Parametric disjunctive timed networks

Étienne André

Université Sorbonne Paris Nord, LIPN, CNRS UMR 7030, Villetaneuse, France
Institut universitaire de France (IUF), France

Swen Jacobs

CISPA Helmholtz Center for Information Security, Germany

Engel Lefaucheux

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Abstract

We consider distributed systems with an arbitrary number of processes, modelled by timed automata that communicate through location guards: a process can take a guarded transition if at least one other process is in a given location. In this work, we introduce parametric disjunctive timed networks, where each timed automaton may contain timing parameters, i.e., unknown constants. We investigate two problems: deciding the emptiness of the set of parameter valuations for which 1) a given location is reachable for at least one process (local property), and 2) a global state is reachable where all processes are in a given location (global property). Our main positive result is that the first problem is decidable for networks of processes with a single clock and without invariants; this result holds for arbitrarily many timing parameters—a setting with few known decidability results. However, it becomes undecidable when invariants are allowed, or when considering global properties, even for systems with a single parameter. This highlights the significant expressive power of invariants in these networks. Additionally, we exhibit further decidable subclasses by restraining the syntax of guards and invariants.

2012 ACM Subject Classification Software and its engineering → Model checking

Keywords and phrases parametrised verification, parametric timed automata

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

Parametrised verification [2, 6] consists in verifying a system's behaviour across all possible configurations of a certain parameter, such as the number of processes. It is most commonly used in the context of networks of identical finite-state processes, and it involves proving that a property holds for any number of processes. This type of verification is crucial for distributed systems, where an arbitrary number of identical agents may be interacting, and ensures that system correctness is maintained no matter the scale of the system.

When timing constraints are involved, more powerful formalisms are needed. Timed automata (TAs) [7] extend finite-state automata with clocks (measuring the time elapsing, and constraining the way to remain in locations or to take transitions), and offer a powerful framework for the verification of real-time systems. Clock constraints are used to constrain the time to remain in a location ("invariant") or to take a transition ("guard").

Several works consider parametrised verification for networks of timed automata [5, 4, 1, 3], showing that it quickly hits undecidability, notably when multiple clocks are involved. Decidability in the presence of multiple clocks can be preserved by restricting the communication between processes, e.g., to communication via *location guards*: a process can take a transition guarded by a location ℓ if at least one other process currently occupies ℓ. In disjunctive timed networks, where identical processes communicate via such location guards, local reachability and safety properties can be decided for any number of clocks [32, 12], even in the presence of invariants [13].

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:25
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

When timing constants are not known with full precision (or completely unknown, e.g., at the beginning of the design phase), timed automata may become impractical. Parametric timed automata (PTAs) [8] address this issue by allowing the modelling and verification of real-time systems with unknown or variable timing constraints modelled as *timing parameters*. This flexibility enables the analysis of system behaviour across a range of parameter valuations, ensuring correctness under diverse conditions and facilitating optimization of parameters.

Common decision problems for parametric timed automata also quickly hit undecidability: emptiness of the parameter valuations set for which a given location is reachable ("reachability-emptiness"), for a single PTA, is undecidable with as few as 3 clocks and a single timing parameter (see [9] for a survey).

Contributions

In this paper, we address the verification of systems with unknown timing constants over an arbitrary number of processes. In that sense, this parametrised parametric timed setting can be seen as having parameters in two dimensions: timing parameters, and number of processes. We introduce *parametric disjunctive timed networks* (PDTNs) as networks of identical parametric timed processes resembling PTAs, and communicating via location guards. A combination of two types of parameters appears natural, especially when designing and verifying communication protocols. These protocols must function regardless of the number of participants (hence the parametric size of networks), while timing parameters allow designers to adjust critical time constraints in each process during early stages of development, where timing is of paramount importance.

Motivating example. We consider an example inspired by applications in the verification of asynchronous programs [21, 13]. In this setting, processes (or threads) can be "posted" at runtime to solve a task, and will terminate upon completing the task. Our example, depicted in Figure 1, features one clock x per process; symbols σ_i are transition labels of the automaton. An unbounded number of processes start in the initial location init. In the inner loop, a process can move to location listen in order to see whether an input channel carries data. Once it determines that this is the case (in our example this always happens after some time), it moves to location post, which gives the command to post a process that actually reads the data, and then can return to init. In the outer loop, if there is a process that gives the command to read data, i.e., a process that is in post, then another process can accept that command and move to reading. After reading for some time, the process will either determine that all the data has been read and move to done, or it will timeout and move to post to ask another process to carry on reading. However, this scheme may run into an error if there are processes in done and reading at the same time, modelled by a transition from reading to error that can only be taken if done is occupied. The time to move to error is parametric, and should be greater than the (unknown) duration p. A natural problem is to identify valuations of p for which error is unreachable regardless of the number of processes.

Problems. We focus here on the *parametrised reachability-emptiness problem*: decide the emptiness of the set of timing parameter valuations for which there exists a number of processes such that a given configuration is reachable.

We consider both *local* properties (reachability condition involving one process), and *global* properties (which can typically express the absence of deadlocks, or the fact that *all* processes must reach a given location).

We also distinguish the presence or the absence of invariants—and we will see that this makes a critical difference in decidability.

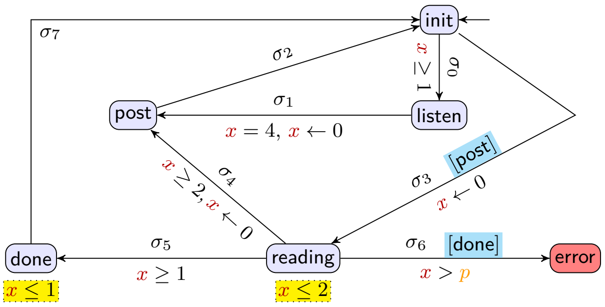


Figure 1 Asynchronous data read example (variation from [13])

In this paper, we prove several results regarding parametrised reachability-emptiness:

- undecidability of local properties for PDTNs with 1 clock, 1 parameter, with invariants (Section 5.2).
- undecidability of global properties for PDTNs with 1 clock, 1 parameter, with or without invariants (Section 5.3).
- decidability for fully parametric PDTNs with 1 parameter (Section 6.1);
- decidability for PDTNs when parameters are partitioned into lower-bound and upper-bound parameters (Section 6.2);
- decidability of local properties for PDTNs with 1 clock, arbitrarily many parameters, without invariants (Section 6.3).

The most surprising result emphasizes the high expressive power of invariants (something which has no impact in single PTAs): local properties in PDTNs are decidable without but become undecidable in their presence, for only 1 clock. In addition, both local and global properties are undecidable with invariants for a single clock and a single parameter—a setting decidable in the context of PTAs taken in isolation: that is to say, while the communication primitive is weak, it is sufficiently expressive to encode models such a 2-counter machines. We also note that local properties in 1-clock PDTNs are decidable for an unbounded number of timing parameters.

Additionally, both as a proof ingredient and as an interesting result *per se*, we show in Section 5.1 that the reachability-emptiness problem is undecidable for 1 clock, 1 parameter, with and without invariants and any *a priori* fixed number (≥ 3) of processes—even though the parametrised version of this problem, i.e., for any (non *a priori* fixed) number of processes, is decidable.

Related work. The concept of identical processes in a timed setting was mainly addressed in networks of processes that either communicate via k-wise synchronization [5, 4, 3] or via location guards [32, 12, 13]. The former model is equivalent to a variant of timed Petri nets [30, 28, 22, 17], whereas the latter would be equivalent to a form of timed Petri nets restricted to immediate observation steps (as in [20, 31]), which however have not been studied separately to the best of our knowledge.

Very few works study decidability results when combining two types of parameters, i.e., discrete (number of processes) and continuous (timing parameters). In [18, 27], security protocols are studied with unknown timing constants, and an unbounded number of participants. However, the focus is not on decidability, and the general setting is undecidable. In [14], action parameters (that can be seen as Boolean variables) and continuous timing parameters are combined (only linearly though) in an extension of PTAs; the mere emptiness of the sets of action and timing parameters for which a location is reachable is undecidable. In contrast, we exhibit in this work some decidable cases.

The closest work to ours, and presumably the only one to consider timing parameters in the setting of parametrised verification, is in [11] where parametric timed broadcast protocols (PTBPs) are introduced. Our contributions differ in the communication setting: while we consider location guards, [11] considers broadcast in cliques (in which every message reaches every process) and in reconfigurable topologies (in which the set of receivers is chosen non-deterministically). Moreover, in this work we study the power of invariants, which are absent from [11]. Our decidability results are also significantly better than in [11]: In [11], parametrised reachability-emptiness (for local properties) is undecidable for PTBPs composed of general PTAs, even with a single clock and without invariants, both in the reconfigurable semantics and in the clique semantics. The only decidable subcase (for both semantics) for reachability-emptiness in [11] is severely restricted with 3 conditions that must hold simultaneously: a single clock per process, parameters partitioned into lower-bound and upper-bound parameters, and bounded (possibly rational-valued) parameters—relaxing any of these conditions leads to undecidability. In contrast, we prove here decidability for general PDTNs with 1 clock and arbitrarily many parameters, *or* for parameters partitioned into lower-bound and upper-bound. Also note that our results do not reuse any proof ingredients from [11] due to the different communication.

Outline. We recall the necessary material in Section 2. We formalize parametric disjunctive timed networks in Section 3, and our problem in Section 4. We prove undecidability results in Section 5 and decidability results in Section 6. We conclude in Section 7.

2 Parametric timed automata

We denote by $\mathbb{N}, \mathbb{N}_{>0}, \mathbb{Z}, \mathbb{R}_{\geq 0}$ the sets of non-negative integers, strictly positive integers, integers, and non-negative reals, respectively. Let $\bowtie \in \{<, \leq, =, \geq, >\}$.

Clocks are real-valued variables that all evolve over time at the same rate. Throughout this paper, we assume a set $\mathbb{X} = \{x_1, \dots, x_H\}$ of *clocks*. A *clock valuation* is a function $\mu : \mathbb{X} \to \mathbb{R}_{\geq 0}$, assigning a non-negative value to each clock. We write $\vec{0}$ for the clock valuation assigning 0 to all clocks. Given $R \subseteq \mathbb{X}$, we define the *reset* of a valuation μ, denoted by $[\mu]_R$, as follows: $[\mu]_R(x) = 0$ if $x \in R$, and $[\mu]_R(x) = \mu(x)$ otherwise. Given a constant $d \in \mathbb{R}_{\geq 0}$, $\mu + d$ denotes the valuation s.t. $(\mu + d)(x) = \mu(x) + d$, for all $x \in \mathbb{X}$.

A *(timing) parameter* is an unknown integer-valued constant. Throughout this paper, we assume a set $\mathbb{P} = \{p_1, \dots, p_M\}$ of *parameters*. A *parameter valuation* v is a function $v : \mathbb{P} \to \mathbb{N}$.

A *constraint* C is a conjunction of inequalities over $\mathbb{X} \cup \mathbb{P}$ of the form $x \bowtie \sum_{1 \leq i \leq M} \alpha_i \times p_i + d$, with $x \in \mathbb{X}$, $p_i \in \mathbb{P}$, and $\alpha_i, d \in \mathbb{Z}$. We call d a *constant term*. Given C, we write $\mu \models v(C)$ if the expression obtained by replacing each x with $\mu(x)$ and each p with $v(p)$ in C evaluates to true. Let $\Phi(\mathbb{X} \cup \mathbb{P})$ denote the set of constraints over $\mathbb{X} \cup \mathbb{P}$. Let *True* denote the constraint made of no inequality, i.e., representing the whole set of clock and parameter valuations.

▶ **Definition 1** (PTA [8]). *A PTA $\mathcal{A}$ is a tuple $\mathcal{A} = (\Sigma, L, \ell_0, \mathbb{X}, \mathbb{P}, I, E)$, where: 1) Σ is a finite set of actions; 2) L is a finite set of locations; 3) $\ell_0 \in L$ is the initial location; 4) $\mathbb{X}$ is a finite set of clocks; 5) $\mathbb{P}$ is a finite set of parameters; 6) $I : L \to \Phi(\mathbb{X} \cup \mathbb{P})$ is the invariant, assigning to every $\ell \in L$ a constraint $I(\ell)$ over $\mathbb{X} \cup \mathbb{P}$; 7) $E \subseteq L \times \Phi(\mathbb{X} \cup \mathbb{P}) \times \Sigma \times 2^{\mathbb{X}} \times L$ is a finite set of edges $\tau = (\ell, g, a, R, \ell')$ where $\ell, \ell' \in L$ are the source and target locations, g is a constraint (called* guard*), $a \in \Sigma$, and $R \subseteq \mathbb{X}$ is a set of clocks to be reset. We say that a location ℓ* does not have an invariant *if $I(\ell) = True$.*

▶ **Definition 2** (Valuation of a PTA). *Given a PTA $\mathcal{A}$ and a parameter valuation v, we denote by $v(\mathcal{A})$ the non-parametric structure where all occurrences of a parameter p_i have been replaced by $v(p_i)$. $v(\mathcal{A})$ is a* timed automaton*.*

We recall the concrete semantics of a TA using a timed transition system (TTS).

▶ **Definition 3** (Semantics of a TA). *Given a PTA $\mathcal{A} = (\Sigma, L, \ell_0, \mathbb{X}, \mathbb{P}, I, E)$ and a parameter valuation v, the semantics of $v(\mathcal{A})$ is given by the TTS $\mathfrak{T}_{v(\mathcal{A})} = (\mathfrak{S}, \mathfrak{s}_0, \Sigma \cup \mathbb{R}_{\geq 0}, \rightarrow)$, with*

1. $\mathfrak{S} = \big\{(\ell, \mu) \in L \times \mathbb{R}_{\geq 0}^H \mid \mu \models v(I(\ell))\big\}$, $\mathfrak{s}_0 = (\ell_0, \vec{0})$,
2. *$\rightarrow$ consists of the discrete and (continuous) delay transition relations:*
   a. *discrete transitions: $(\ell, \mu) \overset{\tau}{\mapsto} (\ell', \mu')$, if $(\ell, \mu), (\ell', \mu') \in \mathfrak{S}$, and there exists $\tau = (\ell, g, a, R, \ell') \in E$, such that $\mu' = [\mu]_R$, and $\mu \models v(g)$.*
   b. *delay transitions: $(\ell, \mu) \overset{d}{\mapsto} (\ell, \mu + d)$, with $d \in \mathbb{R}_{\geq 0}$, if $\forall d' \in [0, d], (\ell, \mu + d') \in \mathfrak{S}$.*

Moreover we write $(\ell, \mu) \overset{(d,\tau)}{\longrightarrow} (\ell', \mu')$ for a combination of a delay and a discrete transition if $\exists \mu'' : (\ell, \mu) \overset{d}{\mapsto} (\ell, \mu'') \overset{\tau}{\mapsto} (\ell', \mu')$.

Given a TA A with concrete semantics $\mathfrak{T}_A$, we refer to the states of $\mathfrak{S}$ as the *concrete states* of A. A *run* of A is an alternating sequence of concrete states of A and pairs of delays and edges starting from the initial state $\mathfrak{s}_0$ of the form $(\ell_0, \mu_0), (d_0, \tau_0), (\ell_1, \mu_1), \cdots$ with $i = 0, 1, \dots$, $\tau_i \in E$, $d_i \in \mathbb{R}_{\geq 0}$ and $(\ell_i, \mu_i) \overset{(d_i,\tau_i)}{\longrightarrow} (\ell_{i+1}, \mu_{i+1})$. Given a TA A, we say that a location ℓ is *reachable* if there exists a state (ℓ, μ) that appears on a run of A.

Reachability-emptiness. Given a PTA $\mathcal{A}$ and a location ℓ, the *reachability-emptiness problem* asks whether the set of parameter valuations v such that ℓ is reachable in $v(\mathcal{A})$ is empty.

▶ **Definition 4** (L/U-PTA [24]). *An* L/U-PTA *(lower-bound/upper-bound PTA) is a PTA where $\mathbb{P}$ is partitioned into $\mathbb{P} = \mathbb{P}_L \uplus \mathbb{P}_U$, where $\mathbb{P}_L$ (resp. $\mathbb{P}_U$) denotes lower-bound (resp. upper-bound) parameters, so that each lower-bound (resp. upper-bound) parameter p_i must be such that, for every constraint $x \bowtie \sum_{1 \leq i \leq M} \alpha_i \times p_i + d$, we have: 1) $\bowtie \in \{\leq, <\}$ implies $\alpha_i \leq 0$ (resp. $\alpha_i \geq 0$), and 2) $\bowtie \in \{\geq, >\}$ implies $\alpha_i \geq 0$ (resp. $\alpha_i \leq 0$).*

A PTA is *fully parametric* whenever it has no constant term (apart from 0):

▶ **Definition 5** (Fully parametric PTA [24, Definition 4.6]). *A PTA $\mathcal{A}$ is* fully parametric *if every constraint in $\mathcal{A}$ is of the form $x \bowtie \sum_{1 \leq i \leq M} \alpha_i \times p_i$, with $p_i \in \mathbb{P}$ and $\alpha_i \in \mathbb{Z}$.*

Our subsequent undecidability proofs work by reduction from the halting problem for 2-counter machines. A deterministic 2-counter machine ("2CM") [29] has two non-negative counters $\mathcal{C}_1$ and $\mathcal{C}_2$, a finite number of states and a finite number of transitions, which can be of the form (for $l \in \{1, 2\}$): *i*) "when in state q_i, increment $\mathcal{C}_l$ and go to q_j"; or *ii*) "when in state q_i, if $\mathcal{C}_l = 0$ then go to q_k, otherwise decrement $\mathcal{C}_l$ and go to q_j".

The 2CM starts in state q_0 with the counters set to 0. The machine follows a deterministic transition function, meaning for each combination of state and counter conditions, there is exactly one action to take. The *halting problem* consists in deciding whether some distinguished state called q_{halt} can be reached or not. This problem is known to be undecidable [29].

3 Parametric disjunctive timed networks

We extend guarded TAs defined in [12] with timing parameters as in PTAs. We follow the terminology and use abbreviations from [32, 12].

▶ **Definition 6** (Guarded Parametric Timed Automaton (gPTA)). *A gPTA $\mathcal{A}$ is a tuple $\mathcal{A} = (\Sigma, L, \ell_0, \mathbb{X}, \mathbb{P}, I, E)$, where: 1) Σ is a finite set of actions; 2) L is a finite set of locations; 3) $\ell_0 \in L$ is the initial location; 4) $\mathbb{X}$ is a finite set of clocks; 5) $\mathbb{P}$ is a finite set of parameters; 6) $I : L \to \Phi(\mathbb{X} \cup \mathbb{P})$ is the invariant, assigning to every $\ell \in L$ a constraint $I(\ell)$ (called* invariant*); 7) $E \subseteq L \times \Phi(\mathbb{X} \cup \mathbb{P}) \times (L \cup \{\top\}) \times \Sigma \times 2^{\mathbb{X}} \times L$ is a finite set of edges $\tau = (\ell, g, \gamma, a, R, \ell')$ where $\ell, \ell' \in L$ are the source and target locations, g is a constraint (called* guard*), γ is the location guard, $a \in \Sigma$, and $R \subseteq \mathbb{X}$ is a set of clocks to be reset.*

Intuitively, an edge $\tau = (\ell, g, \gamma, a, R, \ell') \in E$ takes the automaton from location ℓ to ℓ'; τ can only be taken if *guard* g and *location guard* γ are both satisfied, and it resets all clocks in R. Note that satisfaction of location guards is only meaningful in a *network* of gPTAs (defined below). Intuitively, a location guard γ is satisfied if it is $\top$ or if another automaton in the network currently occupies location γ.

▶ **Example 7.** In Figure 1, the transition to error is guarded both by a guard $x > p$ and by a location guard [done]. In contrast, the location guard to done is $\top$ (and omitted in the figure), i.e., it can be taken without assumption on the location of other processes.

A gPTA is an *L/U-gPTA* if parameters are partitioned into lower-bound and upper-bound parameters (as in Definition 4). A gPTA is *fully parametric* whenever it has no constant term (apart from 0) (as in Definition 5).

Recalling the semantics of NTAs. A gPTA with $\mathbb{P} = \emptyset$ is called a guarded timed automaton (gTA) [12]. Given a gPTA $\mathcal{A}$ and a parameter valuation v, we denote by $v(\mathcal{A})$ the non-parametric structure where all occurrences of a parameter p_i have been replaced by $v(p_i)$; $v(\mathcal{A})$ is a gTA.

Let A be a gTA. We denote by A^n the parallel composition $A \parallel \cdots \parallel A$ of n copies of A, also called a *network of timed automata* (NTA) of size n. Each copy of A in the NTA A^n is called a *process*. A *configuration* $\mathfrak{c}$ of an NTA A^n is a tuple $\mathfrak{c} = \big((\ell_1, \mu_1), \dots, (\ell_n, \mu_n)\big)$, where every (ℓ_i, μ_i) is a concrete state of A. The semantics of A^n can be defined as a TTS $(\mathfrak{C}, \hat{\mathfrak{c}}, T)$, where $\mathfrak{C}$ denotes the set of all configurations of A^n, $\hat{\mathfrak{c}}$ is the unique initial configuration $(\ell_0, \mathbf{0})^n$, and the transition relation T is the union of the following delay and discrete transitions:

delay transition $\big((\ell_1, \mu_1), \dots, (\ell_n, \mu_n)\big) \xrightarrow{d} \big((\ell_1, \mu_1 + d), \dots, (\ell_n, \mu_n + d)\big)$, with $d \in \mathbb{R}_{\geq 0}$, if $\forall i \in \{1, \dots, n\}, \forall d' \in [0, d] : \mu_i + d' \models I(\ell_i)$, i.e., we can delay $d \in \mathbb{R}_{\geq 0}$ units of time if all clock invariants are satisfied until the end of the delay.

discrete transition $\big((\ell_1, \mu_1), \dots, (\ell_n, \mu_n)\big) \xrightarrow{(i,a)} \big((\ell'_1, \mu'_1), \dots, (\ell'_n, \mu'_n)\big)$ for some $i \in \{1, \dots, n\}$ if 1) $(\ell_i, \mu_i) \xrightarrow{\tau} (\ell'_i, \mu'_i)$ is a discrete transition[1] of A with $\tau = (\ell_i, g, \gamma, a, R, \ell'_i)$ for some g, γ and R, 2) $\gamma = \top$ or $\ell_j = \gamma$ for some $j \in \{1, \dots, n\} \setminus \{i\}$, and 3) $\ell'_j = \ell_j$ and $\mu'_j = \mu_j$ for all $j \in \{1, \dots, n\} \setminus \{i\}$.

That is, location guards γ are interpreted as disjunctive guards: unless $\gamma = \top$, at least one of the other processes needs to occupy location γ in order for process i to pass this guard.

We write $\mathfrak{c} \xrightarrow{d,(i,a)} \mathfrak{c}''$ for a delay transition $\mathfrak{c} \xrightarrow{d} \mathfrak{c}'$ followed by a discrete transition $\mathfrak{c}' \xrightarrow{(i,a)} \mathfrak{c}''$. Then, a *timed path* of A^n is a finite sequence $\pi = \mathfrak{c}_0 \xrightarrow{d_0,(i_0,a_0)} \cdots \xrightarrow{d_{l-1},(i_{l-1},a_{l-1})} \mathfrak{c}_l$. A timed path π of A^n is a *computation* if $\mathfrak{c}_0 = \hat{\mathfrak{c}}$. We say that $\mathfrak{c}_l$ is *reachable* in A^n.

[1] Strictly speaking, $(\ell_i, \mu_i) \xrightarrow{\tau} (\ell'_i, \mu'_i)$ is a transition of the TA obtained from A by replacing location guards with $\top$.

We write $\ell \in \mathfrak{c}$ if $\mathfrak{c} = \big((\ell_1, \mu_1), \dots, (\ell_n, \mu_n)\big)$ and $\ell = \ell_i$ for some $i \in \{1, \dots, n\}$. We say that a location ℓ is *reachable* in A^n if there exists a reachable configuration $\mathfrak{c}$ s.t. $\ell \in \mathfrak{c}$.

Given a gPTA $\mathcal{A}$, we denote by $\mathcal{A}^n$ the parallel composition $\mathcal{A} \parallel \cdots \parallel \mathcal{A}$ of n copies of $\mathcal{A}$, also called a *network of parametric timed automata* (NPTA). Given an NPTA $\mathcal{A}^n$ and a parameter valuation v, we denote by $v(\mathcal{A}^n)$ the non-parametric structure where all occurrences of a parameter p_i have been replaced by $v(p_i)$; note that $v(\mathcal{A}^n)$ is an NTA.

▶ **Definition 8.** *A given gPTA $\mathcal{A}$ induces a* parametric disjunctive timed network (PDTN) *$\mathcal{A}^\infty$, defined as the following family of NPTAs: $\mathcal{A}^\infty = \{\mathcal{A}^n \mid n \in \mathbb{N}_{>0}\}$.*

Given a parametric disjunctive timed network $\mathcal{A}^\infty$ and a parameter valuation v, $(v(\mathcal{A}))^\infty$ is a *disjunctive timed network* (DTN) [32].

Given a location ℓ of a gPTA $\mathcal{A}$, given a parameter valuation v, we say that ℓ is reachable in the DTN $(v(\mathcal{A}))^\infty$ if there exists $n \in \mathbb{N}_{>0}$ such that ℓ is reachable in $(v(\mathcal{A}))^n$.

Subclasses of PDTNs. A PDTN $\mathcal{A}^\infty$ induced by a gPTA $\mathcal{A}$ is an *L/U-PDTN* if $\mathcal{A}$ is an L/U-gPTA. Similarly, $\mathcal{A}^\infty$ is a *fully parametric PDTN* if $\mathcal{A}$ is a fully parametric gPTA.

4 Problems for parametric disjunctive timed networks

Reachability. In [12], the parametrised reachability problem (PR) consists in deciding whether, given a gTA A and a location ℓ, there exists $n \in \mathbb{N}$ such that ℓ is reachable in A^n. Here, we consider a parametric version. The emptiness extension consists in asking whether the set of timing parameter valuations for which PR holds is empty.

> **Parameterized reachability-emptiness problem (PR-e):**
> INPUT: a gPTA $\mathcal{A}$ and a location ℓ
> PROBLEM: Decide the emptiness of the set of timing parameter valuations v for which ℓ is reachable in $(v(\mathcal{A}))^\infty$.

▶ **Example 9.** In Figure 1, PR-e with error as target location does *not* hold: for $v(p) = 1$, location error can be reached for ≥ 3 processes.

Without invariants, $|L|$ is a *cutoff* (i.e., a number of processes above which the reachability is homogeneous—better cutoffs are known for local properties). Intuitively, this is because, without invariants, a single process that reaches such a location can stay there and enable the guard forever. However, with invariants, such a simple cutoff does not work (even in the absence of timing parameters), since invariants might force a process to leave a location and lots of different processes might be needed to occupy a location at different points in time.

Global reachability. Global properties refer to the numbers of processes in given locations; we express these using constraints. Formally, a *global reachability property* is defined by a constraint φ in the following grammar: $\varphi ::= \#\ell \geq 1 \mid \#\ell = 0 \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$, where $\ell \in L$ and $\#\ell$ refers to the number of processes in ℓ. Note that the "$\#\ell = 0$" term is responsible for the "global" nature of such properties, i.e., it must hold for *all* processes that they are not in ℓ. The satisfaction of a constraint φ by $\mathfrak{c} = \big((\ell_1, \mu_1), \dots, (\ell_n, \mu_n)\big)$ is defined naturally: $\mathfrak{c} \models \#\ell \geq 1$ if $\ell \in \mathfrak{c}$; $\mathfrak{c} \models \#\ell = 0$ if $\ell \notin \mathfrak{c}$; and as usual for Boolean combinations. The parametrised global reachability problem (PGR) consists in deciding whether, given a gTA A and a global reachability property φ, there exists $n \in \mathbb{N}$ such that a configuration $\mathfrak{c}$ with $\mathfrak{c} \models \varphi$ is reachable in A^n. Again, we extend this problem to the emptiness of the set of timing parameters:

Parameterized global reachability-emptiness problem (PGR-e):
INPUT: a gPTA $\mathcal{A}$ and a global reachability property φ
PROBLEM: Decide the emptiness of the set of timing parameter valuations v for which a configuration $\mathfrak{c}$ with $\mathfrak{c} \models \varphi$ is reachable in $(v(\mathcal{A}))^\infty$.

As special cases, the parametrised global reachability problem includes *control-state reachability* (where φ is $\#\ell \geq 1$ for a single location ℓ; also expressible as a PR-e problem) and *detection of timelocks that are due to location guards* (where φ is a disjunction over conjunctions of the form $\#\ell \geq 1 \land \#\ell_1 = 0 \land \cdots \land \#\ell_m = 0$, where ℓ can only be left through edges guarded by one of the $\ell_1, \ldots, \ell_m$). We will be particularly interested in the global property asking whether *all* processes reach a given final configuration (where φ is of the form $\#\ell_1 = 0 \land \cdots \land \#\ell_m = 0$ with $m = |L| - 1$), sometimes called the *TARGET problem* [19].

▶ **Example 10.** In Figure 1, consider the global property φ stating that all processes must be in error. This property can be written as $\#\mathsf{init} = 0 \land \#\mathsf{listen} = 0 \land \#\mathsf{post} = 0 \land \#\mathsf{reading} = 0 \land \#\mathsf{done} = 0$; then, PGR-e holds: no parameter valuation can allow all processes to be simultaneously in error, whatever the number of processes (this comes from the fact that the transition to error is guarded by done, so at least one process must be elsewhere).

5 Undecidability results

5.1 Fixed number of processes

We first consider a fixed number of processes; the problem addressed here is therefore not (yet) parametrised reachability-emptiness, but only reachability-emptiness. These results will be used as important proof ingredients for the results in Sections 5.2 and 5.3. (Then, the three results in Section 6 use 3 different proof techniques, completely different from this one.)

5.1.1 Undecidability for 3 processes

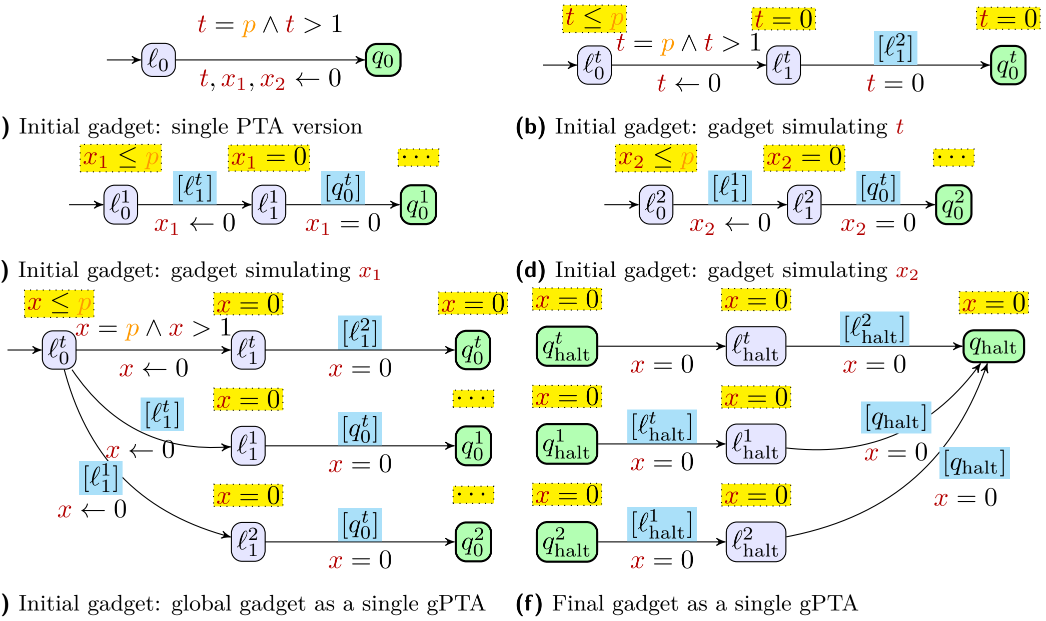
In the following, we show that the emptiness of the parameter valuations set for which a location is reachable in an NPTA made of exactly 3 processes ("reachability-emptiness") is undecidable. We first prove the result with invariants (Proposition 13), and then show it also holds without (Proposition 14). The idea is that invariants are not needed for 3 processes, but will be necessary when proving undecidability for an unbounded number of processes (Theorem 16).

Before that, we first reprove a well-known result stating that reachability-emptiness is undecidable for PTAs with 3 clocks and a single parameter. This result was already proved (with 3 or more clocks) in, e.g., [8, 16, 15] with various proof flavors. The construction we introduce in the proof of Lemma 11 will be then reused and modified in the proof of Proposition 13, which is why we give it first with full details.

▶ **Lemma 11.** *Reachability-emptiness is undecidable for PTAs with 3 clocks and 1 parameter.*

Proof. We reduce from the halting problem of 2-counter machines, which is undecidable [29]. Given a 2CM $\mathcal{M}$, we encode it as a PTA $\mathcal{A}$. Let us describe this encoding in detail, as we will modify it in the subsequent proofs.

Each state q_i of the machine is encoded as a location of the PTA, which we call q_i. The counters are encoded using clocks t, x_1 and x_2 and one integer-valued parameter p, with the following relations with the values c_1 and c_2 of counters $\mathcal{C}_1$ and $\mathcal{C}_2$: when $t = 0$, we have $x_1 = c_1$ and $x_2 = c_2$. This clock encoding is classical for integer-valued parameters, e.g., [16, 15]. The parameter typically encodes the maximal value of the counters along a run.



(a) Initial gadget: single PTA version

(b) Initial gadget: gadget simulating t

(c) Initial gadget: gadget simulating x_1

(d) Initial gadget: gadget simulating x_2

(e) Initial gadget: global gadget as a single gPTA

(f) Final gadget as a single gPTA

Figure 2 Initial and final gadgets

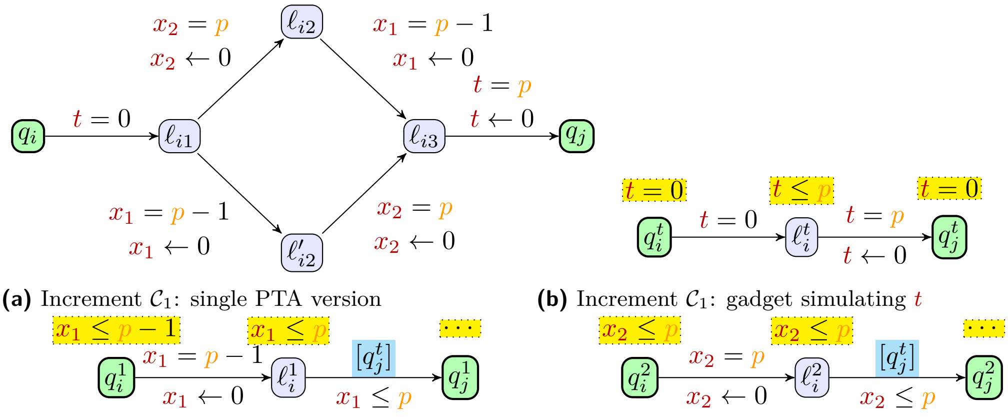
We initialize the clocks with the gadget in Figure 2a (that also blocks the case where $p \leq 1$). Note that, throughout the paper, we highlight in thick green style the locations of the PTA corresponding to a state of the 2CM (in contrast with other locations added in the encoding to maintain the matching between the clock values and the counter values). Since all clocks are initially 0, in Figure 2a clearly, when in q_0 with $t = 0$, we have $x_1 = x_2 = 0$, which indeed corresponds to counter values 0.

We now present the gadget encoding the increment instruction of $\mathcal{C}_1$ in Figure 3a. The edge from q_i to ℓ_{i1} only serves to clearly indicate the entry in the increment gadget and is done in 0 time unit. Since every edge is guarded by one equality, there are really only two timed paths that go through the gadget: one going through ℓ_{i2} and one through ℓ'_{i2}, depending on the respective order between c_1 and c_2. Observe that on both timed paths the gadget lasts exactly p time units (due to the guards and resets of t). In addition, x_2 is reset exactly when it equals p, hence its value when entering the gadget is identical to its value when reaching q_j. Therefore c_2 is unchanged. Now, x_1 is reset when it equals $p - 1$, hence its value after the gadget of duration p is incremented by 1 compared to its value when entering the gadget. Therefore c_1 is incremented by 1 when reaching q_j, as expected.

Let us now consider the 0-test and decrement gadget. Decrement is done similarly to increment, by replacing guards $x_1 = p - 1$ with $x_1 = p + 1$, as shown in Figure 4a. In addition, the 0-test is obtained by simply testing that $x_1 = 0$ whenever $t = 0$ (which ensures that $c_1 = 0$), which is done on the guard from q_j to ℓ_{k1}; we then force exactly p time units to elapse (and reset each clock when it reaches p), which means that the values of the clocks when leaving the gadget are identical to their value when entering. This is not strictly speaking needed here, but this time elapsing will simplify the proof of Proposition 13. Dually, the guard from q_i to ℓ_{i1} ensures that decrement is done only when the counter is not null.

All those gadgets also work for counter $\mathcal{C}_2$ by swapping x_1 and x_2.

The actions associated with the edges do not matter; we can assume a single action a on all edges (omitted in all figures).



(a) Increment $\mathcal{C}_1$: single PTA version

(b) Increment $\mathcal{C}_1$: gadget simulating t

(c) Increment $\mathcal{C}_1$: gadget simulating x_1

(d) Increment $\mathcal{C}_1$: gadget simulating x_2

Figure 3 Increment gadget

We now prove that the machine halts iff there exists a parameter valuation v such that $v(\mathcal{A})$ reaches location q_{halt}. First note that if $p \le 1$ the initial gadget cannot be passed, and so the machine does not halt. Assume $p > 1$. Consider two cases:

1. either the value of the counters is not bounded (and the 2CM does not halt). Then, for any parameter valuation, at some point during an increment of, say, $\mathcal{C}_1$ we will have $c_1 > p$ and hence $x_1 > p - 1$ when taking the edge from ℓ_{i2} to ℓ_{i3} and the PTA will be blocked. Therefore, there exists no parameter valuation for which the PTA can reach q_{halt}.
2. or the value of the counters remains bounded. Let c_{max} be their maximal value. Let us consider two subcases:
   a. either the machine reaches q_{halt}: in that case, if $c_{max} \le p$, then the PTA valuated with such parameter valuations correctly simulates the machine, yielding a (unique) run reaching location q_{halt}.
   b. or the machine does not halt. Then again, for a sufficiently large parameter valuation (i.e., $p > c_{max}$), the machine is properly simulated, and since the machine does not halt, then the PTA never reaches q_{halt}. For other values of p, the machine will block at some point in an increment gadget, because p is not large enough and the guard in the increment gadget cannot be satisfied.

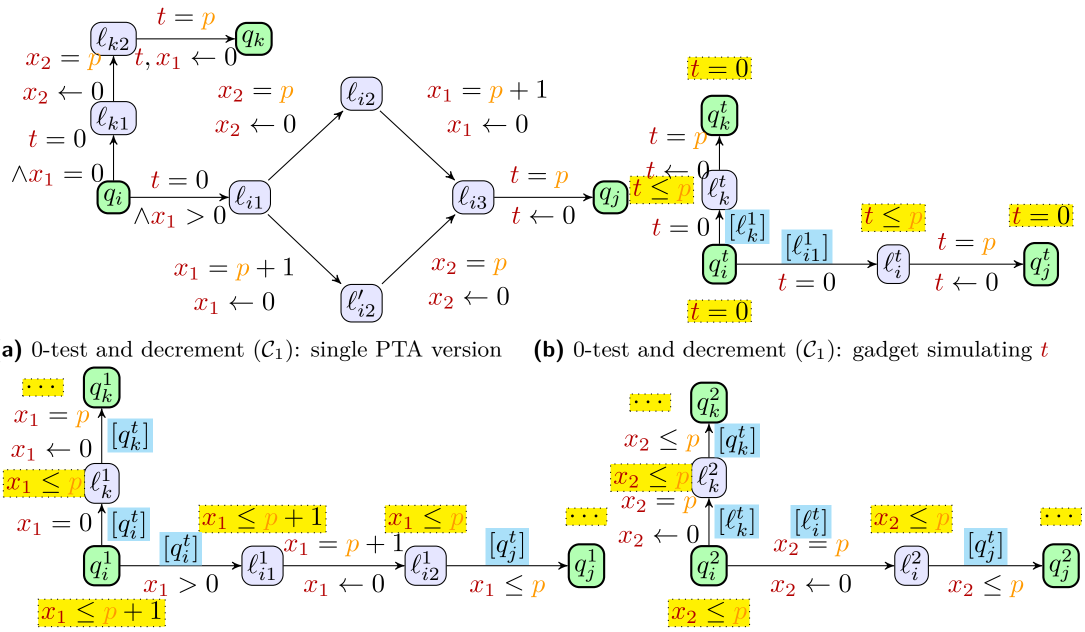
Hence the machine halts iff there exists a parameter valuation v such that $v(\mathcal{A})$ reaches q_{halt}. ◀

▶ Remark 12. Observe that the proof of Lemma 11 does not require invariants, so undecidability holds without invariants as well—a result known since [8].

We now prove undecidability of the reachability-emptiness in NTA with exactly 3 processes, by rewriting the 2CM encoding from the former proof.

▶ **Proposition 13.** *Assume a gPTA $\mathcal{A}$ with invariants, with a single clock and a single parameter, and ℓ one of its locations. Deciding the emptiness of the set of timing parameter valuations v for which ℓ is reachable in the NTA $v(\mathcal{A})^3$ is undecidable, for all such $\mathcal{A}$.*

Proof. The proof main technicality is to rewrite the 2CM encoding of Lemma 11 (made of a PTA with 3 clocks) using 3 processes with 1 clock each. This is not trivial as the communication model between our gPTAs is rather weak. Recall that the parameter encodes



(a) 0-test and decrement ($\mathcal{C}_1$): single PTA version **(b)** 0-test and decrement ($\mathcal{C}_1$): gadget simulating t

(c) 0-test and decrement ($\mathcal{C}_1$): gadget simulating x_1 **(d)** 0-test and decrement ($\mathcal{C}_1$): gadget simulating x_2

Figure 4 0-test and decrement gadget

typically the maximal valuation of the counters, and can therefore be arbitrarily large; we can assume without loss of generality that $p > 1$. Here, we use three different gPTAs (each featuring a single clock) synchronizing together; of course, strictly speaking, we can use only a single structure, but with three "subparts", and we ensure that each of the 3 processes will go into a different subpart of the gPTA. This is ensured by the initial gadget in Figure 2e: in order for process 1 to reach location q_0^t, then, due to the location guards, exactly one other process must have selected the second subpart (location ℓ_1^1) while the third process must have selected the third subpart (location ℓ_1^2).

Each instruction of the 2CM is encoded into a gadget connected with each other, for each of the three subparts.

In the rest of the proof, we therefore give, for each instruction of the 2CM, one gadget for each of the three subparts. Obviously, the (unique) gPTA features a single clock ("x" in Figure 2e); however, for sake of readability, and for consistency with the encoding given in Lemma 11, we use clocks with different names in the 3 subparts: that is, the initial gadget in Figure 2e can be described by three gadgets for the three subparts, given in Figures 2b–2d, with clock names t, x_1 and x_2. Also recall that from Definition 6, clocks are not shared—they cannot be read from another process.

Let us consider the increment gadget: the decomposition into three "subparts" (for each of the three processes) is given in Figures 3b–3d. An invariant "$\cdots$" denotes the fact that the actual invariant is given in the "next" gadget starting from that location. Note that it takes exactly p time units to move from q_i^t to q_j^t. In addition, the outgoing transition from a given q_i^t is always done in 0-time, which is enforced by the invariant $t = 0$; therefore, the location guard $[q_j^t]$ together with the guard $x_2 \leq p$ is exactly equivalent to $t = p \wedge x_2 \leq p$, forcing the subparts simulating x_1 and x_2 to properly synchronize with the subpart simulating t (or getting stuck forever in ℓ_i^1 or ℓ_i^2 and blocking the whole computation). Therefore, all three gadgets synchronize as expected, simulating the original increment gadget in Figure 3a.

Let us now consider the 0-test and decrement gadget: the decomposition into three "subparts" (for each of the three processes) is given in Figures 4b–4d. This time, compared to the former gadgets, we need a better synchronization between gadgets to ensure the correct branching depending on the value of c_1. If $c_1 = 0$, then $x_1 = 0$ when $t = 0$, and therefore the process simulating x_1 can move in 0-time to ℓ^1_k, since q^t_i is occupied; then, again in 0-time, the process simulating t can move to ℓ^t_k, ensuring the correct synchronization between both processes in the correct branch. Note that, because of the invariant in q^t_i, the process simulating t can only stay 0 time unit in q^t_i, and therefore in the process simulating x_1 the guard $x_1 = 0$ together with location guard $[q^t_j]$ is exactly equivalent to $x_1 = 0 \wedge t = 0$, which correctly encodes the 0-test. Conversely, if $c_1 > 0$, then the process simulating x_1 can move in 0-time to ℓ^1_{i1}, since q^t_i is occupied; then, again in 0-time, the process simulating t can move to ℓ^t_i. Similarly, the process simulating x_2 follows the right branch thanks to location guards $[\ell^t_i]$ and $[\ell^t_k]$. The rest of the decrement part (from i to j) works similarly to the increment gadget, and the correctness argument is the same. Recall that the rest of the 0-test gadget forces time to elapse for p units, without modifying the value of the three clocks when leaving the gadget (except of course for x_1 in case of decrement). In particular, in the gadget simulating x_2 (Figure 4d), x_2 has exactly the same value when entering q^2_k as when entering q^2_i due to location guard $[q^t_k]$, and the intermediate location resetting x_2 when it is exactly p; the same holds in the decrement part of the gadget.

In the encoding in the proof of Lemma 11, we showed that the 2CM reaches q_{halt} iff the location q_{halt} is reachable. Here, due to the split of the encoding into three subparts, we need a final gadget, given in Figure 2f (we assume the clock is reset when entering locations q^t_{halt}, q^1_{halt} and q^2_{halt}). In order for the first process to reach the (new) location q_{halt}, we need all three processes to reach their respective final location (i.e., q^t_{halt}, q^1_{halt} and q^2_{halt}) together, which is easily achieved thanks to the various location guards done in 0-time. (Note that this gadget actually allows *all* processes to reach q_{halt}; so far, we only need one process to do so.)

Now, first note that the subpart simulating t progresses in its gadgets, only synchronizing with the two other processes in the initial and final gadgets, as well as in the 0-test and decrement. In contrast, the two subparts simulating x_1 and x_2 constantly synchronize with the location guards from the first subpart; while nothing forces them to take these transitions (notably those guarded by $[q^t_j]$), failing in taking such a transition will immediately lead to a timelock due to the invariants and to the fact that the next time such a location guard will be available is necessarily in p time units. More in detail, recall that any guard location (location used in a location guard) q^t_i in the subpart simulating t has an invariant $t = 0$, and only outgoing transitions guarded with $t = 0$. In addition, at least p time units must elapse between two guard locations in the subpart simulating t: so, due to the invariants in the two other subparts, failing to take a location guard renders impossible to take it the next time the first subpart will be in a guard location.

For all these reasons, if one process reaches q_{halt}, then from Figure 2f, two other processes must have traversed the two other subparts (simulating x_1 and x_2) correctly, by synchronizing with the first process. Hence, the 2CM is correctly simulated.

The rest of the reasoning follows the proof of Lemma 11: the 2CM halts iff there exists a parameter valuation v such that $v(\mathcal{A})^3$ reaches q_{halt}. ◀

We show that the absence of invariants does not avoid undecidability, for exactly 3 processes.

▶ **Proposition 14.** *Assume a gPTA $\mathcal{A}$ with a single clock and a single parameter, and ℓ one of its locations. Deciding the emptiness of the set of timing parameter valuations v for which*

ℓ is reachable in the NTA $v(\mathcal{A})^3$ is undecidable, even without invariants, for all such $\mathcal{A}$.

Proof sketch. The idea is that, if a process "chooses" to not take some outgoing transition in the absence of invariants, then the whole computation is stuck, and q_{halt} is unreachable. Put it differently, while the absence of invariants may add some runs, none of these runs can reach q_{halt}, and the final correctness argument of the proof of Proposition 13 still holds. See Appendix A.1. ◀

5.1.2 Undecidability for any fixed number of processes (with or without invariants)

By modifying the constructions in the proof of Proposition 14, it is easy to show that, *for any fixed number of processes $n \geq 3$*, the reachability-emptiness problem is undecidable, even without invariants. Note that our construction differs for any $n \geq 3$, and hence this does not prove the undecidability of the parametrised-reachability-emptiness problem (which is actually *decidable* without invariants, see Theorem 20).

▶ **Proposition 15.** *Let $\mathcal{A}$ be a gPTA, and ℓ one of its locations. For any $n \geq 3$, deciding the emptiness of the set of timing parameter valuations v for which ℓ is reachable in the NTA $v(\mathcal{A})^n$ is undecidable, even when $\mathcal{A}$ contains a single clock and a single parameter without invariants.*

Proof sketch. The idea is to reuse the construction of the proof of Proposition 14, by "killing" any process except 3 by sending them to "sink" locations. See Appendix A.2. ◀

5.2 Parameterized reachability-emptiness with invariants

Let us now consider the parametrised reachability-emptiness problem in PDTNs. In the absence of invariants, the 2CM encoding used in the previous proofs would become incorrect for a parametric number of processes: recall that we avoided the use of invariants by letting the system get stuck if a transition is not taken at the "right" time. However, in a system with a parametric number of processes, a single process that gets stuck will not cause the whole computation to get stuck, since another process may have taken the transition at the "right" time. Moreover, a process that gets stuck will ruin the synchronization that guarantees simulation of the 2CM, since now taking a transition with location guard $[q_i^t]$ will be possible at any time after the process gets stuck in q_i^t. We now prove that undecidability holds however *with* invariants.

▶ **Theorem 16.** *PR-emptiness is undecidable for general PDTNs, even with a single clock and a single parameter (with invariants).*

Proof sketch. We rely on the construction given in the proof of Proposition 13, and we show that any additional processes will just simulate one of the existing three processes, due to the presence of invariants. Most importantly, no process can remain "idle" in a location forever, and therefore the location guards still guarantee that the 2CM is simulated correctly. See Appendix A.3. ◀

5.3 Parameterized global reachability-emptiness

We show here that our former constructions can get rid of invariants providing we consider *global* properties. That is, without invariants, global properties make parametrised reachability-emptiness undecidable while it is decidable for local properties (see Section 6.3).

▶ **Theorem 17.** *PGR-emptiness is undecidable for general PDTNs, even with a single clock and a single parameter, with or without invariants.*

Proof sketch. The proof technical idea is that we ask whether *all* processes can reach the target location q_{halt}, which can only be done if the 2CM was properly simulated. See Appendix A.4. ◀

6 Decidability results

We will first show that, for two subclasses of PDTNs, i.e., fully parametric PDTNs with a single parameter (Section 6.1) and L/U-PDTNs (Section 6.2), deciding PR-emptiness (resp. PGR-emptiness) is equivalent to checking parametrised reachability (resp. PGR) in a non-parametric DTN—which is decidable. The third positive result, addressing one clock, arbitrarily many parameters and no invariants (Section 6.3), is more involved and requires different techniques.

6.1 Fully parametric PDTNs with a single parameter

▶ **Theorem 18.** *PR-emptiness (resp. PGR-emptiness) for fully parametric PDTNs with 1 parameter and arbitrarily many clocks is equivalent to PR (resp. PGR) for DTNs.*

Proof sketch. The idea is to test the satisfaction of the property on only two non-parametric DTNs, viz., these valuating the only parameter with 0 and 1. We show that any other non-zero valuation is equivalent (up to rescaling) to the valuation 1: indeed, without constant terms in the model (other than 0), multiplying the value of the (unique) parameter by n will only impact the value of the duration of the runs by n, but will not impact reachability. See Appendix B.1. ◀

6.2 L/U-PDTNs with arbitrarily many clocks and parameters

Given $d \in \mathbb{Z}$, let $v_{0/d}$ denote the valuation such that for all $p \in \mathbb{P}_L$, $v(p) = 0$ and for all $p \in \mathbb{P}_U$, $v(p) = d$. We will consider the special valuation $v_{0/\infty}$: strictly speaking, $v_{0/\infty}$ is not a proper parameter valuation due to ∞, but valuating a PTA with $v_{0/\infty}$ consists in removing the inequalities involving upper-bound parameters with a positive coefficient.

▶ **Theorem 19.** *PR-emptiness (resp. PGR-emptiness) for L/U-PDTNs is equivalent to PR (resp. PGR) for DTNs.*

Proof sketch. We first show that networks of L/U-gPTAs are *monotonic*: that is, given $\mathcal{A}$ an L/U-gPTA and v a parameter valuation, any computation of $(v(\mathcal{A}))^n$ is a computation of $(v'(\mathcal{A}))^n$, for all v' such that upper-bound parameters are larger than or equal to their value in v and lower-bound parameters are smaller than or equal to their value in v. We then show that, given φ a global reachability property over $\mathcal{A}$, PGR-emptiness does not hold iff φ is satisfied in a configuration reachable in the DTN $(v_{0/\infty}(\mathcal{A}))^\infty$, as this DTN contains the behaviours for *all* parameter valuations, due to the monotonicity. Therefore, deciding PGR-emptiness for L/U-PDTNs amounts to deciding satisfaction of φ in the DTN $(v_{0/\infty}(\mathcal{A}))^\infty$; an additional technicality is necessary to show that if φ is satisfied in a configuration reachable in $(v_{0/\infty}(\mathcal{A}))^\infty$, then it is also reachable in $(v(\mathcal{A}))^\infty$ for some concrete (non-∞) parameter valuation v. See Appendix B.2. ◀

Table 1 Decidability of PR-e and PGR-e for PDTNs with 1 clock ($\surd$ = decidability, × = undecidability)

	Local properties	Global properties
Without invariants	$\surd$Theorem 20	×Theorem 17
With invariants	×Theorem 16	×

Since PR and PGR can be solved in EXPSPACE and 2-EXPSPACE respectively in DTNs [13], then solving PR-emptiness and PGR-emptiness can be done with the same complexities for fully parametric PDTNs with a single parameter and for L/U-PDTNs.

6.3 PDTNs with one clock, arbitrarily many parameters and no invariants

Our last decidability result closes the gap of Section 5: while PR-emptiness for PDTNs even with a single clock and one parameter is undecidable for local properties with invariants, and is undecidable for global properties even without invariants, we show that the problem becomes decidable for local properties without invariants. This result highlights the power of invariants in PDTNs.

▶ **Theorem 20.** *PR-emptiness is decidable for PDTNs with a single clock, arbitrarily many parameters, and no invariants.*

Proof sketch. We use here a completely different construction: we write a formula in the existential fragment of the first order theory of the integers with addition (a.k.a. Presburger arithmetic) enhanced with the divisibility operand (a decidable logic [25]), which will include the computation of the reachable durations of the locations used in guards in parallel to the reachability of the final location. In order to write this formula, we rely on results relating affine parametric semi-linear sets (apSl sets), a parametric extension of semi-linear sets, to durations in a PTA. Solving PR-emptiness then boils down to deciding the truth of the formula. See Appendix B.3. ◀

7 Conclusion

We investigated models with uncertainty over timing parameters, and parametrised in their number of components. We showed that the emptiness of the parameter valuations set for which a given location is reachable for some number of processes is decidable for networks with a single clock in each process and arbitrarily many parameters, provided no invariants are used, and for local properties only; this positive result is tight, in the sense that adding invariants or considering global properties makes this problem undecidable. We summarize the results for 1 clock in Table 1. Beside emphasizing the strong power of invariants and global properties, our results show an interesting fact on the expressive power of the communication model: while reachability is decidable for PTAs with 1 or 2 clocks and one parameter [8, 15, 23], it becomes undecidable in our setting with a single parameter. We exhibited two further decidable subclasses, by restricting the use of the parameters, without restriction of the number of clocks and parameters.

Perspectives. First, it can be easily shown that our negative results from Section 5 all hold as well over discrete time. Second, our undecidability results are expressed so far over integer-valued parameters. While the undecidability of integer-valued parameters implies undecidability for rational-valued parameters, the case of *bounded* rational-valued parameters (e.g., in a closed interval) remains open; we conjecture it remains undecidable using a different encoding of our undecidability proofs. However, we have no hint regarding the extension of our decidable results (notably Theorem 20), as our proof techniques heavily rely on the parameter integerness. Whether extending the decidable case of Theorem 20 to two clocks preserves decidability remains open too.

An interesting future work is the question of the *parameter synthesis* (for the decidable cases). That is, beyond deciding the emptiness of the valuations set for which some properties hold, can we *synthesize* them?

Finally, considering *universal* properties is an interesting perspective: this can include the existence of a parameter valuation for which *all* numbers of processes satisfy a property or, conversely, the fact that *all* parameter valuations are such that some number of processes satisfies a property.

References

1 Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jonathan Cederberg. Timed lossy channel systems. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *FSTTCS*, volume 18 of *LIPIcs*, pages 374–386. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012. `doi:10.4230/LIPICS.FSTTCS.2012.374`.

2 Parosh Aziz Abdulla and Giorgio Delzanno. Parameterized verification. *STTT*, 18(5):469–473, 2016. `doi:10.1007/s10009-016-0424-3`.

3 Parosh Aziz Abdulla, Giorgio Delzanno, Othmane Rezine, Arnaud Sangnier, and Riccardo Traverso. Parameterized verification of time-sensitive models of ad hoc network protocols. *TCS*, 612:1–22, 2016. `doi:10.1016/j.tcs.2015.07.048`.

4 Parosh Aziz Abdulla, Johann Deneux, and Pritha Mahata. Multi-clock timed networks. In *LiCS*, pages 345–354. IEEE Computer Society, 2004. `doi:10.1109/LICS.2004.1319629`.

5 Parosh Aziz Abdulla and Bengt Jonsson. Model checking of systems with many identical timed processes. *TCS*, 290(1):241–264, 2003. `doi:10.1016/S0304-3975(01)00330-9`.

6 Parosh Aziz Abdulla, A. Prasad Sistla, and Muralidhar Talupur. Model checking parameterized systems. In Edmund M Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 685–725. Springer, 2018. `doi:10.1007/978-3-319-10575-8_21`.

7 Rajeev Alur and David L. Dill. A theory of timed automata. *TCS*, 126(2):183–235, April 1994. `doi:10.1016/0304-3975(94)90010-8`.

8 Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *STOC*, pages 592–601, New York, NY, USA, 1993. ACM. `doi:10.1145/167088.167242`.

9 Étienne André. What's decidable about parametric timed automata? *STTT*, 21(2):203–219, April 2019. `doi:10.1007/s10009-017-0467-0`.

10 Étienne André, Johan Arcile, and Engel Lefaucheux. Execution-time opacity problems in one-clock parametric timed automata. In Siddharth Barman and Sławomir Lasota, editors, *FSTTCS*, volume 323 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, December 2024. `doi:10.4230/LIPIcs.FSTTCS.2024.3`.

11 Étienne André, Benoît Delahaye, Paulin Fournier, and Didier Lime. Parametric timed broadcast protocols. In Constantin Enea and Ruzica Piskac, editors, *VMCAI*, volume 11388 of *LNCS*, pages 491–512. Springer, 2019. `doi:10.1007/978-3-030-11245-5_23`.

12 Étienne André, Paul Eichler, Swen Jacobs, and Shyam Lal Karra. Parameterized verification of disjunctive timed networks. In Rayna Dimitrova and Ori Lahav, editors, *VMCAI*, volume 14499 of *LNCS*, pages 124–146. Springer, 2024. `doi:10.1007/978-3-031-50524-9_6`.

13 Étienne André, Swen Jacobs, Shyam Lal Karra, and Ocan Sankur. Parameterized verification of timed networks with clock invariants. Technical Report abs/2408.05190, arXiv, 2024. URL: `https://arxiv.org/abs/2408.05190`.

14 Étienne André, Michał Knapik, Wojciech Penczek, and Laure Petrucci. Controlling actions and time in parametric timed automata. In Jörg Desel and Alex Yakovlev, editors, *ACSD*, pages 45–54. IEEE Computer Society, 2016. `doi:10.1109/ACSD.2016.20`.

15 Étienne André, Didier Lime, and Nicolas Markey. Language preservation problems in parametric timed automata. *LMCS*, 16(1), January 2020. `doi:10.23638/LMCS-16(1:5)2020`.

16 Nikola Beneš, Peter Bezděk, Kim Guldstrand Larsen, and Jiří Srba. Language emptiness of continuous-time parametric timed automata. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP, Part II*, volume 9135 of *LNCS*, pages 69–81. Springer, July 2015. `doi:10.1007/978-3-662-47666-6_6`.

17 Olaf Burkart and Bernhard Steffen. Composition, decomposition and model checking of pushdown processes. *Nordic Journal of Computing*, 2(2):89–125, 1995.

18 Giorgio Delzanno and Pierre Ganty. Automatic verification of time sensitive cryptographic protocols. In Kurt Jensen and Andreas Podelski, editors, *TACAS*, volume 2988 of *LNCS*, pages 342–356. Springer, 2004. doi:10.1007/978-3-540-24730-2_27.

19 Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In Paul Gastin and François Laroussinie, editors, *CONCUR*, volume 6269 of *LNCS*, pages 313–327. Springer, 2010. doi:10.1007/978-3-642-15375-4_22.

20 Javier Esparza, Mikhail A. Raskin, and Chana Weil-Kennedy. Parameterized analysis of immediate observation Petri nets. In Susanna Donatelli and Stefan Haar, editors, *Petri Nets*, volume 11522 of *LNCS*, pages 365–385. Springer, 2019. doi:10.1007/978-3-030-21571-2_20.

21 Pierre Ganty and Rupak Majumdar. Algorithmic verification of asynchronous programs. *ACM Transactions on Programming Languages and Systems*, 34(1):6:1–6:48, 2012. doi:10.1145/2160910.2160915.

22 Carlo Ghezzi, Dino Mandrioli, Sandro Morasca, and Mauro Pezzè. A unified high-level Petri net formalism for time-critical systems. *TSE*, 17(2):160–172, 1991. doi:10.1109/32.67597.

23 Stefan Göller and Mathieu Hilaire. Reachability in two-parametric timed automata with one parameter is expspace-complete. *TCS*, 68(4):900–985, 2024. doi:10.1007/S00224-023-10121-3.

24 Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. Linear parametric model checking of timed automata. *JLAP*, 52-53:183–220, 2002. doi:10.1016/S1567-8326(02)00037-1.

25 Antonia Lechner, Joël Ouaknine, and James Worrell. On the complexity of linear arithmetic with divisibility. In *LiCS*, pages 667–676. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.67.

26 Engel Lefaucheux. When are two parametric semi-linear sets equal? Technical Report hal-04172593, HAL, 2024. URL: https://inria.hal.science/hal-04172593.

27 Li Li, Jun Sun, Yang Liu, and Jin Song Dong. Verifying parameterized timed security protocols. In Nikolaj Bjørner and Frank S. de Boer, editors, *FM*, volume 9109 of *LNCS*, pages 342–359. Springer, 2015. doi:10.1007/978-3-319-19249-9_22.

28 Philip Meir Merlin and David J. Farber. Recoverability of communication protocols-implications of a theoretical study. *IEEE Transactions on Communications*, 24(9):1036–1043, 1976. doi:10.1109/TCOM.1976.1093424.

29 Marvin L. Minsky. *Computation: Finite and infinite machines.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1967.

30 Chander Ramchandani. *Analysis of asynchronous concurrent systems by timed Petri nets.* PhD thesis, Massachusetts Institute of Technology, USA, 1973. URL: http://hdl.handle.net/1721.1/13739.

31 Mikhail A. Raskin, Chana Weil-Kennedy, and Javier Esparza. Flatness and complexity of immediate observation Petri nets. In Igor Konnov and Laura Kovács, editors, *CONCUR*, volume 171 of *LIPIcs*, pages 45:1–45:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CONCUR.2020.45.

32 Luca Spalazzi and Francesco Spegni. Parameterized model checking of networks of timed automata with Boolean guards. *TCS*, 813:248–269, 2020. doi:10.1016/j.tcs.2019.12.026.

A Proofs of Section 5

A.1 Proof of Proposition 14

▶ **Proposition 14.** *Assume a gPTA $\mathcal{A}$ with a single clock and a single parameter, and ℓ one of its locations. Deciding the emptiness of the set of timing parameter valuations v for which ℓ is reachable in the NTA $v(\mathcal{A})^3$ is undecidable, even without invariants, for all such $\mathcal{A}$.*

Proof. Let us show that the absence of invariants in the proof of Proposition 13 does not harm the encoding of the 2CM. The overall idea is that, if a process "chooses" to not take some outgoing transition in the absence of invariants, then we show that the whole computation is stuck, and q_{halt} is unreachable. Put it differently, while the absence of invariants may add some runs, none of these runs can reach q_{halt}, and the final correctness argument of the proof of Proposition 14 still holds.

In the initial gadget in Figure 2e, if the process encoding x_1 (resp. x_2) fails in taking the transition from ℓ_0^t to ℓ_1^1 (resp. from ℓ_0^t to ℓ_1^2) after exactly p time units, then the first process encoding t is stuck forever in ℓ_1^t, and q_{halt} is unreachable.

Now, consider the subpart encoding the clock t (in all gadgets): in the absence of invariants, no transition is forced to be taken. However, since all outgoing transitions from the q_i^t locations are guarded by $t = 0$, staying more than 0 time unit blocks this process forever, and q_{halt} becomes unreachable. The same applies to intermediate locations, which all have tight guards (e.g., $t = p$). Therefore, to reach q_{halt}, then the process encoding clock t must take all its guards at the appropriate time—which we assume from now on.

Then, consider the two subparts encoding clocks x_1 and x_2. For the same reason, failing in taking a transition will block the process forever and, due to the final gadget in Figure 2f ensuring transitions are guarded by $x = 0$, will block the first process, and therefore q_{halt} will be unreachable.

This concludes the proof. ◀

A.2 Proof of Proposition 15

▶ **Proposition 15.** *Let $\mathcal{A}$ be a gPTA, and ℓ one of its locations. For any $n \geq 3$, deciding the emptiness of the set of timing parameter valuations v for which ℓ is reachable in the NTA $v(\mathcal{A})^n$ is undecidable, even when $\mathcal{A}$ contains a single clock and a single parameter without invariants.*

Proof. The idea is to reuse the construction of the proof of Proposition 14, by "killing" any process except 3 by sending them to "sink" locations, from which they cannot interfere with the encoding of the 2-counter machine. Fix $n > 3$. The modified initial gadget is given in Figure 5. The first three processes behave as in Figure 2e. Then, all remaining processes are sent to "sink" locations (ℓ_s^3 to ℓ_s^{n-1}); for one process i to reach its sink location ℓ_s^i, its predecessor $i - 1$ must have reached its own sink location ℓ_s^{i-1}, due to the location guard "$[\ell_s^{i-1}]$". Finally note that the first process (simulating t) must make sure that the last process has reached its sink location (location guard "$[\ell_s^{n-1}]$") before reaching the starting location q_0^t. This ensures that all processes $i > 3$ are in sink locations, where they cannot interfere with the encoding.

The rest of the proof then follows the reasoning of the proof of Proposition 14. ◀

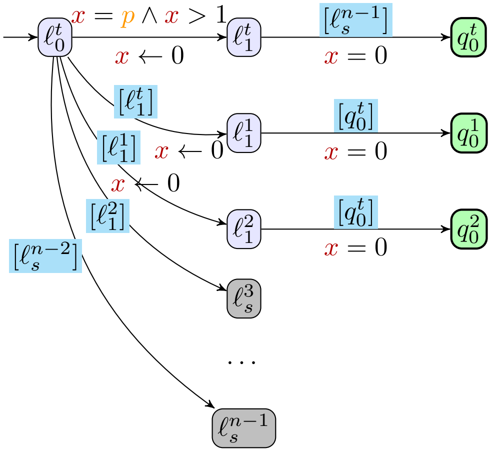


Figure 5 Modified initial gadget for exactly n processes

A.3 Proof of Theorem 16

▶ **Theorem 16.** *PR-emptiness is undecidable for general PDTNs, even with a single clock and a single parameter (with invariants).*

Proof. First note that, in order to traverse the gadgets from the proof of Proposition 13, one needs (at least) 3 processes, due to the interdependency between the location guards. We will show that any process beyond the 3 required processes can only "mimic" the behaviour of one of the 3 required processes.

Consider a process beyond the first three processes. In the initial gadget (Figure 2e), it can "choose" any of the three branches leading to the three subparts, as the first three processes make them all available.

First assume this extra process follows the first subpart, simulating clock t: observe that, in all the gadgets of the first subpart (Figures 2b, 3b, and 4b), the behaviour is almost completely deterministic (e.g., all locations with invariant "$t \leq p$" are followed by a transition guarded by "$t = p$"). The only exception is in the branching to either ℓ_i^t or ℓ_k^t in Figure 4b; assuming exactly one of the location guards $[\ell_{i1}^1]$ or $[\ell_k^1]$ holds (which we discuss below), then this aforementioned extra process will follow exactly the behaviour of the first process.

Alternatively, assume that this extra process follows the second subpart (simulating x_1) and let us show that it will follow the second process (the third subpart is similar). In most locations of our gadgets (Figures 2c, 3c, and 4c), the outgoing guards are *tight* w.r.t. the invariant (e.g., invariant "$x_1 \leq p + 1$" followed by a transition guarded by "$x_1 = p + 1$"), and this extra process is forced to follow the second process. In the remaining locations followed by non-tight guards (e.g., "$x_1 \leq p$" in the increment gadget in Figure 3c), the guard is always additionally guarded by the location guard $[q_j^t]$, which makes its behaviour deterministic since the processes simulating clock t are deterministic. Therefore, this extra process is again forced to follow the second process. The 0-test and decrement gadget is similar, with one additional crucial observation: since the aforementioned extra process mimicked the second process so far, then in q_i^1, due to the location guard $[q_i^t]$ ensuring a transition in 0-time, then either $x_1 = 0$ *or* $x_1 > 0$ holds, and therefore either both processes simulating x_1 reach ℓ_{i1}^1, *or* both of them reach ℓ_k^1—which justifies the assumption made earlier.

For these reasons, q_{halt} is reachable only if the first three processes correctly simulate the 2CM; since any additional processes will just simulate one of the first three processes, we can just apply the correctness argument from the proof of Proposition 13. ◀

A.4 Proof of Theorem 17

▶ **Theorem 17.** *PGR-emptiness is undecidable for general PDTNs, even with a single clock and a single parameter, with or without invariants.*

Proof. We prove the result *without* invariants. (The case with invariants is simpler and follows immediately.)

The idea is that we ask whether *all* processes can reach the target location q_{halt}, which can only be done if the 2CM was properly simulated. First note that asking whether *all* processes can reach the target location q_{halt} is a property satisfying the definition of global property in Section 4, by simply checking that the number of all locations but q_{halt} is exactly 0 (called "target problem").

As in the proof of Theorem 16, in order to traverse the gadgets from the proof of Proposition 13, one needs (at least) 3 processes, due to the interdependency between the location guards. We show here that any process beyond the 3 required processes can only "mimic" the behaviour of one of the 3 required processes, or block the system due to a timelock if it fails to correctly mimic it.

First note that if any of the first three processes does not correctly encode the 2CM (which is a possibility due to the absence of invariants), then an outgoing guard will not be firable, and therefore this process will be blocked forever in its current location, and the global property that all processes must reach q_{halt} cannot hold. So we assume that the first three processes correctly encode the 2CM (w.l.o.g. we assume that process i encodes the i-th subpart).

Now consider a process beyond the first three processes. In the initial gadget (Figure 2e), it can "choose" any of the three branches leading to the three subparts, as the first three processes make them all available. Due to the absence of invariants, the process can also choose to not pass the initial gadget—but failing to take such a guard will block it forever in a location due to the guards $x = 0$, and the global property that all processes must reach q_{halt} will never hold.

First assume this extra process follows the first subpart, simulating clock t: just as in the proof of Theorem 16, because in all the gadgets of the first subpart (Figures 2b, 3b, and 4b), the guards are always punctual (i.e., involve equalities), then this extra process will either follow exactly the behaviour of the first process, or fail in taking some guard—therefore remaining forever in its location and violating the global reachability property.

Alternatively, assume that this extra process follows the second subpart (simulating x_1). As in the former reasoning, either that extra process will exactly mimic the behaviour of the second process, or will fail in taking some guard, and therefore being blocked in its location, thus violating again the global reachability property.

The case of the final gadget is similar: if any of the processes arrive too early or too late, they will be blocked due to the urgent guards (of the form $x = 0$ together with the location guards), and the global reachability will be violated.

For these reasons, q_{halt} is reachable only if the first three processes correctly simulate the 2CM and if all additional processes simulate exactly one of the first three processes. Finally, we can again apply the correctness argument from the proof of Proposition 14. ◀

B Proofs of Section 6

B.1 Proof of Theorem 18

▶ **Theorem 18.** *PR-emptiness (resp. PGR-emptiness) for fully parametric PDTNs with 1 parameter and arbitrarily many clocks is equivalent to PR (resp. PGR) for DTNs.*

Proof. Let $\mathcal{A}$ be a fully parametric gPTA with 1 parameter p. $\mathcal{A}^\infty$ is therefore a fully parametric PDTN. Let v_0 and v_1 denote the valuations such that $v_0(p) = 0$ and $v_1(p) = 1$.

We prove the result for global reachability properties (PGR-emptiness), as local properties are a subcase. Fix a global property φ. Let us show that PGR-emptiness does not hold iff φ is satisfied in a configuration reachable in $(v_0(\mathcal{A}))^\infty$ or in $(v_1(\mathcal{A}))^\infty$.

The following lemma derives easily from [24, Proposition 4.7], adapted to the semantics of NTAs (Section 3), and comes from the fact that, whenever no constant terms are used in a gPTA, then rescaling the parameter valuation does not impact the satisfaction of reachability properties.

▶ **Lemma 21** (Multiplication of constants). *Let $\mathcal{A}$ be a fully parametric gPTA with a single parameter p. Fix $n \in \mathbb{N}$. Let φ be a global property. Then for all parameter valuations v, a configuration $\mathfrak{c}$ with $\mathfrak{c} \models \varphi$ is reachable in $(v(\mathcal{A}))^n$ iff $\forall t \in \mathbb{Q}_{>0}$ such that $t \times v(p) \in \mathbb{N}$, a configuration $\mathfrak{c}'$ with $\mathfrak{c}' \models \varphi$ is reachable in $((t \times v)(\mathcal{A}))^n$, where $t \times v$ denotes the valuation such that $(t \times v)(p) = t \times (v(p))$.*

We can now proceed to the proof of Theorem 18.

- $\Rightarrow$ Assume PGR-emptiness does not hold for $\mathcal{A}^\infty$, i.e., there exists v such that there exists $n \in \mathbb{N}_{>0}$ such that φ is satisfied in a reachable configuration in $(v(\mathcal{A}))^n$. Let us show that φ is satisfiable in a configuration reachable in $(v_0(\mathcal{A}))^n$ or in $(v_1(\mathcal{A}))^n$.
  If $v(p) = 0$, then the result is immediate. If $v(p) \neq 0$, then from Lemma 21, φ is satisfied in a configuration reachable in $(v_1(\mathcal{A}))^n$ (by choosing some appropriate t, i.e., $\frac{1}{v(p)}$).
- $\Leftarrow$ Assume φ is satisfied in a reachable configuration in $(v_0(\mathcal{A}))^\infty$ or in $(v_1(\mathcal{A}))^\infty$. That is, there exists $n \in \mathbb{N}_{>0}$ such that there is a computation π of $(v_0(\mathcal{A}))^n$ or of $(v_1(\mathcal{A}))^n$ reaching a configuration $\mathfrak{c}$ s.t. $\mathfrak{c} \models \varphi$. Therefore PGR-emptiness does not hold.

Therefore, it suffices to test the satisfaction of φ in $(v_0(\mathcal{A}))^\infty$ and $(v_1(\mathcal{A}))^\infty$.

Finally, the hardness argument is immediate, considering a PDTN without parameter, and replacing constants different from 1 with additional clocks and locations. ◀

B.2 Proof of Theorem 19

Recall that $\mathbb{P} = \mathbb{P}_L \uplus \mathbb{P}_U$. Given v, v', we write $v' \preceq v$ whenever $\forall p \in \mathbb{P}_L, v'(p) \leq v(p)$ and $\forall p \in \mathbb{P}_U, v'(p) \geq v(p)$.

▶ **Lemma 22** (Monotonicity). *Let $\mathcal{A}$ be an L/U-gPTA. Let v be a parameter valuation. For any v' such that $v' \preceq v$, for any $n \in \mathbb{N}_{>0}$, any computation of $(v(\mathcal{A}))^n$ is a computation of $(v'(\mathcal{A}))^n$.*

Proof. From the fact that any valuation $v' \preceq v$ will only *add* behaviours due to the enlarged guards. ◀

▶ **Theorem 19.** *PR-emptiness (resp. PGR-emptiness) for L/U-PDTNs is equivalent to PR (resp. PGR) for DTNs.*

Proof. We prove the result for global reachability properties (PGR-emptiness), as local properties are a subcase. Let $\mathcal{A}$ be an L/U-gPTA and φ a global reachability property over $\mathcal{A}$. $\mathcal{A}^\infty$ is therefore an L/U-PDTN. Consider the DTN $(v_{0/\infty}(\mathcal{A}))^\infty$. Let us show that PGR-emptiness does not hold iff φ is satisfied in a configuration reachable in $(v_{0/\infty}(\mathcal{A}))^\infty$.

$\Rightarrow$ Assume PR-emptiness does not hold for $\mathcal{A}^\infty$, i.e., there exists v such that there exists $n \in \mathbb{N}_{>0}$ such that φ is satisfied in $(v(\mathcal{A}))^n$. That is, there exists a computation π of $(v(\mathcal{A}))^n$ reaching a configuration $\mathfrak{c}$ such that $\mathfrak{c} \models \varphi$. From Lemma 22, π is a computation of $(v'(\mathcal{A}))^n$, for any $v' \preceq v$. And by extension, completely removing the upper-bound guards (i.e., valuating upper-bound parameters with ∞) only adds behaviour, and therefore π is a computation of $(v_{0/\infty}(\mathcal{A}))^n$. Hence $\mathfrak{c}$ is reachable in $(v_{0/\infty}(\mathcal{A}))^\infty$, and hence φ is satisfied.

$\Leftarrow$ Assume there exists a configuration $\mathfrak{c}$ reachable in $(v_{0/\infty}(\mathcal{A}))^\infty$ such that $\mathfrak{c} \models \varphi$. That is, there exists $n \in \mathbb{N}_{>0}$ such that there is a computation π of $(v_{0/\infty}(\mathcal{A}))^n$ reaching a configuration $\mathfrak{c}$ s.t. $\mathfrak{c} \models \varphi$. Now, $v_{0/\infty}$ is not a proper parameter valuation, so we need to exhibit a parameter valuation assigning to each parameter an integer value. We reuse the same concrete parameter valuation for upper-bound parameters as exhibited in [24, Proposition 4.4]: let T' be the smallest constant occurring in the L/U-gPTA $\mathcal{A}$, and let T be the maximum clock valuation along π. Fix $D = T + |T'| + 1$. (We add T' to compensate for potentially negative constant terms "d" in guards and invariants of $\mathcal{A}$.) Since the maximum clock valuation along π is T, any guard of the form $x \leq \sum_{1\leq i\leq M} \alpha_i \times p_i + d$, that was replaced with $x < \infty$ in $(v_{0/\infty}(\mathcal{A}))^n$, can be equivalently replaced with $x \leq \sum_{1\leq i\leq M} \alpha_i \times D + d$ without harming the satisfaction of the guard. Therefore, $\mathfrak{c}$ is reachable in $(v_{0/D}(\mathcal{A}))^n$, and hence in $(v_{0/D}(\mathcal{A}))^\infty$. Therefore, since $\mathfrak{c} \models \varphi$, PGR-emptiness does not hold.

Therefore, deciding PGR-emptiness for L/U-PDTNs amounts to deciding satisfaction of φ in the DTN $(v_{0/D}(\mathcal{A}))^\infty$.

The case of local properties follows a similar reasoning. ◀

B.3 Proof of Theorem 20

▶ **Theorem 20.** *PR-emptiness is decidable for PDTNs with a single clock, arbitrarily many parameters, and no invariants.*

In [12], the shortest time to reach a location could be computed, allowing to replace the location guards one by one. That is, for each location appearing in a location guard, we send one process to this location as quickly as possible, and which then remains in this location forever. Hence, from the time this location can be reached, the location guard remains satisfied forever. Note that this method only works thanks to the absence of invariants—which allows processes to "die" in every location.

However, this method cannot be reused here in the presence of parameters, as the notion of a "shortest time" is not entirely well-defined in this setting. As such, we do not want to remove the location guards once at a time. Instead, we will write a formula of the first order theory of the integers with addition (a.k.a. Presburger arithmetic) enhanced with the divisibility operand, which will include the computation of the reachable durations of the locations used in guards in parallel to the reachability of the final location. Hence, solving PR-emptiness will boil down to deciding the truth of the formula.

In order to write this formula, we rely on results relating affine parametric semi-linear sets (apSl sets), a parametric extension of semi-linear sets, to durations in a PTA. An apSl sets is a function associating to a vector of parameter values $\mathbf{p}$ a semi-linear set of vectors of integers. We will not need the specific shape of apSl sets here, but instead two important properties they have.

First, given a PTA $\mathcal{A}$ with one clock and arbitrarily many parameters and given two locations ℓ, ℓ' of $\mathcal{A}$, one can compute [10] the set of parametric durations of runs reaching ℓ' from ℓ, and represent it using a one-dimensional apSl set. It is interesting to note that the apSl set representation only contain integers, while the actual durations are a set of real values. The idea of the apSl representation is that the value $2i$ is in $S(\mathbf{p})$ iff the integer i is a reachable duration, and $2i+1$ is in $S(\mathbf{p})$ iff all the values of the interval $(i, i+1)$ are reachable durations. This representation hence strongly requires that the parameters range over integers. We also note that the construction of this apSl set can easily be modified so that given two locations ℓ, ℓ' and n edges $t_1, \dots, t_n$ of $\mathcal{A}$, one could include the information of when the edge t_1 was crossed on the way to ℓ' for the first time. For instance, if $n = 1$, $(2i+1, 2j)$ belongs to the corresponding apSl set iff ℓ' can be reached from ℓ in j time units, and a path achieving this takes t_1 for the first time during the interval $(i, i+1)$.

Second, it was also shown in [10] that given an apSl set S, one can build a formula in the existential fragment of Presburger arithmetic with divisibility (a decidable logic [25]) ϕ_S such that given parameter values $\mathbf{p}$, $S(\mathbf{p})$ is not empty iff $\exists x \in \phi_S(x, \mathbf{p})$ is true.

We can now move to the proof.

Proof. Let $\mathcal{A} = (\Sigma, L, \ell_0, \{x\}, \mathbb{P}, I, E)$ be a gPTA and ℓ_f be a target location.

We first guess a sequence $e_1, \dots, e_m$ of different edges of E which contains a location guard (we trivially have that $m \leq |E|$). These are intuitively the edges with location guards which will be needed, either by the process reaching ℓ_f, or by the processes which will reach the locations used in location guards. We assume these edges are ordered by the date at which they will be taken for the first time. In practice, this guess can be achieved by complete enumeration.

For all $i \leq m$, we set ℓ_i to be the location appearing in the guard of edge e_i and set $\ell_{m+1} = \ell_\mathrm{f}$. In order to reach ℓ_i, some edges from $e_1, \dots e_{i-1}$ may be needed. Let $e_{j_1^r}, \dots, e_{j_{m_i}^r}$ be those edges, in the order they first appear in the run. We build the PTA $\mathcal{A}_i = (\Sigma, L_i, (\ell_0, 0), (\ell_i, m_i), \{x\}, \mathbb{P}, I_i, E_i)$ where

- $L_i = \big\{(\ell, k) \mid \ell \in L \wedge k \in \{0, \dots, m_i\}\big\}$,
- for any $(\ell, k) \in L'$, $I_i(\ell, k) = I(\ell)$,
- $\big((\ell, k), g, a, R, (\ell', k')\big) \in E'$ iff there exists $e = (\ell, g, \gamma, a, R, \ell') \in E$ such that one of the following holds
  - $k = k'$ and $\gamma = \top$ or there exists $r \leq k$ such that $e = e_{j_r^i}$,
  - $k' = k + 1$ and $e = e_{j_{k+1}^r}$.

In other words, $\mathcal{A}_i$ consists in $m_i + 1$ successive copies of $\mathcal{A}$ where the k'th copy blocks every edge with location guard except e_r^i with $r \leq k+1$, and in particular taking e_{k+1}^i leads to the next copy. This way, a run of $\mathcal{A}_i$ ending in the final location is forced to follow the guessed structure with respect to the usage of edges with location guards. It however loses information about when those edges are available.

By applying the previously mentioned result from [10, 26], we can build a semilinear set T_i of $m_i + 1$-tuples parametric values representing the durations of runs going from $(\ell_0, 0)$ to (ℓ_i, m_i), storing the intervals of the first firing of the edges $e_{j^{r_s}}$.

As previously mentioned, the construction of T_i does not take into account the constraints brought by the location guards. Assuming that for $k < i$ location ℓ_k is reached at time $h_i(\mathbf{p})$, we can see that deciding the existence of parameter values $\mathbf{p}$ such that ℓ_i is reached in $\mathcal{A}$ is equivalent to solving the formula[2]

$$\exists \mathbf{p}, \exists \mathbf{d_1}, d'_1, \ldots \mathbf{d_{m_i+1}}, d'_{m_i+1}, (\mathbf{d_1 p} + d'_1, \ldots, \mathbf{d_{m_i+1} p} + d'_{m_i+1}) \in T_i,$$

$$\bigwedge_{k=1}^{m_i} \mathbf{d_{j^i_k}}\, \mathbf{p} + d'_{j^i_k} \geq h_{j^i_k}(\mathbf{p})$$

Moreover, note that the value $\mathbf{d_{m_i+1} p} + d'_{m_i+1}$ built here is a possible value for $h_{i+1}(\mathbf{p})$.

Hence by combining the formulas obtained for each i, removing the comparison to h_i (which becomes redundant once the variables are shared by every formula), and verifying that the orders of each formula is compatible[3] we have that PR-emptiness is equivalent to the falsity of

$$\exists \mathbf{p}, \exists \mathbf{d_1}, d'_1, \ldots \mathbf{d_{m+1}}, d'_{m+1}, \bigwedge_{i=1}^{m+1} (\mathbf{d_{j^r_1}}\, \mathbf{p} + d'_{j^r_1}, \ldots, \mathbf{d_{j^r_{m_i}}}\, \mathbf{p} + d'_{j^r_{m_i}}, \mathbf{d_i p} + d'_i) \in T_i.$$

From [25], as this formula is expressed in the existential fragment of Presburger arithmetic with divisibility, it is decidable. ◀

▶ Remark 23. Let us quickly discuss the complexity of this algorithm. The formulas produced by [10] are at worst doubly exponential. The modifications we apply to them, combining a polynomial number of those formulas, remains doubly exponential. We then rely on the decidability of the existential fragment of Presburger arithmetic with divisibility which can be solved in NEXPTIME [25]. The nondeterminism allowed through this last step combines with the nondeterministic guesses of transition sequences without additional cost. As a consequence, our algorithm lies in 3-NEXPTIME.

[2] The characters in bold are vectors of variables. In order to avoid complexifying the formula, we did not indicate the transformation from row to column vector which is necessary to multiply the two vectors and produce a single term.

[3] For example, if the path to location ℓ goes through a location guard on ℓ', then ℓ cannot reciprocally be on a location guard encountered on the way to ℓ'. This condition on order is not directly handled in the formula in the case where both location guards can be reached within the same interval of time. A more precise decomposition of time units, would allow including this condition into the formula.