On the existence of cutoffs for model checking disjunctive timed networks

Luca Spalazzi and Francesco Spegni

DII, Università Politecnica delle Marche

Abstract. Given a logical formula and a system described as the composition of arbitrarily many copies of some process template, the parameterized model checking problem wants to establish whether the system satisfies the formula. The focus is on the fact that the property should not depend on the actual number of participating processes. Exactly this, makes the problem equivalent to verifying an infinite state system, and thus undecidable problem in general.
Several authors identified relevant sub-classes of systems or formulae to be model checked. In this work we study the parameterized model checking problem of real-time systems against real-time temporal logics. In particular we study the possibility of finding an upper bound to the size of the system, known as *cutoff*, ensuring that adding more participants does not change the set of satisfiable formulae. A distinction exists between *dynamic cutoffs*, depending both on the process templates and the formula, and *static cutoffs*, that only consider the templates.
We start by introducing *disjunctive timed networks*. We show they do not admit static cutoffs, in general. Then we identify a subfamily that admits static cutoff, implying that the parameterized model checking problem is decidable.

1 Motivations

Many types of system are naturally described as the combination of several agents or processes cooperating in order to reach a common task (distributed algorithms, protocols, ...). When the number of participants is a parameter of the system, which is expected to behave correctly despite its exact value, we say that we are handling a *parameterized system*.

The parameterized verification problem is the problem of checking whether a specification holds in a parameterized system, for any number of participants. This is known to be an undecidable problem in general, but there exists a variety of restrictions that isolate families of systems and properties whose parameterized model checking problem is decidable.

In this work we focus on a specific subset of systems, viz. real-time systems where processes are finite state and communicate by means of transitions with disjunctive boolean guards referring to the neighbor locations . This family of systems, that we call *disjunctive timed networks*, is a combination of timed networks by Abdulla et al. [2] and Emerson and Kahlon's disjunctively guarded processes [11].

We also focus on a restriction of the parameterized verification problem via model checking of systems up to a maximum number c of participants. Such maximum number c must have the property that any system with more than c instances satisfies exactly the same logical formulae that are satisfied by systems up to c instances. If this is the case, c is called a *cutoff*. If the cutoff depends on both the structure of the processes and the formula, we call it *dynamic*, following Kaiser et al. terminology [16]. By contrast, if the cutoff only depends on the structure of the processes we call it *static*.

Contribution. In this work we first show that the family of disjunctive timed networks does not admit static cutoffs, in the most general case. Next, we show a subset of such family that instead admits such cutoff. Roughly speaking, such subset contains timed processes that are allowed to stay in their locations as long as they wish. Finally, we determine the complexity of the parameterized model checking problem of such subset.

2 Real-time temporal logics

We define iMTL, an indexed extension of real-time (linear) temporal logic MTL [9, 10]. Let us denote with $\mathbb{T}$ the *time domain*, i.e. the set of all possible time values considered by the logics, and let us consider it a *dense* set, i.e. $\mathbb{T} = \mathbb{Q}_{\geq 0}$. In the following AP denotes a finite set of atomic propositions.

Definition 1 (Syntax of iMTL). *Let $\{S_1, \ldots, S_k\}$ be a finite set of sorts and V a finite set of variables. Let p be an atomic proposition in AP.* iMTL *is the multi-sorted logic whose set of formulae is inductively defined as follows:*

$$\begin{aligned} \Phi &::= \forall i : S_l.\ \Phi \mid \phi \\ \phi &::= \top \mid p\,\langle i \rangle \mid p\,\langle S_l, a \rangle \mid \phi \ \wedge\ \phi \mid \neg\phi \mid \phi\ \mathsf{U}_{\sim c}\ \phi \end{aligned}$$

where $\sim \in \{<, \leq, \geq, >, =\}$, $l \in [1, k]$, $a \in \mathbb{N}_{>0}$, $i \in V$, $p \in AP$ and $c \in \mathbb{T}$.

Let us call the pair $\langle S_l, a \rangle$ in the above grammar a *concrete instance identifier* (or just *instance identifier*), while $i \in V$ denotes an *instance variable*. Let us assume only closed formulae, where variables are all bound to a sort. The notation $\phi[i \leftarrow a]$ represents the usual replacement operation, where all free occurrences of variable i of sort S_l in the sub-formula ϕ are replaced by the instance identifier $\langle S_l, a \rangle$. Missing boolean operators $(\vee, \rightarrow, \ldots)$ and temporal operators $(\mathsf{G}, \mathsf{F}, \mathsf{W}, \ldots)$ can be defined in the usual ways. [1]

Examples. Given a sort P representing a process, and a set of propositions $AP = \{cs, \ldots\}$, the familiar mutual exclusion property can be expressed with the formula $\forall i : P.\ \forall j : P.\ \mathsf{G}_{\geq 0} \neg(cs\,\langle i \rangle \wedge cs\,\langle j \rangle)$. The alternative formula $\forall i : P.\ \forall j : P.\ \mathsf{G}_{\geq 3} \neg(cs\,\langle i \rangle \wedge cs\,\langle j \rangle)$ expresses that the mutual exclusion property is ensured after a transient time of three time units. The formula $\forall i : P.\ \mathsf{G}_{\geq 0}(cs\,\langle i \rangle \rightarrow$

[1] The usual "next" operator (X) does not appear, since iMTL assumes a dense time domain, thus it does not make sense to talk about a next state in time.

$\mathsf{F}_{\leq 3}\neg cs\,\langle i\rangle$) is an example of *bounded response property*; namely it states that any instance of the process must exit the critical section within three time units.

In this work, executions are represented as sequences of time values together with the propositions that hold at that time. To this aim, let us call *interval sequence* over $\mathbb{T}$ any infinite sequence $I = I_0 I_1 \ldots$ of non-empty intervals of $\mathbb{T}$ with the following properties:

- *(adjacency)* the intervals $I_i = [a, b)$ and $I_{i+1} = [b, c)$ are adjacent, for all $i \geq 0$ and $a, b, c \in \mathbb{T}$ s.t. $a < b < c$;
- *(progress)* for every $t \in \mathbb{T}$, there exists $j \geq 0$ such that $t \in I_j$.

Given the set of propositions AP and sorts $S_1, \ldots, S_k$, let us call *state* any subset of the following set: $\{p\,\langle S_l, a\rangle \ \ s.t.\ p \in AP, l \in [1,k], a \in \mathbb{N}_{>0}\}$. For instance, given a sort P representing a process and a set of propositions $AP = \{a, b, c\}$, the set $\{a\,\langle P, 1\rangle, b\,\langle P, 1\rangle, c\,\langle P, 2\rangle\}$ represents the state with two copies of P: in the former a and b holds, while in the latter only c holds. Let us call *state sequence* any infinite sequence $\sigma = \sigma_0\sigma_1 \ldots$ of states. Finally, let us call *timed state sequence* a pair $\rho = (\sigma, I)$ where I is an interval sequence, and σ is a state sequence. Let us write $\rho(t)$ to denote the state σ' at time t, formally: given $\rho = (\sigma, I)$, where $\sigma = \sigma_0\sigma_1 \ldots$ and $I = I_0 I_1 \ldots$, then $\sigma' = \sigma_i$ if $t \in I_i$, for some $i \in \mathbb{N}_{\geq 0}$. Let us now introduce the satisfiability relation of iMTL.

Definition 2 (Satisfiability of iMTL).
Assume $t \in \mathbb{T}$ and let ρ be a timed state sequence. The satisfiability relation of an iMTL *formula is defined inductively on the structure of the formula itself:*

$$
\begin{array}{lll}
\rho, t \models \forall i : S_l.\ \Phi & \textit{iff} & \rho, t \models \Phi[i \leftarrow a] \textit{ for any } a \in \mathbb{N}_{>0} \\
\rho, t \models \top & & \\
\rho, t \models p\,\langle S_l, a\rangle & \textit{iff} & p\,\langle S_l, a\rangle \in \rho(t) \\
\rho, t \models \phi_1 \ \wedge\ \phi_2 & \textit{iff} & \rho, t \models \phi_1 \textit{ and } \rho, t \models \phi_2 \\
\rho, t \models \neg\phi_1 & \textit{iff} & \rho, t \not\models \phi_1 \\
\rho, t \models \phi_1\ \mathsf{U}_{\sim c}\ \phi_2 & \textit{iff} & \exists t' > t : t' \sim c \textit{ and } \rho, t' \models \phi_2 \textit{ and} \\
 & & \quad \forall t'' \in [t, t').\ \rho, t'' \models \phi_1
\end{array}
$$

In this work we will consider also the following fragments of iMTL viz.:

- iMITL. It is the restriction of iMTL to formulae where punctual intervals are not allowed (e.g. cannot use the $\mathsf{U}_{=c}$ operator).
- iUpp. It is an extension of the Uppaal specification language to sorted variables. It can be defined as the following subset of iMTL:

$$
\begin{array}{l}
\Phi ::= \forall i : S_l.\ \Phi \mid \phi \\
\phi ::= \top \mid p\,\langle i\rangle \mid p\,\langle S_l, a\rangle \mid \phi \ \wedge\ \phi \mid \neg\phi \mid \\
\qquad \mathsf{G}_{\sim c} p\,\langle i\rangle \mid \mathsf{F}_{\sim c} p\,\langle i\rangle \mid \mathsf{G}_{\sim c}(p\,\langle i\rangle \rightarrow \mathsf{F}_{\sim c} p\,\langle i\rangle) \mid \\
\qquad \mathsf{G}_{\sim c} p\,\langle S_l, a\rangle \mid \mathsf{F}_{\sim c} p\,\langle S_l, a\rangle \mid \mathsf{G}_{\sim c}(p\,\langle S_l, a\rangle \rightarrow \mathsf{F}_{\sim c} p\,\langle S_l, a\rangle)
\end{array}
$$

where $\sim \in \{<, \leq, \geq, >, =\}$, $l \in [1,k]$, $a \in \mathbb{N}_{>0}$, $i \in V$, $p \in AP$, and $c \in \mathbb{T}$. [2]

[2] Let us underline that usually the Uppaal specification logic is presented as a fragment of the real-time (branching time) temporal logic TCTL [7]. Nevertheless, the syntax of formulae is so limited to be expressible also as a fragment of MTL

Let us denote with iMTL_h iMITL_h and iUpp_h, for $h \in \mathbb{N}$, the subsets of the logics iMTL iMITL and iUpp, respectively, where formulae use at most h sorted variables. We also denote with iLTL (resp. iLTL_h) the subset of iMTL (resp. iMTL_h) using only universal time intervals $[0, \infty)$.

3 Timed automata

Assume a set of clock variables C. We call *temporal constraints* $TC(C)$ the terms of the grammar: $TC(C) ::= \top \mid \neg\, TC(C) \mid TC(C) \ \vee \ TC(C) \mid C \sim C \mid C \sim \mathbb{T}$, where $\sim \in \{<, \leq, >, \geq, =\}$ is a comparison operator and $\mathbb{T}$ is the same dense time domain introduced in Sec. 2.

Definition 3 (Timed automaton template). *A timed automaton template U_l is a tuple $\langle Q_l, \hat{q}_l, C_l, \Gamma_l, \tau_l, I_l \rangle$ where Q_l is a finite set of locations, $\hat{q}_l \in Q_l$ is a distinguished initial location, C_l is a finite set of clock variables, Γ_l is a finite set of synchronization labels, $\tau_l \subseteq Q_l \times TC(C_l) \times 2^{C_l} \times \Gamma_l \times Q_l$ is a finite set of transitions, $I_l : Q_l \to TC(C_l)$ maps locations to temporal constraints.*

In the following we write $|U_l|$ to denote the number of locations $|Q_l|$ in the template. Let us remark that the temporal constraint $I_l(q)$ for some location q will also be referred to as the *invariant* of location q.

We call *network of timed automata* any finite combination of templates, and we will call *timed network* the family of networks of arbitrary size.

Definition 4 (Network of timed automata). *Assume the TA templates $U_1, \ldots, U_k$. Let $(n_1, \ldots, n_k)$ be a tuple of positive natural numbers. Then*

$$(U_1, \ldots, U_k)^{(n_1, \ldots, n_k)}$$

is a network of timed automata *(NTA for short) denoting the asynchronous parallel composition of timed automata $U_1^1 \parallel \ldots \parallel U_1^{n_1} \parallel \ldots \parallel U_k^1 \parallel \ldots \parallel U_k^{n_k}$, such that U_l^i is the $i-th$ disjoint copy of U_l, for each $l \in [1, k]$ and $i \in [1, n_l]$.*

Definition 5 (Timed networks). *Given any set of timed automaton templates $U_1, \ldots, U_k$, it induces a* timed network *(TN for short) defined as the following family of networks of timed automata:*

$$\{(U_1, \ldots, U_k)^{(n_1, \ldots, n_k)} \ : \ n_1, \ldots, n_k \in \mathbb{N}_{>0}\}$$

Let us denote a TN with the tuple: $(U_1, \ldots, U_k)$.

The core idea of TNs with disjunctive guards is that their Γ component is a restricted (disjunctive) boolean formula allowing to look at neighbor locations before deciding to take a step.

Definition 6 (Disjunctive templates). *Given any template U_l, it is a disjunctive template iff the following holds:*

– Γ_l *is a set of boolean guards of the form:*
$\bigvee_{h\in[1,k]} \exists i : h \neq l \ \vee \ i \neq self.\ \left(q_h^1(i) \vee \cdots \vee q_h^r(i)\right)$ *where* $\{q_h^1, \ldots, q_h^r\} \subseteq Q_h$ *for every* $h \in [1,k]$ *and some* $r \in \mathbb{N}_{\geq 0}$*;*
– $I_l(\hat{q}_l) = \top$.

We call *disjunctive timed network* (or DTN) a TN made of only disjunctive templates. The formal definition of NTA operational semantics is omitted for the sake of spacee. Here we informally describe the rules, well known from the theory of (safety) timed automata [8]: *delay* transitions cause all clock variables to increase by the same amount $\delta \in \mathbb{T}$, provided that at any point in time no process falsifies a location invariant; *discrete* transition causes a single process to make a move, provided that: (i) the associated disjunctive boolean guard is satisfied by the neighbors locations, (ii) the clock constraint associated to the guard is satisfied, (iii) the moving process does not falsify the location invariant after the move, (iv) if specified, some clock variables are reset.

Let us define the parameterized model checking problem of DTN on top of the usual definitions of model checking NTAs.

Definition 7 (NTA model checking problem). *Let assume a NTA* $(U_1, \ldots, U_k)^{(n_1,\ldots,n_k)}$ *and a* iMTL *formula* Φ*. The* model checking problem *(MCP for short) is defined as follows:*

$$IN : (U_1, \ldots, U_k)^{(n_1,\ldots,n_k)}, \Phi$$
$$OUT : \mathsf{yes} \text{ or } \langle \mathsf{no}, x\rangle$$

such that the output are:

– $\langle \mathsf{no}, x\rangle$ *if* x *is an execution in it and* $x, 0 \not\models \Phi$*, and*
– yes *otherwise.*

Let us consider MTL, MITL, and Upp as the non-indexed restrictions of the logics presented in Section 2. Let us report known decidability of their MCP.

*Property 1 (***Decidability of MCP).** The MCP for MTL is undecidable [4]. The MCP for MITL is decidable, it has space complexity EXPSPACE-Complete [4], and thus it has time complexity in 2-EXPTIME. The MCP for Upp can be reduced to reachability in timed networks, which is decidable, it has space complexity PSPACE-complete [3], and thus it has time complexity in EXPTIME.

Let us now extend the model checking problem to timed networks.

Definition 8 (DTN parameterized model checking problem). *Let assume a DTN* $(U_1, \ldots, U_k)$ *and a* iMTL *formula* Φ*. Its* parameterized model checking problem *(PMCP) is defined as follows:*

$$IN : (U_1, \ldots, U_k), \Phi$$
$$OUT : \mathsf{yes} \text{ or } \langle \mathsf{no}, (n_1, \ldots, n_k), x\rangle$$

such that the output are:

– $\langle \mathsf{no}, (n_1, \ldots, n_k), x\rangle$ *if* $(U_1, \ldots, U_k)^{(n_1,\ldots,n_k)}$ *is a disjunctive NTA,* x *is an execution in it and* $x, 0 \not\models \Phi$*,*
– yes *otherwise.*

4 Cutoffs

We introduce *blueprints* to delineate the property of a tuple being a cutoff.

Definition 9. (Blueprint) *Let us call* k-blueprint *a triple* $(\mathcal{P}, k, \mathcal{F})$ *where* $k \in \mathbb{N}_{>0}$*,* $\mathcal{P}^k$ *is a family of systems with* k *templates, and* $\mathcal{F}$ *a family of temporal logic formulae. A* blueprint *is a pair:* $(\mathcal{P}, \mathcal{F}) = \{(\mathcal{P}, k, \mathcal{F}) : k \in \mathbb{N}_{>0}\}$.

In Section 2, we have seen three families of logic formulae, namely iMTL, iMITL and iUpp. In Section 3, we have introduced a family of system templates, viz. disjunctive TNs. We will use DTN^k to denote the set of disjunctive timed networks with k templates and define $\mathsf{DTN} = \bigcup_{k \in \mathbb{N}_{>0}} \mathsf{DTN}^k$. In Section 5, we will introduce a sub-family of the latter.

We assume the following partial ordering on tuples of natural numbers: for any natural k and tuples $(a_1, \ldots, a_k)$ and $(a'_1, \ldots, a'_k)$ in $\mathbb{N}^k$, we will write $(a_1, \ldots, a_k) \preceq (a'_1, \ldots, a'_k)$ iff $a_i \leq a'_i$ for all $i \in [1, k]$.

In order to say that a tuple of positive natural numbers *is a cutoff* for a blueprint, we introduce a relation between tuples of natural numbers and blueprints. One may read this property also as *the blueprint admits a cutoff.*

Definition 10. (Dynamic cutoff) *Let* $(\mathcal{P}, k, \mathcal{F})$ *be a* k*-blueprint,* $\overline{U} \in \mathcal{P}^k$ *a system template, and* $\Phi \in \mathcal{F}$ *a formula. A* dynamic cutoff *is any tuple* $\overline{c} \in \mathbb{N}^k_{>0}$*:*

$$(\forall \overline{m} \preceq \overline{c}.\ \overline{U}^{\overline{m}} \models \Phi) \Leftrightarrow (\forall \overline{n} \in \mathbb{N}^k_{>0}.\ \overline{U}^{\overline{n}} \models \Phi)$$

Definition 11. (Dynamic cutoff relation) *Let* $(\mathcal{P}, k, \mathcal{F})$ *be a* k*-blueprint. Any* $R^k_{(\mathcal{P},\mathcal{F})} \subseteq \mathbb{N}^k_{>0} \times \mathcal{P}^k \times \mathcal{F}$ *is a* dynamic cutoff relation *iff* $\overline{c}$ *is a* dynamic cutoff *w.r.t. system template* $\overline{U}$ *and formula* Φ*, for any* $(\overline{c}, \overline{U}, \Phi) \in R^k_{(\mathcal{P},\mathcal{F})}$.

We call this type of cutoff *dynamic* since it may return different values depending on both the system template *and* the formula to be verified.

Definition 12. (Static cutoff) *Let* $(\mathcal{P}, k, \mathcal{F})$ *be a* k*-blueprint and* $\overline{U} \in \mathcal{P}^k$ *a system template. A* static cutoff *is any tuple* $\overline{c} \in \mathbb{N}^k_{>0}$*:*

$$\forall \Phi \in \mathcal{F}. \left((\forall \overline{m} \preceq \overline{c}.\ \overline{U}^{\overline{m}} \models \Phi) \Leftrightarrow (\forall \overline{n} \in \mathbb{N}^k_{>0}.\ \overline{U}^{\overline{n}} \models \Phi) \right)$$

Definition 13. (Static cutoff relation) *Let* $(\mathcal{P}, k, \mathcal{F})$ *be a* k*-blueprint. Any* $R^k_{\mathcal{P}} \subseteq \mathbb{N}^k_{>0} \times \mathcal{P}^k$ *is a* static cutoff relation *iff* $\overline{c}$ *is a* static cutoff *w.r.t. system template* $\overline{U}$*, for any* $(\overline{c}, \overline{U}) \in R^k_{\mathcal{P}}$.

We call this second type of cutoff *static* since it only depends on the system template structure, thus relating the same cutoff tuple for any logical formula.

Fixing a system template and a specification, it is immediate to understand that, whenever a tuple is a cutoff for them, than any bigger tuple is also a cutoff. In other words, any system template and specification either admit infinitely many cutoffs or none. The same holds for static cutoff relations on k-blueprints.

Let us observe that by fixing both a system template and a formula, there always exists a tuple that is a cutoff for them: the idea is that either the specification does hold for any size of the system, thus the cutoff is $(1, \ldots, 1)$, or it does not hold and thus the cutoff is the smallest system size that falsifies it.

*Property 2 (***Dynamic cutoffs always exist).** Let $(\mathcal{P}, k, \mathcal{F})$ be a k-blueprint. Then there exists some dynamic cutoff relation $R^k_{(\mathcal{P},\mathcal{F})}$:

$$\forall \overline{U} \in \mathcal{P}^k.\ \forall \Phi \in \mathcal{F}.\ \exists \overline{c} \in \mathbb{N}^k_{>0}.\ (\overline{c}, \overline{U}, \Phi) \in R^k_{(\mathcal{P},\mathcal{F})}.$$

The proof of the property above reduces the problem of finding a cutoff to the PMCP. If the PMCP is decidable, we also obtain an algorithm for computing it. Nevertheless, the cutoff obtained in this way is of little practical use for verification purposes, since usually one desires to know the cutoff *to the aim* of deciding the PMCP.

*Property 3 (***Static cutoffs imply dynamic cutoffs).** Let $(\mathcal{P}, k, \mathcal{F})$ be a k-blueprint and $R^k_{\mathcal{P}} \in \mathbb{N}^k_{>0} \times \mathcal{P}^k$ a static cutoff relation. Then, there exists a dynamic cutoff relation $R^k_{(\mathcal{P},\mathcal{F})}$.

The proof is immediate, since we can define $R^k_{(\mathcal{P},\mathcal{F})} = \{(\bar{c}, \bar{U}, \Phi)\ :\ (\bar{c}, \bar{U}) \in R^k_{\mathcal{P}}, \Phi \in \mathcal{F}\}$, for any given static cutoff relation $R^k_{\mathcal{P}}$. By definition of static cutoff, it is implied that $R^k_{(\mathcal{P},\mathcal{F})}$ is a dynamic cutoff for the blueprint.

Let us underline that a cutoff relation does not require to include *all* tuples that are cutoffs for the given system template and formula. Notice also that any total and computable function $f : \mathcal{P}^k \times \mathcal{F} \to \mathbb{N}^k_{>0}$ (resp. $f : \mathcal{P}^k \to \mathbb{N}_{>0}$) returning a tuple which is a dynamic (resp. static) cutoff for the input, induces a dynamic (resp. static) cutoff relation. Indeed, for every input, it is enough to take all the tuples that are greater than the returned one to have a cutoff relation. We call any such f a *dynamic cutoff algorithm* (resp. *static cutoff algorithm*).

Definition 14. (Cutoff existence) *We say a* blueprint $(\mathcal{P}, \mathcal{F})$ admits a dynamic cutoff *iff for every k-blueprint $(\mathcal{P}, k, \mathcal{F})$ there exists a dynamic cutoff relation $R^k_{(\mathcal{P},\mathcal{F})}$. We say it admits a* static cutoff relation *iff for every k-blueprint $(\mathcal{P}, k, \mathcal{F})$ there exists a static cutoff relation $R^k_{\mathcal{P}}$.*

In the next section, we will see that the blueprint formed by disjunctive timed networks and iMTL does not admit a static cutoff.

Such negative result does not imply that all the system templates in the blueprint cannot have a static cutoff for iMTL formulae. Indeed, the existence of the static cutoff can be "recovered" for a subset of system templates. Later we are going to exploit exactly this observation and we are going to introduce a subfamily of disjunctive timed networks for which a static cutoff can be computed.

5 Cutoff theorems for timed networks

Let us introduce a witness disjunctive timed network and a family of formulae that can count the number of running processes in the system.

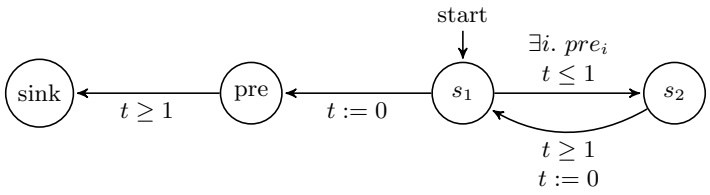

Fig. 1. Witness template for Theorem 1. Location *pre* has the invariant: $I(pre) = t \leq 1$

Theorem 1. *There exists a TA template U such that for every number $c \in \mathbb{N}_{>0}$ there exists a formula $\Phi_c \in \mathsf{iMTL}$ with the following property:*

$$(\forall c' \in \mathbb{N}_{>0} : c' \leq c.\ (U)^{(c')} \models \Phi_c) \wedge (U)^{(c+1)} \not\models \Phi_c$$

Fig. 1 shows a template P proving the above theorem, together with the following family of formulae: $\forall i : P.\ \neg(\underbrace{s_1(i)\ \mathsf{U}\ s_2(i)\ \mathsf{U}\ \ldots\ \mathsf{U}\ s_1(i)\ \mathsf{U}\ s_2(i)}_{c\ alternations})$. For every value of c, the formula Φ_c states that it is *not* possible to alternate c times between states s_1 and s_2. By simulating timed networks of growing sizes built with the given template, one sees that the formula is falsified only by networks of size c or more. Thus, for every candidate cutoff c, at least one formula Φ_{c+1} holds for networks up to size c and is falsified by bigger networks. This implies that every mapping $f : \mathsf{DTN} \to \mathbb{N}^k_{>0}$, for any $k \in \mathbb{N}_{>0}$, cannot be a static cutoff algorithm.

Lemma 1. *Let $k, h \in \mathbb{N}_{>0}$. Let $(\mathsf{DTN}, k, \mathsf{iMTL})$ be any k-blueprint. Let f be any function such that $f : \mathsf{DTN}^k \to \mathbb{N}^k_{>0}$. Then f is not a static cutoff algorithm for the k-blueprint.*

This lemma is proven by contradiction, exploiting Thm. 1. Let us now introduce DTN^- as the subfamily of DTN such that every TA template $U = \langle Q, \hat{q}, C, \Gamma, \tau, I\rangle$ satisfies the following property: for every location $q \in Q$, either $I(q) = \top$ or location q cannot appear in the transition guards of any template. Later we show that blueprint $(\mathsf{DTN}^-, \mathsf{iMTL})$ admits static cutoffs.

Let us fix a family of *static cutoff algorithm* for DTN's, inspired by the algorithm used by Emerson and Kahlon to prove the cutoff theorems for disjunctive processes [11]. The algorithm is the total mapping: $dc^h_k : {\mathsf{DTN}^-}^k \to \mathbb{N}^k_{>0}$ such that $dc^h_k((U_1, \ldots, U_k)) = (c_1, \ldots, c_k)$ and for every $l \in [1, k]$, $c_l = |U_l| + h$.

Such total mappings return cutoffs which are basically given by the number of process locations augmented by a constant factor h.

We introduce two properties of DTN^-: the former shows that adding instances does not cause to loose counterexamples, while the latter shows that for any counterexample found in systems bigger than the cutoff it is possible to find a similar counterexample in the system with as many instances as the cutoff. Together they imply that a property holds in the system whose size equals the cutoff if and only if it holds in any bigger instance.

Lemma 2 (DTN^- monotonicity).
For any $k, h \in \mathbb{N}_{>0}$, let Φ be any iMTL_h formula, let $(U_1, \ldots, U_k)^{(n_1,\ldots,n_k)}$ and $(U_1, \ldots, U_k)^{(m_1,\ldots,m_k)}$ be two NTAs in DTN^-, such that $(n_1, \ldots, n_k) \preceq (m_1, \ldots, m_k)$. Then the following holds:

$$(U_1, \ldots, U_k)^{(n_1,\ldots,n_k)} \not\models \Phi \Rightarrow (U_1, \ldots, U_k)^{(m_1,\ldots,m_k)} \not\models \Phi$$

The proof is immediate: it is enough to leave the $m_l - n_l$ added instances of template l in the initial state, and replay in the "big" system the counterexample of the "small" system.

Lemma 3 (DTN^- bounding).
For any $k, h \in \mathbb{N}_{>0}$, let Φ be any iMTL_h formula, let $(U_1, \ldots, U_k)^{(n_1,\ldots,n_k)}$ and $(U_1, \ldots, U_k)^{(c_1,\ldots,c_k)}$ be two NTAs in DTN^- such that $(c_1, \ldots, c_k) = dc_k^h((U_1, \ldots, U_k))$ and $(c_1, \ldots, c_k) \preceq (n_1, \ldots, n_k)$. Then the following holds:

$$(U_1, \ldots, U_k)^{(n_1,\ldots,n_k)} \not\models \Phi \Rightarrow (U_1, \ldots, U_k)^{(c_1,\ldots,c_k)} \not\models \Phi$$

Intuitively, we have to prove any counterexample in a "big" system has a core of processes whose behavior can be replicated exactly in a system whose size equals the cutoff. Given the falsified formula Φ with j sorted variables, the core of the counterexample is made of j processes that falsify the formula Φ itself. We call them *core processes*. In the proof we use $|U_i|$ additional processes for each template $i \in [1, k]$ to enable the moves of the core processes. We call the latter *enabling processes*. One key observation is that any step of the computation is enabled by checking that *some* process of template i is in some location q. The second key observation is that working with disjunctive guards of the family DTN^-, a process is never "forced" to leave its current location. This is the key feature that is not available, in general, in DTN processes. The latter, in fact, may have invariants that force a process to leave its current state. Given these assumptions, the core processes can replay (modulo some stuttering) the steps taken in the "big" system to falsify the formula Φ. Lemmata 2 and 3 imply the following results.

Lemma 4 (DTN^- cutoff).
For any $k, h \in \mathbb{N}_{>0}$, let Φ be any iMTL_h formula, let $(U_1, \ldots, U_k)^{(n_1,\ldots,n_k)}$ and $(U_1, \ldots, U_k)^{(c_1,\ldots,c_k)}$ be two NTAs in DTN^- such that $(c_1, \ldots, c_k) = dc_k^h((U_1, \ldots, U_k))$ and $(c_1, \ldots, c_k) \preceq (n_1, \ldots, n_k)$. Then the following holds:

$$(U_1, \ldots, U_k)^{(n_1,\ldots,n_k)} \models \Phi \Leftrightarrow (U_1, \ldots, U_k)^{(c_1,\ldots,c_k)} \models \Phi$$

Corollary 1 (Cutoff existence for $(\mathsf{DTN}^-, \mathsf{iMTL})$). *The blueprint $(\mathsf{DTN}^-, \mathsf{iMTL})$ admits a static cutoff relation.*

6 Complexity of PMCP for timed networks

The undecidability of the MCP for timed automata against MTL specifications (see Property 1) yields the following.

Corollary 2. *For any $k, h \in \mathbb{N}_{>0}$, the parameterized model checking problems of k-blueprints* $(\mathsf{DTN}, k, \mathsf{iMTL}_h)$ *are* undecidable.

Let us now lift complexity of MCP for timed automata and sub-families of iMTL to the PMCP of DTN^{-}.

Lemma 5. *Assume a k-blueprint $(\mathcal{P}, k, \mathcal{F})$ and a template cutoff algorithm $f : \mathcal{P}^k \to \mathbb{N}^k_{>0}$. Call $O(\mathsf{TIME}_{\mathsf{MCP}}(n))$ the upper bound on time computational complexity of the model checking problem with system of size n. Similarly, call $O(\mathsf{SPACE}_{\mathsf{MCP}}(n))$ the upper bound on the space complexity of the same problem. The PMCP of $(U_1, \ldots, U_k)$ and formula Φ has the following complexities:*

- $O(\mathsf{SPACE}_{\mathsf{MCP}}(\mathbf{U}^{k \cdot c_{\mathbf{U}}}))$,
- $(c_{\mathbf{U}})^k \cdot O(\mathsf{TIME}_{\mathsf{MCP}}(\mathbf{U}^{k \cdot c_{\mathbf{U}}}))$

where $\mathbf{U} = \max\{|U_1|, \ldots, |U_k|\}$, $(c_1, \ldots, c_k) = f(U_1, \ldots, U_k)$, *and* $c_{\mathbf{U}} = \max\{c_1, \ldots, c_k\}$.

Lemma 5 together with Property 1 yield the following.

Theorem 2. *The parameterized model checking problem of blueprint* $(\mathsf{DTN}^{-}$, iMITL) *has space complexity 2-*EXPSPACE*, and time complexity in 3-*EXPTIME.

Theorem 3. *The parameterized model checking problem of blueprint* $(\mathsf{DTN}^{-}$, iUpp) *has space complexity* EXPSPACE*, and time complexity in 2-*EXPTIME.

7 Conclusions and related work

In this work we studied different types of cutoffs to the aim of verifying properties on timed networks of arbitrary size. We discovered an interesting boundary between systems that only admit dynamic cutoffs w.r.t. those that can admit static cutoffs. Here we share the questions that motivated our research, as a suggestions for future works on this direction. Given a blueprint, one may ask:

RQ1 Does a dynamic cutoff relation exist that does not assume the decidability of the corresponding parameterized model checking problem?
RQ2 Does a static cutoff relation exist?
RQ3 Does a cutoff algorithm return the smallest possible cutoff, for the blueprint and specification in input?
RQ4 Is the PMCP of a blueprint with unknown cutoff decidable?

It is remarkable that, to the best of our knowledge, almost all cutoff results in literature consists of finding static cutoffs, i.e. answering RQ2. For instance, Emerson et al. [11–13] present examples of static cutoffs for (untimed) processes communicating via token passing and arranged in rings, or communicating via either conjunctive or disjunctive boolean guards and arranged in cliques. Aminof et al. [5] generalize Emerson template cutoff algorithms to a wider set of process topologies. The latter also showed that the blueprint composed of pairwise

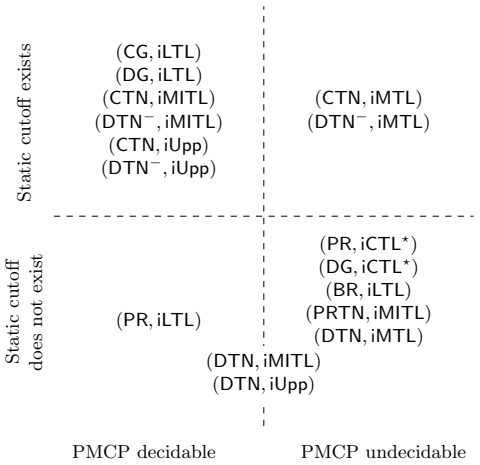

Fig. 2. Compare decidability of PMCP vs. existence of static cutoffs.

rendez-vous processes (PR for short) and iLTL_h formulae do not admit a static cutoff (even for $h = 1$). In contrast, in a previous work [17] we have shown that the blueprint $(\mathsf{CTN}, \mathsf{iMITL})$, i.e. conjunctive timed networks and iMITL specifications, admits static cutoffs and its PMCP is decidable.

A notable exception is Kaiser et al. [16], that addressed RQ1, showing that dynamic cutoffs exist for the blueprint formed by finite state programs with shared variables and reachability properties.

We provided a negative answer to RQ2 for the blueprint $(\mathsf{DTN}, \mathsf{iMITL})$ and a positive answer for $(\mathsf{DTN}^-, \mathsf{iMITL})$. To the best of our knowledge, it is still an open question whether there exist other blueprints that answer negatively to RQ2. It is also open RQ1 for PR and DTN. While RQ4 is answered positively for $(\mathsf{PR}, \mathsf{iLTL})$ [15], it is still open for $(\mathsf{DTN}, \mathsf{iMITL})$.

We underline that even blueprints admitting static cutoffs are worth investigating for existence of dynamic cutoffs considering the verified specification. It may lead to obtaining smaller cutoffs, thus reducing the number of invocations of the model checker, and perhaps reducing the complexity of the PMCP.

Figure 2 contains an overview of several blueprints, describing either timed or untimed systems. In it we compare whether each blueprint admits static cutoffs and whether its PMCP is decidable. The blueprints crossing the decidability line denote the fact that RQ4 has not been set. Blueprints $(\mathsf{CG}, \mathsf{iLTL})$ and $(\mathsf{DG}, \mathsf{iLTL})$ denote untimed systems with conjunctive and disjunctive guards, respectively, and the existence of static cutoffs for them have been proven by Emerson and Kahlon [11]. Non existence of static cutoffs for pairwise-rendezvous systems, i.e. $(\mathsf{PR}, \mathsf{iLTL})$, as well as undecidability of PMCP for $(\mathsf{PR}, \mathsf{iCTL}^\star)$ and $(\mathsf{DG}, \mathsf{iCTL}^\star)$ have been proved by Aminof et al. [5]. Decidability of PMCP for $(\mathsf{PR}, \mathsf{iLTL})$ have been proved by German and Sistla [15]. The system template PRTN represents an alternative definition of timed networks synchronizing via pairwise

rendezvous, introduced by Abdulla et al. who also proved the undecidability of their PMCP for logics capable of expressing liveness specifications [1, 2]. The system template BR represents networks of untimed processes communicating via broadcast, whose undecidability of PMCP for liveness specifications have been proven by Esparza et al. [14]. All the other results in the Figure have been proven in the current work. The PMCP of several other blueprints have been proven decidable or undecidable in literature, but are not represented in the Figure for the sake of conciseness. We leave as future work to determine whether the computed cutoffs for DTN^{-} are the minimal static cutoffs (RQ3), by appealing at existing approaches for untimed systems [5, 6].

References

1. Abdulla, P., Deneux, J., Mahata, P.: Multi-clock timed networks. In: Logic in Computer Science, 2004. Proceedings of the 19th Annual IEEE Symposium on. pp. 345–354 (July 2004)
2. Abdulla, P.A., Jonsson, B.: Model checking of systems with many identical timed processes. Theoretical Computer Science 290(1), 241–264 (Jan 2003)
3. Alur, R., Dill, D.L.: A Theory of Timed Automata. Theoretical Computer Science 126(2), 183–235 (Apr 1994)
4. Alur, R., Feder, T., Henzinger, T.: The benefits of relaxing punctuality. Journal of the ACM (JACM) (1996)
5. Aminof, B., Kotek, T., Rubin, S., Spegni, F., Veith, H.: Parameterized Model Checking of Rendezvous Systems. In: CONCUR 2014, pp. 109–124. Springer (2014)
6. Außerlechner, S., Jacobs, S., Khalimov, A.: Tight cutoffs for guarded protocols with fairness. In: VMCAI, Proceedings. pp. 476–494 (2016)
7. Behrmann, G., David, A., Larsen, K.: A tutorial on UPPAAL. In: Formal methods for the design of real-time systems, pp. 33–35. No. November, Springer (2004)
8. Bengtsson, J., Yi, W.: Timed Automata: Semantics, Algorithms and Tools. Tech. Rep. 316 (2004)
9. Bouyer, P.: Model-checking timed temporal logics. Electronic Notes in Theoretical Computer Science 231, 323–341 (2009)
10. Bouyer, P., Chevalier, F., Markey, N.: On the expressiveness of TPTL and MTL. Information and Computation 208(2), 97–116 (Feb 2010)
11. Emerson, A.E., Kahlon, V.: Reducing model checking of the many to the few. Automated Deduction-CADE-17 pp. 236–254 (2000)
12. Emerson, A.E., Trefler, R.J., Wahl, T.: Reducing Model Checking of the Few to the One pp. 94–113 (2006)
13. Emerson, E.A., Namjoshi, K.S.: On reasoning about rings. Int. J. Found. Comput. Sci. 14(4), 527–550 (2003)
14. Esparza, J., Finkel, a., Mayr, R.: On the verification of broadcast protocols. Proceedings. 14th Symposium on Logic in Comp. Sci. (July) (1999)
15. German, S.M., Sistla, A.P.: Reasoning about systems with many processes. J. ACM 39(3), 675–735 (1992)
16. Kaiser, A., Kroening, D., Wahl, T.: Dynamic cutoff detection in parameterized concurrent programs. In: CAV, Proceedings. pp. 645–659. Springer (2010)
17. Spalazzi, L., Spegni, F.: Parameterized Model-Checking of Timed Systems with Conjunctive Guards. In: Verified Software: Theories, Tools and Experiments, pp. 235–251. Springer (2014)