Model Checking Parameterised Multi-Token Systems via the Composition Method*

Benjamin Aminof[1] and Sasha Rubin[2]

[1] Technische Universiät Wien, Austria
benj@forsyte.at
[2] Università di Napoli "Federico II", Italy
sasha.rubin@unina.it

Abstract. We study the model checking problem of parameterised systems with an arbitrary number of processes, on arbitrary network-graphs, communicating using multiple multi-valued tokens, and specifications from indexed-branching temporal logic. We prove a composition theorem, in the spirit of Feferman-Vaught (1959) and Shelah (1979), and a finiteness theorem, and use these to decide the model checking problem. Our results assume two constraints on the process templates, one of which is the standard fairness assumption introduced in the cornerstone paper of Emerson and Namjoshi (1995). We prove that lifting any of these constraints results in undecidability. The importance of our work is three-fold: i) it demonstrates that the composition method can be fruitfully applied to model checking complex parameterised systems; ii) it identifies the most powerful model, to date, of parameterised systems for which model checking indexed *branching-time* specifications is decidable; iii) it tightly marks the borders of decidability of this model.

1 Introduction

Many concurrent systems consist of identical processes running in parallel, such as peer-to-peer systems, sensor networks, multi-agent systems, etc. [14,27,29]. Model checking is a successful technique for establishing correctness of such systems: model a system as the product transition system $\mathbf{P}^G$, where $\mathbf{P}$ is a transition system representing the process, and G is a network-graph describing the communication lines [9]. If the number of processes is not known, or too large for model-checking tools, it is appropriate to express correctness as a parameterised model checking (PMC) problem: decide if a given temporal logic specification holds irrespective of the number of processes [8,22]. That is, for a fixed infinite set $\mathcal{G}$ of network-graphs (e.g., $\mathcal{G}$ may be the set of all ring network-graphs), decide, given process $\mathbf{P}$ and specification φ if $\mathbf{P}^G \models \varphi$ for all $G \in \mathcal{G}$. Not surprisingly, PMC is a hard problem, i.e., even for a given $\mathbf{P}$, PMC consists of

* Benjamin Aminof is supported by the Vienna Science and Technology Fund (WWTF) through grant ICT12-059. Sasha Rubin is a Marie Curie fellow of the Istituto Nazionale di Alta Matematica.

model-checking infinitely many systems; in other words, it can be thought of as model checking infinite-state systems [15,23]. It quickly becomes undecidable [8], even if the participating processes are finite-state [32], and even if they do not communicate with each other at all [25]. Thus, much work has focused on proving decidability for restricted systems, i.e., by limiting both the communication mechanism and the specification logic [20,17,13,1,2,3,10,6,7].

We consider specifications in indexed branching temporal logic without the "next-time" operator X (formulas without X are stuttering-insensitive, and are thus natural for specifying asynchronous concurrent systems [18]). More specifically, we use formulas of prenex indexed-$\mathsf{CTL}^*_d\backslash\mathsf{X}$ (CTL^* without X in which there at most $d \in \mathbb{N}$ nested path quantifiers). These are formulas of the form $\forall x_1 \exists x_2 \cdots \forall x_k \phi$ where the variables x_i vary over processes, and ϕ is a $\mathsf{CTL}^*_d\backslash\mathsf{X}$ formula where atomic propositions are paired with the index variables x_i [30]. This specification language allows one to express many natural properties, e.g. mutual-exclusion. Non-prenex temporal logic is so powerful that its parameterised model checking is undecidable already for indexed $\mathsf{LTL}\backslash\mathsf{X}$ specifications, even for non-communicating processes [25]. We consider systems with an arbitrary number of processes, on arbitrary network-graphs, communicating using multiple multi-valued tokens. Such systems arise in various contexts: multiple tokens are a means to resolve conflicts over multiple shared resources such as in the drinking philosophers problem [12], they can represent mobile finite-state agents [29,4,5], and tokens are used in self-stabilisation algorithms [24]. We further allow the edges of the network-graph to carry directions, called local port-numberings, along which the processes may send and receive tokens. Such network-graphs are typical in the distributed computing literature, for instance in mobile finite-state agents, e.g., [14,27,26]. Note that even slightly more powerful communication primitives such as pairwise-rendezvous have undecidable PMC for expressive logics such as prenex indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ [3].

The Compositional Method for Parameterised Model-Checking. Composition theorems, pioneered in the seminal work of Feferman and Vaught [21] and Shelah [31], are tools that reduce reasoning about compound structures to reasoning about their component parts. Unfortunately, composition theorems for product systems are not easy to come by [28]. Nonetheless, we successfully apply the composition method to multi-token systems and prenex indexed-$\mathsf{CTL}^*_d\backslash\mathsf{X}$ specification languages. Our composition result (Theorem 3) states, roughly, that if two processes $\mathbf{X}, \mathbf{Y}$ are bisimilar, and if two network-graphs G, H with k visible vertices $\bar{g}, \bar{h}$ (i.e., vertices that formulas can talk about) are $\mathsf{CTL}^*_d\backslash\mathsf{X}$-equivalent, then the product systems $\mathbf{X}^G, \mathbf{Y}^H$ with visible processes at $\bar{g}, \bar{h}$ are $\mathsf{CTL}^*_d\backslash\mathsf{X}$-equivalent. We complement this with a finiteness result (Theorem 5) that states, roughly, that for every $d, k \in \mathbb{N}$, there are only finitely many $\mathsf{CTL}^*_d\backslash\mathsf{X}$-types of network-graphs G with k visible vertices (even though, over all graphs, there are infinitely many logically inequivalent $\mathsf{CTL}^*_d\backslash\mathsf{X}$ formulas, already for $d = 1$). Combining the composition and finiteness we reduce reasoning about $\mathbf{P}^G$ for all $G \in \mathcal{G}$ to reasoning about finitely many $G \in \mathcal{G}$, and thus decide the PMC.

Our systems employ two fairness conditions: the standard assumption (introduced in [18]) that processes that make infinitely many transitions must make infinitely many token passing transitions; and the assumption that from every state from which a process can send (resp. receive) a token, it can also reach a state in which it can send (resp. receive) the token in any other given direction and value. We show that if either of the fairness conditions is removed PMC becomes undecidable; furthermore, it remains undecidable even if other very restrictive assumptions are added. It is notable that until now it was not known if the standard fairness assumption was necessary for decidability. Thus, our results answer this question in the affirmative. Due to space constraints, some proofs are only sketched or omitted.

2 Definitions

A *labeled transition system (LTS)* is a tuple $\langle \mathsf{AP}, \Sigma, Q, Q_0, \delta, \lambda \rangle$ where AP is a finite set of *atomic propositions* (also called *atoms*), Σ is a finite set of *actions*, Q is a finite set of *states*, $Q_0 \subseteq Q$ is a set of *initial states*, $\delta \subseteq Q \times \Sigma \times Q$ is a *transition relation*, and $\lambda : Q \to 2^{\mathsf{AP}}$ is a *labeling* function. We write $q \xrightarrow{\sigma} q'$ if $(q, \sigma, q') \in \delta$, and write $q \to q'$ if $(q, \sigma, q') \in \delta$ for some $\sigma \in \Sigma$. An LTS is *total* if for every $q \in Q$ there exists $q' \in Q$ such that $q \to q'$. A *transition system (TS)* is a tuple $\langle \mathsf{AP}, Q, Q_0, \delta, \lambda \rangle$ like an LTS, except that $\delta \subseteq Q \times Q$. A *path* of an LTS is a finite string $q_0 q_1 \dots q_n \in Q^+$ or an infinite string $q_0 q_1 \dots \in Q^\omega$ such that $q_i \to q_{i+1}$ for all i. An *edge-path* of an LTS is a (finite or infinite) sequence of transitions $(q_0, \sigma_0, q_1)(q_1, \sigma_1, q_2) \dots$ of δ. Every edge-path $(q_0, \sigma_0, q_1)(q_1, \sigma_1, q_2) \dots$ *induces* the path $q_0 q_1 q_2 \dots$. A path is *simple* if no vertex repeats, and it is a *simple cycle* if the (only) two equal vertices are the first and last. An edge-path is *simple (resp. simple cycle)* if the induced path is. A *run* of an LTS is a maximal path starting in an initial state. An LTS can be translated into a TS by simply removing the actions from transitions. We will implicitly use this translation.

System Model. Informally, a token-passing system is an LTS obtained by taking some finite edge-labeled graph (called a *topology* or *network-graph*), placing one process at each of its vertices, and having all processes execute the same code (given in the form of a finite-state *process template*). Processes synchronize by sending one of finitely many tokens along the edges of the topology, which are labeled with a send direction, a receive direction, and a token-value.[3] In the most general model, processes can choose the direction to send the token, from which direction to receive a token, and the value of the token. In case there is more than one possible recipient for a token, one is chosen nondeterministically

In what follows, we use a finite non-empty set of *token values* Σ_{val}, finite disjoint non-empty sets Σ_{snd} of *send directions* and Σ_{rcv} of *receive directions*, and an integer $T > 0$ (the number of tokens in the system). Since these data are usually fixed, we do not mention them if they are clear from the context.

[3] The direction-labels on the edges (also called a local orientation) represent network port numbers [14,27]. All of our results also hold for the case that each edge has a single direction-label that combines send and receive directions, e.g., "clockwise".

Process Template. Fix a countable set $\mathcal{AP}$ of atomic propositions for use by all process templates. We assume that $\mathcal{AP}$ also contains, for every integer $i \geq 0$, the special proposition tok^i. A *process template* (w.r.t. $T \in \mathbb{N}, \Sigma_{\mathsf{val}}, \Sigma_{\mathsf{snd}}, \Sigma_{\mathsf{rcv}}$) is a total LTS $\mathbf{P} = \langle \mathsf{AP}_{\mathsf{pr}}, \Sigma_{\mathsf{pr}}, Q, Q_0, \delta, \lambda \rangle$ such that: (i) $\mathsf{AP}_{\mathsf{pr}} \subset \mathcal{AP}$ is a finite set containing tok^i for $0 \leq i \leq T$; (ii) for every $q \in Q$ there is exactly one i such that $\mathsf{tok}^i \in \lambda(q)$ (and we say that q *has* i *tokens*); (ii) $\Sigma_{\mathsf{pr}} = \{\mathsf{int}\} \cup [(\Sigma_{\mathsf{snd}} \cup \Sigma_{\mathsf{rcv}}) \times \Sigma_{\mathsf{val}}]$; (iv) $Q_0 = \{\iota_T, \iota_0\}$ where ι_T has T tokens, and ι_0 has 0 tokens; (v) For every transition $q \xrightarrow{\sigma} q'$: if $\sigma \in \Sigma_{\mathsf{snd}} \times \Sigma_{\mathsf{val}}$ then $\exists i > 0$ such that q has i tokens and q' has $(i-1)$ tokens; if $\sigma \in \Sigma_{\mathsf{rcv}} \times \Sigma_{\mathsf{val}}$ then $\exists i < T$ such that q has i tokens and q' has $(i+1)$ tokens; and if $\sigma = \mathsf{int}$ then q and q' have the same number of tokens.

Notation. We say that the *initial states* of two templates $\mathbf{X}, \mathbf{Y}$ *are bisimilar* if, writing $\iota_0^{\mathbf{Z}}, \iota_T^{\mathbf{Z}}$ for the initial states of template $\mathbf{Z}$, we have that $\iota_\epsilon^{\mathbf{X}} \sim \iota_\epsilon^{\mathbf{Y}}$ for $\epsilon \in \{0, T\}$, where $\sim$ is a bisimulation relation between $\mathbf{X}$ and $\mathbf{Y}$. The elements of Q are called *local states* and the transitions in δ are called *local transitions*. A transition (q, σ, q') is called a *local send transition* if $\sigma \in \Sigma_{\mathsf{snd}} \times \Sigma_{\mathsf{val}}$; it is called a *local receive transition* if $\sigma \in \Sigma_{\mathsf{rcv}} \times \Sigma_{\mathsf{val}}$; and it is called a *local internal transition* if $\sigma = \mathsf{int}$. The local send/receive transitions are collectively known as *local token-passing transitions*. A local state q for which there exists a local send-transition (resp. receive-transition) $(q, (d, m), q')$ is called *ready to send (resp. receive) in direction* d *and value* m; it is also called *ready to send (resp. receive) in direction* d, *ready to send (resp. receive) value* m, or simply *ready to send (resp. receive)*.

Fairness Notions. A template $\mathbf{P}$ is *fair* if every infinite path $q_1 q_2 \cdots$ in $\mathbf{P}$ satisfies that for infinitely many i the transition from q_i to q_{i+1} is a local token passing transition [18]. Other restrictions that we consider involve treating different directions and/or different token values in an unbiased way, and thus "fairly". Formally, a state q of $\mathbf{P}$ having i tokens that is ready to send (resp. receive) is called an i-*sending* (resp. i-*receiving*) state. A path in $\mathbf{P}$ is an i-path if it only mentions states having i tokens. A template $\mathbf{P}$ is *direction/value-fair* if for every $\mathsf{d} \in \Sigma_{\mathsf{rcv}}$, $\mathsf{e} \in \Sigma_{\mathsf{snd}}$ and $\mathsf{m} \in \Sigma_{\mathsf{val}}$, for every i-receiving (resp. i-sending) state q there is a finite i-path from q ending in a state that is ready to receive (resp. send) in direction d (resp. from direction e) and value m. We denote by $\mathcal{P}^{\mathsf{FDV}}$ the set of fair and direction/value-fair process-templates; and by $\mathcal{P}^{\mathsf{FD}}$ the set of fair, direction-fair, and valueless (i.e., with $|\Sigma_{\mathsf{val}}| = 1$) process templates. As noted in the introduction, the undecidability results (Section 4) show that the limitations of $\mathcal{P}^{\mathsf{FDV}}$ are, in a strong sense, minimal limitations one can impose and still obtain a decidable parameterized model checking problem.

Topology/Network-Graph. LTS $G = \langle \emptyset, \Sigma_{\mathsf{snd}} \times \Sigma_{\mathsf{rcv}}, V, \{\mathsf{init}\}, E, \lambda \rangle$ is a *topology* (w.r.t. $\Sigma_{\mathsf{snd}}, \Sigma_{\mathsf{rcv}}$) if: (i) $V = [n]$ for some $n \in \mathbb{N}$ is a set of *vertices* (or *process indices*); (ii) $\mathsf{init} \in V$ is an *initial vertex*; (iii) $E \subseteq V \times (\Sigma_{\mathsf{snd}} \times \Sigma_{\mathsf{rcv}}) \times V$ is called the *edge relation*, (iv) and λ is the constant function $\lambda(v) = \emptyset$. We abbreviate and write $G = \langle V, E, \mathsf{init} \rangle$ or $G = \langle V_G, E_G, \mathsf{init}_G \rangle$. The *underlying graph of* G has vertex set V and edge (v, w) iff $\exists \mathsf{d}, \mathsf{e}. (v, (\mathsf{d}, \mathsf{e}), w) \in E$. We assume that the underlying graph is irreflexive, contains no vertices without outgoing edges, and that every vertex $v \in V$ is reachable from init. These are natural assumptions since paths in the topology represent the paths along which the tokens can move.

Parameterized Topology $\mathcal{G}$. Let $\mathcal{G}$ denote a countable set of topologies. For example, the set of all pipelines, or the set of all rings.

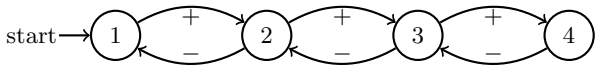

Fig. 1: example pipeline topology: + signifies a $(\mathsf{snd_E}, \mathsf{rcv_W})$ label, and − $(\mathsf{snd_W}, \mathsf{rcv_E})$.

Token-Passing System. Given a process template $\mathbf{P} = \langle \mathsf{AP_{pr}}, \Sigma_{\mathsf{pr}}, Q, Q_0, \delta, \lambda \rangle$ and a topology $G = \langle V, E, \mathsf{init} \rangle$, we define the *token-passing system* (or TPS for short) to be the LTS $\mathbf{P}^G = \langle \mathsf{AP_{sys}}, \Sigma_{\mathsf{sys}}, S, S_0, \Delta, \Lambda \rangle$. Informally, the system $\mathbf{P}^G$ can be thought of as the interleaving parallel composition of $\mathbf{P}$ over G. The tokens start with process init. Time is discrete: at each step either exactly one process makes a local internal transition; or exactly two processes, say at vertices v, w, simultaneously make local token-passing transitions as the process at v sends a token in direction d with value m, and the one at w receives the token from direction e with value m. Such a transition can occur only if $(v, (\mathsf{d}, \mathsf{e}), w) \in E$.

Formally: (1) $\mathsf{AP_{sys}} := \mathsf{AP_{pr}} \times V$ is the set of *indexed atomic propositions* (because it is standard notation, we sometimes write p_i instead of (p, i)); (2) $\Sigma_{\mathsf{sys}} := \{\mathsf{int}\} \cup (\Sigma_{\mathsf{snd}} \times \Sigma_{\mathsf{rcv}} \times \Sigma_{\mathsf{val}})$ is the set of actions; (3) The set of *global states* is $S := Q^V$, i.e., all functions from V to Q (informally, if $s \in Q^V$ is a global state then $s(i)$ denotes the local state of the process with index i); (4) The set of *global initial states* S_0 consists of the unique global state $s \in Q_0^V$ such that $s(\mathsf{init}) = \iota_T$, and for all $i \neq \mathsf{init}$, $s(i) = \iota_0$; (5) 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 $i \in V$ (informally, p_i is true at s if and only if p is true at the corresponding local state of the process with index i);
(6) The *global transition relation* $\Delta \subseteq S \times \Sigma_{\mathsf{sys}} \times S$ consists of global internal transitions and global token-passing transitions (collectively called *global transitions*), defined as follows: *(6a)* The *global internal transitions* are elements of the form (s, int, s') for which there exists a process index $v \in V$ such that $s(v) \overset{\mathsf{int}}{\rightarrow} s'(v)$ is a local internal transition of $\mathbf{P}$, and for all $w \in V \setminus \{v\}$, $s(w) = s'(w)$. Such a global transition is said to *involve* v. *(6b)* The *global token-passing transitions* are elements of the form $(s, (\mathsf{d}, \mathsf{e}, \mathsf{m}), s')$ for which there exist process indices $v, w \in V$ such that: $(v, (\mathsf{d}, \mathsf{e}), w) \in E$; there is a local send transition $s(v) \overset{(\mathsf{d},\mathsf{m})}{\rightarrow} s'(v)$ and a local receive transition $s(w) \overset{(\mathsf{e},\mathsf{m})}{\rightarrow} s'(w)$ of $\mathbf{P}$; and for every $u \in V \setminus \{v, w\}$, $s'(u) = s(u)$. Such a transition is said to have v *sending in direction* d *an* m*-valued token to* w *from direction* e.

Observe that the definition above specifies that at the beginning all tokens are at the initial state of the topology. This is not a real restriction since a system which allows the tokens to start already distributed in a nondeterministic way among multiple vertices can be simulated by adding to the topology a new initial state that starts with all the tokens and from which they are later distributed. One can even accommodate many fixed initial distributions by modifying the specification formula to remove from consideration unwanted distributions.

For a global state s, let tokens(s) be the set of v such that $s(v)$ has one or more tokens. If $T = 1$, we define tokens(s) to be the vertex that has the token.

Specification Language. For the syntax and semantics of CTL^* see [9]. In this work, TL denotes a syntactic fragment of CTL^*, such as $\mathsf{CTL}^*\backslash\mathsf{X}$ (i.e., CTL^* without the "next" operator), or the fragment $\mathsf{CTL}^*_d\backslash\mathsf{X}$ of $\mathsf{CTL}^*\backslash\mathsf{X}$ in which the nesting-depth of the path quantifiers E, A is at most $d \in \mathbb{N}_0$ (see [30]).

A *partition* of an infinite path $\pi = \pi_1\pi_2\ldots$ is an infinite sequence $B_1, B_2, \ldots$ of finite intervals of $\mathbb{N}$ such that there exist integers $m_1 < m_2 < \ldots$ with $m_1 = 1$ and for all $i \in \mathbb{N}$, $B_i = [m_i, m_{i+1} - 1]$. The intervals B_i are called *blocks*.

Definition 1. *[30] For TSs $M = \langle AP, S, S_0, \Delta, \Lambda\rangle$, $M' = \langle AP, S', S'_0, \Delta', \Lambda'\rangle$ (over the same set of atomic propositions AP), and non-negative integer d, define relations $\equiv_d \subseteq S \times S'$ as follows:* (i) *$s \equiv_0 s'$ if $\Lambda(s) = \Lambda'(s')$; and* (ii) *$s \equiv_{d+1} s'$ if for every infinite path π in M from s there exists an infinite path π' in M' from s' (and vice versa) and a partition $B_1B_2\ldots$ of π and a partition $B'_1B'_2\ldots$ of π' such that for every $i \in \mathbb{N}$ and every $b \in B_i, b' \in B'_i$ we have that $\pi_b \equiv_d \pi'_{b'}$.*

In case we need to stress the LTSs, we write $\equiv_d^{M,M'}$ instead of $\equiv_d$. Say that M is *TL-equivalent* to M', denoted by $M \equiv_{\mathsf{TL}} M'$, if they agree on all TL formulas, i.e., for every TL formula φ over AP it holds that $M \models \varphi$ iff $M' \models \varphi$. The next proposition characterizes $\mathsf{CTL}^*_d\backslash\mathsf{X}$-equivalence, denoted $\equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}}$.

Proposition 1. *[30] For every integer d, TS M with a single initial state s, and TS M' with a single initial state s': $M \equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}} M'$ if and only if $s \equiv_d s'$.*

Indexed Temporal Logics (ITLs) were introduced in [11,19,18] to model specifications of systems with multiple processes. ITL formulas are built from TL formulas by adding the ability to quantify over process indices using the universal and existential process quantifiers $\forall x_{cond}$ and $\exists x_{cond}$ (generally written as Qx). Accordingly, the atoms are $\mathcal{AP} \times \mathsf{Vars}$, where $\mathsf{Vars} = \{x, y, z, \ldots\}$ is some fixed infinite set of index variables (we write p_x instead of $(p, x) \in \mathcal{AP} \times \mathsf{Vars}$). For example, the formula $\forall x \forall y_{x \neq y}.\, \mathsf{A} \neg \mathsf{F}\, c_x \wedge c_y$ specifies mutual exclusion, i.e., that it is never the case that two different processes simultaneously satisfy atom c. Syntactically, *Indexed-CTL^* formulas* are formed by adding the following to the syntax of CTL^* formulas over atomic propositions $\mathsf{AP} \times \mathsf{Vars}$: if φ is an indexed-CTL^* state (resp. path) formula then so are the formulas $\forall x_{cond}.\varphi$ and $\exists x_{cond}.\varphi$, where $x, y \in \mathsf{Vars}$, and *cond* is Boolean combination over predicates of the form true, $(x, y) \in E$, $(y, x) \in E$, and $x = y$.

Semantics. Indexed-CTL^* formulas over variables Vars and atomic propositions $\mathsf{AP}_{\mathsf{pr}}$ are interpreted over a token-passing system $\mathbf{P}^G$, where $\mathbf{P}$ has atomic propositions $\mathsf{AP}_{\mathsf{pr}}$, and $G = \langle V, E, \mathsf{init}\rangle$. A *valuation* is a function $e : \mathsf{Vars} \to V_G$. An *$x$-variant* of e is a valuation e' with $e'(y) = e(y)$ for all $y \in \mathsf{Vars} \setminus x$. First we inductively define what it means for *valuation e to satisfy cond*, written $e \models cond$: $e \models \mathsf{true}$ (for all e); $e \models x = y$ iff $e(x) = e(y)$; $e \models (x, y) \in E$ iff $(e(x), e(y)) \in E$; $e \models \neg cond$ iff $e \not\models cond$; $e \models cond \wedge cond'$ iff $e \models cond$ and $e \models cond'$.

For a TPS $\mathbf{P}^G = \langle \mathsf{AP}_{\mathsf{sys}}, \Sigma_{\mathsf{sys}}, S, S_0, \Delta, \Lambda\rangle$, a global state s, a state formula φ, and a valuation e, define $(\mathbf{P}^G, s) \models \varphi[e]$ inductively:

– $(\mathbf{P}^G, s) \models p_x[e]$ iff $p_{e(x)} \in \Lambda(s)$,
– $(\mathbf{P}^G, s) \models \mathsf{E}\,\psi[e]$ iff $(\mathbf{P}^G, \pi) \models \psi[e]$ for some infinite path π form s in $\mathbf{P}^G$,
– $(\mathbf{P}^G, s) \models \forall x_{cond}.\varphi[e]$ (resp. $(\mathbf{P}^G, s) \models \exists x_{cond}.\varphi[e]$) iff for all (resp. for some) x-variants e' of e that satisfy *cond*, it holds that $(\mathbf{P}^G, s) \models \varphi[e']$,
– $(\mathbf{P}^G, s) \models \varphi \wedge \varphi'[e]$ iff $(\mathbf{P}^G, s) \models \varphi[e]$ and $(\mathbf{P}^G, s) \models \varphi'[e]$, and
– $(\mathbf{P}^G, s) \models \neg\varphi[e]$ iff it is not the case that $(\mathbf{P}^G, s) \models \varphi[e]$.

Path formulas are interpreted similarly, but over $(\mathbf{P}^G, \pi)$, where π is an infinite path. An indexed CTL^* formula is a *sentence* if every atom is in the scope of a process quantifier. Let φ be an indexed-CTL^* state formula. For a valuation e, define $\mathbf{P}^G \models \varphi[e]$ if $(\mathbf{P}^G, s_0) \models \varphi[e]$, where s_0 is the initial state of $\mathbf{P}^G$. If φ is also a sentence, define $\mathbf{P}^G \models \varphi$ if for all valuations e (equivalently, for some valuation) it holds that $(\mathbf{P}^G, s_0) \models \varphi[e]$. Similarly, define $(\mathbf{P}^G, s) \models \varphi$ iff for all valuations (equivalently, for some valuation) $e : \mathsf{Vars} \to V_G$ it holds that $(\mathbf{P}^G, s) \models \varphi[e]$. We use the usual shorthands, e.g., $\forall x.\varphi$ is shorthand for $\forall x_{\mathsf{true}}.\varphi$.

Prenex indexed-TL is a syntactic fragment of indexed-TL in which all the processes' index quantifiers are at the front of the formula, e.g., prenex indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ consists of formulas of the form $(Q_1 x_1) \ldots (Q_k x_k)\, \varphi$ where φ is a $\mathsf{CTL}^*\backslash\mathsf{X}$ formula over atoms $\mathsf{AP} \times \{x_1, \ldots, x_k\}$, and the $Q_i x_i$s are index quantifiers. Such formulas with k quantifiers are called *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. The remainder of this paper deals with prenex indexed-$\mathsf{CTL}^*_d\backslash\mathsf{X}$.

Parameterized Model Checking Problem $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}, \mathcal{F})$. The *parameterized model checking (PMC) problem* is to decide, given $\mathbf{P} \in \mathcal{P}$ and $\varphi \in \mathcal{F}$, whether or not for all $G \in \mathcal{G}$, $\mathbf{P}^G \models \varphi$. Here $\mathcal{P}$ is a set of process templates, and $\mathcal{F}$ is a set of ITL formulas.

Cutoffs and Decidability. A *cutoff* for $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}, \mathcal{F})$ is a natural number c such that for every $\mathbf{P} \in \mathcal{P}$ and $\varphi \in \mathcal{F}$, if $\mathbf{P}^G \models \varphi$ for all $G \in \mathcal{G}$ with $|V_G| \leq c$ then $\mathbf{P}^G \models \varphi$ for all $G \in \mathcal{G}$. Note: if $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}, \mathcal{F})$ has a cutoff, then it is decidable. Note that the existence of a cutoff only implies the existence of a decision procedure. For instance, the statement "for every $k \in \mathbb{N}$, $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}, \{\forall, \exists\}^k$-$\mathsf{TL})$ has a cutoff" does not imply, a priori, that $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}, \{\forall, \exists\}^*$-$\mathsf{TL})$ is decidable.

3 Decidability Results

In this section we prove that token-passing systems have decidable PMC problem for specifications from k-indexed $\mathsf{CTL}^*_d\backslash\mathsf{X}$ for fair and direction/value-fair process templates. We begin with some definitions.

Notation. A *k-tuple over V_G*, written $\bar{g}$, denotes a tuple $(g_1, \ldots, g_k)$ of elements of V_G. We write $v \in \bar{g}$ if $v = g_i$ for some i. Given a valuation $e : \mathsf{Vars} \to V_G$, the relevant part of e for a $\mathsf{CTL}^*_d\backslash\mathsf{X}$ formula with k free variables (w.l.o.g. called $x_1, \ldots x_k$) can be described by a k-tuple $\bar{g}$ over V_G (with $g_i = e(x_i)$ for $1 \leq i \leq k$).

The restriction $\mathbf{P}^G|\bar{g}$. Fix process template $\mathbf{P}$, topology G, and nodes $\bar{g} \in V_G^k$. Define the *restriction of* $\mathbf{P}^G = \langle AP_{\mathsf{sys}}, S, S_0, \Delta, \Lambda\rangle$ *onto* $\bar{g}$, written $\mathbf{P}^G|\bar{g}$, as the LTS $(\mathsf{AP}_@, S, S_0, \Delta, L)$ over atomic propositions $\mathsf{AP}_@ = \{p@i : p \in \mathsf{AP}_{\mathsf{pr}}, i \in [k]\}$,

where for all $s \in S$ the labeling $L(s)$ is defined as follows: $L(s) := \{p@i : p_{g_i} \in \Lambda(s), i \in [k]\}$. Informally, $\mathbf{P}^G|\bar{g}$ is the LTS $\mathbf{P}^G$ with a modified labeling that, for every $g_i \in \bar{g}$, replaces the indexed atom p_{g_i} by the atom $p@i$ (i.e., process indices are replaced by their *positions* in $\bar{g}$); all other atoms are removed. Intuitively, $p@i$ means that the atom $p \in \mathsf{AP}_{\mathsf{pr}}$ holds in the process with index (i.e., at the vertex) g_i. Note that $\mathbf{P}^G$ and $\mathbf{P}^G|\bar{g}$ only differ in their labelling. It is not hard to see that given a k-indexed formula $\theta := Q_1 x_1 \dots Q_k x_k.\ \varphi$, the truth value of φ in $\mathbf{P}^G$, with respect to a valuation for $x_1, \dots x_k$ described by a k-tuple $\bar{g}$, can be deduced by reasoning instead on $\mathbf{P}^G|\bar{g}$ (since for this evaluation φ only "sees" the atomic propositions of processes in vertices in $\bar{g}$).

The valuation TS $G[\![\bar{g}]\!]$. The idea is to annotate the topology G by atoms that allow logical formulae to talk about the movement of tokens in and out of vertices in $\bar{g}$. In order to capture the directions involved in such movements, we insert new nodes in the middle of any edge of G that is incident with a vertex in $\bar{g}$. Thus, $G[\![\bar{g}]\!]$ is a TS formed as follows: i) the atoms true at v are the positions that v appears in $\bar{g}$, if any; ii) split each edge labeled (d, e) involving (one or two) vertices from $\bar{g}$ by inserting a state whose atoms label the directions to or from the vertices from $\bar{g}$ that are involved; iii) remove all edge labels.

Formally, let $G = \langle V, E, \mathsf{init}\rangle$ be a topology, and let $\bar{g}$ be a k-tuple over V. Define the *valuation TS* $G[\![\bar{g}]\!]$ as the TS $\langle \mathsf{AP}, Q, Q_0, \delta, \lambda\rangle$ where

- $\mathsf{AP} = [k] \cup \Sigma_{\mathsf{snd}} \cup \Sigma_{\mathsf{rcv}}$,
- $Q = V \cup \{[v, \mathsf{d}, \mathsf{e}, w] \mid (v, (\mathsf{d}, \mathsf{e}), w) \in E$ and either $v \in \bar{g}$ or $w \in \bar{g}\}$,
- $Q_0 = \{\mathsf{init}\}$,
- $\delta \subset Q \times Q$ is the union of $\{(v, v') : \exists \mathsf{d}, \mathsf{e}.(v, (\mathsf{d}, \mathsf{e}), v') \in E, v \notin \bar{g} \wedge v' \notin \bar{g}\}$ and $\{(v, [v, \mathsf{d}, \mathsf{e}, w]) : [v, \mathsf{d}, \mathsf{e}, w] \in Q\}$ and $\{([v, \mathsf{d}, \mathsf{e}, w], w) : [v, \mathsf{d}, \mathsf{e}, w] \in Q\}$.
- $\lambda(v) := \{i \in [k] : v = g_i\}$ (for $v \in V$); and $\lambda([v, \mathsf{d}, \mathsf{e}, w])$ is $\{\mathsf{d}\}$ if $v \in \bar{g}, w \notin \bar{g}$, is $\{\mathsf{e}\}$ if $w \in \bar{g}, v \notin \bar{g}$, and is $\{\mathsf{d}, \mathsf{e}\}$ if $v \in \bar{g}, w \in \bar{g}$.

Since Σ_{snd} and Σ_{rcv} are disjoint, the label of $(v, \mathsf{d}, \mathsf{e}, w)$ determines which of v and w is in $\bar{g}$. A valuation TS $G[\![\bar{g}]\!]$ with $|\bar{g}| = k$ is called a *k-valuation TS*. Every edge-path $\xi \in G$ naturally induces a path $\mathtt{map}(\xi)$ in $G[\![\bar{g}]\!]$. Observe that $\mathtt{map}(\xi)$ starts in a node of V_G, and if ξ is finite also ends in a node of V_G. Formally: $\mathtt{map}((v, \sigma, v'))$ is defined to be vv' if $v \notin \bar{g}$ and $v' \notin \bar{g}$, and $v \cdot [v, \sigma, v'] \cdot v'$ otherwise; and $\mathtt{map}(\xi \cdot (v, \sigma, v'))$ is defined to be $\mathtt{map}(\xi) \cdot v'$ if $v \notin \bar{g}$ and $v' \notin \bar{g}$, and is $\mathtt{map}(\xi) \cdot [v, \sigma, v'] \cdot v'$, otherwise. Note that for a path ρ in $G[\![\bar{g}]\!]$ that begins in a node of V_G (and, if ρ is finite, also ends in V_G), the set $\mathtt{map}^{-1}(\rho)$ is non-empty.

We can now define the Composition and Finiteness properties.

– Composition Property for $\langle \mathsf{TL}, \mathcal{P}, \mathcal{G}\rangle$: For every $k \in \mathbb{N}$, processes $\mathbf{X}, \mathbf{Y} \in \mathcal{P}$, topologies $G, H \in \mathcal{G}$, and k-tuples $\bar{g} \in V_G^k$, $\bar{h} \in V_H^k$: if $G[\![\bar{g}]\!] \equiv_{\mathsf{TL}} H[\![\bar{h}]\!]$ and the initial states of $\mathbf{X}$ and $\mathbf{Y}$ are bisimilar, then $\mathbf{X}^G|\bar{g} \equiv_{\mathsf{TL}} \mathbf{Y}^H|\bar{h}$. In words, the composition property states that if the initial states of $\mathbf{X}$ and $\mathbf{Y}$ are bisimilar then one can deduce the logical equivalence of the restrictions $\mathbf{X}^G|\bar{g}, \mathbf{Y}^H|\bar{h}$ from the logical equivalence of the valuation TSs $G[\![\bar{g}]\!], H[\![\bar{h}]\!]$.

– Finiteness Property for $\langle \mathsf{TL}, \mathcal{G}\rangle$: For every $k \in \mathbb{N}$, the set $\mathcal{M} := \{G[\![\bar{g}]\!] : G \in \mathcal{G}, \bar{g} \in V_G^k\}$ has only finitely many $\equiv_{\mathsf{TL}}$ equivalence classes.

Later in this section we will prove the COMPOSITION and FINITENESS properties with $\mathsf{TL} = \mathsf{CTL}^*_d\backslash\mathsf{X}$ (for fixed $d \in \mathbb{N}$), $\mathcal{P} = \mathcal{P}^{\mathsf{FD}}$. Note that if instead of using valuation TSs one uses arbitrary TSs then the finiteness property does not hold even for $\mathsf{TL} = \mathsf{CTL}^*_1\backslash\mathsf{X}$.[4] Thus, the proof of the finiteness property for $\mathsf{CTL}^*_d\backslash\mathsf{X}$ must, and does, exploit properties of valuation TSs; in particular, the fact that the number of atoms is bounded and no atom is true in more than one state of $G[\![\bar{g}]\!]$ in every path between two vertices in $\bar{g}$.

We now state the main theorem of this section.

Theorem 1. *$\mathit{PMCP}_{\mathcal{G}}(\mathcal{P}^{\mathsf{FDV}}, \{\forall,\exists\}^k\text{-}\mathit{CTL}^*_d\backslash\mathsf{X})$ is decidable for every parameterised topology $\mathcal{G}$ and every $d, k \in \mathbb{N}$.*

The proof is in two steps. First, one removes the token values by encoding them in the directions. Thus, in the statement of Theorem 1, we may replace $\mathcal{P}^{\mathsf{FDV}}$ by $\mathcal{P}^{\mathsf{FD}}$. In step two, we show that (for every $\mathcal{G}, k, d$) the PMC problem has a cutoff using the composition method, following the recipe from [2]:

Theorem 2. *If $\langle \mathit{TL}, \mathcal{P}, \mathcal{G}\rangle$ has the* COMPOSITION *property and $\langle \mathit{TL}, \mathcal{G}\rangle$ has the* FINITENESS *property, then for all $k \in \mathbb{N}$, $\mathit{PMCP}_{\mathcal{G}}(\mathcal{P}, \{\forall,\exists\}^k\text{-}\mathit{TL})$ has a cutoff.*

Proof (sketch). The truth value of a $\{\forall,\exists\}^k$-TL formula $\theta := Q_1x_1 \dots Q_kx_k.\ \varphi$ in a system $\mathbf{P}^G$ is a Boolean combination of the truth values of the (non-indexed) TL formula φ, resulting from different valuations of the variables $x_1, \dots, x_k$. By the COMPOSITION property, two different topologies G, H, with corresponding valuations $\bar{g}, \bar{h}$, that yield TL-equivalent valuation TSs will admit the same truth values of φ in $\mathbf{P}^G$, $\mathbf{P}^H$. By the FINITENESS property, all the valuation TSs fall into finitely many TL-equivalence classes. Hence, given G, evaluating θ in $\mathbf{P}^G$ amounts to evaluating a Boolean function (that depends only on $G, Q_1, \dots Q_k$) over finitely many variables (one variable for each representative valuation TS); and evaluating θ with respect to $\mathcal{G}$ amounts to evaluating a set of such functions (all using the same variables). Since there are only finitely many Boolean functions over a finite set of variables we obtain a cutoff.[5] □

3.1 The Composition Theorem

Theorem 3 (Composition). *For all $d, k \in \mathbb{N}$, topologies G, H, processes $\mathbf{X}, \mathbf{Y} \in \mathcal{P}^{\mathsf{FD}}$, $\bar{g} \in V_G^k$ and $\bar{h} \in V_H^k$: if $G[\![\bar{g}]\!] \equiv_{\mathit{CTL}^*_d\backslash\mathsf{X}} H[\![\bar{h}]\!]$ and the initial states of $\mathbf{X}$ and $\mathbf{Y}$ are bisimilar, then $\mathbf{X}^G|\bar{g} \equiv_{\mathit{CTL}^*_d\backslash\mathsf{X}} \mathbf{Y}^H|\bar{h}$.*

[4] Indeed, there are infinitely many $\mathsf{CTL}^*_1\backslash\mathsf{X}$ formulas that are pairwise logically-inequivalent. E.g., every finite word over $\{0,1\}$ can be represented as an LTS, which itself can be axiomatised by a $\mathsf{CTL}^*_1\backslash\mathsf{X}$ formula that uses the U operator.

[5] The existence of a cutoff is independent of whether $\mathcal{G}$ is computable. However, deciding whether a given number is a cutoff may not be easy. Consider for example the limited setting of [2]: there exists a computable $\mathcal{G}$ and a fixed $\mathbf{P}$ such that it is impossible, given $k, d \in \mathbb{N}$ (even fixing $d = 1$), to compute a cutoff [2]. Nonetheless, by [3], in the same setting (and we believe that also in our broader setting) one can compute a cutoff for many natural parameterized topologies $\mathcal{G}$.

Proof (sketch). The proof has the following outline. Let s_0 and s'_0 be the initial states of $\mathbf{X}^G|\bar{g}$ and $\mathbf{Y}^H|\bar{h}$, respectively. By Proposition 1, it is enough to show that if the assumption of the theorem holds then $s_0 \equiv_d s'_0$. This is done by induction on d. For stating the inductive hypothesis we first need the following definition: given a system $\mathbf{P}^G$, a function $\beta : [T] \to \text{tokens}(s)$ that maps token numbers to vertices in G is a *token assignment at* s if, for every $v \in \text{tokens}(s)$, the number of tokens mapped to v is equal to the number of tokens at v according to s, i.e., $(\mathsf{tok}^{|\beta^{-1}(v)|}, v) \in \Lambda(s)$, where Λ is the labelling of $\mathbf{P}^G$.

The dth Inductive Hypothesis. *For every global state s of $\mathbf{X}^G|\bar{g}$ and global state s' of $\mathbf{Y}^H|\bar{h}$, conclude that $s \equiv_d s'$ if the following two conditions hold:*

1. *$s(g_i) \sim s'(h_i)$ for all $i \in [k]$, and*
2. *there exists a token assignment β at s, and there exists a token assignment β' at s', such that for all $i \in [T]$ we have that $\beta(i) \equiv_d \beta'(i)$.*

The first condition says that s and s' assign bisimilar states to matching processes in $\bar{g}$ and $\bar{h}$; the second condition says that s and s' have their tokens in nodes of G, H (respectively) that are equivalent according to $\equiv_d^{G[\![\bar{g}]\!], H[\![\bar{h}]\!]}$. The theorem follows by showing that s_0, s'_0 satisfy these assumptions.

For the induction base, observe that (by the first assumption in the inductive hypothesis) s and s' assign bisimilar local states to matching processes in $\bar{g}, \bar{h}$ and thus, s, s' have the same labelling and are indistinguishable by a $\mathsf{CTL}_0^*\backslash\mathsf{X}$ formula; now apply Proposition 1.

The main work in proving the inductive step is to satisfy the second condition in the definition of $\equiv_d$ (Definition 1). This requires that, for every path π in $\mathbf{X}^G|\bar{g}$ starting in s, one can find a $(d-1)$-matching path π' in $\mathbf{Y}^H|\bar{h}$ starting at s' (and vice versa). Note that since our setup is symmetric we can ignore the "vice-versa" and only find π' given π. We construct π' using the general scheme graphically depicted in Figure 2.

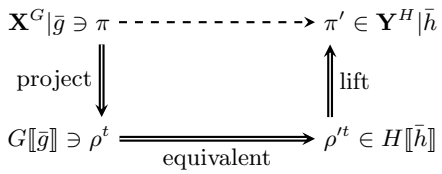

Fig. 2: Proving the Composition property via Definition 1.

First, given π, we assign to each token a unique number from 1 through T. This allows us to track the movements of individual tokens in G (according to the token-passing transitions of π) which we arbitrarily assume obey the following rule: all processes start in the initial vertex, and during a global token-passing transition, the smallest numbered token that the sending process has is the one being sent. Using this rule, for $i \in \mathbb{N}$ and $t \in [T]$ we can define the function $\mathsf{token}_i : [T] \to V_G$ such that $\mathsf{token}_i(t)$ is the vertex in which the token numbered t is located in the global state π_i. However, in order to construct π' so that it mimics π, we also need to know the directions the tokens take when entering and leaving nodes (which is information that is not explicitly present in π). The

reason this is needed is that processes in $\mathbf{X}^G|\bar{g}$ can change state based on the direction a token is sent to or received from (this is possible even if the process is direction-fair). To solve this, we arbitrarily choose some edge-path ξ in $\mathbf{X}^G|\bar{g}$ that induces π — being an edge-path, ξ contains the directions as part of each edge. As it turns out, we only need to know the directions of token-passing transitions affecting processes in $\bar{g}$. These are the send (resp. receive) directions of edges in G that start (resp. end) in a state in $\bar{g}$, and are captured by the extra nodes added to G to construct $G[\![\bar{g}]\!]$. Thus, for each $t \in [T]$ we obtain from ξ a path ρ^t in $G[\![\bar{g}]\!]$ that records the movement of token t along π.

For $t \in [T]$, by the assumption in the inductive hypothesis, $\beta(t) \equiv_d \beta'(t)$. Apply Definition 1 to the path ρ^t (that starts in $\beta(t)$) of $G[\![\bar{g}]\!]$ to get a $(d-1)$-matching path ρ'^t that starts in $\beta'(t)$ of $H[\![\bar{h}]\!]$. The paths $\rho'^1, \ldots, \rho^T$ will serve as the paths of tokens' movements for the path π'.

We construct π' by mimicking the transitions of π: an internal transition in $\mathbf{Y}^H|\bar{h}$ that involves a process $g_i \in \bar{g}$ is mimicked by a bisimilar internal transition using h_i; and a token passing transition[6] of the token number t (for $t \in [T]$) is mimicked as follows: we take the portion w of ρ^t this transition corresponds to, match it to a portion w' of ρ'^t (using the partitioning of ρ^t and ρ'^t into matching blocks), and then push the token t along an edge-path in H that induces w'.

The following lemma says that this last "pushing" step is possible.

Lemma.[7] *Let $\mathbf{P} \in \mathcal{P}^{\mathsf{FD}}$, let p, q be states of $\mathbf{P}$, and let G be a topology. Let $\rho = (v_1, (\mathsf{d}_1, \mathsf{e}_1), v_2)(v_2, (\mathsf{d}_2, \mathsf{e}_2), v_3) \ldots (v_{m-1}, (\mathsf{d}_{m-1}, \mathsf{e}_{m-1}), v_m)$ be an edge-path that is a simple path (or a simple cycle) in G, and let s be a state of $\mathbf{P}^G$ with $v_1 \in \mathrm{tokens}(s)$. There exists a finite edge-path $\alpha = (f_0, \sigma_0, f_1)(f_1, \sigma_1, f_2) \ldots (f_{h-1}, \sigma_{h-1}, f_h)$ in $\mathbf{P}^G$, with $f_0 = s$, such that:*

1. *α has $m-1$ token-passing transitions and in the ith token-passing transition v_i sends a token in direction $\mathsf{d_i}$ to v_{i+1} from direction $\mathsf{e_i}$ (for $0 \leq i < m$);*
2. *If vertex $x \in V_G$ is not on the path ρ, then no transition of α involves x.*

This lemma makes crucial use of the fact that $\mathbf{P}$ is fair and direction-fair. Fairness ensures that tokens can always be made to flow in and out of a process, and direction-fairness ensures that tokens can always flow in any given direction. Indeed, the lemma is not true without both assumptions which is the main reason that without them the composition theorem does not hold and, as we show in Section 4, the PMC problem becomes undecidable. □

3.2 Finiteness property for $\mathsf{CTL}^*_d\backslash\mathsf{X}$

Our aim in this section is to prove the Finiteness property for $\mathsf{CTL}^*_d\backslash\mathsf{X}$. We begin by recursively defining, given positive integers k, d and a k-valuation TS $G[\![\bar{g}]\!] = \langle \mathsf{AP}, Q, \{\mathsf{init}\}, \delta, \lambda \rangle$, a marking function Ξ^k_d. This function associates with each vertex $v \in Q$ a $k+1$-dimensional vector $\Xi^k_d(v)$ whose ith coordinate $\Xi^k_d(v)[i]$ is a set of strings over the alphabet $\cup_{u \in Q}\{\Xi^k_{d-1}(u)\}$. The marking function Ξ^k_d

[6] Fortunately, we only have to mimic such transitions that cross blocks in ρ^t.

[7] The full version of this lemma contains two more conclusions.

will help us later in defining the $\mathsf{CTL}^*_d\backslash\mathsf{X}$-*character* of a valuation TS, which succinctly captures the $\mathsf{CTL}^*_d\backslash\mathsf{X}$-equivalence class of this valuation TS.

Notation. The *positions* of a string v is the set $\{1,\ldots,|v|\}$ if v is finite, and $\mathbb{N}$ otherwise. A string w is *ultimately constant* if $|w| = \infty$ and $w_i = w_j$ for all $j \geq i$, for some i. Recall that the destuttering of a string is formed by removing identical consecutive letters. Define a mapping $pos_v : [|v|] \to [|v|]$ as follows: $pos_v(1) = 1$ and; for $i > 1$, $pos_v(i) := pos_v(i-1)$ if $v_{i-1} = v_i$, and otherwise $pos_v(i) := pos_v(i-1) + 1$. Intuitively, pos_v maps a position i of the string v to its corresponding position in $\mathsf{destut}(v)$. Note that the image of pos_v is of the form $[L]$ for some $L \leq |v|$. Formally, $\mathsf{destut}(v)$ is the string w of length L such that for all $i \leq L$, $w_i = v_{\min\{j:pos_v(j)=i\}}$. Thus, $v_i = w_{pos_v(i)}$ for all $i \leq L$. **The Marking Ξ^k_d.** Fix $k, d \in \mathbb{N}$, topology G, and k-tuple $\bar{g}$ over V_G. Let $G[\![\bar{g}]\!]$ be $\langle \mathsf{AP}, Q, Q_0, \delta, \lambda\rangle$. For every vertex $v \in Q$, let $v^{\rightsquigarrow}$ be the set of maximal paths in G starting in v that have no intermediate nodes in $\bar{g}$. Formally, a (finite or infinite) path $\pi = \pi_1\pi_2\ldots$ is in $v^{\rightsquigarrow}$ iff: $\pi_1 = v$ and, for all $1 < i < |\pi|$ we have $\pi_i \notin \bar{g}$ and, if π is finite then $|\pi| \geq 2$ and $\pi_{|\pi|} \in \bar{g}$. We write $v^{\rightsquigarrow 0} := \{\pi \in v^{\rightsquigarrow} \mid |\pi| = \infty\}$ for the infinite paths in $v^{\rightsquigarrow}$; also, for every $i \in [k]$, we write $v^{\rightsquigarrow i} := \{\pi \in v^{\rightsquigarrow} \mid \pi_{|\pi|} = g_i\}$ for the set of paths in $v^{\rightsquigarrow}$ that end in g_i.

In the definition below, for a (finite or infinite) path π, we write $\Xi^k_{d-1}(\pi) := \Xi^k_{d-1}(\pi_1)\Xi^k_{d-1}(\pi_2)\ldots$ for the concatenation of the $d-1$ markings of the nodes of π. We define the marking Ξ^k_d of a node inductively (on d) as follows: $\Xi^k_0(v) := \lambda(v)$ and, for $d > 0$, $\Xi^k_d(v)$ is the vector $(\Xi^k_d(v)[0],\ldots,\Xi^k_d(v)[k])$, where $\Xi^k_d(v)[i] := \cup_{\pi\in v^{\rightsquigarrow 0}}\{\mathsf{destut}(\Xi^k_{d-1}(\pi)\}$ if $i = 0$; and $\Xi^k_d(v)[i] := \cup_{\pi\in v^{\rightsquigarrow i}}\{\mathsf{destut}(\Xi^k_{d-1}(\pi')) \mid \pi' = \pi_1\ldots\pi_{|\pi|-1}\}$, for $1 \leq i \leq k$. That is, for $d = 0$, the marking $\Xi^k_d(v)$ is the label $\lambda(v)$; and for $d > 1$ the marking $\Xi^k_d(v)$ is a vector of sets of strings, where the ith coordinate of the vector contains the set of strings obtained by de-stuttering the Ξ^k_{d-1} markings of the nodes of paths in $v^{\rightsquigarrow}$ (excluding the last node if $i > 0$) that end in g_i (if $i > 0$), or never visit any node in $\bar{g}$ (if $i = 0$). Observe that, for every $0 \leq i \leq k$ and every $d > 0$, the marking $\Xi^k_d(v)[i]$ is a set of strings over the alphabet[8] $\cup_{u\in Q}\{\Xi^k_{d-1}(u)\}$, and that all strings in $\Xi^k_d(v)[i]$ start with the letter $\Xi^k_{d-1}(v)$.

Since, for all $0 \leq i \leq k$ and $d > 0$, all strings in $\Xi^k_d(v)[i]$ start with the letter $\Xi^k_{d-1}(v)$, and since $\Xi^k_d(v)[i] = \emptyset$ iff $v^{\rightsquigarrow i} = \emptyset$, we get the following lemma:

Lemma 1. *For every $d > 0$, if v, u are nodes (of possibly different k-valuation TSs) such that $\Xi^k_d(v) = \Xi^k_d(u)$, then for all $0 < j \leq d$ we have that $\Xi^k_j(v) = \Xi^k_j(u)$. If, in addition, $v^{\rightsquigarrow} \neq \emptyset$ then also $\Xi^k_0(v) = \Xi^k_0(u)$.*

The $\mathsf{CTL}^*_d\backslash\mathsf{X}$-*character* of a valuation TS. Given a k-valuation TS $G[\![\bar{g}]\!] = \langle \mathsf{AP}, Q, \{\mathsf{init}\}, \delta, \lambda\rangle$, the $\mathsf{CTL}^*_d\backslash\mathsf{X}$-*character* of $G[\![\bar{g}]\!]$ is defined as the following vector $(\langle\lambda(\mathsf{init}), \Xi^k_d(\mathsf{init})\rangle, \langle\lambda(g_1), \Xi^k_d(g_1)\rangle, \ldots, \langle\lambda(g_k), \Xi^k_d(g_k)\rangle)$ of the pairs of labels and Ξ^k_d markings of the initial state and the states in $\bar{g}$.

The following theorem relates the $\mathsf{CTL}^*_d\backslash\mathsf{X}$-character of a valuation TS and its $\mathsf{CTL}^*_d\backslash\mathsf{X}$-equivalence class.

[8] Here, the empty set $\emptyset$ is a letter in $2^{[k]}$, not to be confused with the empty string ϵ.

Theorem 4. *For every $k, d \in \mathbb{N}$, if $G[\![\bar{g}]\!], H[\![\bar{h}]\!]$ are two k-valuation TSs with the same $\mathsf{CTL}^*_d\backslash\mathsf{X}$-character, then $G[\![\bar{g}]\!] \equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}} H[\![\bar{h}]\!]$.*

Our next goal is to prove that there are (for given $k, d \in \mathbb{N}$) only finitely many $\mathsf{CTL}^*_d\backslash\mathsf{X}$-characters for all k-valuation TSs. We do this by showing that (for fixed alphabets $\Sigma_{\mathsf{snd}}, \Sigma_{\mathsf{rcv}}$) for all k, d, all k-valuation TSs $G[\![\bar{g}]\!]$, and all $v \in G$, we have that $\Xi^k_d(v)$ ranges over finitely many values. This is clearly true for $d = 0$. For $d > 0$, we prove this by defining a finite poset Υ^k_d, which depends only on k and d, and showing that $\Xi^k_d(v) \in \Upsilon^k_d$. We begin by defining a relation $\preceq$ between sets of strings.

Definition of $\preceq$. For sets of strings $X, Y \subseteq (\Sigma^+ \cup \Sigma^\omega)$, define $X \preceq Y$ if for all $x \in X$ there exists $y \in Y$ such that x is a (not necessarily proper) suffix of y.

It is easy to verify that the relation $\preceq$ is reflexive and transitive, but that it may not be antisymmetric (consider for example $X = \{b, ab\}$ and $Y = \{ab\}$).

Lemma 2. *Given a k-valuation TS $G[\![\bar{g}]\!]$, and a path $\pi_1 \dots \pi_t$ in it satisfying $\pi_l \notin \bar{g}$ for all $1 < l \leq t$, we have that: $\Xi^k_d(\pi_j)[i] \preceq \Xi^k_d(\pi_h)[i]$ for every $0 \leq i \leq k$, and $d > 0$, and $1 \leq h < j \leq t$.*

The relation $\preceq$ is antisymmetric when restricted to the domain consisting of sets of strings Z such that: *(i)* all strings in Z start with the same letter $first(Z)$ (i.e., there exists $first(Z) \in \Sigma$ such that for all $w \in Z$, $w_1 = first(Z)$); *(ii)* in every string in Z the letter $first(Z)$ appears only once (i.e., for all $w \in Z$, $i > 1$ implies $w_i \neq first(Z)$). Given an alphabet Σ, let $\mathbb{P}_\Sigma \subset 2^{\Sigma^+ \cup \Sigma^\omega}$ denote the set of all sets of strings Z (over Σ) satisfying the above two conditions. We have:

Lemma 3. *$(\mathbb{P}_\Sigma, \preceq)$ is a partially ordered set.*

Definition of $(\Upsilon^k_d, \preceq_d)$. The definition is by induction on d: for $d = 0$ we have $\Upsilon^k_0 := 2^{\mathsf{AP}}$ (recall that $\mathsf{AP} = [k] \cup \Sigma_{\mathsf{snd}} \cup \Sigma_{\mathsf{rcv}}$); and $\preceq_0$ is the transitive closure of the relation obtained by having, for every $X \in 2^{[k]}$, every $\mathsf{d} \in \Sigma_{\mathsf{snd}}$, and every $\mathsf{e} \in \Sigma_{\mathsf{rcv}}$, that: $\{\mathsf{d}\} \preceq_0 X, \{\mathsf{d}, \mathsf{e}\} \preceq_0 X, \emptyset \preceq_0 \{\mathsf{d}\}$, and $\{\mathsf{e}\} \preceq_0 \emptyset$. For $d > 0$, let: $\Upsilon^k_d = \{X \in (\mathbb{P}_{\Upsilon^k_{d-1}})^{k+1} \mid w \in X[i]$ implies $w_{j+1} \prec_{d-1} w_j$ for all $0 \leq i \leq k$ and $1 \leq j < |w|\}$ and take $\preceq_d$ to be the point-wise ordering of vectors, i.e., $X \prec_d Y$ iff $X[i] \preceq Y[i]$ for every $0 \leq i \leq k$, where $\preceq$ is the ordering defined earlier for sets of strings. Intuitively, $X \in \Upsilon^k_d$ iff every coordinate of X contains strings over the alphabet Υ^k_{d-1} that all start with the same letter and are all strictly decreasing chains of the poset $(\Upsilon^k_{d-1}, \preceq_{d-1})$. Observe that if Υ^k_{d-1} is a finite set then there are finitely many strictly decreasing chains (each of finite length) in $(\Upsilon^k_{d-1}, \preceq_{d-1})$, implying that Υ^k_d is also finite. Since Υ^k_0 is finite, we can conclude, for every $d \geq 0$, that Υ^k_d is a finite set of finite strings.

The following lemma states that for fixed k, d (recall that we assume fixed alphabets $\Sigma_{\mathsf{snd}}, \Sigma_{\mathsf{rcv}}$) the domain of Ξ^k_d is contained in Υ^k_d (and is thus finite). Note that this also implies that even though the strings in $\Xi^k_d(v)[0]$ are obtained by de-stuttering markings of infinite paths in $v^{\rightsquigarrow 0}$ they are all finite strings.

Lemma 4. *For all k, d, if v is a vertex of a k-valuation TS then $\Xi^k_d(v) \in \Upsilon^k_d$.*

We conclude with the finiteness theorem for $\mathsf{CTL}^*_d\backslash\mathsf{X}$.

Theorem 5 (Finiteness). *For every $k, d \in \mathbb{N}$, the set $\{G[\![\bar{g}]\!] : G$ is a topology, $\bar{g} \in V_G^k\}$ has only finitely many $\equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}}$ equivalence classes.*

Proof. The theorem follows immediately from the fact that the $\mathsf{CTL}^*_d\backslash\mathsf{X}$-character of a valuation TS is a finite vector, Lemma 4, and Theorem 4.

4 Undecidability

The positive decidability results appearing in Section 3 are the strongest one can hope for. Indeed, we prove that if one drops any of the restrictions that were imposed on the process template, namely of fairness and direction/value-fairness, then PMC becomes undecidable. Furthermore, these undecidability results hold even if multiple other strong restrictions are put instead (such as having a single token, having no values, having one send or one receive direction, etc.)

Our proofs reduce the non-halting problem for counter-machines (CMs) to the PMC problem. The basic encoding uses one process (the *controller*) to orchestrate the simulation and store the line number of the CM, and many *memory* processes, each having one bit for each counter. The main difficulty we face, compared to other reductions that follow this basic encoding (e.g., in [32,18,20,2]), is how to make sure that the controller's commands are executed by the memory processes given that the restrictions imposed in the theorems prevent the controller from communicating its commands to the memory processes.

Theorem 6. *Let $\mathcal{P}^{\mathsf{DV}}$ denote the set of process templates that are direction/value-fair but not necessarily fair. There exists $\mathcal{G}$ such that $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}^{\mathsf{DV}}, \{\forall\}^5\text{-}\mathsf{LTL}\backslash\mathsf{X})$ is undecidable, even if one limits the processes to have a single valueless token (i.e. $T = 1$ and $|\Sigma_{\mathsf{val}}| = 1$), and with a single receive direction (i.e., $|\Sigma_{\mathsf{rcv}}| = 1$). The same holds replacing "receive" by "send"; furthermore, $\mathcal{G}$ is computable.*

A template $\mathbf{P}$ is *receive-direction fair* if for every i-sending state q and for every $\mathsf{d} \in \Sigma_{\mathsf{rcv}}$, there is a finite i-path from q ending in a state that is ready to receive in direction d; it is *send-direction fair* if the previous condition holds with "send(ing)" replacing "receive(ing)" and Σ_{snd} replacing Σ_{rcv}; it is *direction-fair* if it is both receive- and send-direction fair. A template $\mathbf{P}$ is *value-fair* if for every i-receiving (resp. i-sending) state q, and for every token-value $\mathsf{m} \in \Sigma_{\mathsf{val}}$, there is a finite i-path from q ending in a state that is ready to receive (resp. send) value m. It is important to note that a template that is both direction-fair and value-fair is *not*, in general, direction/value-fair. The difference is that while the former can correlate values with directions, the latter cannot. For example, it may be that from every state it can only receive/send in direction a if the value of the token is 0, and receive/send in direction b only if the token value is 1. This kind of behaviour is not allowed if the template is direction/value-fair.

Theorem 7. *Let $\mathcal{P}^{\mathsf{F}}$ be the set of process templates that are fair but not necessarily direction/value-fair. There exists $\mathcal{G}$ such that $\mathsf{PMCP}_{\mathcal{G}}(\mathcal{P}^{\mathsf{F}}, \{\forall\}^5\text{-}\mathsf{LTL}\backslash\mathsf{X})$ is undecidable, even for direction fair and value fair templates with $|\Sigma_{\mathsf{rcv}}| = 1$; furthermore, $\mathcal{G}$ is computable.*

5 Discussion

The literature contains PMC decidability results of token-passing systems with a single token [18,13,16,2,3], and with multiple tokens [16,22].[9] However, the results on multiple tokens (and their proofs) only apply to linear-time specifications, and only to ring or clique network-graphs. In contrast, our results apply to branching-time specifications and to general network-graphs.

The proof of our decidability result follows the framework outlined in [2] (inspired by [13,18]) which suggests combining composition and finiteness results. Moreover, our work inherits from [13,2] the non-uniformity of the decision problem. We leave for future work the problem of calculating explicit cutoffs for concrete classes of network-graphs, as was done in [3].

Rabinovich [28] also uses the composition method for solving PMC. He considers the PMC problem for propositional modal logic assuming the parameterized network-graphs $\mathcal{G}$ have a decidable monadic-second order validity problem. While the systems in [28] are very general, the specification language, i.e., modal logic, is orthogonal to ours (e.g., it can not express liveness properties).

To the best of our knowledge, [28,2] are the only other works that use composition to establish decidability of PMC of distributed systems. While proving composition and finiteness may not be easy, we find the methodology to be elegant and powerful. Indeed, in all of these cases, no other method is known (e.g., automata, tableaux) for proving decidability. We leave for future work the intriguing problem of applying this methodology to other problems.

References

1. P.A. Abdulla, G. Delzanno, O. Rezine, A. Sangnier, and R. Traverso. On the verification of timed ad hoc networks. In *FORMATS*, pages 256–270, 2011.
2. B. Aminof, S. Jacobs, A. Khalimov, and S. Rubin. Parameterized model checking of token-passing systems. In *VMCAI*, pages 262–281, 2014.
3. B. Aminof, T. Kotek, F. Spegni, S. Rubin, and H. Veith. Parameterized model checking of rendezvous systems. In *CONCUR*, pages 109–124, 2014.
4. B. Aminof, A. Murano, S. Rubin, and F. Zuleger. Verification of asynchronous mobile-robots in partially-known environments. In *PRIMA 2015*, pages 185–200, 2015.
5. B. Aminof, A. Murano, S. Rubin, and Florian Zuleger. Automatic verification of multi-agent systems in parameterised grid-environments. In *AAMAS*, 2016.
6. B. Aminof, S. Rubin, F. Spegni, and F. Zuleger. Liveness of parameterized timed networks. In *ICALP*, pages 375–387, 2015.
7. B. Aminof, S. Rubin, and F. Zuleger. On the expressive power of communication primitives in parameterised systems. In *LPAR*, pages 313–328, 2015.
8. K. Apt and D. Kozen. Limits for automatic verification of finite-state concurrent systems. *Information Processing Letters*, pages 307–309, 1986.
9. C. Baier and J-P. Katoen. *Principles of model checking.* MIT Press, 2008.

[9] Communication in [22] is by rendezvous, powerful enough to express token-passing.

10. R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, and J. Widder. *Decidability of Parameterized Verification*. Synthesis Lectures on Distributed Computing Theory. M&C, 2015.
11. M. C. Browne, E. M. Clarke, and O. Grumberg. Reasoning about networks with many identical finite state processes. *Inf. Comput.*, pages 13–31, April 1989.
12. K. M. Chandy and J. Misra. The drinking philosophers problem. *ACM TOPLAS*, 6(4):632–646, 1984.
13. E. Clarke, M. Talupur, T. Touili, and H. Veith. Verification by network decomposition. In *CONCUR 2004*, pages 276–291, 2004.
14. S. Das. Mobile agents in distributed computing: Network exploration. *Bull. EATCS*, pages 54–69, 2013.
15. S. Demri and D. Poitrenaud. Verification of infinite-state systems. In S. Haddad, F. Kordon, L. Pautet, and L. Petrucci, editors, *Models and Analysis in Distributed Systems*, chapter 8, pages 221–269. Wiley, 2011.
16. E. A. Emerson and Vineet Kahlon. Parameterized model checking of ring-based message passing systems. In *CSL*, pages 325–339. Springer, 2004.
17. E.A. Emerson and V. Kahlon. Model checking guarded protocols. In *LICS*, pages 361–370. IEEE, 2003.
18. E.A. Emerson and K.S. Namjoshi. Reasoning about rings. In *POPL*, pages 85–94, 1995. Journal version: Int. J. Found. Comp. Sci., 14 (4), 2003.
19. E.A. Emerson and A. Sistla. Symmetry and model checking. In *CAV*, pages 463–478, 1993.
20. J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *LICS*, pages 352–359. IEEE, 1999.
21. S. Feferman and R.L. Vaught. The first-order properties of algebraic systems. *Fund. Math.*, 47:57–103, 1959.
22. S. German and A. Sistla. Reasoning about systems with many processes. *JACM*, 39(3):675–735, 1992.
23. S. Ghilardi, E. Nicolini, S. Ranise, and D. Zucchelli. Combination methods for satisfiability and model-checking of infinite-state systems. In F. Pfenning, editor, *Automated Deduction – CADE-21*, pages 362–378, 2007.
24. T. Herman. Probabilistic self-stabilization. *Inf. Process. Lett.*, 35(2):63–67, 1990.
25. A. John, I. Konnov, U. Schmid, H. Veith, and J. Widder. Parameterized model checking of fault-tolerant distributed algorithms by abstraction. In *FMCAD*, pages 201–209, 2013.
26. A. Kosowski. *Time and Space-Efficient Algorithms for Mobile Agents in an Anonymous Network*. Habilitation, U. Sciences et Technologies - Bordeaux I, 2013.
27. E. Kranakis, D. Krizanc, and S. Rajsbaum. Computing with mobile agents in distributed networks. In S. Rajasekaran and J. Reif, editors, *Handbook of Parallel Computing: Models, Algorithms, and Applications*. CRC Press, 2007.
28. A. Rabinovich. On compositionality and its limitations. *ACM TOCL*, 8(1), 2007.
29. S. Rubin. Parameterised verification of autonomous mobile-agents in static but unknown environments. In *AAMAS*, pages 199–208, 2015.
30. S. Shamir, O. Kupferman, and E. Shamir. Branching-depth hierarchies. *ENTCS*, 39(1):65 – 78, 2003.
31. S. Shelah. The monadic theory of order. *Ann. of Math.*, pages 379–419, 1975.
32. I. Suzuki. Proving properties of a ring of finite-state machines. *Inf. Process. Lett.*, 28(4):213–214, 1988.