Parameterized Model Checking of Token-Passing Systems

Benjamin Aminof[1], Swen Jacobs[2], Ayrat Khalimov[2], Sasha Rubin[1,3] ⋆

[1]IST Austria (first.last@ist.ac.at), [2]TU Graz (first.last@iaik.tugraz.at), [3]TU Wien

Abstract. We revisit the parameterized model checking problem for token-passing systems and specifications in indexed $\mathsf{CTL}^*\backslash\mathsf{X}$. Emerson and Namjoshi (1995, 2003) have shown that parameterized model checking of indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ in uni-directional token rings can be reduced to checking rings up to some *cutoff* size. Clarke et al. (2004) have shown a similar result for general topologies and indexed $\mathsf{LTL}\backslash\mathsf{X}$, provided processes cannot choose the directions for sending or receiving the token.

We unify and substantially extend these results by systematically exploring fragments of indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ with respect to general topologies. For each fragment we establish whether a cutoff exists, and for some concrete topologies, such as rings, cliques and stars, we infer small cutoffs. Finally, we show that the problem becomes undecidable, and thus no cutoffs exist, if processes are allowed to choose the directions in which they send or from which they receive the token.

1 Introduction

As executions of programs and protocols are increasingly distributed over multiple CPU cores or even physically separated computers, correctness of concurrent systems is one of the primary challenges of formal methods today. Many concurrent systems consist of an arbitrary number of identical processes running in parallel. 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.

The PMCP is undecidable in many cases. For example, it is undecidable already for safety specifications and finite-state processes communicating by passing a binary-valued token around a uni-directional ring [14,7]. However, decidability may be regained by restricting the communication primitives, the topologies under consideration (i.e., the underlying graph describing the comunication paths between the processes), or the specification language. In particular, previous results have shown that parameterized model checking can sometimes be reduced to model checking a finite number of instances of the system, up to some *cutoff* size.

⋆ This work was supported by the Austrian Science Fund through grant P23499-N23 and through the RiSE network (S11403, S11405, S11406, S11407-N23); ERC Starting Grant (279307: Graph Games); Vienna Science and Technology Fund (WWTF) grants PROSEED, ICT12-059, and VRG11-005.

For token-passing systems (TPSs) with uni-directional ring topologies, such cutoffs are known for specifications in the prenex fragment of indexed CTL^* without the next-time operator ($\mathsf{CTL}^*\backslash\mathsf{X}$) [9,7]. For token-passing in general topologies, cutoffs are known for the prenex fragment of indexed $\mathsf{LTL}\backslash\mathsf{X}$, provided that processes are not allowed to choose the direction to which the token is sent or from which it is received [4]. In this paper we generalize these results and elucidate what they have in common.

Previous Results. In their seminal paper [7], Emerson and Namjoshi consider systems where the token does not carry messages, and specifications are in prenex indexed temporal logic — i.e., quantifiers $\forall$ and $\exists$ over processes appear in a block at the front of the formula. They use the important concept of a *cutoff* — a number c such that the PMCP for a given class of systems and specifications can be reduced to model checking systems with up to c processes. Clearly, if there is a cutoff for a given class and processes are finite state, then the PMCP for this class is decidable. Conversely, if the PMCP is undecidable, then there can be no cutoff for such systems.

For uni-directional rings, Emerson and Namjoshi provide cutoffs for formulas with a small number k of quantified index variables, and state that their proof method allows one to obtain cutoffs for other quantifier prefixes. In brief, cutoffs exist for the branching-time specification language prenex-indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ and the highly regular topology of uni-directional rings.

Clarke et al. [4] consider the PMCP for token-passing systems arranged in general topologies. Their main result is that the PMCP for systems with arbitrary topologies and k-indexed $\mathsf{LTL}\backslash\mathsf{X}$ specifications (i.e., specifications with k quantifiers over processes) can be reduced to combining the results of model-checking finitely many topologies of size at most $2k$ [4, Theorem 4]. Their proof implies that, for each k, the PMCP for linear-time specifications in k-indexed $\mathsf{LTL}\backslash\mathsf{X}$ and general topologies has a cutoff.

Questions. Comparing these results, an obvious question is: are there cutoffs for branching time temporal logics and arbitrary topologies (see Table 1)? Clarke et al. already give a first answer [4, Corollary 3]. They prove that there is no cutoff for token-passing systems with arbitrary topologies and specifications from 2-indexed $\mathsf{CTL}\backslash\mathsf{X}$. However, their proof makes use of formulas with unbounded nesting-depth of path quantifiers. This lead us to the first question.

	Uni-Ring Topologies	Arbitrary Topologies
indexed $\mathsf{LTL}\backslash\mathsf{X}$	–	[4]
indexed $\mathsf{CTL}^*\backslash\mathsf{X}$	[7]	this paper

Table 1: Direction-Unaware TPSs.

	Bi-Ring Topologies	Arbitrary Topologies
indexed $\mathsf{LTL}\backslash\mathsf{X}$	[6]	this paper
indexed $\mathsf{CTL}^*\backslash\mathsf{X}$	this paper	this paper

Table 2: Direction-Aware TPSs.

Question 1. Is there a way to stratify k-indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ such that for each level of the stratification there is a cutoff for systems with arbitrary topologies? In particular, does stratification by nesting-depth of path quantifiers do the trick?

Cutoffs for k-indexed temporal logic fragments immediately yield that for each k there is an algorithm (depending on k) for deciding the PMCP for k-indexed temporal logic. However, this does not imply that there is an algorithm that can compute the cutoff for a given k. In particular, it does not imply that PMCP for full prenex indexed temporal logic is decidable.

Question 2. For which topologies (rings, cliques, all?) can one conclude that the PMCP for the full prenex-indexed temporal logic is decidable?

Finally, an important implicit assumption in Clarke et al. [4] is that processes are not *direction aware*, i.e., they cannot sense or choose in which direction the token is sent, or from which direction it is received. In contrast, Emerson and Kahlon [6] show that cutoffs exist for certain direction-aware systems in bi-directional rings (see Section 8). We were thus motivated to understand to what extent existing results about cutoffs can be lifted to direction-aware systems, see Table 2.

Question 3. Do cutoffs exist for direction-aware systems on arbitrary topologies and k-indexed temporal logics (such as $\mathsf{LTL}\backslash\mathsf{X}$ and $\mathsf{CTL}\backslash\mathsf{X}$)?

Our contributions. In this paper, we answer the questions above, unifying and substantially extending the known cutoff results:

Answer to Question 1. Our main positive result (Theorem 7) states that for arbitrary parameterized topologies $\mathbf{G}$ there is a cutoff for specifications in k-indexed $\mathsf{CTL}^*_d\backslash\mathsf{X}$ — the cutoff depends on $\mathbf{G}$, the number k of the process quantifiers, and the nesting depth d of path quantifiers. In particular, indexed $\mathsf{LTL}\backslash\mathsf{X}$ is included in the case $d = 1$, and so our result generalizes the results of Clarke et al. [4].

Answer to Question 2. We prove (Theorem 14) that there exist topologies for which the PMCP is undecidable for specifications in prenex-indexed $\mathsf{CTL}\backslash\mathsf{X}$ or $\mathsf{LTL}\backslash\mathsf{X}$. Note that this undecidability result does not contradict the existence of cutoffs (Theorem 7), since cutoffs may not be computable from k, d (see the note on decidability in Section 2.4). However, for certain topologies our positive result is constructive and we can compute cutoffs given k and d (Theorem 15). To illustrate, we show that rings have a cutoff of $2k$, cliques of $k+1$, and stars of $k+1$ (independent of d). In particular, PMCP is decidable for these topologies and specifications in prenex-indexed $\mathsf{CTL}^*\backslash\mathsf{X}$.

Answer to Question 3. The results just mentioned assume that processes are not direction-aware. Our main negative result (Theorem 18) states that if processes can control at least one of the directions (i.e., choose in which direction to send or from which direction to receive) then the PMCP for arbitrary topologies and k-indexed logic (even $\mathsf{LTL}\backslash\mathsf{X}$ and $\mathsf{CTL}\backslash\mathsf{X}$) is undecidable, and therefore does not have cutoffs. Moreover, if processes can control both in- and out-directions, then the PMCP is already undecidable for bi-directional rings and 1-indexed $\mathsf{LTL}\backslash\mathsf{X}$.

Technical contributions relative to previous work. Our main positive result (Theorem 7) generalizes proof techniques and ideas from previous results [7,4]. We observe that in both of these papers the main idea is to abstract a TPS by simulating the quantified processes exactly and simulating the movement of the token between these processes. The relevant information about the movement of the token is this: whether there is a direct edge, or a path (through unquantified processes) from one quantified process to another. This abstraction does not work for $\mathsf{CTL}^*_d\backslash\mathsf{X}$ and general topologies since the formula can express branching properties of the token movement. Our main observation is that the relevant information about the branching-possibilities of the token can be expressed in $\mathsf{CTL}^*_d\backslash\mathsf{X}$ over the topology itself. We develop a composition-like theorem, stating that if two topologies (with k distinguished vertices) are indistinguishable by $\mathsf{CTL}^*_d\backslash\mathsf{X}$ formulas, then the TPSs based on these topologies and an arbitrary process template P are indistinguishable by $\mathsf{CTL}^*_d\backslash\mathsf{X}$ (Theorem 9). The machinery involves a generalization of stuttering trace-equivalence [13], a notion of d-contraction that serves the same purpose as the connection topologies of Clarke et al. [4, Proposition 1], and also the main simulation idea of Emerson and Namjoshi [7, Theorem 2].

Our main negative result, undecidability of PMCP for direction-aware systems (Theorem 18), is proven by establishing a reduction from the non-halting problem for 2-counter machines (as is typical in this area [7,10]). Due to the lack of space, full proofs are omitted, and can be found in the full version [1].

2 Definitions and Existing Results

Let $\mathbb{N}$ denote the set of positive integers. Let $[k]$ for $k \in \mathbb{N}$ denote the set $\{1, \ldots, k\}$. The concatenation of strings u and w is written uw or $u \cdot w$.

Let AP denote a countably infinite set of *atomic propositions* or *atoms*. A *labeled transition system (LTS) over* AP is a tuple $(Q, Q_0, \Sigma, \delta, \lambda)$ where Q is the set of *states*, $Q_0 \subseteq Q$ are the *initial states*, Σ is the set of *transition labels* (also called *action labels*), $\delta \subseteq Q \times \Sigma \times Q$ is the *transition relation*, and $\lambda : Q \to 2^{\mathsf{AP}}$ is the *state-labeling* and satisfies that $\lambda(q)$ is finite (for every $q \in Q$). Transitions $(q, \sigma, q') \in \delta$ may be written $q \xrightarrow{\sigma} q'$.

A *state-action path* of an LTS $(Q, Q_0, \Sigma, \delta, \lambda)$ is a finite sequence $q_0\sigma_0q_1\sigma_1 \ldots q_n \in (Q\Sigma)^*Q$ or an infinite sequence $q_0\sigma_0q_1\sigma_1 \cdots \in (Q\Sigma)^\omega$ such that $(q_i, \sigma_i, q_{i+1}) \in \delta$ (for all i). A *path* of an LTS is the projection $q_0q_1 \ldots$ of a state-action path onto states Q. An *action-labeled path* of an LTS is the projection $\sigma_0\sigma_1 \ldots$ of a state-action path onto transition labels Σ.

2.1 System Model (Direction-Unaware)

In this section we define the LTS P^G — it consists of replicated copies of a process P placed on the vertices of a graph G. Transitions in P^G are either internal (in which exactly one process moves) or synchronized (in which one

process sends the token to another along an edge of G). The token starts with the process that is at the initial vertex of G.

Fix a countably infinite set of (local) atomic propositions $\mathsf{AP}_{\mathsf{pr}}$ (to be used by the states of the individual processes).

Process Template P. Let Σ_{int} denote a finite non-empty set of *internal-transition labels*. Define Σ_{pr} as the disjoint union $\Sigma_{\mathsf{int}} \cup \{\mathsf{rcv}, \mathsf{snd}\}$ where rcv and snd are new symbols.

A *process template* P is a LTS $(Q, Q_0, \Sigma_{\mathsf{pr}}, \delta, \lambda)$ over $\mathsf{AP}_{\mathsf{pr}}$ such that:

i) the state set Q is finite and can be partitioned into two non-empty sets, say $T \cup N$. States in T are said to *have the token*.
ii) The initial state set is $Q_0 = \{\iota_t, \iota_n\}$ for some $\iota_t \in T, \iota_n \in N$.
iii) Every transition $q \xrightarrow{\mathsf{snd}} q'$ satisfies that q has the token and q' does not.
iv) Every transition $q \xrightarrow{\mathsf{rcv}} q'$ satisfies that q' has the token and q does not.
v) Every transition $q \xrightarrow{\mathsf{a}} q'$ with $\mathsf{a} \in \Sigma_{\mathsf{int}}$ satisfies that q has the token if and only if q' has the token.
vi) The transition relation δ is total in the first coordinate: for every $q \in Q$ there exists $\sigma \in \Sigma_{\mathsf{pr}}, q' \in Q$ such that $(q, \sigma, q') \in \delta$ (i.e., the process P is non-terminating).
vii) Every infinite action-labeled path $a_0 a_1 \ldots$ is in the set $(\Sigma_{\mathsf{int}}^* \ \mathsf{snd} \ \Sigma_{\mathsf{int}}^* \ \mathsf{rcv})^\omega \cup (\Sigma_{\mathsf{int}}^* \ \mathsf{rcv} \ \Sigma_{\mathsf{int}}^* \ \mathsf{snd})^\omega$ (i.e., snd and rcv actions alternate continually along every infinite action-labeled path of P). [1]

The elements of Q are called *local states* and the transitions in δ are called *local transitions (of P)*. A local state q such that the only transitions are of the form $q \xrightarrow{\mathsf{snd}} q'$ (for some q') is said to be *send-only*. A local state q such that the only transitions are of the form $q \xrightarrow{\mathsf{rcv}} q'$ (for some q') is said to be *receive-only*.

Topology G. A *topology*[2] is a directed graph $G = (V, E, x)$ where $V = [k]$ for some $k \in \mathbb{N}$, vertex $x \in V$ is the *initial vertex*, $E \subseteq V \times V$, and $(v, v) \notin E$ for every $v \in V$. Vertices are called *process indices*.

We may also write $G = (V_G, E_G, x_G)$ if we need to disambiguate.

Token-Passing System P^G. Let $\mathsf{AP}_{\mathsf{sys}} := \mathsf{AP}_{\mathsf{pr}} \times \mathbb{N}$ be the *indexed atomic propositions*. For $(p, i) \in \mathsf{AP}_{\mathsf{sys}}$ we may also write p_i. Given a process template $P = (Q, Q_0, \Sigma_{\mathsf{pr}}, \delta, \lambda)$ over $\mathsf{AP}_{\mathsf{pr}}$ and a topology $G = (V, E, x)$, define the *token-passing system (TPS) P^G* as the finite LTS $(S, S_0, \Sigma_{\mathsf{int}} \cup \{\mathsf{tok}\}, \Delta, \Lambda)$ over atomic propositions $\mathsf{AP}_{\mathsf{sys}} := \mathsf{AP}_{\mathsf{pr}} \times \mathbb{N}$, where:

- The set S of *global states* is Q^V, i.e., all functions from V to Q. If $s \in Q^V$ is a global state then $s(i)$ denotes the local state of the process with index i.[3]
- The set of *global initial states* S_0 consists of the unique global state $s_0 \in Q_0^V$ such that only $s_0(x)$ has the token.

[1] This restriction was introduced by Emerson and Namjoshi in [7]. Our positive results that cutoffs exist also hold for a more liberal restriction (see Section 7).

[2] There does not seem to be standard terminology for this object. The paper [4] uses 'network graph' and reserves 'topology' for a set of network graphs.

[3] In [7,4] a global state is defined to be a vector of local states, which is a notational variation of the functional notation we prefer.

- The labeling $\Lambda(s) \subset \mathsf{AP_{sys}}$ for $s \in S$ is defined as follows: $p_i \in \Lambda(s)$ if and only if $p \in \lambda(s(i))$, for $p \in \mathsf{AP_{pr}}$ and $i \in V$.
- The *global transition relation* Δ is defined to consist of the set of all internal transitions and synchronous transitions:
  - An *internal transition* is an element (s, a, s') of $S \times \Sigma_{\mathsf{int}} \times S$ for which there exists a process index $v \in V$ such that
    i) $s(v) \xrightarrow{\mathsf{a}} s'(v)$ is a local transition of P, and
    ii) for all $w \in V \setminus \{v\}$, $s(w) = s'(w)$.
  - A *token-passing transition* is an element (s, tok, s') of $S \times \{\mathsf{tok}\} \times S$ for which there exist process indices $v, w \in V$ such that $(v, w) \in E$ and
    i) $s(v) \xrightarrow{\mathsf{snd}} s'(v)$ is a local transition of P,
    ii) $s(w) \xrightarrow{\mathsf{rcv}} s'(w)$ is a local transition of P, and
    iii) for every $u \in V \setminus \{v, w\}$, $s'(u) = s(u)$.

In words, the system P^G can be thought of the asynchronous parallel composition of P over topology G. The token starts with process x. 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.

2.2 System Model (Direction-Aware)

Inspired by direction-awareness in the work of Emerson and Kahlon [6], we extend the definition of TPS to include additional labels on edges, called *directions*. The idea is that processes can restrict which directions are used when they send or receive the token. Fix finite non-empty disjoint sets $\mathsf{Dir_{snd}}$ of *sending directions* and $\mathsf{Dir_{rcv}}$ of *receiving directions*. A *direction-aware token-passing system* is a TPS with the following modifications.

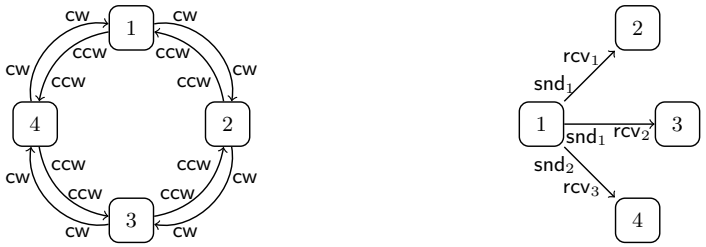
Direction-aware Topology. A *direction-aware topology* is a topology $G = (V, E, x)$ with labeling functions $\mathsf{dir_{rcv}} : E \to \mathsf{Dir_{rcv}}$, $\mathsf{dir_{snd}} : E \to \mathsf{Dir_{snd}}$.

Direction-aware Process Template. In direction-aware systems, process templates use transition labels from $\Sigma_{\mathsf{pr}} := \Sigma_{\mathsf{int}} \cup \mathsf{Dir_{snd}} \cup \mathsf{Dir_{rcv}}$. The definition of a *direction-aware process template* is like that in Section 2.1, except that in item iii) snd is replaced by $\mathsf{d} \in \mathsf{Dir_{snd}}$, in iv) rcv is replaced by $\mathsf{d} \in \mathsf{Dir_{rcv}}$, and in vii) snd is replaced by $\mathsf{Dir_{snd}}$ and rcv by $\mathsf{Dir_{rcv}}$.

Direction-aware Token Passing System. Fix $\mathsf{Dir_{snd}}$ and $\mathsf{Dir_{rcv}}$, let G be a direction-aware topology and P a direction-aware process template. Define the *direction-aware token-passing system* P^G as in Section 2.1, except that token-passing transitions are now direction-aware:

Direction-aware token-passing transitions are elements (s, tok, s') of $S \times \{\mathsf{tok}\} \times S$ for which there exist process indices $v, w \in V$ with $(v, w) \in E$, $\mathsf{dir_{snd}}(v, w) = \mathsf{d}$, and $\mathsf{dir_{rcv}}(v, w) = \mathsf{e}$, such that:

i) $s(v) \xrightarrow{\mathsf{d}} s'(v)$ is a local transition of P.
ii) $s(w) \xrightarrow{\mathsf{e}} s'(w)$ is a local transition of P.
iii) For every $u \in V \setminus \{v, w\}$, $s'(u) = s(u)$.



(a) Bi-directional ring with directions.

(b) Directed topology with multiple edges of same direction.

Fig. 1: Direction-aware Topologies.

Notations $\mathcal{P}_{\mathsf{u}}$, $\mathcal{P}_{\mathsf{snd}}$, $\mathcal{P}_{\mathsf{rcv}}$, $\mathcal{P}_{\mathsf{sndrcv}}$. Let $\mathcal{P}_u$ denote the set of all process templates for which $|\mathsf{Dir}_{\mathsf{snd}}| = |\mathsf{Dir}_{\mathsf{rcv}}| = 1$. In this case P^G degenerates to a direction-unaware TPS as defined in Section 2.1. If we require $|\mathsf{Dir}_{\mathsf{rcv}}| = 1$, then processes cannot choose from which directions to receive the token, but possibly in which direction to send it. Denote the set of all such process templates by $\mathcal{P}_{snd}$. Similarly define $\mathcal{P}_{rcv}$ to be all process templates where $|\mathsf{Dir}_{\mathsf{snd}}| = 1$ — processes cannot choose where to send the token, but possibly from which direction to receive it. Finally, let $\mathcal{P}_{sndrcv}$ be the set of all direction-aware process templates.

Examples. Figure 1(a) shows a bi-directional ring with directions cw (clockwise) and ccw (counterclockwise). Every edge e is labeled with an outgoing direction $dir_{\mathsf{snd}}(e)$ and an incoming direction $dir_{\mathsf{rcv}}(e)$.[4] Using these directions, a process that has the token can choose whether he wants to send it in direction cw or ccw. Depending on its local state, a process waiting for the token can also choose to receive it only from direction cw or ccw.

Figure 1(b) depicts a topology in which process 1 can choose between two outgoing directions. If it sends the token in direction snd_1, it may be received by either process 2 or 3. If however process 2 blocks receiving from direction rcv_1, the token can only be received by process 3. If 3 additionally blocks receiving from rcv_2, then this token-passing transition is disabled.

2.3 Indexed Temporal Logics

Indexed temporal logics (ITL) were introduced in [3,8,7] to model specifications of certain distributed systems. Subsequently one finds a number of variations of indexed temporal-logics in the literature (linear vs. branching, restrictions on the quantification). Thus we introduce Indexed-CTL^* which has these variations (and those in [4]) as syntactic fragments.

Syntactic Fragments of CTL^*, and $\equiv_{\mathsf{TL}}$. We assume the reader is familiar with the syntax and semantics of CTL^*, for a reminder see [2]. For $d \in \mathbb{N}$ let $\mathsf{CTL}^*_d\backslash\mathsf{X}$ denote the syntactic fragment of $\mathsf{CTL}^*\backslash\mathsf{X}$ in which the nesting-depth of path quantifiers is at most d (for a formal definition see [13, Section 4]).

[4] For notational simplicity, we denote both outgoing direction $\mathsf{snd}_{\mathsf{cw}}$ and incoming direction $\mathsf{rcv}_{\mathsf{cw}}$ by cw, and similarly for ccw.

Let TL denote a temporal logic (in this paper these are fragments of $\mathsf{CTL}^*\backslash\mathsf{X}$). For temporal logic TL and LTSs M and N, write $M \equiv_{\mathsf{TL}} N$ to mean that for every formula $\phi \in \mathsf{TL}$, $M \models \phi$ if and only if $N \models \phi$.

Indexed-CTL^*. Fix an infinite set $\mathsf{Vars} = \{x, y, z, \dots\}$ of index variables, i.e., variables with values from $\mathbb{N}$. These variables refer to vertices in the topology.

Syntax. The *Indexed-CTL^* formulas over variable set Vars and atomic propositions AP* are formed by adding the following rules to the syntax of CTL^* over atomic propositions $\mathsf{AP} \times \mathsf{Vars}$. We write p_x instead of $(p, x) \in \mathsf{AP} \times \mathsf{Vars}$.

If ϕ is an indexed-CTL^* state (resp. path) formula and $x, y \in \mathsf{Vars}, Y \subset \mathsf{Vars}$ then the following are also indexed-CTL^* state (resp. path) formulas:

- $\forall x.\phi$ (meaning that for all vertices in the topology ϕ should hold),
- $\forall x.x \notin Y \rightarrow \phi$ (meaning that ϕ holds for all vertices that are different to those that are designated by variables in Y),
- $\forall x.x \in Y \rightarrow \phi$,
- $\forall x.x \in E(y) \rightarrow \phi$ (meaning that ϕ holds for all vertices to which there is an edge from the vertex designated by the variable y), and
- $\forall x.x \notin E(y) \rightarrow \phi$.

Shorthand. As usual, write $\exists x.\phi$ as shorthand for $\neg\forall x.\neg\phi$, write $\forall x \notin Y.\phi$ as shorthand for $\forall x.x \notin Y \rightarrow \phi$, and $\forall x \in E(y).\phi$ as shorthand for $\forall x.x \in E(y) \rightarrow \phi$. All these prefixes $\exists x, \forall x, \exists x \in E(y), \forall x \in E(y), \exists x \in Y, \cdots$ are called *index quantifiers*. We use Qx to denote an index quantifier.

Semantics. Indexed temporal logic is interepreted over a system instance P^G (with P a process template and G a topology). The formal *semantics* are in the full version of the paper [1]. Here we give some examples. The formula $\forall i.\,\mathsf{E}\,\mathsf{F}\,p_i$ states that for every process there exists a path such that eventually that process is in a state that satisfies atom p. The formula $\mathsf{E}\,\mathsf{F}\,\forall i.p_i$ states that there is a path such that eventually all processes satisfy atom p simultaneously. We now define the central fragment of ITL which includes the former example and not the latter.

Prenex indexed TL and $\{\forall, \exists\}^k$-TL. *Prenex indexed temporal-logic* is a syntactic fragment of indexed temporal-logic in which all quantifiers are at the front of the formula, e.g., prenex-indexed $\mathsf{LTL}\backslash\mathsf{X}$ consists of formulas of the form $(Q_1x_1)\dots(Q_kx_k)\ \varphi$ where φ is an $\mathsf{LTL}\backslash\mathsf{X}$ formula over atoms $\mathsf{AP} \times \{x_1, \dots, x_k\}$, and the Q_ix_is are index quantifers. Such formulas with k quantifiers will be refered to as *k-indexed*, collectively written $\{\forall, \exists\}^k$-TL. The union of $\{\forall, \exists\}^k$-TL for $k \in \mathbb{N}$ is written $\{\forall, \exists\}^*$-TL and called (full) prenex indexed TL.

2.4 Parameterized Model Checking Problem, Cutoffs, Decidability

A *parameterized topology* $\mathbf{G}$ is a countable set of topologies. E.g., the set of unidirectional rings with all possible initial vertices is a parameterized topology.

$\mathsf{PMCP}_{\mathbf{G}}(-,-)$. The *parameterized model checking problem (PMCP)* for parameterized topology $\mathbf{G}$, processes from $\mathcal{P}$, and parameterized specifications from

$\mathcal{F}$, written $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \mathcal{F})$, is the set of pairs $(\varphi, P) \in \mathcal{F} \times \mathcal{P}$ such that for all $G \in \mathbf{G}$, $P^G \models \varphi$.

A solution to $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \mathcal{F})$ is an algorithm that, given a formula $\varphi \in \mathcal{F}$ and a process template $P \in \mathcal{P}$ as input, outputs 'Yes' if for all $G \in \mathbf{G}$, $P^G \models \varphi$, and 'No' otherwise.

Cutoff. A *cutoff* for $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \mathcal{F})$ is a natural number c such that for every $P \in \mathcal{P}$ and $\varphi \in \mathcal{F}$, the following are equivalent:[5]
- $P^G \models \varphi$ for all $G \in \mathbf{G}$ with $|V_G| \leq c$;
- $P^G \models \varphi$ for all $G \in \mathbf{G}$.

Thus $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \mathcal{F})$ *does not have a cutoff* iff for every $c \in \mathbb{N}$ there exists $P \in \mathcal{P}$ and $\varphi \in \mathcal{F}$ such that $P^G \models \varphi$ for all $G \in \mathbf{G}$ with $|V_G| \leq c$, and there exists $G \in \mathbf{G}$ such that $P^G \not\models \varphi$.

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

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

Note About Decidability. The following statements are not, a priori, equivalent (for given parameterized topology $\mathbf{G}$ and process templates $\mathcal{P}$):
- For every $k \in \mathbb{N}$, $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \{\forall, \exists\}^k\text{-}\mathsf{TL})$ is decidable.
- $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \{\forall, \exists\}^*\text{-}\mathsf{TL})$ is decidable.

The first item says that for every k there *exists* an algorithm A_k that solves the PMCP for k-indexed TL. This does not imply the second item, which says that there exists a single algorithm that solves the PMCP for $\cup_{k \in \mathbb{N}} \{\forall, \exists\}^k\text{-}\mathsf{TL}$. If the function $k \mapsto A_k$ is also *computable* (e.g., Theorem 15) then indeed the second item follows: given P, φ, extract the size k of the prenex block of φ, *compute* a description of A_k, and run A_k on P, φ.

For instance, the result of Clarke et al. (that there are cutoffs for k-index $\mathsf{LTL}\backslash\mathsf{X}$ and arbitrary topologies) does not imply that the PMCP for $\{\forall, \exists\}^*\text{-}\mathsf{LTL}\backslash\mathsf{X}$ and arbitrary topologies is decidable. Aware of this fact, the authors state (after Theorem 4) "Note that the theorem does not provide us with an effective means to find the reduction [i.e. algorithm]...".

In fact, we prove (Theorem 14) that there is some parameterized topology such that PMCP is undecidable for prenex-indexed $\mathsf{LTL}\backslash\mathsf{X}$.

Existing Results. We restate the known results using our terminology.
A *uni-directional ring* $G = (V, E, x)$ is a topology with $V = [n]$ for some $n \in \mathbb{N}$, there are edges $(i, i+1)$ for $1 \leq i \leq n$ (arithmetic is modulo n), and $x \in V$. Let $\mathbf{R}$ be the parameterized topology consisting of all uni-directional rings.

Theorem 2 (Implict in [7]). *For every $k \in \mathbb{N}$, there is a cutoff for the problem $PMCP_{\mathbf{R}}(\mathcal{P}_u, \{\forall, \exists\}^k\text{-}\mathsf{CTL}^*\backslash\mathsf{X})$.* [6]

[5] Of course the first item always follows from the second item.

[6] The paper explicitly contains the result that 4 is a cutoff for $\forall\forall\text{-}\mathsf{CTL}^*\backslash\mathsf{X}$ on rings. However the proof ideas apply to get the stated theorem.

Although Clarke et al. [4] do not explicitly state the following theorem, it follows from their proof technique, which we generalise in Section 3.

Theorem 3 (Implicit in [4]). *For every parameterized topology* $\mathbf{G}$*, and every* $k \in \mathbb{N}$*, the problem* $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall, \exists\}^k\text{-}\mathsf{LTL}\backslash\mathsf{X})$ *has a cutoff. A corollary [12, Corollary 2] states that* $2k$ *is a cutoff if* $\mathbf{G}$ *is taken to be* $\mathbf{R}$ [7]*.*

Theorem 4 ([4, Corollary 3]). *There exists a parameterized topology* $\mathbf{G}$ *and process* $P \in \mathcal{P}_u$ *such that the problem* $\mathsf{PMCP}_{\mathbf{G}}(\{P\}, \{\exists\}^2\text{-}\mathsf{CTL}\backslash\mathsf{X})$ *does not have a cutoff.*

The proof of this theorem defines $\mathbf{G}$, process P, and for every $c \in \mathbb{N}$ a formula φ_c, such that if $G \in \mathbf{G}$ then $P^G \models \varphi_c$ if and only if $|V_G| \leq c$. The formula φ_c is in 2-indexed $\mathsf{CTL}\backslash\mathsf{X}$ and has nesting depth of path quantifiers equal to c.

3 Method for Proving Existence of Cutoffs

We give a method for proving cutoffs for direction-*un*aware TPSs that will be used to prove Theorem 7. The method abstracts from the ideas of Clarke et al. [4].

In a k-indexed TL formula $Q_1 x_1 \ldots Q_k x_k.\ \varphi$, every valuation of the variables $x_1, \ldots, x_k$ designates k nodes of the underlying topology G, say $\bar{g} = g_1, \ldots, g_k$. The formula φ can only talk about (the processes in) $\bar{g}$. In order to prove that the PMCP has a cutoff, it is sufficient (as the proof of Theorem 5 will demonstrate) to find conditions on two topologies G, G' and $\bar{g}, \bar{g}'$ that allow one to conclude that P^G and $P^{G'}$ are indistinguishable with repect to φ.

We define two abstractions for a given TPS P^G. The first abstraction simulates P^G, keeping track only of the local states of processes indexed by $\bar{g}$. We call it the *projection* of P^G onto $\bar{g}$.[8] The second abstraction only simulates the movement of the token in G, restricted to $\bar{g}$. We call it the *graph LTS* of G and $\bar{g}$.

Notation. Let $\bar{g}$ denote a tuple $(g_1, \ldots, g_k)$ of *distinct* elements of V_G, and $\bar{g}'$ a k-tuple of distinct elements of $V_{G'}$. Write $v \in \bar{g}$ if $v = g_i$ for some i.

The projection $P^G|\bar{g}$. Informally, the *projection* of P^G onto a tuple of process indices $\bar{g}$ is the LTS P^G and a new labeling that, for every $g_i \in \bar{g}$, replaces the indexed atom p_{g_i} by the atom $p@i$; all other atoms are removed. Thus $p@i$ means that the atom $p \in \mathsf{AP}_{\mathsf{pr}}$ holds in the process with index g_i. In other words, process indices are replaced by their *positions* in $\bar{g}$.

Formally, fix process P, topology G, and k-tuple $\bar{g}$ over V_G. Define the *projection of* $P^G = (S, S_0, \Sigma_{\mathsf{int}} \cup \{\mathsf{tok}\}, \Delta, \Lambda)$ *onto* $\bar{g}$, written $P^G|\bar{g}$ as the LTS $(S, S_0, \Sigma_{\mathsf{int}} \cup \{\mathsf{tok}\}, \Delta, L)$ over atomic propositions $\{p@i : p \in \mathsf{AP}_{\mathsf{pr}}, i \in [k]\}$, where for all $s \in S$ the labeling $L(s)$ is defined as follows: $L(s) := \{p@i : p_{g_i} \in \Lambda(s), i \in [k]\}$.

[7] There is an error in [12, Corollary 2]: $2k$ is the cutoff only for formula with no quantifier alternations. See Remark 17 in Section 5.

[8] Emerson and Namjoshi [7, Section 2.1] define the related notion "LTS projection".

The graph LTS $G|\bar{g}$. Informally, $G|\bar{g}$ is an LTS where states are nodes of the graph G, and transitions are edges of G. The restriction to $\bar{g}$ is modeled by labeling a state with the position of the corresponding node in $\bar{g}$.

Let $G = (V, E, x)$ be a topology, and let $\bar{g}$ be a k-tuple over V_G. Define the *graph LTS* $G|g$ as the LTS $(Q, Q_0, \Sigma, \Delta, \Lambda)$ over atomic propositions $\{1, \ldots, k\}$, with state set $Q := V$, initial state set $Q_0 := \{x\}$, action set $\Sigma = \{\mathsf{a}\}$, transition relation with $(v, \mathsf{a}, w) \in \Delta$ iff $(v, w) \in E$, and labeling $\Lambda(v) := \{i\}$ if $v = g_i$ for some $1 \leq i \leq k$, and $\emptyset$ otherwise.

Fix a non-indexed temporal logic TL, such as $\mathsf{CTL}^*\backslash\mathsf{X}$. We now define what it means for TL to have the reduction property and the finiteness property. Informally, the reduction property says that if G and G' have the same connectivity (with respect to TL and only viewing k-tuples $\bar{g}, \bar{g}'$) then P^G and $P^{G'}$ are indistinguishable (with respect to TL-formulas over process indices in $\bar{g}, \bar{g}'$).

REDUCTION property for TL

For every positive integer k, process $P \in \mathcal{P}_u$, topologies G, G', k-tuples $\bar{g}, \bar{g}'$, if $G|\bar{g} \equiv_{\mathsf{TL}} G'|\bar{g}'$ then $P^G|\bar{g} \equiv_{\mathsf{TL}} P^{G'}|\bar{g}'$.

FINITENESS property for TL

For every positive integer k, there are finitely many equivalence classes $[G|\bar{g}]_{\equiv_{\mathsf{TL}}}$ where G is an arbitrary topology, and $\bar{g}$ is a k-tuple over V_G.

Theorem 5 (REDUCTION & FINITENESS $\Longrightarrow$ Cutoffs exist for $\{\forall, \exists\}^k$-TL). *If TL satisfies the* REDUCTION *property and $\mathbf{G}$ satisfies the* FINITENESS *property with respect to TL then for every k, $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall, \exists\}^k\text{-}\mathsf{TL})$ has a cutoff.*

Proof. Fix quantifier prefix $Q_1 x_1 \ldots Q_k x_k$. We prove that there exist finitely many topologies $G_1, \cdots, G_N \in \mathbf{G}$ such that for every $G \in \mathbf{G}$ there is an $i \leq N$ such that for all $P \in \mathcal{P}_u$, and all TL-formulas φ over atoms $\mathsf{AP}_{\mathsf{pr}} \times \{x_1, \cdots, x_k\}$

$$P^G \models Q_1 x_1 \ldots Q_k x_k.\ \varphi \iff P^{G_i} \models Q_1 x_1 \ldots Q_k x_k.\ \varphi$$

In particular, $c := \max\{|V_{G_i}| : 1 \leq i \leq N\}$ is a cutoff for $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall, \exists\}^k\text{-}\mathsf{TL})$.

Suppose for simplicity of exposition that $Q_i x_i$ is a quantifier that also expresses that the value of x_i is different from the values of $x_j \in \{x_1, \ldots, x_{i-1}\}$.[9] Fix representatives of $[G|\bar{g}]_{\equiv_{\mathsf{TL}}}$ and let r map $G|\bar{g}$ to the representative of $[G|\bar{g}]_{\equiv_{\mathsf{TL}}}$. Define a function *rep* that maps $P^G|\bar{g}$ to $P^H|\bar{h}$, where $r(G|\bar{g}) = H|\bar{h}$.

For every $\equiv_{\mathsf{TL}}$-representative $H|\bar{h}$ (i.e., $H|\bar{h} = r(G|\bar{g})$ for some topology G and k-tuple $\bar{g}$), introduce a new Boolean proposition $q_{H|\bar{h}}$. By the FINITENESS property of TL there are finitely many such Boolean propositions, say n.

Define a valuation e_φ (that depends on φ) of these new atoms by

$$e_\varphi(q_{H|\bar{h}}) := \begin{cases} \top & \text{if } P^H|\bar{h} \models \varphi[p_{x_j} \mapsto p@j] \\ \bot & \text{otherwise.} \end{cases}$$

[9] All the types of quantifiers defined in Section 2.3, such as $\exists x \in E(y)$, can be dealt with similarly at the cost of notational overhead.

For every $G \in \mathbf{G}$ define Boolean formula $B_G := (\overline{Q_1} g_1 \in V_G) \dots (\overline{Q_k} g_k \in V_G)\ q_{r(G|\bar{g})}$, where $\overline{Q}$ is the Boolean operation corresponding to Q, e.g., $\overline{\exists} g_i \in V_G$ is interpreted as $\bigvee_{g \in V_G \setminus \{1,\dots,i-1\}}$.[10]

Then (for all P, G and φ)

$$\begin{aligned}
&P^G \models Q_1 x_1 \dots Q_k x_k.\ \varphi \\
&\iff Q_1 g_1 \in V_G \dots Q_k g_k \in V_G : P^G \models \varphi[p_{x_j} \mapsto p_{g_j}] \\
&\iff Q_1 g_1 \in V_G \dots Q_k g_k \in V_G : P^G|\bar{g} \models \varphi[p_{x_j} \mapsto p@j] \\
&\iff Q_1 g_1 \in V_G \dots Q_k g_k \in V_G : rep(P^G|\bar{g}) \models \varphi[p_{x_j} \mapsto p@j] \\
&\iff e_\varphi(B_G) = \top
\end{aligned}$$

Here $\varphi[p_{x_j} \mapsto p_{g_j}]$ is the formula resulting from replacing every atom in φ of the form p_{x_j} by the atom p_{g_j}, for $p \in \mathsf{AP_{pr}}$ and $1 \le j \le k$. Similarly $\varphi[p_{x_j} \mapsto p@j]$ is defined as the formula resulting from replacing (for all $p \in \mathsf{AP_{pr}}, j \le k$) every atom in φ of the form p_{x_j} by the atom $p@j$. The first equivalence is by the definition of semantics of indexed temporal logic; the second is by the definition of $P^G|\bar{g}$; the third is by the REDUCTION property of TL; the fourth is by the definition of e_φ and *rep*.

Fix $B_{G_1}, \dots, B_{G_N}$ (with $G_i \in \mathbf{G}$) such that every B_G ($G \in \mathbf{G}$) is logically equivalent to some B_{G_i}. Such a finite set of formulas exists since there are 2^{2^n} Boolean formulas (up to logical equivalence) over n Boolean propositions, and thus at most 2^{2^n} amongst the B_G for $G \in \mathbf{G}$.

By the equivalences above conclude that for every $G \in \mathbf{G}$ there exists $i \le N$ such that $P^G \models Q_1 x_1 \dots Q_k x_k.\ \varphi$ if and only if $P^{G_i} \models Q_1 x_1 \dots Q_k x_k.\ \varphi$. Thus '$\forall G \in \mathbf{G}, P^G \models \varphi$' is equivalent to '$\bigwedge_{i \le N} e_\varphi(B_{G_i})$' and so the integer $c := \max\{|V_{G_i}| : 1 \le i \le N\}$ is a cutoff for $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall, \exists\}^k\text{-}\mathsf{TL})$. □

Remark 6. Fix parameterized topology $\mathbf{G}$, and assume that one can decide if $P^G \models \varphi$ given $\varphi \in \mathcal{F}, G \in \mathbf{G}, P \in \mathcal{P}$. Then the theorem implies that for every $k \in \mathbb{N}$, $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \{\forall, \exists\}^k\text{-}\mathsf{TL})$ is decidable. Further, suppose that given k one could compute the finite set $G_1, \cdots, G_N$. Then by the last sentence in the proof one can compute the cutoff c. In this case, $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \{\forall, \exists\}^*\text{-}\mathsf{TL})$ is decidable.

4 Existence of Cutoffs for k-indexed $\mathsf{CTL}^*_d\backslash\mathsf{X}$

The following theorem answers Question 1 from the introduction.

Theorem 7. *Let* $\mathbf{G}$ *be a parameterized topology. Then for all* $k, d \in \mathbb{N}$*, the problem* $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall, \exists\}^k\text{-}\mathsf{CTL}^*_d\backslash\mathsf{X})$ *has a cutoff.*

Corollary 8. *Let* $\mathbf{G}$ *be a parameterized topology. Then for all* $k, d \in \mathbb{N}$*, the problem* $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall, \exists\}^k\text{-}\mathsf{CTL}^*_d\backslash\mathsf{X})$ *is decidable.*

[10] Note that in the Boolean propositions G is fixed while $\bar{g} = (g_1, \dots, g_n)$ ranges over $(V_G)^k$ and is determined by the valuation of the variables g_i.

To prove the Theorem it is enough, by Theorem 5, to show that $\{\forall, \exists\}^k$-$\mathsf{CTL}^*_d\backslash\mathsf{X}$ has the REDUCTION property, and that an arbitrary $\mathbf{G}$ has the FINITENESS property with respect to $\{\forall, \exists\}^k$-$\mathsf{CTL}^*_d\backslash\mathsf{X}$.

Reduction Theorem. Considering previous results, the following theorem identifies the key reason that a formula holds in a system on a big ring iff it holds in a system on a small ring (cf. [7]), and that a formula holds on a system on an arbitrary topology G iff that formula holds in a system on the network topology of G (in the notation of [4]).

Theorem 9 (Reduction). *For all $d, k \in \mathbb{N}$, topologies G, G', processes $P \in \mathcal{P}_u$, k-tuples $\bar{g}$ over V_G and k-tuples $\bar{g}'$ over $V_{G'}$:*
If $G|\bar{g} \equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}} G'|\bar{g}'$ then $P^G|\bar{g} \equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}} P^{G'}|\bar{g}'$.

The idea behind the proof is to show that paths in P^G can be simulated by paths in $P^{G'}$ (and vice versa). Given a path π in P^G, first project it onto G to get a path ρ that records the movement of the token, then take an equivalent path ρ' in G' which exists since $G|\bar{g} \equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}} G'|\bar{g}'$, and then lift ρ' up to get a path π' in $P^{G'}$ that is equivalent to π. This lifting step uses the assumption that process P is in $\mathcal{P}_u$, i.e., P cannot control where it sends the token, or from where it receives it. The proof can be found in the full version of the paper [1].

Remark 10. As immediate corollaries we get that the REDUCTION property holds with $\mathsf{TL} = \mathsf{LTL}\backslash\mathsf{X}$ (take $d = 1$), $\mathsf{CTL}^*\backslash\mathsf{X}$ (since if the assumption holds with $\mathsf{TL} = \mathsf{CTL}^*\backslash\mathsf{X}$ then the conclusion holds with $\mathsf{TL} = \mathsf{CTL}^*_d\backslash\mathsf{X}$ for all $d \in \mathbb{N}$, and thus also for $\mathsf{TL} = \mathsf{CTL}^*\backslash\mathsf{X}$) and, if P is finite, also for $\mathsf{TL} = \mathsf{CTL}\backslash\mathsf{X}$ (since $\mathsf{CTL}\backslash\mathsf{X}$ and $\mathsf{CTL}^*\backslash\mathsf{X}$ agree on finite structures).

Finiteness Theorem. Theorem 4 ([4, Corollary 3]) states that there exists $\mathbf{G}$ such that the problem $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \exists\exists\text{-}\mathsf{CTL}^*\backslash\mathsf{X})$ does not have a cutoff. We observed that the formulas from their result have unbounded nesting depth of path quantifiers. This leads to the idea of stratifying $\mathsf{CTL}^*\backslash\mathsf{X}$ by nesting depth.

Recall from Section 2.3 that i) $\mathsf{CTL}^*_d\backslash\mathsf{X}$ denotes the syntactic fragment of $\mathsf{CTL}^*\backslash\mathsf{X}$ in which formulas have path-quantifier nesting depth at most d; ii) $M \equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}} N$ iff M and N agree on all $\mathsf{CTL}^*_d\backslash\mathsf{X}$ formulas. Write $[M]_{\mathsf{CTL}^*_d\backslash\mathsf{X}}$ for the set of all LTSs N such that $M \equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}} N$.

Following the method of Section 3 we prove that the following FINITENESS property holds (where k represents the number of process-index quantifiers in the prenex indexed temporal logic formula).

Remark 11. For ease of exposition we sketch a proof under the assumption that path quantifiers in formulas ignore runs in which the token does not visit every process infinitely often. This is an explicit restriction in [4] and implicit in [7]. In the full version [1] we remove this restriction. For the purpose of this paper this restriction only affects the explicit cutoffs in Theorem 15.

Theorem 12 (Finiteness). *For all positive integers k and d, there are finitely many equivalence classes $[G|\bar{g}]_{\equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}}}$ where G is an arbitrary topology, and $\bar{g}$ is a k-tuple over V_G.*

Proof Idea. We provide an algorithm that given positive integers k, d, topology G, k-tuple $\bar{g}$ over V_G, returns a LTS $\textsc{con}_d G|\bar{g}$ such that $G|\bar{g} \equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}} \textsc{con}_d G|\bar{g}$. Moreover, we prove that for fixed k, d the range of $\textsc{con}_{d\text{-}}|_{\text{-}}$ is finite.

Recursively define a marking function μ_d that associates with each $v \in V_G$ a finite set (of finite strings over alphabet $\mu_{d-1}(V_G)$). For the base case define $\mu_0(v) := \Lambda(v)$, the labeling of $G|\bar{g}$. The marking $\mu_d(v)$ stores (representatives) of all strings of μ_{d-1}-labels of paths that start in v and reach some element in $\bar{g}$. The idea is that $\mu_d(v)$ determines the set of $\mathsf{CTL}^*_d\backslash\mathsf{X}$ formulas that hold in $G|\bar{g}$ with initial vertex v, as follows: stitch together these strings, using elements of $\bar{g}$ as stitching points, to get the $\mathsf{CTL}^*_d\backslash\mathsf{X}$ types of the infinite paths starting in v. This allows us to define a topology, called the d-contraction $\textsc{con}_d G|\bar{g}$, whose vertices are the μ_d-markings of vertices in G. In the full version [1] we prove that $G|\bar{g}$ is $\mathsf{CTL}^*_d\backslash\mathsf{X}$-equivalent to its d-contraction, and that the number of different d-contractions is finite, and depends on k and d.

Definition of d-contraction $\textsc{con}_d G|\bar{g}$. Next we define d-contractions.

Marking μ_d. Fix $k, d \in \mathbb{N}$, topology G, and k-tuple $\bar{g}$ over V_G. Let Λ be the labeling-function of $G|\bar{g}$, i.e., $\Lambda(v) = \{i\}$ if $v = g_i$, and $\Lambda(v) = \emptyset$ for $v \notin \bar{g}$. For every vertex $v \in V_G$ define a set $X(v)$ of paths of G as follows: a path $\pi = \pi_1 \dots \pi_t$, say of length t, is in $X(v)$ if π starts in v, none of $\pi_1, \dots, \pi_{t-1}$ is in $\bar{g}$, and $\pi_t \in \bar{g}$. Note that $X(g_i) = \{g_i\}$.

Define the marking μ_d inductively:

$$\mu_d(v) := \begin{cases} \Lambda(v) & \text{if } d = 0 \\ \{\mathsf{destutter}(\mu_{d-1}(\pi_1) \dots \mu_{d-1}(\pi_t)) : \pi_1 \dots \pi_t \in X(v), t \in \mathbb{N}\} & \text{if } d > 0, \end{cases}$$

where $\mathsf{destutter}(w)$ is the maximal substring s of w such that for every two consecutive letters s_i and s_{i+1} we have that $s_i \neq s_{i+1}$. Informally, remove identical consecutive letters of w to get the 'destuttering' $\mathsf{destutter}(w)$.

Note that the elements of $\mu_d(v)$ $(d > 0)$ are finite strings over the alphabet $\mu_{d-1}(V_G)$. For instance, strings in $\mu_1(v)$ are over the alphabet $\{\{1\}, \{2\}, \dots, \{k\}, \emptyset\}$.

Equivalence relation $\sim_d$. Vertices $v, u \in V_G$ are *d-equivalent*, written $u \sim_d v$, if $\mu_d(v) = \mu_d(u)$. We say that $\sim_d$ *refines* $\sim_j$ if $u \sim_d v$ implies $u \sim_j v$.

Lemma 13. *If $0 \leq j < d$, then $\sim_d$ refines $\sim_j$.*

Indeed, observe that for all nodes v, all strings in $\mu_d(v)$ start with the letter $\mu_{d-1}(v)$. Thus $\mu_d(v) = \mu_d(u)$ implies that $\mu_{d-1}(v) = \mu_{d-1}(u)$. In other words, if $u \sim_d v$ then $u \sim_{d-1} v$, and thus also $u \sim_j v$ for $0 \leq j < d$.

d-contraction $\textsc{con}_d G|\bar{g}$. Define an LTS $\textsc{con}_d G|\bar{g}$ called the d-contraction of $G|\bar{g}$ as follows. The nodes of the contraction are the $\sim_d$-equivalence classes. Put an edge between $[u]_{\sim_d}$ and $[v]_{\sim_d}$ if there exists $u' \in [u]_{\sim_d}, v' \in [v]_{\sim_d}$ and an edge in G from u' to v'. The initial state is $[x]_{\sim_d}$ where x is the initial vertex of G. The label of $[u]_{\sim_d}$ is defined to be $\Lambda(u)$ — this is well-defined because, by Lemma 13, $\sim_d$ refines $\sim_0$.

In the full version [1] we prove that $G|\bar{g}$ is $\mathsf{CTL}^*_d\backslash\mathsf{X}$-equivalent to its d-contraction, and that the number of different d-contractions is finite, and depends on k and d. □

5 Cutoffs for k-index $\mathsf{CTL}^*\backslash\mathsf{X}$ and Concrete Topologies

The following two Theorems answer Question 2 from the introduction regarding PMCP for specifications from $\{\forall,\exists\}^*$-$\mathsf{CTL}^*\backslash\mathsf{X}$.

First, PMCP is undecidable for certain (pathological) parameterized topologies $\mathbf{G}$ and specifications from $\{\forall,\exists\}^*$-$\mathsf{CTL}^*\backslash\mathsf{X}$.

Theorem 14. *There exists a process $P \in \mathcal{P}_u$, and parameterized topologies $\mathbf{G}$, $\mathbf{H}$, such that the following PMCP problems are undecidable*
1. $\mathit{PMCP}_{\mathbf{G}}(\{P\}, \{\forall,\exists\}^*\text{-}\mathit{LTL}\backslash\mathit{X})$.
2. $\mathit{PMCP}_{\mathbf{H}}(\{P\}, \{\forall,\exists\}^2\text{-}\mathit{CTL}\backslash\mathit{X})$.

Moreover, $\mathbf{G}$ and $\mathbf{H}$ can be chosen to be computable sets of topologies.

Second, PMCP is decidable for certain (regular) parameterized topologies and specifications from $\{\forall\}^*$-$\mathsf{CTL}^*\backslash\mathsf{X}$. This generalises results from Emerson and Namjoshi [7] who show this result for $k = 1, 2$ and uni-directional ring topologies. By Remark 11 these cutoffs apply under the assumption that we ignore runs that do not visit every process infinitely often.

Theorem 15. *If $\mathbf{G}$ is as stated, then $\mathit{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall\}^k\text{-}\mathit{CTL}^*\backslash\mathit{X})$ has the stated cutoff.*
1. *If $\mathbf{G}$ is the set of uni-directional rings, then $2k$ is a cutoff.*
2. *If $\mathbf{G}$ is the set of bi-directional rings, then $2k$ is a cutoff.*
3. *If $\mathbf{G}$ is the set of cliques, then $k+1$ is a cutoff.*
4. *If $\mathbf{G}$ is the set of stars, then $k+1$ is a cutoff.*

Consequently, for each $\mathbf{G}$ listed, $\mathit{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall\}^*\text{-}\mathit{CTL}^*\backslash\mathit{X})$ is decidable.

This theorem is proved following Remark 6: given k, d, we compute a set $G_1, \ldots, G_N \in \mathbf{G}$ such that every B_G for $G \in \mathbf{G}$ is logically equivalent to some B_{G_i}, where the Boolean formula B_G is defined as $\bigwedge_{\bar{g}} q_{\text{CON}_d G|\bar{g}}$. To do this note that B_G is logically equivalent to B_H if and only if $\{\text{CON}_d G|\bar{g} : \bar{g} \in V_G\} = \{\text{CON}_d H|\bar{h} : \bar{h} \in V_H\}$ (this is where we use that there is no quantifier alternation). So it is sufficient to prove that, if c is the stated cutoff,

$$|V_G|, |V_H| \geq c \implies \{\text{CON}_d G|\bar{g} : \bar{g} \in V_G\} = \{\text{CON}_d H|\bar{h} : \bar{h} \in V_H\}$$

To illustrate how to do this we analyse the case of uni-directional rings and cliques (the other cases are similar).

Uni-directional rings. Suppose $\mathbf{G}$ are the uni-directional rings and let $G \in \mathbf{G}$. Fix a k-tuple of distinct elements of V_G, say $(g_1, g_2, \ldots, g_k)$. Define a function $f : V_G \to \{g_1, \ldots, g_k\}$ that maps v to the first element of $\bar{g}$ on the path $v, v+1, v+2, \ldots$ (addition is mod $|V_G|$). In particular $f(g_i) = g_i$ for $i \in [k]$. In

the terminology of the proof of Theorem 12, $X(v)$ consists of the simple path $v, v+1, \cdots, f(v)$.

We now describe μ_d. Clearly $\mu_d(g_i) = \{\mu_{d-1}(g_i)\}$. By induction on d one can prove that if $v \notin \bar{g}$ with $f(v) = g_j$ then $\mu_d(v) = \{\mu_{d-1}(v) \cdot \mu_{d-1}(g_j)\}$. So for every $d > 1$, the equivalences $\sim_d$ and $\sim_1$ coincide.

d	0	1	2	...
$\mu_d(v)$ for $v = g_i$	$\{i\}$	$\{\{i\}\}$	$\{\{\{i\}\}\}$	...
$\mu_d(v)$ if $v \notin \bar{g}$ and $f(v) = g_j$	$\emptyset$	$\{\emptyset \cdot \{j\}\}$	$\{\{\emptyset \cdot \{j\}\} \cdot \{\{j\}\}\}$	...

Thus for every $k \in \mathbb{N}$, the d-contraction $\textsc{con}_d G|\bar{g}$ is a ring of size at most $2k$ (in particular, it is independent of d). In words, the d-contraction of G is the ring resulting by identifying adjacent elements not in $\bar{g}$. It is not hard to see that if G, H are rings such that $|V_G|, |V_H| \geq 2k$ then for every $\bar{g}$ there exists $\bar{h}$ such that $\textsc{con}_d G|\bar{g} = \textsc{con}_d H|\bar{h}$.

Cliques. Fix $n \in \mathbb{N}$. Let G be a clique of size n. That is: $V_G = [n]$ and $(i, j) \in E_G$ for $1 \leq i \neq j \leq n$. Fix a k-tuple of distinct elements of V_G, say $(g_1, g_2, \ldots, g_k)$. We now describe $\mu_d(v)$. Clearly $\mu_d(g_i) = \{\mu_{d-1}(g_i)\}$ and for $v \notin \bar{g}$ we have $\mu_d(v) = \{\mu_{d-1}(v) \cdot \mu_{d-1}(j) : j \in [k]\}$.

It is not hard to prove that for every $k \in \mathbb{N}$, the d-contraction $\textsc{con}_d G|\bar{g}$ is the clique of size $k+1$. In words, the d-contraction of G results from G by identifying all vertices not in $\bar{g}$. It is not hard to see that if G, H are cliques such that $|V_G|, |V_H| \geq k+1$ then for every $\bar{g}$ there exists $\bar{h}$ such that $\textsc{con}_d G|\bar{g} = \textsc{con}_d H|\bar{h}$.

Remark 16. For cliques and stars $k+1$ is also the cutoff for $\{\forall, \exists\}^k$-$\mathsf{CTL}^*\backslash\mathsf{X}$.

Remark 17. $2k$ is *not* the cutoff for uni-rings and $\{\forall, \exists\}^k$-$\mathsf{CTL}^*\backslash\mathsf{X}$ as stated in [12, Corollary 2]. The formula $\forall i \forall j \exists k. adj(k, i) \vee adj(k, j)$, where $adj(k, i) := tok_i \rightarrow tok_i \,\mathsf{U}\, tok_k \vee tok_k \rightarrow tok_k \,\mathsf{U}\, tok_i$, holds in ring of size 6 but not in 7.

6 There are No Cutoffs for Direction-Aware Systems

In the following, we consider systems where processes can choose which directions are used to send or receive the token, i.e., process templates are from $\mathcal{P}_{snd}$, $\mathcal{P}_{rcv}$, or $\mathcal{P}_{sndrcv}$. Let $\mathbf{B}$ be the parameterized topology of all bi-directional rings, with directions cw (clockwise) and ccw (counter-clockwise). The following theorem answers Question 3 from the introduction.

Theorem 18. *1. The problem $\mathsf{PMCP}_{\mathbf{B}}(\mathcal{P}_{sndrcv}, \forall\text{-}\mathsf{LTL}\backslash\mathsf{X})$ is undecidable.*
2. For $\mathcal{F}$ equal to $\{\forall\}^9$-$\mathsf{LTL}\backslash\mathsf{X}$ or $\{\exists\}^9$-$\mathsf{CTL}\backslash\mathsf{X}$, and $\mathcal{P} \in \{\mathcal{P}_{snd}, \mathcal{P}_{rcv}\}$, there exists a parameterized topology $\mathbf{G}$ such that $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \mathcal{F})$ is undecidable.

Proof Idea. In all cases we reduce the non-halting problem of two-counter machines (2CMs) to the PMCP. The idea is that one process, the *controller*, simulates the finite-state control of the 2CM. The other processes, arranged in a chain or a ring, are *memory processes*, collectively storing the counter values with a

fixed memory per process. This allows a given system to simulate a 2CM with bounded counters. Since a 2CM terminates if and only if it terminates for some bound on the counter values, we can reduce the non-halting problem of 2CMs to the PMCP. The main work is to show that the controller can issue commands, such as 'increment counter 1' and 'test counter 1 for zero'. We give a detailed proof sketch for part 1 of the theorem, and then outline a proof for part 2.

1. $\forall$*-LTL*$\backslash X$ *and* $\mathcal{P}_{sndrcv}$ *in bi-directional rings.*

The process starting with the token becomes the controller, all others are memory, each storing one bit for each counter of the 2CM. The current value of a counter c is the total number of corresponding bits (c-bits) set to 1. Thus, a system with n processes can store counter values up to $n-1$.

Fix a numbering of 2CM-commands, say $0 \mapsto$ 'increment counter 1', $1 \mapsto$ 'decrement counter 1', $2 \mapsto$ 'test counter 1 for zero', etc. Every process has a command variable that represents the command to be executed when it receives the token from direction ccw.

If the controller sends the token in direction cw, the memory processes will increment (mod 6) the command variable, allowing the controller to encode which command should be executed. Every process just continues to pass the token in direction cw, until it reaches the controller again.

If the controller sends the token in direction ccw, then the memory processes try to execute the command currently stored. If it is an 'increment counter c' or 'decrement counter c' command, the memory process tries to execute it (by incrementing/decrementing its c-bit). If the process cannot execute the command (because the c-bit is already 1 for an increment, or 0 for a decrement), then it passes the token along direction ccw and remembers that a command is being tried. If the token reaches a memory process which can execute the command, then it does so and passes the token back in direction cw. The processes that remembered that a command is being tried will receive the token from direction cw, and know that the command has been successfully executed, and so will the controller. If the controller gets the token from ccw, the command failed. In this case, the controller enters a loop in which it just passes the token in direction cw (and no more commands are executed).

If the command stored in the memory processes is a 'test for zero counter c', then the processes check if their c-bit is 0. If this is the case, it (remembers that a command is being tried and) passes the token to the next process in direction ccw. If the token reaches a process for which the c-bit is 1, then this process sends the token back in direction cw. Other memory processes receiving it from cw (and remembering that the command is being tried), pass it on in direction cw. In this case, the controller will receive the token from cw and know that counter c is not zero. On the other hand, if all memory processes store 0 in their c-bit, then they all send the token in direction ccw. Thus, the controller will receive it from ccw and knows that counter c currently is zero. To terminate the command, it sends the token in direction cw, and all processes (which remembered that a command is being tried), know that execution of this command is finished.

With the description above, a system with $n-1$ memory processes can simulate a 2CM as long as counter values are less than n. Let $HALT$ be an atomic proposition that holds only in the controller's halting states. Then solving the PMCP for $\forall i\, \mathsf{G}\, \neg HALT_i$ amounts to solving the non-halting problem of the 2CM.

2. $\{\forall\}^9$*-LTL\X and* $\mathcal{P}_{snd}$.

We give a proof outline. In this case there are $2n$ memory processes, n for each counter $c \in \{1, 2\}$. The remaining 9 processes are special and called 'controller', 'counter c is zero', 'counter c is not zero', 'counter c was incremented', and 'counter c was decremented'. When the controller wants to increment or decrement counter c, it sends the token nondeterministically to some memory process for counter c. When the controller wants to test counter c for zero, it sends the token to the first memory process. When a memory process receive the token it does not know who sent it, and in particular does not know the intended command. Thus, it non-deterministically takes the appropriate action for one of the possible commands. If its bit is set to 0 then it either i) increments its bit and sends the token to a special process 'counter c was incremented', or ii) it sends the token to the next memory node in the chain, or to the special process 'counter c is zero' if it is the last in the chain. If its bit is set to 1 then it either i) decrements its bit and sends the token to a special process 'counter c was decremented', or ii) sends the token to a special process 'counter c is not zero'.

Even though incoming directions are not available to the processes, we can write the specification such that, out of all the possible non-deterministic runs, we only consider those in which the controller receives the token from the expected special node (the formula requires one quantified index variable for each of the special nodes). So, if the controller wanted to increment counter c it needs to receive the token from process 'counter c was incremented'. If the controller receives the token from a different node, it means that a command was issued but not executed correctly, and we disregard this run. Otherwise, the system of size $2n+1$ correctly simulates the 2CM until one of the counter values exceeds n. As before, parameterized model checking of this construction would allow us to solve the non-halting problem for 2CMs. □

7 Extensions

There are a number of extensions of direction-unaware TPSs for which the theorems that state existence of cutoffs (Theorems 7 and 15) still hold. We describe these in order to highlight assumptions that make the proofs work:

1. Processes can be infinite-state.
2. The EN-restriction on the process template P can be relaxed: replace item *vii*) in Definition 2.1 by "For every state q that has the token there is a finite path $q \dots q'$ such that q' does not have the token, and for every q that does not have the token there is a finite path $q \dots q'$ such that q' has the token".
3. One can further allow *direction-sensing* TPSs, which is a direction-aware TPS with an additional restriction on the process template: "If $q \xrightarrow{\mathsf{d}} q' \in \delta$ for some direction $d \in \mathsf{Dir_{snd}}$, then for every $d \in \mathsf{Dir_{snd}}$ there exists a transition

$q \xrightarrow{\mathsf{d}} q'' \in \delta$"; and a similar statement for $\mathsf{Dir}_{\mathsf{rcv}}$. Informally: we can allow processes to change state according to the direction that the token is (non-deterministically) sent to or received, but the processes are not allowed to block any particular direction.

4. One can further allow the token to carry a value but with the strong restriction that from every state that has the token and every value v there is a path of internal actions in P which eventually sends the token with value v, and the same for receiving.

These conditions on P all have the same flavor: they ensure that a process can not choose what information to send/receive, whether that information is a value on the token or a direction for the token.

8 Related work

This paper is based on [14,7,4] (as described in the introduction). However, there are several other relevant papers.

Emerson and Kahlon [6] consider token-passing in uni- and bi-directional rings, where processes are direction-aware and tokens carry messages (but can only be changed a bounded number of times). However, the provided cutoff theorems only hold for specifications that talk about two processes (in a uni-directional ring) or one process (in a bi-directional ring), and process implementations need to be deterministic. Furthermore, cutoffs depend on the complexity of the process implementation.

German and Sistla [11] provide cutoffs for the PMCP for systems with pairwise synchronization. Although pairwise synchronization can simulate token-passing, their cutoff results are restricted to cliques and 1-indexed LTL. Moreover, their proof uses vector-addition systems with states and their cutoff depends on the process template and the specification formula. German and Sistla [11, Section 6] also consider non-prenex formulas with the restriction that exactly one index variable name occurs, e.g., $\exists i.Req_i \rightarrow \mathsf{F}\,\exists i.Grant_i$, and prove that the resulting PMCP is undecidable.

Delzanno et al. [5] study a model of broadcast protocols on arbitrary topologies, in which a process can synchronize with all of its available neighbours 'at once' by broadcasting a message (from a finite set of messages). They prove undecidability of PMCP for systems with arbitrary topologies and 1-indexed safety properties, and that the problem becomes decidable if one restricts the topologies to 'graphs with bounded paths' (such as stars). Their proof uses the machinery of well-structured transitions systems, and no cutoffs are provided. They also show undecidability of the PMCP in the case of non-prenex indexed properties of the form $\mathsf{G}(\exists i.s(i) \in B)$ on general and the restricted topologies.

9 Summary

The goal of this work was to find out under what conditions there are cutoffs for temporal logics and token-passing systems on general topologies. We found that

stratifying prenex indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ by nesting-depth of path quantifiers allowed us to recover the existence of cutoffs; but that there are no cutoffs if the processes are allowed to choose the direction of the token. In all the considered cases where there is no cutoff we show that the PMCP problem is actually undecidable.

Our positive results are provided by a construction that generalizes and unifies the known positive results, and clearly decomposes the problem into two aspects: tracking the movement of the token through the underlying topology, and simulating the internal states of the processes that the specification formula can see. The construction yields small cutoffs for common topologies (such as rings, stars, and cliques) and specifications from prenex indexed $\mathsf{CTL}^*\backslash\mathsf{X}$.

Acknowledgments. We thank Roderick Bloem for detailed comments on numerous drafts and Krishnendu Chatterjee for important comments regarding the structure of the paper. We thank Roderick Bloem, Igor Konnov, Helmut Veith, and Josef Widder for discussions at an early stage of this work that, in particular, pointed out the relevance of direction-unawareness in [4].

References

1. Aminof, B., Jacobs, S., Khalimov, A., Rubin, S.: Parameterized Model Checking of Token-Passing Systems (2013), pre-print on arxiv.org
2. Baier, C., Katoen, J.P., et al.: Principles of model checking, vol. 26202649. MIT press Cambridge (2008)
3. Browne, M.C., Clarke, E.M., Grumberg, O.: Reasoning about networks with many identical finite state processes. Inf. Comput. 81, 13–31 (April 1989)
4. Clarke, E., Talupur, M., Touili, T., Veith, H.: Verification by network decomposition. In: CONCUR 2004. vol. 3170, pp. 276–291 (2004)
5. Delzanno, G., Sangnier, A., Zavattaro, G.: Parameterized verification of ad hoc networks. In: CONCUR. LNCS, vol. 6269, pp. 313–327 (2010)
6. Emerson, E.A., Kahlon, V.: Parameterized model checking of ring-based message passing systems. In: CSL. LNCS, vol. 3210, pp. 325–339. Springer (2004)
7. Emerson, E.A., Namjoshi, K.S.: On reasoning about rings. Int. J. Found. Comput. Sci. 14(4), 527–550 (2003)
8. Emerson, E.A., Sistla, A.P.: Symmetry and model checking. In: CAV. pp. 463–478 (1993)
9. Emerson, E., Namjoshi, K.: Reasoning about rings. In: POPL. pp. 85–94 (1995)
10. Esparza, J., Finkel, A., Mayr, R.: On the verification of broadcast protocols. Logic in Computer Science, Symposium on 0, 352 (1999)
11. German, S.M., Sistla, A.P.: Reasoning about systems with many processes. J. ACM 39(3), 675–735 (1992)
12. Khalimov, A., Jacobs, S., Bloem, R.: Towards efficient parameterized synthesis. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) VMCAI, LNCS, vol. 7737, pp. 108–127. Springer Berlin Heidelberg (2013)
13. Shamir, S., Kupferman, O., Shamir, E.: Branching-depth hierarchies. ENTCS 39(1), 65 – 78 (2003)
14. Suzuki, I.: Proving properties of a ring of finite-state machines. Inf. Process. Lett. 28(4), 213–214 (Jul 1988)