

Centre Fédéré en Vérification

Technical Report number 2002.1

Durations, Parametric Model-Checking in Timed Automata with Pressburger Arithmetic

Véronique Bruyère, Emmanuel Dall'Olio and Jean-François Raskin



This work was partially supported by a FRFC grant: 2.4530.02

<http://www.ulb.ac.be/di/ssd/cfv>

Durations, Parametric Model-Checking in Timed Automata with Presburger Arithmetic

VÉRONIQUE BRUYÈRE
Institut d'Informatique*
Université de Mons-Hainaut

EMMANUEL DALL'OLIO, JEAN-FRANÇOIS RASKIN
Département d'Informatique†
Université Libre de Bruxelles

Abstract

We consider the problem of model-checking a parametric extension of the logic TCTL over timed automata and establish its decidability. We show that the notion of durations between regions defined by a timed automaton is expressible in the arithmetic of Presburger (when the time domain is discrete) or in the theory of the reals (when the time domain is dense). When this notion of duration is formalized in those theories, we show that the parametric model-checking problem for the logic TCTL can easily be solved.

*Université de Mons-Hainaut, Le Pentagone, Avenue du Champ de Mars 6, B-7000 Mons, Belgium, Email: Veronique.Bruyere@umh.ac.be

†Université Libre de Bruxelles, Boulevard du Triomphe CP 212, B-1050-Bruxelles, Belgium Email: Emmanuel.Dallolio@ulb.ac.be, Jean-Francois.Raskin@ulb.ac.be

1 Introduction

For more than twenty years, temporal logics [14, 6] and automata theory play a central role in the foundations for the formal verification of reactive systems. Reactive systems often have to meet real-time constraints. As a consequence, in the early nineties, temporal logics and automata theory have been extended to explicitly refer to real-time. Timed automata [2], an extension of usual automata with clocks, have been proposed as a natural model for systems that must respect real-time constraints. Temporal logics have also been extended to express quantitative real-time properties [1, 3, 11].

The model-checking problem, given a timed automaton \mathcal{A} and a real-time logic formula ϕ , consists in answering by YES or NO if the executions of \mathcal{A} verify the property expressed by ϕ . So the model-checking problem tries to answer the question $\mathcal{A} \models? \phi$. If the answer is NO then the model-checking algorithm is able to exhibit an example of execution where the property is falsified.

Response property is a typical example of a real-time property: “if every request reaches the server within two time units, then every request must be accepted or rejected within four time units”. This property is easily expressed in TCTL as follows:

$$\forall \square(\text{send} \rightarrow \forall \Diamond_{\leq 2} \text{request}) \rightarrow \forall \square(\text{send} \rightarrow \forall \Diamond_{\leq 4} \text{granted} \vee \text{rejected})$$

The main drawback with this formulation of the response property is that it refers to fixed constants. In the context of real-time systems, it is often more natural to use parameters instead of constants to express the delays involved in the model. In particular, we would like to formalize the above property in a more abstract way like “if every request reaches the server within α time units, then there should exist a bound of time γ within which every request must be accepted or rejected, that bound should be less than 2α ”. In this way, the property is less dependent on the particular model on which it is checked. This is particularly important when the model is not completely known a priori. The parametric property above can be formalized using an extension of TCTL with parameters and quantification over parameters as follows:

$$\forall \alpha \exists \gamma \cdot \gamma \leq 2\alpha \wedge [\forall \square(\text{send} \rightarrow \forall \Diamond_{\leq \alpha} \text{request}) \rightarrow \forall \square(\text{send} \rightarrow \forall \Diamond_{\leq \gamma} \text{granted} \vee \text{rejected})]$$

The use of parameters can also be very useful to *learn* about a model. For example, parameters in conjunction with temporal logics can be used to express questions that are not expressible in usual temporal logics. As an example, consider the following question: “Is there a bound on the delay to reach a σ state from an initial state?”. This property is easily expressed using the TCTL syntax extended with parameters: $\exists \alpha \cdot \forall \Diamond_{\leq \alpha} \sigma$. The answer to the question is YES if the formula is true. We can even be more ambitious and ask to characterize the set of parameter valuations for which a formula is true on a given timed model. As an example, consider the following TCTL formula with parameter γ : $\forall \Diamond_{\leq \gamma} \sigma$. If there is no bound on the time needed to reach a σ state from an initial state, then the answer to the parameter valuation problem would be “empty”, while it would be all the valuations v such that $v(\alpha) \geq c$ if all paths starting from an initial state reach a σ state within c time units.

We show in this paper that the problems outlined above are decidable for discrete- and dense-timed automata (if parameters range over natural numbers). When those results were partially known, see [15, 16, 10], the main contributions of this paper are as follows. First, we show that the *natural* and *intuitive* notion of *path duration* (between regions of a timed automaton) is central and sufficient to answer the parametric TCTL verification problem. Second, we show that Presburger arithmetic (or the additive theory of reals) and classical automata are *elegant* tools to formalize the notion of path durations between regions of a timed automaton. Finally, using those simple and clean concepts from logic and automata theory, we are able to prove the decidability of the model-checking problem for a parametric version of TCTL that strictly subsumes, to the best of our knowledge, all the parametric extensions of TCTL proposed so far. Furthermore, the technique to establish the decidability results is very similar if the time domain is discrete or dense.

Structure of the paper. The paper is organized as follows. In Section 2, we review some basic notions about timed automata. We consider both discrete and dense time domains. In Section 3, we introduce the Parametric Timed CTL logic, PTCTL for short. In Section 4, we details known results about the finite region graphs underlying any timed automaton. In Section 5, we define the notion of paths duration between

two regions and show that the set of possible path durations between two regions is effectively Presburger definable when considering a discrete time domain, and effectively definable in the additive theory of reals when considering a dense time domain. In Section 6, we show that the model-checking problem for PTCTL can be solved using the notion of path duration and its effective formalization into Presburger arithmetic (or in the additive theory of reals). In Section 7, we study in details the complexity of our method. In Section 8, we compare our work with existing related works.

2 Timed Automata

In this section, we recall the classical notion of timed automaton [2].

Notations. Throughout the paper, we denote by $X = \{x_1, \dots, x_n\}$ a set of n clocks. We use the same notation $x = (x_1, \dots, x_n)$ for the *variables* and for an *assignment* of values to these variables. Depending on whether the timed automata are *dense*-timed or *discrete*-timed, the values of the variables are taken in domain \mathbb{T} equal to the set \mathbb{R}^+ of nonnegative reals or to the set \mathbb{N} of natural numbers. Given a clock assignment x and $\tau \in \mathbb{T}$, $x + \tau$ is the clock assignment $(x_1 + \tau, \dots, x_n + \tau)$. A *simple guard* is an expression of the form $x_i \sim c$ where x_i is a clock, $c \in \mathbb{N}$ is an integer constant, and \sim is one of the symbols $\{<, \leq, =, >, \geq\}$. A *guard* is any finite conjunction of simple guards. We denote by \mathcal{G} the set of guards. Let g be a guard and x be a clock assignment, notation $x \models g$ means that x satisfies g . A *reset function* r indicates which clocks are reset to 0. Thus either $r(x)_i = 0$ or $r(x)_i = x_i$. The set of reset functions is denoted by \mathcal{R} . We use notation Σ for the set of *atomic propositions*.

Definition 1 A *timed automaton* is a quadruple $\mathcal{A} = (L, X, E, I)$ with the following components: (i) L is a finite set of *locations* together with a *labeling function* $\mathcal{L} : L \rightarrow 2^\Sigma$, (ii) X is a set of *clocks*, (iii) $E \subseteq L \times \mathcal{G} \times \mathcal{R} \times L$ is a finite set of *edges*, (iv) $I : L \rightarrow \mathcal{G}$ assigns an *invariant* to each location.

Definition 2 A timed automaton $\mathcal{A} = (L, X, E, I)$ generates a *transition system* $T_{\mathcal{A}} = (Q, \rightarrow)$ with a set of *states* Q equal to $\{(l, x) \mid l \in L, x \in \mathbb{T}^n, x \models I(l)\}$ and a *transition relation* $\rightarrow = \bigcup_{\tau \in \mathbb{T}} \xrightarrow{\tau}$ defined as follows:

$$(l, x) \xrightarrow{\tau} (l', x')$$

- if $\tau > 0$, then $l = l'$ and $x' = x + \tau$ (*elapse of time* at location l)
- if $\tau = 0$, then $(l, g, r, l') \in E$, $x \models g$ and $r(x) = x'$ (*instantaneous switch*)

The states (l, x) of $T_{\mathcal{A}}$ are shortly denoted by q . Given $q = (l, x) \in Q$ and $\tau \in \mathbb{T}$, we denote by $q + \tau$ the state $(l, x + \tau)$. Note that if \mathcal{A} is a discrete-timed automaton, the elapse of time is discrete with $\tau = 1, 2, 3, \dots$. If \mathcal{A} is a dense-timed automaton, then τ is any positive real number. Note also that given a transition $q \rightarrow q'$ of $T_{\mathcal{A}}$, it is easy to compute the unique τ such that $q \xrightarrow{\tau} q'$.

Definition 3 Given a transition system $T_{\mathcal{A}}$, a *run* $\rho = (q_i)_{i \geq 0}$ is an infinite path in $T_{\mathcal{A}}$

$$\rho = q_0 \xrightarrow{\tau_0} q_1 \xrightarrow{\tau_1} q_2 \cdots q_i \xrightarrow{\tau_i} q_{i+1} \cdots$$

such that $\sum_{i \geq 0} \tau_i = \infty$. A *finite run* $\rho = (q_i)_{0 \leq i \leq j}$ is any finite path in $T_{\mathcal{A}}$. A *position* in ρ is a state $q_i + \tau$, where $i \geq 0$, $\tau \in \mathbb{T}$ and either $\tau = 0$, or $0 < \tau < \tau_i$. The *duration* $t = D(q)$ at *position* $q = q_i + \tau$ is equal to $t = \tau + \sum_{0 \leq i' < i} \tau_{i'}$.

So, if \mathcal{A} is a discrete-timed automaton, then D is a function which assigns a natural number to any position of the run. If \mathcal{A} is a dense-timed automaton, the assignment is in \mathbb{R}^+ . As we allow *several* consecutive instantaneous switches in the definition of a run, different positions q can have the same duration $D(q)$. The set of positions in a run ρ can be totally ordered as follows. Let $q = q_i + \tau$ and $q' = q_{i'} + \tau'$ be two positions of ρ . Then $q < q'$ iff either $i < i'$ or $i = i'$ and $\tau < \tau'$.

First type formulae:

$q \models_v \sigma$	iff	$\sigma \in \mathcal{L}(l)$
$q \models_v \neg\varphi$	iff	$q \not\models_v \varphi$
$q \models_v \varphi \vee \psi$	iff	$q \models_v \varphi$ or $q \models_v \psi$
$q \models_v \exists\bigcirc\varphi$	iff	there exists a transition $q \rightarrow q'$ in $T_{\mathcal{A}}$ such that $q' \models_v \varphi$
$q \models_v \varphi\exists\mathbf{U}_{\sim\alpha}\psi$	iff	there exists a run $\rho = (q_i)_{i \geq 0}$ in $T_{\mathcal{A}}$ with $q = q_0$, there exists a position p in ρ such that $\mathbf{D}(p) \sim v(\alpha)$, $p \models_v \psi$ and $p' \models_v \varphi$ for all $p' < p$
$q \models_v \varphi\forall\mathbf{U}_{\sim\alpha}\psi$	iff	for any run $\rho = (q_i)_{i \geq 0}$ in $T_{\mathcal{A}}$ with $q = q_0$, there exists a position p in ρ such that $\mathbf{D}(p) \sim v(\alpha)$, $p \models_v \psi$ and $p' \models_v \varphi$ for all $p' < p$

Second type formulae:

$q \models_v \theta \sim \beta$	iff	$v(\theta) \sim v(\beta)$
$q \models_v \neg f$	iff	$q \not\models_v f$
$q \models_v f \vee g$	iff	$q \models_v f$ or $q \models_v g$
$q \models_v \exists\theta f$	iff	there exists $c \in \mathbb{N}$ such that $q \models_{v'} f$ where v' is defined on P_f by $v' = v$ on $P_{\exists\theta f}$ and $v'(\theta) = c$

Table 1: Semantics of PTCTL

3 Parametric Timed CTL Logic

Let $P = \{\theta_1, \dots, \theta_m\}$ be a fixed set of *parameters*. A *parameter valuation* for P is a function $v : P \rightarrow \mathbb{N}$ which assigns a natural number to each parameter $\theta \in P$. In the sequel, α means any element of $P \cup \mathbb{N}$ and β means any linear term $\sum_{i \in I} c_i \theta_i + c$, with $c_i, c \in \mathbb{N}$ and $I \subseteq \{1, \dots, m\}$. A parameter valuation v is naturally extended to linear terms by defining $v(c) = c$ for any $c \in \mathbb{N}$.

The *syntax* of *Parametric Timed CTL logic*, PTCTL logic for short, is defined in two steps: we first define formulae of first type, and then formulae of second type. We propose two logics, the discrete PTCTL logic and the dense PTCTL logic. The first one is dedicated to discrete-timed automata whereas the second one is dedicated to dense-timed automata. Notation σ means any atomic proposition $\sigma \in \Sigma$.

Definition 4 *Discrete* PTCTL formulae φ of *first type* are given by the following grammar

$$\varphi ::= \sigma \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists\bigcirc\varphi \mid \varphi\exists\mathbf{U}_{\sim\alpha}\varphi \mid \varphi\forall\mathbf{U}_{\sim\alpha}\varphi$$

and formulae f of *second type* by

$$f ::= \varphi \mid \theta \sim \beta \mid \neg f \mid f \vee f \mid \exists\theta f$$

Dense PTCTL formulae are defined in the same way, except that operator $\exists\bigcirc$ is forbidden.

In this definition, in second type formula $\exists\theta f$, it is assumed that θ is a free parameter of formula f . The set of free parameters of f is denoted by P_f . Note that usual operators $\exists\mathbf{U}$ and $\forall\mathbf{U}$ are obtained as $\exists\mathbf{U}_{\geq 0}$ and $\forall\mathbf{U}_{\geq 0}$. We now give the *semantics* of PTCTL logic.

Definition 5 Let $T_{\mathcal{A}}$ be the transition graph of a discrete-timed automaton \mathcal{A} and $q = (l, x)$ be a state of $T_{\mathcal{A}}$. Let f be a discrete PTCTL formula and v be a parameter valuation on P_f . Then the *satisfaction* relation $q \models_v f$ is defined inductively as indicated in Table 1.

If \mathcal{A} is a dense-timed automaton and f is a dense PTCTL formula, then the satisfaction relation is defined in the same way, except that $q \models_v \exists\bigcirc\varphi$ does not exist.

Problem 6 The *model-checking* problem is the following. Given a timed automaton \mathcal{A} and a state q of $T_{\mathcal{A}}$, given a PTCTL formula f , is there a parameter valuation v on P_f such that $q \models_v f$? The model-checking problem is called *discrete* if \mathcal{A} is a discrete-timed automaton and f a discrete PTCTL formula. It is called *dense* if \mathcal{A} is a dense-timed automaton and f a dense PTCTL formula.

4 Region Graphs

In this section we recall the definition of region graph [2]. It is usually given for dense-timed automata. It can also be applied to discrete-timed automata. Thus in the sequel $\mathcal{A} = (L, X, E, I)$ is a discrete- or a dense-timed automaton.

We first recall the usual equivalence on clock assignments and its extension to states of the transition system $T_{\mathcal{A}}$ generated by \mathcal{A} . For clock x_i , let c_i be the largest constant that x_i is compared with in any guard g of E . For $\tau \in \mathbb{T}$, $\text{fract}(\tau)$ denotes its fractional part and $\lfloor \tau \rfloor$ denotes its integral part.

Definition 7 Two clock assignments x and x' are equivalent, $x \approx x'$, iff the following conditions hold

- For any i , $1 \leq i \leq n$, either $\lfloor x_i \rfloor = \lfloor x'_i \rfloor$ or $x_i, x'_i > c_i$.
- For any $i \neq j$, $1 \leq i, j \leq n$ such that $x_i \leq c_i$ and $x_j \leq c_j$,
 1. $\text{fract}(x_i) \leq \text{fract}(x_j)$ iff $\text{fract}(x'_i) \leq \text{fract}(x'_j)$,
 2. $\text{fract}(x_i) = 0$ iff $\text{fract}(x'_i) = 0$.

The equivalence relation \approx is extended to the states of $T_{\mathcal{A}}$ as follows:

$$q = (l, x) \approx q' = (l', x') \quad \text{iff} \quad l = l' \text{ and } x \approx x'.$$

We use $[x]$ (resp. $[q]$) to denote the equivalence class to which x (resp. q) belongs. A *region* is an equivalence class $[q]$. The set of all regions is denoted by R . A region $[q]$ is called *boundary* if $q + \tau \not\approx q$ for any $\tau > 0$. It is called *unbounded* if it satisfies $q = (l, x)$ with $x_i > c_i$ for all i . It is well-known [2] that \approx is *back-stable* on Q : if $q \approx q'$ and $p \rightarrow q$, then $\exists p' \approx p$ such that $p' \rightarrow q'$. It is proved in [1] that states belonging to the same region satisfy the same set of TCTL formulae. The proof is easily adapted to PTCTL formulae.

Proposition 8 Let q, q' be two states of $T_{\mathcal{A}}$ such that $q \approx q'$. Let f be a PTCTL formula and v be a parameter valuation. Then $q \models_v f$ iff $q' \models_v f$.

Problem 6 can be thus restated as follows. The interest is obviously the finite number of regions.

Problem 9 Given a timed automaton \mathcal{A} and a region r of R , given a PTCTL formula f , is there a parameter valuation v on P_f such that $r \models_v f$?

We proceed with the definition of region graph. Given two regions $r = [q]$, $r' = [q']$ such that $r \neq r'$, we say that r' is a *successor* of r , $r' = \text{succ}(r)$, if $\exists \tau \in \mathbb{T}$, $q + \tau \in r'$, and $\forall \tau' < \tau$, $q + \tau' \in r \cup r'$.

Definition 10 Let \mathcal{A} be a timed automaton. The *region graph* $R_{\mathcal{A}} = (R, F)$ is a finite graph with R as vertex set and its edge set F defined as follows. Given two regions $r, r' \in R$, the edge (r, r') belongs to F : (i) if $q \xrightarrow{\tau} q'$ in $T_{\mathcal{A}}$, with $\tau = 0$, $r = [q]$ and $r' = [q']$, or (ii) if $r' = \text{succ}(r)$, or (iii) if $r = r'$ is an unbounded region.

We recall that the size $|R_{\mathcal{A}}|$ of the region automaton $R_{\mathcal{A}}$ is in $O(2^{|\mathcal{A}|})$ [2].

There is a simple correspondence between runs of $T_{\mathcal{A}}$ and paths in $R_{\mathcal{A}}$. More precisely, let $\rho = (q_i)_{i \geq 0}$ be a run. Consider $q_i \xrightarrow{\tau_i} q_{i+1}$. If $\tau_i = 0$ or if $[q_i] = [q_{i+1}]$ is an unbounded region, then $([q_i], [q_{i+1}])$ is an edge of $R_{\mathcal{A}}$. Otherwise, there is a path $(r_{i,k})_{1 \leq k \leq n_i}$ in $R_{\mathcal{A}}$ such that $r_{i,1} = [q_i]$, $r_{i,n_i} = [q_{i+1}]$ and $r_{i,k+1} = \text{succ}(r_{i,k})$ for all k , $1 \leq k < n_i$. This path can be empty (when $[q_i] = [q_{i+1}]$). In this way, an infinite path denoted $\pi(\rho)$ of $R_{\mathcal{A}}$ corresponds to the run ρ of $T_{\mathcal{A}}$. We say that $\pi(\rho)$ is the path *associated* to ρ . Such paths are called *progressive* since time progresses without bound along ρ (see Definition 3). On the other hand, for any progressive path of $R_{\mathcal{A}}$, we can find a corresponding run of $T_{\mathcal{A}}$ [2]. This run is not unique. Let us look at the region graph $R_{\mathcal{A}}$ when \mathcal{A} is discrete-timed. In Definition 7, only the first condition is usefull. Thus given a clock x_i , the possible values in an equivalence class are $1, 2, \dots, c_i$ and $c_i^+ = \{c \in \mathbb{N} \mid c > c_i\}$. If r, r' are two regions such that $r' = \text{succ}(r)$, then any clock assignment has been increased by 1.

5 Durations

To solve the model-checking problem (see Problems 6 and 9), we want an algorithm that, given a timed automaton \mathcal{A} and a region r of $R_{\mathcal{A}}$, given a PTCTL formula f , tests whether there exists a parameter valuation v on P_f such that $r \models_v f$.

Our approach is the following. Let $P_f = \{\theta_1, \dots, \theta_m\}$. In the case of the discrete model-checking problem, we are going to construct a formula $\Delta(f, r)$ of Presburger arithmetic with free variables $\theta_1, \dots, \theta_m$ such that $r \models_v f$ for some valuation v iff the sentence $\exists \theta_1 \dots \exists \theta_m \Delta(f, r)$ is true. Our algorithm follows because Presburger arithmetic has a decidable theory.

The *main tool* of our approach is a description, given two regions s, s' of $R_{\mathcal{A}}$, of all the possible values of duration $D(q_j)$ for any finite run from q_0 to q_j in $T_{\mathcal{A}}$ such that $[q_0] = s$, $[q_j] = s'$ (see Definition 11 below). The description is given by a Presburger arithmetic formula. The construction of $\Delta(f, r)$ is then easily performed by induction on the formula f .

The approach is exactly the same for the dense model-checking problem but with a real arithmetic instead of Presburger arithmetic.

Definition 11 Let \mathcal{A} be a timed automaton and $R_{\mathcal{A}} = (R, F)$ be its region graph. Let $s, s' \in R$ and $S \subseteq R$. Then $\lambda_{s, s'}^S$ is the set of $t \in \mathbb{T}$ such that

- there exists a finite run $\rho = (q_i)_{0 \leq i \leq j}$ in $T_{\mathcal{A}}$ with duration $t = D(q_j)$,
- let $\pi(\rho) = (r_l)_{0 \leq l \leq k}$ be the path in $R_{\mathcal{A}}$ associated with ρ , then $s = r_0$, $s' = r_k$ and $r_l \in S$ for any l , $0 \leq l < k$.

In this definition, s belongs to S and s' may not belong to S .

This section is devoted to the study of set $\lambda_{s, s'}^S$. In the next section, we solve the model-checking problem.

5.1 Discrete Time

We recall that *Presburger arithmetic*, PA for short, is the set of first-order formulae of $\langle \mathbb{N}, +, <, 0, 1 \rangle$. Terms are built from variables, the constants 0, 1, and the symbol $+$. Atomic formulae are either equations $t_1 = t_2$ or inequations $t_1 < t_2$ between terms t_1, t_2 . Formulae are built from atomic formulae using first-order quantifiers and the usual connectives. Formulae are interpreted over the natural numbers, with the usual interpretation of $+$, $<$, 0 and 1. Presburger *theory* is the set of all the sentences of Presburger arithmetic i.e., formulae without free variables. It is well-known that Presburger theory is decidable with a complexity in 3EXPTIME in the size of the sentence [13]. A set $X \subseteq \mathbb{N}$ is *definable* by a PA formula $\varphi(x)$ if X is exactly the set of assignments of variable x making formula φ true.

Proposition 12 Let $\mathcal{A} = (L, X, E, I)$ be a discrete-timed automaton and $R_{\mathcal{A}} = (R, F)$ be its region graph. Let $s, s' \in R$ and $S \subseteq R$. Then set $\lambda_{s, s'}^S$ is definable by a PA formula. The construction of the formula is effective.

Proof As \mathcal{A} is a discrete-timed automaton, its region graph satisfies the following property. Consider the edge $(r, r') \in F$. Either it corresponds to an instantaneous switch, which takes no time. Either $r' = \text{succ}(r)$ and any clock has been increased by 1 from r to r' . Or $r' = r$ is an unbounded region for which we can suppose that any clock is increased by 1 along (r, r') .

Let a be a fixed symbol meaning an increment by 1 of the clocks. We define a classical [12] automaton \mathcal{C} , as a particular subgraph of $R_{\mathcal{A}}$. It has $S \cup \{s'\}$ as set of states and $F \cap S \times (S \cup \{s'\})$ as set of transitions. Any of its transitions (r, r') is labeled by ϵ (the empty word) if it corresponds to an instantaneous switch, and by a otherwise. It has a unique initial state equal to s and a unique final state equal to s' . The standard subset construction is then applied to \mathcal{C} to get a deterministic automaton without ϵ -transitions. The resulting automaton \mathcal{C}' has the special structure of “frying pan” automaton since the only symbol labeling the transitions is a (see Figure 1). Denote its states by $\{\sigma_0, \dots, \sigma_k, \dots, \sigma_l\}$, with σ_0 the initial state. Note that \mathcal{C}' can have no cycle.

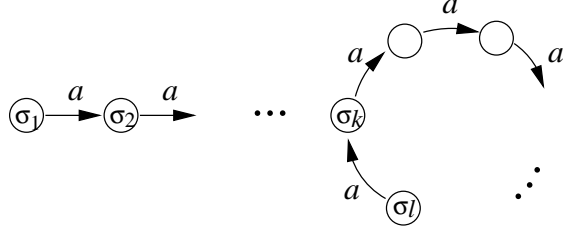


Figure 1: Frying pan automaton

It is not difficult to check that t belongs to $\lambda_{s,s'}^S$ iff t is the length of a path in \mathcal{C}' starting at σ_0 and ending at some final state σ_m . Hence if $m < k$, then $t = m$ and if $k \leq m \leq l$, then $\exists z \in \mathbb{N}, t = m + cz$ where $c = l - k + 1$ is the length of the cycle. Therefore $\lambda_{s,s'}^S$ is definable by a PA formula given by a disjunction of terms like $t = m$ or $\exists z t = m + cz$ where m, c are constants. \square

5.2 Dense Time

To deal with the dense time, we work with the *Real arithmetic*, RA, instead of Presburger arithmetic. It is the set of first-order formulae of $\langle \mathbb{R}, +, <, \mathbb{N}, 0, 1 \rangle$ where \mathbb{N} is a unary predicate. Formulae are interpreted over the reals numbers. The interpretation of \mathbb{N} is defined such that $\mathbb{N}(x)$ holds iff x is a natural number. As for Presburger arithmetic, the Real arithmetic has a decidable theory with a complexity in 3EXPTIME in the size of the sentence [17]. Note that any PA formula is a RA formula thanks to predicate $\mathbb{N}(x)$.

Proposition 13 *Suppose that \mathcal{A} is a dense-timed automaton. Then set $\lambda_{s,s'}^S$ is definable by a RA formula. The construction of the formula is effective.*

The proof of Proposition 13 is in the same vein as the proof of Proposition 12, however with some fitting due to dense time.

The region graph $R_{\mathcal{A}}$ only deals with clock assignments. In a way to also include the total time elapsed, we are going to add a new clock x_0 which is reset to 0 each time it reaches value 1. The new automaton denoted by \mathcal{A}^0 , is obtained from $\mathcal{A} = (L, X, E, I)$ in the following way. First, for each location l , a new term $x_0 \leq 1$ is added to $I(l)$. Second, let $(l, g, r, l') \in E$ be any edge of \mathcal{A} . We replace it by two edges (l, g_1, r_1, l') and (l, g_2, r_2, l') such that

- g_1 is the guard $g \wedge (x_0 < 1)$, r_1 acts as r and lets x_0 unchanged,
- g_2 is the guard $g \wedge (x_0 = 1)$, r_2 acts as r and resets x_0 to 0.

Third, for every location $l \in L$, a new edge (l, g, r, l) is added such that g is the guard $x_0 = 1$, r lets all clocks unchanged except x_0 which is reset to 0, i.e. $r((x_0, x_1, \dots, x_n)) = (0, x_1, \dots, x_n)$.

To any (finite) run ρ of $T_{\mathcal{A}}$ corresponds in a natural way a run of $T_{\mathcal{A}^0}$ such that $x_0 = 0$ at its first state. We denote this run by ρ^0 . Conversely to any (finite) run of $T_{\mathcal{A}^0}$ corresponds a run of $T_{\mathcal{A}}$ by erasing clock x_0 . Note that the run ρ^0 may be longer than ρ as clock x_0 is reset to 0 each time it reaches value 1.

The following property is immediate.

Lemma 14 *Let $\rho = (q_i)_{0 \leq i \leq j}$ and $\rho^0 = (q_i^0)_{0 \leq i \leq j^0}$. Suppose that duration $D(q_j)$ equals t . Then $\text{fract}(t)$ is equal to the value of x_0 at state $q_{j^0}^0$ and $\lfloor t \rfloor$ is equal to the number of times x_0 is reset to 0 along ρ^0 .*

Note that x_0 is reset to 0 along transition $q_i^0 \rightarrow q_{i+1}^0$ iff its value is different from 0 at state q_i^0 and equal to 0 at state q_{i+1}^0 .

The next lemma is the first step to Proposition 13. We index the equivalence relation \approx by \mathcal{A} or \mathcal{A}^0 to emphasize on the automaton that is used. Clearly if $q^0 \approx_{\mathcal{A}^0} q'^0$, then $q \approx_{\mathcal{A}} q'$.

Lemma 15 *Let $t \in \lambda_{s,s'}^S$. If $t \in]c, c + 1[$ for some $c \in \mathbb{N}$, then $]c, c + 1[\subseteq \lambda_{s,s'}^S$.*

Proof Let $t \in \lambda_{s,s'}^S$. By hypothesis, $t = c + y$ with $0 < y < 1$. Let $t' = c + y'$ such that $0 < y' < 1$. We have to prove that $t' \in \lambda_{s,s'}^S$.

By Definition 11, there exists a finite run $\rho = (q_i)_{0 \leq i \leq j}$ in $T_{\mathcal{A}}$ with duration $t = D(q_j)$ such that the associated path $\pi(\rho) = (r_l)_{0 \leq l \leq k}$ in $R_{\mathcal{A}}$ satisfies the following properties: $s = r_0$, $s' = r_k$ and $r_l \in S$ for any l , $0 \leq l < k$. Consider $\rho^0 = (q_i^0)_{0 \leq i \leq j^0}$ in $T_{\mathcal{A}^0}$ and $\pi(\rho^0) = (r_l^0)_{0 \leq l \leq k^0}$ in $R_{\mathcal{A}^0}$.

Suppose that $q_{j^0}^0 = (l, x)$. By Lemma 14, $x_0 = \text{fract}(t) = y$. We then define $q_{j^0}^0 = (l, x')$ such that $x' \approx_{\mathcal{A}^0} x$ and $x'_0 = y'$ (such an x' exists). As relation $\approx_{\mathcal{A}^0}$ is back stable, it follows that there exists a finite run $\rho'^0 = (q_i'^0)_{0 \leq i \leq j^0}$ in $T_{\mathcal{A}^0}$ such that $q_i'^0 \approx_{\mathcal{A}^0} q_i^0$, for any i , $0 \leq i \leq j^0$.

Hence $\pi(\rho^0) = \pi(\rho'^0)$. If we forget the additional clock, we get a finite run $\rho' = (q_i')_{0 \leq i \leq j}$ in $T_{\mathcal{A}}$ such that $\pi(\rho) = \pi(\rho')$.

By construction, the number of times clock x_0 is reset to 0 along ρ'^0 and ρ^0 is identical. By Lemma 14, this number is equal to $\lfloor t \rfloor = c$ and $D(q'_j) = c + y'$. This shows that $t' \in \lambda_{s,s'}^S$. \square

Proof of Proposition 13. Let $R_{\mathcal{A}^0} = (R_0, F_0)$ be the region graph of \mathcal{A}^0 . If r is a region of $R_{\mathcal{A}^0}$, we denote by \bar{r} the region of $R_{\mathcal{A}}$ obtained by erasing clock x_0 .

As in the proof of Proposition 12, we construct a classical automaton \mathcal{C} as a particular subgraph of $R_{\mathcal{A}^0}$. Here symbol a means that clock x_0 has been reset to 0. Formally, the set of states of \mathcal{C} is equal to $\{r \in R_0 \mid \bar{r} \in S \cup \{s'\}\}$ and its set of transitions is equal to $\{(r, r') \in F_0 \mid \bar{r} \in S, \bar{r}' \in S \cup \{s'\}\}$. Any transition (r, r') is labeled by a if x_0 has a non null value at region r and a null value at region r' . Otherwise it is labeled by ϵ . The unique initial state is the region r such that $\bar{r} = s$ and x_0 has value 0. The final states are regions r such that $\bar{r} = s'$. Let \mathcal{C}' be the frying pan automaton obtained when the subset construction is applied to \mathcal{C} . We denote its states by $\{\sigma_0, \dots, \sigma_k, \dots, \sigma_l\}$, with σ_0 as initial state (see Figure 1).

Let $t = \lfloor t \rfloor + y \in \lambda_{s,s'}^S$ with $y \in [0, 1[$. By definition of $\lambda_{s,s'}^S$, there exists a run $\rho = (q_i)_{0 \leq i \leq j}$ in $T_{\mathcal{A}}$ such that $t = D(q_j)$ and the path $\pi(\rho) = (r_l)_{0 \leq l \leq k}$ in $R_{\mathcal{A}}$ satisfies $s = r_0$, $s' = r_k$ and $r_l \in S$ for any l , $0 \leq l < k$. By construction of \mathcal{C} , the related path $\pi(\rho^0)$ in $R_{\mathcal{A}^0}$ can be view as a path of \mathcal{C} starting at the initial state and ending at some final state that we denote by r_f .

Now, in \mathcal{C}' , the latter path leads to a path π' starting at σ_0 and ending at some final state σ_m , the length of which is equal to $\lfloor t \rfloor$ (by Lemma 14). Moreover if the value of x_0 is different from 0 at state r_f , then for any $y' \in]0, 1[$, $t' = \lfloor t \rfloor + y' \in \lambda_{s,s'}^S$ with the same path π' in \mathcal{C}' (see Lemma 15).

Therefore set $\lambda_{s,s'}^S$ is definable by a RA formula given by a disjunction of terms like $t = m$, $\exists z \mathbb{N}(z) \wedge t = m + cz$ where m, c are constants in \mathbb{N} (as in the proof of Proposition 12) and terms like $m < t < m + 1$, $\exists z \mathbb{N}(z) \wedge m + cz < t < m + cz + 1$ (due to the above observation). \square

5.3 Additional Sets

To solve the model-checking problem, we need two auxiliary sets that we present now. We begin with the definition of set $\chi_{s,s'}^S$ which is very close to the definition of $\lambda_{s,s'}^S$.

Definition 16 Let \mathcal{A} be a timed automaton and $R_{\mathcal{A}} = (R, F)$ be its region graph. Let $s, s' \in R$ and $S \subseteq R$. Then $\chi_{s,s'}^S$ is the set of $t \in \mathbb{N}$ such that

- there exists a finite run $\rho = (q_i)_{0 \leq i \leq j}$ in $T_{\mathcal{A}}$ with duration $t = D(q_j)$,
- let $\pi(\rho) = (r_l)_{0 \leq l \leq k}$ be the path in $R_{\mathcal{A}}$ associated with ρ , then $s = r_0$, $s' = r_k$ and $r_l \in S$ for any l , $0 \leq l < k$,
- any position $p < q_j$ satisfies $D(p) < t$.

The differences with $\lambda_{s,s'}^S$ are the following. The duration t is necessarily a natural number. A third condition has been added which means that q_j is the first position in ρ with duration equal to t (remember that several positions can have the same duration in a run).

Proposition 17 Set $\chi_{s,s'}^S$ is definable by a PA formula. The construction of the formula is effective.

The proof of this proposition is given in the appendix. We end this subsection with the definition of set Pr_S .

Definition 18 Let \mathcal{A} be a timed automaton and $R_{\mathcal{A}} = (R, F)$ be its region graph. Let $S \subseteq R$ and $s \in S$. Then $s \in \text{Pr}_S$ iff there exists a progressive path in $R_{\mathcal{A}}$ with all its vertices in S and its first vertex equal to s .

Proposition 19 *It is decidable whether $s \in \text{Pr}_S$.*

Proof Suppose that \mathcal{A} is discrete-timed. The proof is again close to the proof of Proposition 12. Here \mathcal{C} has S as set of states and $F \cap S \times S$ as set of transitions. It has a unique initial state equal to s . Then $s \in \text{Pr}_S$ iff the frying pan automaton \mathcal{C}' has a cycle. The proof is similar if \mathcal{A} is dense-timed. \square

6 Model-Checking

In this section, we solve the model-checking problem. Complexity issues are discussed in Section 7.

6.1 Discrete Model-Checking

Theorem 20 *Let \mathcal{A} be a discrete-timed automaton and r be a region of $R_{\mathcal{A}} = (R, F)$. Let f be a PTCTL formula with $P_f = \{\theta_1, \dots, \theta_m\}$. Then there exists a PA formula $\Delta(f, r)$ such that $r \models_v f$ for some valuation v iff the sentence $\exists \theta_1 \dots \exists \theta_m \Delta(f, r)$ is true. The construction of $\Delta(f, r)$ is effective.*

Corollary 21 *The discrete model-checking problem is decidable.*

Proof of Theorem 20. Let $q = (l, x)$ be a state of $T_{\mathcal{A}}$ and $r = [q]$ be the region containing q . We use both notations $q \models_v f$ and $r \models_v f$ thanks to Proposition 8. The construction of $\Delta(f, r)$ is done by induction on the formula f . The set of free variables of $\Delta(f, r)$ is equal to P_f .

We consider formulae $f = \varphi$ of first type only, since the construction is then immediate for formulae f of second type. For formulae of first type, it is not difficult to check that we can work with the grammar

$$\varphi ::= \sigma \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists \bigcirc \varphi \mid \varphi \exists \text{U}_{\sim\alpha} \varphi \mid \exists \square_{\sim\alpha} \varphi$$

The semantics of the new operator is defined as follows

$$q \models_v \exists \square_{\sim\alpha} \varphi \quad \text{iff} \quad \text{there exists a run } \rho = (q_i)_{i \geq 0} \text{ in } T_{\mathcal{A}} \text{ with } q = q_0 \text{ such that } p \models_v \varphi \text{ for any position } p \text{ in } \rho \text{ with duration } D(p) \sim v(\alpha)$$

The construction of $\Delta(\varphi, r)$ is easy in the next four cases.

$$\begin{array}{ll} \varphi = \sigma & \Delta(\varphi, r) = \text{true iff } \sigma \in \mathcal{L}(l) \\ \varphi = \neg\psi & \Delta(\varphi, r) = \neg\Delta(\psi, r) \\ \varphi = \psi \vee \phi & \Delta(\varphi, r) = \Delta(\psi, r) \vee \Delta(\phi, r) \\ \varphi = \exists \bigcirc \psi & \Delta(\varphi, r) = \bigvee_{(r, r') \in F} \Delta(\psi, r') \end{array}$$

Let us study the case $\varphi = \psi \exists \text{U}_{\sim\alpha} \phi$. Suppose that $q \models_v \varphi$. By Definition 5, there exists a run $\rho = (q_i)_{i \geq 0}$ in $T_{\mathcal{A}}$ with $q = q_0$, there exists a position p in ρ with $D(p) \sim \alpha$ such that $p \models_v \phi$ and $p' \models_v \psi$ for all $p' < p$.

If $p = q$, then $q \models_v \phi$. Otherwise let $t = D(p)$ and consider the path $\pi(\rho) = (r_k)_{k \geq 0}$ in $R_{\mathcal{A}}$. We have $r_0 = r$ and $r_l = [p]$ for some $l > 0$. Denote by r' the region r_l and by S the set $\{r_k \mid 0 \leq k < l\}$. We thus observe that $t \in \lambda_{r, r'}^S$, $r' \models_v \phi$ and $s \models_v \psi$ for all $s \in S$. Moreover, r' is the first vertex of the progressive path $(r_k)_{k \geq l}$, i.e. $r' \in \text{Pr}_R$ (see Definition 18). We get for $\Delta(\varphi, r)$ the next formula

$$\begin{aligned} \Delta(\psi \exists \text{U}_{\sim\alpha} \phi, r) &= (\Delta(\phi, r) \wedge \text{Pr}_R(r) \wedge 0 \sim \alpha) \quad \vee \\ &\quad \bigvee_{r' \in R} \bigvee_{S \subseteq R} (\exists t \sim \alpha \lambda_{r, r'}^S(t) \wedge \Delta(\phi, r') \wedge \bigwedge_{s \in S} \Delta(\psi, s) \wedge \text{Pr}_R(r')) \end{aligned}$$

In this formula, $\lambda_{r, r'}^S(t)$ denotes the PA formula defining set $\lambda_{r, r'}^S$ (see Proposition 12) and $\text{Pr}_R(r)$ is true or false according to r belongs to set Pr_R or not (Proposition 19). The application of valuation v to the set P_φ of free variables of $\Delta(\varphi, r)$ leads to a sentence which is true in Presburger arithmetic.

φ	$\Delta(\varphi, r)$
$\psi \exists U \sim_\alpha \phi$	$(\Delta(\phi, r) \wedge \text{Pr}_R(r) \wedge 0 \sim \alpha) \vee$ $\bigvee_{r' \in R} \bigvee_{S \subseteq R} (\exists t \sim \alpha \lambda_{r, r'}^S(t) \wedge \Delta(\phi, r') \wedge \bigwedge_{s \in S} \Delta(\psi, s) \wedge \text{Pr}_R(r') \wedge (\neg B(r') \rightarrow \Delta(\psi, r')))$
$\exists \square \geq_\alpha \psi$	$\bigvee_{r' \in R} \bigvee_{S \subseteq R} (\chi_{r, r'}^R(\alpha) \wedge \bigwedge_{s \in S} \Delta(\psi, s) \wedge \text{Pr}_S(r'))$
$\exists \square <_\alpha \psi$	$\bigvee_{r' \in R} \bigvee_{S \subseteq R} (\chi_{r, r'}^S(\alpha) \wedge \bigwedge_{s \in S} \Delta(\psi, s) \wedge \text{Pr}_R(r') \wedge (\neg B(r') \rightarrow \Delta(\psi, r')))$
$\exists \square \leq_\alpha \psi$	$\bigvee_{S \subseteq R} \bigvee_{r' \in S} \bigvee_{r'', (r', r'') \in F} (\lambda_{r, r'}^S(\alpha) \wedge \bigwedge_{s \in S} \Delta(\psi, s) \wedge \text{Pr}_R(r''))$
$\exists \square >_\alpha \psi$	$\bigvee_{S \subseteq R} \bigvee_{r' \in R} \bigvee_{r'', (r', r'') \in F} (\lambda_{r, r'}^R(\alpha) \wedge \bigwedge_{s \in S} \Delta(\psi, s) \wedge \text{Pr}_S(r'') \wedge (\neg B(r') \rightarrow \Delta(\psi, r')))$
$\exists \square =_\alpha \psi$	$\bigvee_{S \subseteq R} \bigvee_{r', r'' \in S} \bigvee_{r''', (r', r''') \in F} (\chi_{r, r'}^R(\alpha) \wedge \lambda_{r', r''}^S(0) \wedge \bigwedge_{s \in S} \Delta(\psi, s) \wedge \text{Pr}_R(r'''))$

Table 2: formulae $\Delta(\varphi, r)$ for dense time

On the converse, it is easy to verify that if $\exists \theta_1 \dots \exists \theta_m \Delta(\psi \exists U \sim_\alpha \phi, r)$ is true, then there exists a parameter valuation v such that $r \models_v \varphi$. The main argument is Proposition 8.

We now turn to the case $\varphi = \exists \square \sim_\alpha \psi$. Since time is discrete, we can restrict the study to $\sim \in \{<, \geq, =\}$.

Suppose that $\varphi = \exists \square <_\alpha \psi$. Then $q \models_v \exists \square <_\alpha \psi$ iff there exists a run $\rho = (q_i)_{i \geq 0}$ with $q = q_0$ such that $p \models_v \psi$ for any position p in ρ with duration $D(p) < v(\alpha)$. Consider the *first* position p' in ρ such that $D(p') = v(\alpha)$. Define $r' = [p']$ and $S = \{s \in R \mid s = [p], p < p'\}$. It follows that $D(p') \in \chi_{r, r'}^S$ (see Definition 16). Moreover, for any $s \in S$, $s \models_v \psi$ and r' is the first vertex of a progressive path. Hence

$$\Delta(\exists \square <_\alpha \psi, r) = \bigvee_{r' \in R} \bigvee_{S \subseteq R} (\chi_{r, r'}^S(\alpha) \wedge \bigwedge_{s \in S} \Delta(\psi, s) \wedge \text{Pr}_R(r'))$$

Similarly for $\varphi = \exists \square \geq_\alpha \psi$, we have

$$\Delta(\exists \square \geq_\alpha \psi, r) = \bigvee_{r' \in R} \bigvee_{S \subseteq R} (\chi_{r, r'}^R(\alpha) \wedge \bigwedge_{s \in S} \Delta(\psi, s) \wedge \text{Pr}_S(r'))$$

For $\varphi = \exists \square =_\alpha \psi$, remember that there may exist several positions p in ρ such that $D(p) = v(\alpha)$. In the next formula, r'' is the region $[p'']$ such that p'' is the first position in ρ with $D(p'') = v(\alpha) + 1$.

$$\Delta(\exists \square =_\alpha \psi, r) = \bigvee_{r', r'' \in R} \bigvee_{S \subseteq R} (\chi_{r, r'}^R(\alpha) \wedge \chi_{r', r''}^S(1) \wedge \bigwedge_{s \in S} \Delta(\psi, s) \wedge \text{Pr}_R(r''))$$

□

The previous proof is simple. It is an easy translation to PA of PTCTL semantics, as soon as definability in PA of sets $\lambda_{r, r'}^S$ and $\chi_{r, r'}^S$ is established. The same situation repeats for dense model-checking in the next subsection.

6.2 Dense Model-Checking

Theorem 22 *Let \mathcal{A} be a dense-timed automaton and r be a region of $R_{\mathcal{A}} = (R, F)$. Let f be a PTCTL formula with $P_f = \{\theta_1, \dots, \theta_m\}$. Then there exists a RA formula $\Delta(f, r)$ such that $r \models_v f$ for some valuation v iff the sentence $\exists \theta_1 \dots \exists \theta_m \Delta(f, r)$ is true. The construction of $\Delta(f, r)$ is effective.*

Proof The approach is the same as in the proof of Theorem 20. It is an induction on the first type formulae given by the modified grammar (except that $\exists \square \varphi$ does not exist).

We focus on operators $\exists U \sim_\alpha$ and $\exists \square \sim_\alpha$ only. The related formulae $\Delta(\varphi, r)$ are very similar to the formulae for discrete time. They are given in Table 2. Some comments are given hereafter about the little modifications due to dense time.

Concerning $\exists U \sim_\alpha$, formula $\Delta(\varphi, r)$ is identical to the discrete case, except an additional subformula $\neg B(r') \rightarrow \Delta(\psi, r')$. Given a region r' , $B(r')$ is true iff r' is a boundary region. This new subformula is necessary because if r' is not boundary, then $r' \models_v \psi$.

Since time is dense, we can no longer restrict the study of $\exists \square \sim_\alpha$ to $\sim \in \{<, \geq, =\}$. Formula $\Delta(\varphi, r)$ for $\varphi = \exists \square \geq_\alpha \psi$ is the same as for discrete time. If $\varphi = \exists \square <_\alpha \psi$, an additional subformula $\neg B(r') \rightarrow \Delta(\psi, r')$ is again necessary.

Let us discuss the new case $\varphi = \exists \square \leq_\alpha \psi$. We need to express that r' is the region $[p']$ such that p' is the *last* position in ρ with $D(p') = v(\alpha)$. This is equivalent to say that there is a region $r'' = [p'']$ with p''

some position in ρ , there is an edge (r', r'') in $R_{\mathcal{A}}$ such that $p' \xrightarrow{\tau} p''$ and τ is *strictly* positive. We denote by $F^{>}$ the set of edges $(r', r'') \in F$ with $\tau > 0$. Thus, compared with $\Delta(\exists \square_{<\alpha} \psi, r)$, two new things appear in $\Delta(\exists \square_{\leq \alpha} \psi, r)$: first $\bigvee_{r'', (r', r'') \in F^{>}}$ as just discussed, second $\bigvee_{r \in S}$ instead of $\bigvee_{r \in R}$ to express that $r' \models_v \psi$.

The case $\varphi = \exists \square_{>\alpha} \psi$ is solved similarly. For the last case $\varphi = \exists \square_{=\alpha} \psi$, subformula $\chi_{r', r''}^S(1)$ in discrete time has to be replaced by $\bigvee_{r''', (r'', r''') \in F^{>}, \lambda_{r', r''}^S(0)$ and $r'' \in S$ in dense time. \square

Corollary 23 *The dense model-checking problem is decidable.*

7 Complexity

In this section, we study the complexity of the algorithm that we proposed in the previous section for the model-checking problem. We first establish complexity results for discrete time. We then show that these results remain valid for dense time.

Throughout this section, \mathcal{A} is a timed automaton and $R_{\mathcal{A}} = (R, F)$ its region automaton. We suppose that f is a PTCTL formula constructed by induction on the modified grammar. We also suppose that S is a subset of R and r, r' are two regions of R .

7.1 Discrete Time

To compute the complexity of the algorithm given in Theorem 20, we first study the length of formulae $\lambda_{r, r'}^S(t)$ and $\Delta(f, r)$ and the time to construct them (Propositions 24 and 25).

Proposition 24 *The PA formulae $\lambda_{r, r'}^S(t)$ and $\chi_{r, r'}^S(t)$ have a size, and can be constructed in time, bounded by $\mathcal{O}(2^{2 \cdot |R_{\mathcal{A}}|})$.*

The proof of this proposition is given in the appendix.

Proposition 25 *The PA formula $\Delta(f, r)$ has a size, and can be constructed in time, bounded by $\mathcal{O}(2^{6 \cdot |R_{\mathcal{A}}| \cdot |f|})$.*

The proof of this proposition is given in the appendix.

Corollary 26 *The discrete model-checking problem is in 4EXPTIME in the product of the sizes of $R_{\mathcal{A}}$ and f , and in 5EXPTIME in the product of the sizes of \mathcal{A} and f .*

Proof By Theorem 20, the model-checking problem reduces to checking the satisfiability of the PA formula $\Delta(f, r)$. We recall that the size of $R_{\mathcal{A}}$ is in $\mathcal{O}(2^{|A|})$ and that PA has a decidable theory with complexity 3EXPTIME in the size of the formula. The thesis follows by Proposition 25. \square

We now turn to the fragment of PTCTL logic, denoted by \exists PTCTL, where the universal quantification over parameters is prohibited. More precisely, any \exists PTCTL formula is of the form $\exists \theta_1 \cdots \exists \theta_m g$ where g is a PTCTL formula *without* quantifiers. For this fragment, we have a lower complexity.

Proposition 27 *Let f be a formula of \exists PTCTL and let $r \in R$. Then the model-checking problem for f and r can be solved in 3EXPTIME in the product of the sizes of \mathcal{A} and f .*

The proof of this proposition is given in the appendix.

7.2 Dense Time

We now show that we obtain identical complexities for dense time. The proofs are only sketched since they are similar.

Theorem 28 *The dense model-checking problem for a PTCTL is in 4EXPTIME in the sizes of $R_{\mathcal{A}}$ and f , and in 5EXPTIME in the product of the sizes of \mathcal{A} and f .*

Proof First, the RA formulae $\lambda_{r,r'}^S(t)$ and $\chi_{r,r'}^S(t)$ have a size, and can be constructed in time, bounded by $\mathcal{O}(2^{2 \cdot |R_{\mathcal{A}}|})$. Second, we can prove by induction on f that $|\Delta(f, r)|$ is bounded by $\mathcal{O}(2^{3 \cdot |R_{\mathcal{A}}| \cdot |f|} \cdot |R_{\mathcal{A}}|^{4 \cdot |f|})$. Thus $\Delta(f, r)$ has a size, and can be constructed in time, bounded by $\mathcal{O}(2^{7 \cdot |R_{\mathcal{A}}| \cdot |f|})$. Third, RA has a decidable theory with complexity 3EXPTIME in the size of the formula. The announced result follows. \square

In the next proposition, we consider a formula f of $\exists\text{PTCTL}$ logic. Note that the proof will transform the formula $\Delta(f, r)$ not only into a formula of the existential fragment of RA but also of PA.

Proposition 29 *Let f be a $\exists\text{PTCTL}$ formula and let $r \in R$. Then the model-checking problem for f and r can be solved in 3EXPTIME in the product of the sizes of \mathcal{A} and f .*

The proof of this proposition is given in the appendix.

Corollary 30 *Let f be a $\exists\text{PTCTL}$ formula and let $r \in R$. Then $\Delta(f, r)$ can be constructed as a formula of the existential fragment of PA.*

Concerning lower bounds, there is a gap which needs more research effort. Model-checking for $\exists\text{PTCTL}$ logic is at least PSPACE-HARD since the model-checking problem defined in [1] is a particular case of our problem. Model-checking problem for PTCTL logic is at least 3EXPTIME-HARD since full Presburger arithmetic is already present.

8 Related Works

The work by Wang et al in [15, 16] is closely related to our work. The logic they consider is a strict subset of the logic considered in this paper: in their works the parameters are all implicitly quantified existentially, so their logic corresponds to our fragment $\exists\text{PTCTL}$. The technique they use to establish the decidability result while ingenious is more complex than the technique that we propose in this paper. So the main contribution with regard to their work is, we feel, a simpler proof of decidability of a generalization of their logic. For $\exists\text{PTCTL}$ logic, we obtain a similar bound on the complexity of the model-checking problem.

Emerson et al have also studied an extension of TCTL with parameters in [10]. They make strong assumptions on the timed models used (their timed models are a very limited class of discrete timed automata). They also impose strong restrictions on the use of parameters in the way they constrain the scope of the temporal operators: parameters can only be used to express upper bounds. The main interest of their work is to have identified a fragment of parametric TCTL that have a polynomial time model-checking problem for the restricted class of timed models that they consider.

Alur et al [5] have also studied the extension of real-time logics with parameters but in the context of linear time. In linear time, where the satisfiability problem can be reduced to the model-checking problem, things are harder. In particular, they show that, in linear time context, the use of equality leads to undecidability of the model-checking problem.

Alur et al [4] have studied the introduction of parameters in timed automata. They show that if only one clock is constrained by a parameter then the emptiness problem for the new class of automata is decidable, but when three clocks are used, the problem becomes undecidable. To solve the problem, they also rely on the use of Presburger arithmetic.

Other researchers have also proposed the use of Presburger arithmetic (or Real arithmetic) in the context of timed automata. In particular, Comon et al [7] have studied the use of the arithmetic of the reals to express the reachability relation of timed automata. Their work is more ambitious, since they do not only consider durations between regions but also durations between individual clock valuation. Nevertheless they do not consider properties expressible in temporal logics. Along the same line, [9] have shown that it was possible to extend the results of Comon et al to analyse the binary reachability relation of discrete pushdown automata.

Finally, we can show that our notion of durations and its formalization using Presburger arithmetics (or Real arithmetic) is sufficient to answer the minimal/maximal delay problems solved in [8]. In fact, the minimal delay between two regions r and r' is characterized by the following PA formula with t as free variable:

$$\lambda_{r,r'}^R(t) \wedge \forall t' < t : \neg \lambda_{r,r'}^R(t')$$

References

- [1] R. Alur, C. Courcoubetis, and D.L. Dill. Model checking for real-time systems. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science*, pages 414–425. IEEE Computer Society Press, 1990.
- [2] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
- [3] R. Alur, T. Feder, and T.A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.
- [4] R. Alur, T.A. Henzinger, and M.Y. Vardi. Parametric real-time reasoning. In *Proceedings of the 25th Annual Symposium on Theory of Computing*, pages 592–601. ACM Press, 1993.
- [5] Rajeev Alur, Kousha Etessami, Salvatore La Torre, and Doron Peled. Parametric temporal logic for “model measuring“. *Lecture Notes in Computer Science*, 1644:159–173, 1999.
- [6] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal-logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.
- [7] H. Comon and Y. Jurski. Timed automata and the theory of real numbers. In *Proc. 10th Int. Conf. Concurrency Theory (CONCUR’99), Eindhoven, The Netherlands, Aug. 1999*, volume 1664 of *Lecture Notes in Computer Science*, pages 242–257. Springer, 1999.
- [8] Costas Courcoubetis and Mihalis Yannakakis. Minimum and maximum delay problems in real-time systems. In *Formal Methods in System Design 1(4): 385–415*. Kluwer, 1992.
- [9] Zhe Dang, Oscar H. Ibarra, Tevfik Bultan, Richard A. Kemmerer, and Jianwen Su. Binary reachability analysis of discrete pushdown timed automata. In *Computer Aided Verification*, pages 69–84, 2000.
- [10] E. Allen Emerson and Richard J. Trefler. Parametric quantitative temporal reasoning. In *Logic in Computer Science*, pages 336–343, 1999.
- [11] T.A. Henzinger. It’s about time: Real-time logics reviewed. In D. Sangiorgi and R. de Simone, editors, *CONCUR 98: Concurrency Theory*, Lecture Notes in Computer Science 1466, pages 439–454. Springer-Verlag, 1998.
- [12] Harry Lewis and Christos Papadimitriou. *Elements of the theory of computation*. Prentice Hall, 1998.
- [13] Derek Oppen. A superexponential upper bound on the complexity of presburger arithmetic. *Journal of Comput. System Sci.*, (16):323–332, 1996.
- [14] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE Computer Society Press, 1977.
- [15] Farn Wang. Timing behavior analysis for real-time systems. In *In Proceedings of the Tenth IEEE Symposium on Logic in Computer Science*, pages 112–122, 1995.
- [16] Farn Wang and Pao-Ann Hsiung. Parametric analysis of computer systems. In *Algebraic Methodology and Software Technology*, pages 539–553, 1997.
- [17] Weispfenning. Mixed real-integer linear quantifier elimination. In *ISSAC: Proceedings of the ACM SIGSAM International Symposium on Symbolic and Algebraic Computation*, 1999.

A Appendix

Proposition 17 *Set $\chi_{s,s'}^S$ is definable by a PA formula. The construction of the formula is effective.*

Proof We begin with the case of a discrete-timed automaton \mathcal{A} . The proof is similar to the proof of Proposition 12 with a construction of automaton \mathcal{C} slightly different. We add to $R_{\mathcal{A}} = (R, F)$ a new region s'_c as a copy of s' . We also add a new edge (r, s'_c) as a copy of (r, s') for any $r \in R$ such that clocks have been increased by 1 from r to s' . Then \mathcal{C} is constructed with $S \cup \{s_c\}$ as set of states and $F \cap S \times (S \cup \{s'_c\})$ as set of transitions. The rest of the proof is identical.

We now treat the dense case. The proof is similar to the proof of Proposition 13. The construction of automaton \mathcal{C} is modified as above with copy of edges along which clock x_0 has been reset to 0. As $t \in \mathbb{N}$ in the definition of $\chi_{s,s'}^S$, the situation of Lemma 15 does not occur. We thus obtain a PA formula. \square

Proposition 24 *The PA formulae $\lambda_{r,r'}^S(t)$ and $\chi_{r,r'}^S(t)$ have a size, and can be constructed in time, bounded by $\mathcal{O}(2^{2 \cdot |R_{\mathcal{A}}|})$.*

Proof To construct the formula $\lambda_{r,r'}^S(t)$, as explained in the proof of Proposition 12, we apply the subset construction to some subgraph of $R_{\mathcal{A}}$. The resulting automaton \mathcal{C}' has at most $\mathcal{O}(2^{|R_{\mathcal{A}}|})$ states. We construct for each final state of \mathcal{C}' a formula of the form $t = m$ or $\exists z t = m + cz$. So the number of these formulae is bounded by $|\mathcal{C}'|$. Furthermore, the size of constants c and m are also bounded by $|\mathcal{C}'|$. As a consequence the size of formula $\lambda_{r,r'}^S(t)$, and the time needed to construct it, is bounded by $\mathcal{O}(|\mathcal{C}'|^2) = \mathcal{O}(2^{2 \cdot |R_{\mathcal{A}}|})$. The proof is similar for formula $\chi_{r,r'}^S(t)$. \square

Proposition 25 *The PA formula $\Delta(f, r)$ has a size, and can be constructed in time, bounded by $\mathcal{O}(2^{6 \cdot |R_{\mathcal{A}}| \cdot |f|})$.*

Proof The PA formula $\Delta(f, r)$ is constructed by induction on the structure of f (see the proof of Theorem 20). We prove here by induction on f that $|\Delta(f, r)|$ is bounded by $\mathcal{O}(2^{3 \cdot |R_{\mathcal{A}}| \cdot |f|} \cdot |R_{\mathcal{A}}|^{3 \cdot |f|})$. The thesis will follow.

For the base case, f is either an atomic proposition σ or of the form $\theta \sim \beta$. Formula $\Delta(f, r)$ has the expected length.

For the induction case, we only treat the case where f is of the form $\exists \square_{=\alpha} \psi$ or $\psi \exists \text{U}_{\sim \alpha} \phi$ as the other cases are simpler or similar. We recall the definition of $\Delta(f, r)$

$$\begin{aligned} \Delta(\exists \square_{=\alpha} \psi, r) &= \bigvee_{r', r'' \in R} \bigvee_{S \subseteq R} (\chi_{r,r'}^R(\alpha) \wedge \chi_{r',r''}^S(1) \wedge \bigwedge_{s \in S} \Delta(\psi, s) \wedge \text{Pr}_R(r'')) \\ \Delta(\psi \exists \text{U}_{\sim \alpha} \phi, r) &= (\Delta(\phi, r) \wedge \text{Pr}_R(r) \wedge 0 \sim \alpha) \vee \\ &\quad \bigvee_{r' \in R} \bigvee_{S \subseteq R} (\exists t \sim \alpha \lambda_{r,r'}^S(t) \wedge \Delta(\phi, r') \wedge \bigwedge_{s \in S} \Delta(\psi, s) \wedge \text{Pr}_R(r')) \end{aligned}$$

Using the induction hypothesis and Proposition 24, the size of the above formulae are bounded as follows (with $n = |R_{\mathcal{A}}|$)

$$\begin{aligned} |\Delta(\exists \square_{=\alpha} \psi, r)| &\leq \mathcal{O}(n^2 2^n \cdot (2^{2n} + 2^{2n} + n|\Delta(\psi, s)| + 1)) \\ &\leq \mathcal{O}(n^3 2^n \cdot (2^{2n} + |\Delta(\psi, s)|)) \\ &\leq \mathcal{O}(n^3 2^n \cdot (2^{2n} \cdot |\Delta(\psi, s)|)) \\ &\leq \mathcal{O}(n^{3 \cdot (|\psi|+1)} \cdot 2^{3n \cdot (|\psi|+1)}) \\ &\leq \mathcal{O}(n^{3 \cdot |f|} \cdot 2^{3n \cdot |f|}). \end{aligned}$$

$$\begin{aligned} |\Delta(\psi \exists \text{U}_{\sim \alpha} \phi, r)| &\leq \mathcal{O}(|\Delta(\phi, r)| + 1 + 1 + n 2^n \cdot (1 + 2^{2n} + |\Delta(\phi, r')| + n|\Delta(\psi, s)| + 1)) \\ &\leq \mathcal{O}(n^2 2^n \cdot (2^{2n} + |\Delta(\phi, r')| + |\Delta(\psi, s)|)) \\ &\leq \mathcal{O}(n^3 2^n \cdot (2^{2n} \cdot |\Delta(\phi, r')| \cdot |\Delta(\psi, s)|)) \\ &\leq \mathcal{O}(n^{3 \cdot (|\phi|+|\psi|+1)} \cdot 2^{3n \cdot (|\phi|+|\psi|+1)}) \\ &\leq \mathcal{O}(n^{3 \cdot |f|} \cdot 2^{3n \cdot |f|}). \end{aligned}$$

\square

Proposition 27 *Let f be a formula of $\exists \text{PTCTL}$ and let $r \in R$. Then the model-checking problem for f and r can be solved in 3EXPTIME in the product of the sizes of \mathcal{A} and f .*

Proof We are going to show that the PA formula $\Delta(f, r)$ constructed in the proof of Theorem 20 can be transformed into a formula of the existential fragment of PA. The size of this formula will be again bounded by $\mathcal{O}(2^{6 \cdot |R_{\mathcal{A}}| \cdot |f|})$. The thesis will follow since the existential fragment of PA is known to have a decidable theory in NP [13].

The only problematic case is the existential formula $\exists t \sim \alpha \lambda_{r, r'}^S(t)$ appearing in $\Delta(\psi \exists U \sim \alpha \phi, r)$. Indeed a negation applied to it transforms $\exists t$ into $\forall t$.

Recall that formula $\lambda_{r, r'}^S(t)$ is a disjunction of formulae of the form $t = m$ or $\exists z t = m + cz$ where m and c are integer constants. So $\exists t \sim \alpha \lambda_{r, r'}^S(t)$ is a disjunction of formulae $\exists t t \sim \alpha \wedge t = m$ or $\exists t \exists z t \sim \alpha \wedge t = m + cz$.

In every case except one, existential quantifiers can be eliminated as follows.

$$\begin{aligned} \exists t t \sim \alpha \wedge t = m & \equiv m \sim \alpha \\ \exists t \exists z t > (\geq) \alpha \wedge t = m + cz & \equiv \text{true} \\ \exists t \exists z t < (\leq) \alpha \wedge t = m + cz & \equiv m < (\leq) \alpha \end{aligned}$$

It remains to study the negation of formula $\exists t \exists z t = \alpha \wedge t = m + cz$ equivalent to $\exists z \alpha = m + cz$. We consider two cases. Suppose first that $m < c$. Then $\neg(\exists z \alpha = m + cz)$ is equivalent to $\bigvee_{m' < c, m' \neq m} \alpha = m' \bmod c$ which is expressed in the existential fragment of PA as $\bigvee_{m' < c, m' \neq m} \exists z \alpha = m' + cz$. Suppose now that $m \geq c$. If we consider $m_0 = m \bmod c$, then $\exists z \alpha = m + cz$ is equivalent to $\exists z \alpha = m_0 + cz \wedge \alpha \geq m$. The latter formula can be treated as explained just before.

As the constants c and m are bounded by $|R_{\mathcal{A}}|$ when they are written in binary, we can show that the modified formula $\Delta(f, r)$ has a size still bounded by $\mathcal{O}(2^{6 \cdot |R_{\mathcal{A}}| \cdot |f|})$. \square

Proposition 29 *Let f be a \exists PTCTL formula and let $r \in R$. Then the model-checking problem for f and r can be solved in 3EXPTIME in the product of the sizes of \mathcal{A} and f .*

Proof The proof follows the same schema as for Proposition 27. The main difference is that formula $\exists t \sim \alpha \lambda_{r, r'}^S(t)$ is a disjunction of RA (instead of PA) formulae of the form

$$\begin{aligned} \exists t \quad t \sim \alpha \wedge t = m, \\ \exists t \exists z \quad t \sim \alpha \wedge \mathbb{N}(z) \wedge t = m + cz, \\ \exists t \quad t \sim \alpha \wedge m < t < m + 1, \\ \exists t \exists z \quad t \sim \alpha \wedge \mathbb{N}(z) \wedge m + cz < t < m + cz + 1 \end{aligned}$$

where m and c are constants in \mathbb{N} .

The first two formulae are treated as for Proposition 27. For the third formula, existential quantifiers can be eliminated as follows.

$$\begin{aligned} t = \alpha & : \text{false} \\ t < (\leq) \alpha & : m + 1 \leq \alpha \\ t > (\geq) \alpha & : \alpha \leq m \end{aligned}$$

For the last formula, the elimination is done in the following way.

$$\begin{aligned} t = \alpha & : \text{false} \\ t < (\leq) \alpha & : m + 1 \leq \alpha \\ t > (\geq) \alpha & : \text{true} \end{aligned}$$

So, in any case, we obtain a PA (instead of RA) formula. Looking at the proof of Theorem 22, we see that the transformed formula $\Delta(f, r)$ is formula of the existentiel fragment of Presburger arithmetic. This completes the proof. \square