \neq v$, $f(w) = g(w)$; this is called an *internal transition*,
  - $\mathsf{m} \in \Sigma_{\mathsf{sync}}$ and there exists $v \neq w \in V$ with $(v,w) \in E$ such that $f(v) \xrightarrow{\mathsf{m}!} g(v)$ and $f(w) \xrightarrow{\mathsf{m}?} g(w)$ and for all $z \notin \{v, w\}$, $f(z) = g(z)$; this is called a *synchronous transition*. We say that the process at v *sends the message* m and the process at w *receives the message* m.
- The labeling function $\Lambda : Q \to 2^{\mathsf{AP}_{\mathsf{pr}} \times V}$ is defined by $(p, v) \in \Lambda(f) \iff p \in \Phi_{type(v)}(f(v))$ (for all configurations f, atoms $p \in \mathsf{AP}_{\mathsf{pr}}$ and vertices $v \in V$).

In words then, a topology of size n specifies n-many processes, which processes have the same type, and how the processes are connected. In the internal transition above only the process at vertex v makes a transition, and in the synchronous transition above only the process at vertex v and its neighbour at w make a transition. Let $\pi = f_0 f_1 \cdots$ be a state-labeled path in $\overline{P}^G$. The *projection of π to vertex $v \in V$*, written $proj_v(\pi)$, is the sequence $f_0(v) f_1(v) \cdots$ of states of $P_{type(v)}$. If $type(v) = j$ we say that the *vertex v runs (a copy of) the process P_j*, or that *the process at v is P_j*.

2.2 Disjunctively-Guarded System, and Token Passing System

We define guarded protocols and token-passing systems as restricted forms of pairwise rendezvous systems. In fact, the restrictions are on the system template and the synchronization alphabet. Write $P_i = (S_i, R_i, \{\iota_i\}, \Phi_i, \mathsf{AP}_{\mathsf{pr}}, \Sigma \cup \{\tau\})$.

Disjunctively-Guarded System Template. A system $\overline{P}^G$ is *disjunctively-guarded* if $\overline{P}$ is. A system template $\overline{P}$ is *disjunctively-guarded* if **(i)** The state sets of the process templates are pairwise disjoint, i.e., $S_i \cap S_j = \emptyset$ for $1 \leq i < j \leq r$. **(ii)** The transition-labels alphabet Σ is $\{\tau\} \cup \{\mathsf{q}!, \mathsf{q}? \mid q \in \cup_{i \leq r} S_i\}$ **(iii)** For every state $s \in S$, there is a transition labeled $s \xrightarrow{\mathsf{s}?} s$. **(iv)** For every state $s \in S$, the only transitions labeled $\mathsf{s}?$ are of the form $s \xrightarrow{\mathsf{s}?} s$. Intuitively, in this kind of systems a process can decide to move depending on the local state of some

neighbor process, but it cannot relate the state of any two processes at the same time, nor it can force another process to move from its local state. Our definition of disjunctively-guarded systems on a clique topology is a reformulation of the definition of concrete system in [4, Section 2].

Token Passing System. One can express a token passing system (TPS) as a special case of pairwise rendezvous. In this work we only consider the case of a single valueless token, whose formal definition can be found in [6,2]. A token passing system (TPS) $\overline{P}^G$ can be thought of the asynchronous parallel composition of the processes templates in $\overline{P}$ over topology G according to the types of vertices. At any time during the computation, exactly one vertex has the token. The token starts with the unique process in P_1. At each time step either exactly one process makes an internal transition, or exactly two processes synchronize when one process sends the token to another along an edge of G.

2.3 Indexed Temporal Logic

We assume the reader is familiar with the syntax and semantics of CTL^* and LTL. Indexed temporal logics were introduced by [7] to model specifications of certain distributed systems. They are obtained by adding *vertex quantifiers* to a given temporal logic over indexed atomic propositions. For example, in a system with two process templates, the formula $\forall i : \mathit{type}(i) = 1.\, \mathsf{AG}((\mathit{good}, i))$ states that every process of type 1 on all computations at all points of time satisfies the atom *good*. In a system with one process template, the formula $\forall i \neq j.\, \mathsf{AG}(\neg(\mathit{critical}, i) \vee \neg(\mathit{critical}, j))$ states that it is never the case that two processes both satisfy the atom *critical* at the same time.

Syntax. Fix an infinite set $\mathsf{Vars} = \{i, j, \ldots\}$ of vertex variables (called index variables for the clique topology). A *vertex quantifier* is an expression of the form $\exists x : \mathit{type}(x) = m$ or $\forall x : \mathit{type}(x) = m$ where $m \in \mathbb{N}$. An *indexed CTL^* formula over vertex variables Vars and atomic propositions $\mathsf{AP_{pr}}$* is a formula of the form $Q_1 i_1, \ldots, Q_k i_k : \varphi.$, where each $i_n \in \mathsf{Vars}$, each Q_{i_n} is an index quantifier, and φ is a CTL^* formula over atomic predicates $\mathsf{AP_{pr}} \times \mathsf{Vars}$.

The semantics is fully described in the full version of this paper. For 1-ary systems we may write $\forall x$ instead of $\forall x : \mathit{type}(x) = 1$. In the syntax of indexed formulas we may write $\mathit{type}(x) = P_m$ instead of $\mathit{type}(x) = m$. i-CTL^* denotes the set of all indexed CTL^* sentences, and k-CTL^* for the set of all k-indexed formulas in i-CTL^*, i.e., formulas with k quantifiers. We similarly define indexed versions of various fragments of CTL^*, e.g., i-LTL, k-$\mathsf{LTL}\backslash\mathsf{X}$ and k-$\mathsf{CTL}^*_d\backslash\mathsf{X}$ (k-CTL^* formulas with nesting depth of path quantifiers at most d). We write $\overline{P}^G \equiv_{\text{k-}\mathsf{CTL}^*_d\backslash\mathsf{X}} \overline{P}^{G'}$, if $\overline{P}^G$ and $\overline{P}^{G'}$ agree on all k-$\mathsf{CTL}^*_d\backslash\mathsf{X}$ formulas.

Note. The index variables are bound *outside* of all the temporal path quantifiers (A and E). In particular, for an existentially quantified LTL formula to be satisfied there must exist a valuation of the index variables such that ϕ holds for all runs (and not one valuation for each run). Thus this logic is sometimes called prenex indexed temporal logic. Note that if one allows index quantifiers inside the scope

of temporal path quantifiers then one quickly reaches undecidability even for systems with no communication [8].

For the remainder of this paper specifications only come from i-CTL$^*\backslash$X, i.e., without the next-time operators. It is usual in the context of parameterized systems to consider specification logics without the "next" (X) operator.

2.4 Parameterized Topology, Parameterized System, PMCP, Cutoff

Parameterized Topology $\mathcal{G}$. An *(r-ary) parameterized topology* $\mathcal{G}$ is a set of r-topologies. Moreover, we assume membership in $\mathcal{G}$ is decidable. Typical examples are the set of r-ary cliques or the set of 1-ary rings.

Parameterized Model Checking Problem. Fix an r-ary parameterized topology $\mathcal{G}$, a set of r-ary system templates $\mathcal{P}$, and a set of indexed temporal logic sentences $\mathcal{F}$. The *parameterized model checking problem (PMCP)* for this data, written $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P},\mathcal{F})$, is to decide, given a formula $\varphi \in \mathcal{F}$ and a system template $\overline{P} \in \mathcal{P}$, whether for all $G \in \mathcal{G}$, $\overline{P}^G \models \varphi$. The complexity of the $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P},\mathcal{F})$, where the formula $\varphi \in \mathcal{F}$ is fixed and only the system template is given as an input, is called the *program complexity.*

Cutoff. A *cutoff* for $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P},\mathcal{F})$ is a natural number c such that for every $\overline{P} \in \mathcal{P}$ and $\varphi \in \mathcal{F}$, the following are equivalent: **(i)** $\overline{P}^G \models \varphi$ for all $G \in \mathcal{G}$ with $|V_G| \leq c$; **(ii)** $\overline{P}^G \models \varphi$ for all $G \in \mathcal{G}$.

Lemma 1. *If $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P},\mathcal{F})$ has a cutoff, then $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P},\mathcal{F})$ is decidable*

Proof. If c is a cutoff, let $G_1,\ldots,G_n$ be all topologies G in $\mathcal{G}$ such that $|V_G| \leq c$. The algorithm that solves PMCP takes $\overline{P},\varphi$ as input and checks whether or not $\overline{P}^{G_i} \models \varphi$ for all $1 \leq i \leq n$. □

2.5 Destuttering and Process Executions

The *destuttering of an infinite word* $\alpha \in \Sigma^\omega$ is the infinite word $\alpha^\delta \in \Sigma^\omega$ defined by replacing every maximal finite consecutive sequence of repeated symbols in α by one copy of that symbol. Thus, the destuttering of $(aaba)^\omega$ is $(ab)^\omega$; and the destuttering of aab^ω is ab^ω. The *destuttering of set* $L \subseteq \Sigma^\omega$, written L^δ, is the set $\{\alpha^\delta \mid \alpha \in L\} \subseteq \Sigma^\omega$.

It is known that LTL$\backslash$X can not distinguish between a word and its destuttering, which is the main motivation for the following definition.

Process Executions. For parameterized r-topology $\mathcal{G}$, r-ary system template $\overline{P} = (P_1,\cdots,P_r)$ and $t \leq r$, define the set of *(process) executions (with respect to $t,\overline{P},\mathcal{G}$)*, written t-EXEC$_{\mathcal{G},\overline{P}}$, as the destuttering of the following set:

$$\bigcup_{G\in\mathcal{G}}\{proj_v(\pi) \mid \pi \text{ is a state-labelled run of } \overline{P}^G \text{ and } v \in V_G \text{ is of type } t\}.$$

When $\mathcal{G}$ or $\overline{P}$ is clear from the context we may omit them.

The point is that for universal 1-index LTL$\backslash$X we can reduce the PMCP to model checking a single system whose runs are t-EXEC$_{\mathcal{G},\overline{P}}$. This is explained in details in the full version of this paper.

2.6 Two prominent kinds of pairwise-rendezvous systems

Identical processes. Concurrent systems in which all processes are identical are modeled with system arity $r = 1$. In this case there is a single process template P, and a topology may be thought of as a directed graph $G = (V, E)$ (formally $G = (V, E, T_1)$ with $T_1 = V$). We write USER-EXEC$_{\mathcal{G}}(U)$ for the set of executions of the user processes in a 1-ary system, i.e., 1-EXEC$_{\mathcal{G},U}$.

Identical processes with a controller. Concurrent systems in which all processes are identical except for one process (typically called a controller or the environment) are modeled with system arity $r = 2$, and system templates of the form (P_1, P_2), and we restrict the topologies so that exactly one vertex has type 1 (i.e., runs the controller). We call such topologies *controlled*. We often write (C, U) instead of (P_1, P_2), and $G = (V, E, v)$ instead of $(V, E, \{v\}, V \setminus \{v\})$. We write CONTROLLER-EXEC$_{\mathcal{G}}(C, U)$ for the set of executions of the controller process, i.e., 1-EXEC$_{\mathcal{G},(C,U)}$. We write USER-EXEC$_{\mathcal{G}}(C, U)$ for the set of executions of the user processes in this 2-ary system, i.e., 2-EXEC$_{\mathcal{G},(C,U)}$.

2.7 Classes of parameterized topologies

Here we define the classes of parameterized topologies which we will use in the sequel. The classes we define all have bounded clique-width.

w-terms and clique-width. An *r-ary w-topology* $(V, E, T_1, \ldots, T_r, C_1, \ldots, C_w)$ extends $(V, E, T_1, \ldots, T_r)$ by a partition $(C_1, \ldots, C_w)$ of V. For every $u \in V$, if $u \in C_i$ then we say u has color i. We define the w-terms inductively. ϵ is a w-term. If x, y are w-terms, then $add_{i,t}(x)$, $recol_{i,j}(x)$, $edge_{i,j}(x)$ and $x \sqcup y$ are w-terms for $i, j \in [w]$, $t \in [r]$. Every w-term x has an associated w-topology $[[x]]$:

- $[[\epsilon]]$ has $V = E = \emptyset$ and empty labeling.
- $[[add_{i,t}(x)]]$ is formed by adding a new vertex of color i and type t to $[[x]]$.
- $[[recol_{i,j}(x)]]$ is formed by recoloring every vertex with color i of $[[x]]$ by j.
- $[[edge_{i,j}(x)]]$ is formed from $[[x]]$ by adding an edge from every vertex of color i to every vertex of color j.
- $[[x \sqcup y]]$ is the disjoint union of x and y and the union of the labelings.

A topology G has *clique-width at most w* if there is a w-term ρ such that G is isomorphic to $[[\rho(\epsilon)]]$ (forgetting the coloring $C_1, \ldots, C_w$). Every topology of size n has clique-width at most n. A class of topologies $\mathcal{G}$ has *bounded clique-width* if there exists w such that every graph in $\mathcal{G}$ has clique-width at most w. It is well-known if $\mathcal{G}$ has bounded tree-width, then it has bounded clique-width.

Monadic Second Order Logic MSO. MSO is a powerful logic for graphs and graph-like structures. It is the extension of First Order Logic with set quantification. MSO can define classic graph-theoretic concepts such as planarity, connectivity, c-regularity and c-colorability. We assume the reader is familiar with Monadic Second Order logic as described e.g. in [11]. A parameterized topology $\mathcal{G}$ is *MSO-definable* if there exists an MSO-formula Φ such that $G \in \mathcal{G}$

iff $G \models \Phi$. E.g., $\exists U \forall x \forall y (E(x,y) \rightarrow (U(x) \leftrightarrow \neg U(y)))$ defines the set of bipartite graphs. We denote by $\equiv_q^{\mathsf{MSO}}$ the equivalence relation of topologies of being indistinguishable by MSO-formulas of quantifier rank q.

Theorem 1 (Courcelle's Theorem, see [11]). *Let $w \geq 1$ and let $\varphi \in \mathsf{MSO}$. The MSO theory of r-topologies of clique-width w is decidable. I.e., there is an algorithm that on input $\varphi \in \mathsf{MSO}$, decides whether there is an r-topology G of clique-width at most w such that $G \models \varphi$. Moreover, the number of equivalence classes in $\equiv_q^{\mathsf{MSO}}$ is finite and computable, and a topology belonging to each class is computable.*

We now define a user-friendly and expressive formalism that can be used to generate natural parameterized topologies.

Iteratively constructible parameterized topologies. A parameterized topology is *iteratively constructible* if it can be built from an initial labeled graph by means of a repeated fixed succession of elementary operations involving addition of vertices and edges, deletion of edges, and relabeling. More precisely, an r-ary parameterized topology $\mathcal{G}$ is *iteratively-constructible* if there are w-terms $\rho(x), \sigma(x)$ with one variable x and no use of disjoint union, and a w-graph H_0 such that (i) $G \in \mathcal{G}$ iff $G = \sigma(\rho^n(H_0))$ for some $n \in \mathbb{N}$, where $\rho^0(H) = H$, (ii) exactly one vertex of H_0 has type 1, and (iii) no vertex of type 1 is added in ρ or σ. For terms $\rho(\cdot)$ and $\rho'(\cdot)$ we write $\rho :: \rho'$ instead of $\rho(\rho'(\cdot))$. Intuitively, ρ "builds up" the topology, and σ puts on the "finishing touch" (see examples below). The unique vertex of type 1 can act as the controller if it is assigned a unique process template, and it is the initial token position in TPSs.

Example 1 (Cliques and rings). The set of cliques (irreflexive) is iteratively constructible: let H_0 consist of a single vertex v of color 1 and type 1, let $\rho(x)$ be $edge_{1,1} :: add_{1,2}(x)$, and $\sigma(x)$ be the identity.

The set of uni-directional rings is iteratively constructible: let H_0 consist of two vertices, one of color 1 and type 1 and one of color 2 and type 2 with an edge from 1 to 2. Let $\rho(x)$ be $recol_{4,2} :: recol_{2,3} :: edge_{2,4} :: add_{4,2}$ and $\sigma(x) = edge_{2,1}$.

Clique-like (and controllerless clique-like) parameterized topologies. We now define other sets of topologies of bounded clique-width that generalise cliques and stars, but not rings.

Let H be an r-ary topology with vertex set V_H of size m in which each vertex has a distinct type. Let $\rho_2(x) = add_{1,type(1)} :: \cdots :: add_{m,type(m)}$. Let $\rho_1(x)$ be the m-term obtained by the composition of $edge_{i,j}$ for all $(i,j) \in E_H$ (in an arbitrary order). Let $\rho(x) = \rho_1(x) :: \rho_2(x)$. We have $[[\rho(\epsilon)]] = H$.

An r-ary parameterized topology $\mathcal{G}$ is *clique-like* if there is an r-ary topology H and a partition $B_{sng}, B_{clq}, B_{ind}$ of V_H such that $G \in \mathcal{G}$ iff there exists a function $num : B_{clq} \cup B_{ind} \rightarrow \mathbb{N}$ such that $[[\rho^{num}(\epsilon)]] = G$, and ρ^{num} is obtained from ρ by (i) repeating each $add_{i,type(i)}$ $num(i)$ times rather than once, and (ii) finally performing $edge_{i,i}$ for all $i \in B_{clq}$. Intuitively, G is obtained from H by substituting each vertex in B_{clq} with a clique, each vertex in B_{ind} with an independent set, and leaving every vertex in B_{sng} as a single vertex.

We say that $\mathcal{G}$ is *generated by* H *and* $B_{sng}, B_{clq}, B_{ind}$. The cardinality of B_{sng} is the *number of controllers* in $\mathcal{G}$. In case $B_{sng} = \emptyset$ we say that $\mathcal{G}$ is *controllerless*.

Example. Cliques, stars and complete bipartite graphs. Let H be the 2-topology with vertex set $V_H = \{1, 2\}$ and edge set $\{(1, 2), (2, 1)\}$ and $type(i) = i$ for $i \in [2]$. The set of 2-ary cliques in which exactly one index has type 1 is clique-like using H as defined, $B_{clq} = \{2\}$, $B_{ind} = \emptyset$ and $B_{sng} = \{1\}$. The set of stars in which exactly one index has type 1 is clique-like using H above, $B_{clq} = \emptyset$, $B_{ind} = \{2\}$ and $B_{sng} = \{1\}$. The set of topologies that are complete bipartite graphs is clique-like using H above, $B_{ind} = \{1, 2\}$, and $B_{clq} = B_{sng} = \emptyset$.

Example. Rings are not clique-like. Clique-like parameterized topologies have diameter at most $|V_H|$ unless their diameter is infinite. Rings have unbounded but finite diameter and are therefore not clique-like.

3 Results for Pairwise-Rendezvous Systems

The known decidability results for parameterized pairwise-rendezvous systems are for clique topologies and specifications from 1-indexed $\mathsf{LTL}\backslash\mathsf{X}$. [3]. Thus we might hope to generalise this result in two directions: more general specification languages and more general topologies. We first show, by reducing the non-halting problem of two-counter machines (2CMs) to the PMCP, that allowing branching specifications results in undecidability:

Theorem 2. *$\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}, \mathcal{F})$ is undecidable where $\mathcal{F}$ is the set of 1-indexed $\mathsf{CTL}^*_2\backslash\mathsf{X}$ formulas, $\mathcal{G}$ is the set of 1-ary clique topologies, and $\mathcal{P}$ is the set of 1-ary system templates.*

We conclude that we should restrict the specification logic if we want decidability. In the rest of this section we focus on 1-indexed $\mathsf{LTL}\backslash\mathsf{X}$ and parameterized clique-topologies with or without a controller (note that the PMCP for 1-indexed $\mathsf{LTL}\backslash\mathsf{X}$ is undecidable for topologies that contain uni-directional rings [1,2]).

Pairwise Rendezvous: Complexity of PMCP. The proof of the following theorem extends the technique used in [3, Theorem 3.6] for clique topologies:

Theorem 3. *Fix an r-ary clique-like parameterized topology $\mathcal{G}$, let $\mathcal{F}$ be the set of 1-index $\mathsf{LTL}\backslash\mathsf{X}$ formulas, and $\mathcal{P}$ the set of r-ary system templates. Then $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}, \mathcal{F})$ is decidable in EXPSPACE.*

Thus, using the fact that PMCP is EXPSPACE-hard already for clique topologies and the coverability problem [10], we get:

Theorem 4. *Fix an r-ary clique-like parameterized topology $\mathcal{G}$, let $\mathcal{F}$ be the set of 1-index $\mathsf{LTL}\backslash\mathsf{X}$ formulas, and $\mathcal{P}$ the set of r-ary system templates. Then $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}, \mathcal{F})$ is EXPSPACE-complete. The same holds for program complexity.*

It is known that PMCP for 1-ary cliques is PSPACE-complete (the upper bound is from [3, Section 4], and the lower bound holds already for $\mathsf{LTL}\backslash\mathsf{X}$ model checking a single finite state system P, with no communication). We extend the

upper bound to clique-like topologies in which $B_{sng} = \emptyset$, i.e., controllerless clique-like parameterized topologies. The proof follows [3] and is via a reduction to emptiness of Büchi automata, see Theorem 8.

Theorem 5. *Fix an r-ary controllerless clique-like parameterized topology $\mathcal{G}$, let $\mathcal{F}$ be the set of 1-index $\mathsf{LTL}\backslash\mathsf{X}$ formulas, and $\mathcal{P}$ the set of r-ary system templates. Then $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}, \mathcal{F})$ is PSPACE-complete, and the program complexity is in PTIME.*

Pairwise Rendezvous: Cutoffs.

Theorem 6. *Let $\mathcal{G}$ be the 1-ary parameterized clique topology and let $\mathcal{F}$ be the set of 1-index $\mathsf{LTL}\backslash\mathsf{X}$ formulas. There exists a process template P such that $\mathsf{PMCP}_{\mathcal{G}}(\{P\}, \mathcal{F})$ has no cutoff.*

Proof (Sketch). Define process template $P = (S, R, I, \Phi)$ by $S := \{1, 2, 3\}$, $I = \{1\}$, $R = \{(1, \tau, 1), (1, a!, 2), (2, \tau, 1), (1, a?, 3)\}$, and $\Phi(i) = \{i\}$. Thus in a system with $n + 1$ processes one possible behaviour is, up to stuttering, $(12)^n 1^\omega$. This run does not appear in any system with $\leq n$ processes. Thus take the formula ϕ_n stating that for every process and every path, the initial segment, up to stuttering, is not of the form $(12)^n$ (for instance $1 \wedge (1\ \mathsf{U}\ (2 \wedge (2\ \mathsf{U}\ 1)))$ states that there is an initial prefix of the form $11^*22^*11^*$). □

Pairwise Rendezvous: Equivalence to finite-state systems. The following theorem says that if there is a cutoff for the set of 1-indexed $\mathsf{LTL}\backslash\mathsf{X}$ formulas then the set of executions is ω-regular. The proof uses the fact that 1-indexed $\mathsf{LTL}\backslash\mathsf{X}$ is expressive enough to describe finite prefixes of infinite words, and deducing that since all finite executions of a system of any size must already appear in systems up to the cutoff size, so do the infinite executions. This holds for general topologies, not only for clique-like ones.

Theorem 7. *Fix r-ary parameterized topology $\mathcal{G}$, let $\mathcal{F}$ be the set of 1-index $\mathsf{LTL}\backslash\mathsf{X}$ formulas, and let $\overline{P}$ be an r-ary system template. If $\mathsf{PMCP}_{\mathcal{G}}(\{\overline{P}\}, \mathcal{F})$ has a cutoff, then for every $t \leq r$, the set of executions t-$\mathrm{EXEC}_{\mathcal{G},\overline{P}}$ is ω-regular.*

The following theorem states that the set of executions of each process in a controllerless parameterized clique-like topology is ω-regular, i.e., recognizable by a Non-deterministic Büchi Word automaton (NBW)(see [12] for a definition). This is done by a reduction to the case of a clique topology and using the corresponding result in [3, Section 4][6]

Theorem 8. *For every controllerless clique-like r-ary parameterized topology $\mathcal{G}$, every r-ary system template $\overline{P}$, and every $i \leq r$, there is a linearly sized NBW (computable in PTIME) that recognises the set i-$\mathrm{EXEC}_{\mathcal{G},\overline{P}}$.*

[6] The relevant result in [3, Section 4] is correct. However, its proof has some bugs and some of the statements (e.g., Theorem 4.8) are wrong. In the full version of this paper we give a correct proof for the main result of [3, Section 4].

By constructing an appropriate system template, and using a pumping argument, we are able to show that the set of executions of systems with a controller is not, in general, ω-regular. More precisely:

Theorem 9. *Let $\mathcal{G}$ be the 2-ary parameterized clique topology. There exist a system template (C, U) for which* CONTROLLER-EXEC$_{\mathcal{G}}(C, U)$ *is not ω-regular.*

4 Results for Disjunctive Guards

In the following we will consider parameterized systems as described in Section 2.6, i.e., with an arbitrary number of copies of one template U, and possibly with a unique controller C, arranged in a clique.

The following theorem follows similar lines as Theorem 2, and uses a reduction from the non-halting problem of 2CMs. The main complication here is that, unlike the case of pairwise rendezvous, mutual exclusion is not easily obtainable using disjunctive guards, and thus more complicated gadgets are needed to ensure that the counter operations are simulated correctly.

Theorem 10. *$\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}, \mathcal{F})$ is undecidable where $\mathcal{F}$ is the set of 1-indexed $\mathsf{CTL}^*_2\backslash\mathsf{X}$ formulas, $\mathcal{G}$ is the 1-ary parameterized clique topology, and $\mathcal{P}$ is the set of 1-ary disjunctively-guarded system templates.*

We conclude that we should restrict the specification logic if we want decidability, and in the rest of this section we focus on 1-indexed $\mathsf{LTL}\backslash\mathsf{X}$.

Disjunctive Guards: Cutoffs. By [4], for the r-ary parameterized clique topology and k-indexed $\mathsf{LTL}\backslash\mathsf{X}$ formulae, there is a cutoff of size $|U| + 2$ (where U is the process template). The following proposition shows that this cutoff is tight.

Proposition 1. *Let $\mathcal{G}$ be the r-ary parameterized clique topology, let $\mathcal{F}$ be the set of 1-index $\mathsf{LTL}\backslash\mathsf{X}$ formulas, and let $k > 0$. There is a disjunctively-guarded system template $\mathcal{P}$ of size $\Theta(k)$ such that $\Theta(k)$ is the smallest cutoff for $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}, \mathcal{F})$*

Proof (sketch). We show the case of 1-ary cliques. Similar examples exist for r-ary systems, with or without a controller. Consider the process template: $U = (S_U, R_U, I_U, \Phi_U)$ where $S_U = \{s_1, \ldots, s_k\}$, $R_U = \{(s_i, s_i, s_{i+1}) \mid i < k\} \cup \{(s_k, s_k, s_1)\} \cup \{(s_i, \top, s_i) \mid i \leq k\}$, $I_U = \{s_1\}$, and $\Phi_U(s_i) = \{s_i\}$; and the formula $\phi_k = \forall x. AG((s_k, x) \rightarrow G(s_k, x))$. Evidently, ϕ_k holds in all systems with at most k processes, but false in systems with $k + 1$ or more processes.

Disjunctive Guards: Equivalence to finite-state systems. There are several techniques for solving the PMCP for 1-indexed $\mathsf{LTL}\backslash\mathsf{X}$ formulae for systems using disjunctive guards. One such technique consists in finding an NBW that model-checks the set of all possible executions of the system, for any number of copies of user processes U. We begin by showing that in general, such an automaton is necessarily big. We show the following lower bound by encoding the language of palindromes of length $2k$.

Proposition 2. *Let $\mathcal{G}$ be the 2-ary parameterized controlled clique topology. For every $k > 0$ there exist a disjunctively-guarded system template (C, U) where the sizes of C and U are $\Theta(k)$ such that the smallest NBW whose language is* CONTROLLER-EXEC$_{\mathcal{G}}(C, U)$ *has size at least $2^{\Omega(k)}$.*

On the other hand, the cutoff $|U| + 2$ yields an NBW of size $|C| \times |U|^{\Omega(|U|)}$, and since this cutoff is tight, this technique can not yield a smaller NBW. In the following theorem we prove, surprisingly, that there is a smaller NBW, of size $O(|C| \times 2^{|U|})$.

Theorem 11. *Let $\mathcal{G}$ be the 2-ary parameterized controlled clique topology. For every disjunctively-guarded system template (C, U) there is an NBW of size $O(|C| \times 2^{|U|})$ recognizing the set* CONTROLLER-EXEC$_{\mathcal{G}}(C, U)$. *The same is true for* USER-EXEC$_{\mathcal{G}}(C, U)$.

Intuitively, each state in the NBW pairs the current *controller state* together with a set of *reachable user states*, i.e. sets of states of U that can be reached in some system of finite size, given the actual state of the controller C. In this construction, a state $s \in S_U$ is considered reachable iff it is the target of a sequence of transitions in R_U that (a) are not guarded, or (b) are guarded by other reachable states, or (c) are guarded by the current controller state. The NBW has $O(|C| \times 2^{|U|})$ (abstract) configurations, and it is shown that every path in the NBW can be concretized in some system of some finite size.

Disjunctive Guards: Complexity of PMCP. We inherit the PSPACE-hardness of model-checking $\mathsf{LTL}\backslash\mathsf{X}$ on a single finite-state system. For the upper bound, the construction in Theorem 11 can be done 'on-the-fly'

Theorem 12. *Let $\mathcal{G}$ be the 2-ary parameterized controlled clique topology or the 1-ary parameterized clique topology. Let $\mathcal{F}$ be the set of 1-index $\mathsf{LTL}\backslash\mathsf{X}$ formulas, and let $\mathcal{P}$ be the set of disjunctively guarded system templates (of suitable arity). The complexity of $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}, \mathcal{F})$ is PSPACE-complete.*

We inherit the PTIME program complexity (without controller) from Theorem 8. With a controller, the coNP upper bound results from a fine analysis of Theorem 11, and the coNP-hardness by coding of unsatisfiability (the user processes store an assignment, and the controller verifies it is not satisfying).

Theorem 13. *Fix $\mathcal{F}$ to be the set of 1-index $\mathsf{LTL}\backslash\mathsf{X}$ formulas. If $\mathcal{P}$ is the set of 1-ary disjunctively guarded system templates, and $\mathcal{G}$ is the 1-ary parameterized clique topology, then the program complexity of $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}, \mathcal{F})$ is PTIME.*

If $\mathcal{P}$ is the set of 2-ary disjunctively guarded system templates, and $\mathcal{G}$ is the 2-ary parameterized controlled clique topology, then the program complexity of $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}, \mathcal{F})$ is coNP-complete.

5 Results for Token Passing Systems

Theorem 14. *Let $\mathcal{G}$ be a parameterized topology that is either iteratively-constructible, or MSO-definable and of bounded clique-width. Then (i) The problem*

$\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}, \textit{i-CTL}^*\backslash X)$ is decidable; (ii) There is an algorithm that given k and d outputs a cutoff for k-$\textit{CTL}^*_d\backslash X$.

Decidability. We use the *finiteness* and *reduction* properties of k-$\mathsf{CTL}^*_d\backslash\mathsf{X}$ from [6]. The reduction property essentially says that the process templates in $\overline{P}$ play no role, i.e. we can assume the processes in $\overline{P}_{topo}$ do nothing except send and receive the token. The only atoms are p_j which indicate that j has the token. In a k-$\mathsf{CTL}^*_d\backslash\mathsf{X}$ formula $Q_1x_1 \ldots Q_kx_k.\ \varphi$, every valuation of the variables $x_1, \ldots, x_k$ designates k vertices of the underlying topology G, say $\bar{g} = g_1, \ldots, g_k$. The formula φ can only use the atoms p_{g_j} for $g_j \in \bar{g}$. We denote the structures of φ by $G|\bar{g}$ to indicate (1) that the process templates are $\overline{P}_{topo}$ and (2) that $\bar{g}$ have been assigned to $x_1, \ldots, x_k$ by quantification. The finiteness property says that there is a computable finite set $CON_{d,k}$ such that every $G|\bar{g}$ is $\equiv_{\text{k-}\mathsf{CTL}^*_d\backslash\mathsf{X}}$-equivalent to a member of $CON_{d,k}$. We use the details of the construction of $CON_{d,k}$ to show essentially that $\equiv_{\text{k-}\mathsf{CTL}^*_d\backslash\mathsf{X}}$ is MSO-definable by reducing the quantification on infinite paths in k-$\mathsf{CTL}^*_d\backslash\mathsf{X}$ to MSO quantification on finite simple paths and cycles. Decidability of PMCP is achieved using the decidability of MSO on classes of parameterized topologies of bounded clique-width (Theorem 1). The decidability of PMCP on iteratively constructible parameterized topologies can be shown by employing methods of similar to [9].

Cutoffs. Cuttoffs are derived as the maximal size of a representative topology belonging to a $\equiv^{\mathsf{MSO}}_q$-equivalence class as guaranteed in Theorem 14 and are non-elementary due to the number of equivalence classes. For iterateively-constructible parameterized topologies the cutoffs may be much lower, though there exists a system template $\overline{P}$, and, for all $k \in \mathbb{N}$, an iteratively constructible parameterized topology $\mathcal{G}_k$ of clique-width at most k and a k-indexed $\mathsf{LTL}\backslash\mathsf{X}$ formula φ such that the cutoff of $\mathsf{PMCP}_{\mathcal{G}}(\{\overline{P}\}, \{\varphi\})$ is $2^{\Omega(\sqrt{k})}$.

6 Discussion and related work

The applicability of the reduction of the PMCP to finitely many classical model checking problems as a technique for solving the PMCP depends on the communication primitive, the specification language, and the set of topologies of the system. The wide-ranging nature of our work along these axes gives us some insights which may be pertinent to system models different from our own:

Decidability but no cutoffs. Theorems 3 and 6 show that it can be the case that, for certain sets of specifications formula, cutoffs do not exist yet the PMCP problem is decidable.

Cutoffs may not be optimal. Proposition 1 and Theorem 11 imply that even in cases that cutoffs exist and are computable, they may not yield optimal algorithms for solving the PMCP.

Formalisms for topologies are useful. Many results in Sections 3 and 5 show that decidability and complexity of PMCP can be extended from concrete examples of sets of topologies such as rings and cliques to infinite classes of

topologies given as user-friendly yet powerful formalisms. The formalisms we study may be useful for other system models.

In the context of cutoffs, it is worth noting that we only considered cutoffs with respect to sets of formulas and process templates. As Theorem 6 shows, there is a parameterized topology $\mathcal{G}$, and a system template $\overline{P}$, for which no cutoff exists for the set of 1-indexed $\mathsf{LTL}\backslash\mathsf{X}$ formulas. Note, however, that if the formula φ is also fixed then a cutoff always exists. Indeed, given $\mathcal{G}, \overline{P}, \varphi$, letting $c := |V_G|$ yields a (minimal) cutoff if we choose G to be the smallest for which $\overline{P}^G \not\models \varphi$, or simply the smallest topology in $\mathcal{G}$ if all topologies in $\mathcal{G}$ satisfy φ. We reserve the question of computing the cutoff in such cases to future work.

As previously discussed, this work draws on and generalises the work in [3] on pairwise rendezvous on cliques, the work in [4] on disjunctive guards on cliques, and the work in [6,5,2] on token-passing systems. There are very few published complexity lower-bounds for PMCP (notable exceptions are [10,13]), and to the best of our knowledge, our lower bounds on the sizes of cutoffs are the first proven non-trivial lower bounds for these types of systems.

References

1. Suzuki, I.: Proving properties of a ring of finite-state machines. Inf. Process. Lett. **28** (1988) 213–214
2. Emerson, E.A., Namjoshi, K.S.: On reasoning about rings. Int. J. Found. Comput. Sci. **14** (2003) 527–550
3. German, S.M., Sistla, A.P.: Reasoning about systems with many processes. J. ACM **39** (1992) 675–735
4. Emerson, E.A., Kahlon, V.: Reducing model checking of the many to the few. In: CADE. Springer (2000) 236–254
5. Clarke, E., Talupur, M., Touili, T., Veith, H.: Verification by network decomposition. In: CONCUR 2004. (2004) 276–291
6. Aminof, B., Jacobs, S., Khalimov, A., Rubin, S.: Parameterized model checking of token-passing systems. In: VMCAI, Springer (2014) 262–281
7. Browne, M.C., Clarke, E.M., Grumberg, O.: Reasoning about networks with many identical finite state processes. Inf. Comput. **81** (1989) 13–31
8. John, A., Konnov, I., Schmid, U., Veith, H., Widder, J.: Counter attack on byzantine generals: Parameterized model checking of fault-tolerant distributed algorithms. CoRR **abs/1210.3846** (2012)
9. Fischer, E., Makowsky, J.A.: Linear recurrence relations for graph polynomials. In: Pillars of Computer Science, Springer (2008) 266–279
10. Esparza, J.: Keeping a crowd safe: On the complexity of parameterized verification. In: STACS. (2014)
11. Courcelle, B., Engelfriet, J.: Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach. Volume 138 of Encyclopedia of mathematics and its applications. Cambridge University Press (2012)
12. Vardi, M., Wolper, P.: Automata-theoretic techniques for modal logics of programs. Journal of Computer and System Sciences **32** (1986) 182–221
13. Schmitz, S., Schnoebelen, P.: The power of well-structured systems. In: CONCUR, Springer (2013) 5–24