An Algorithmic approach for abstracting transient states in timed systems

Mohammed Achkari Begdouri, Houda Bel Mokadem and Mohamed El Haddad
Labtic – Ecole Nationale des Sciences Appliqu´ees de Tanger,
BP 1818 Tanger Principal, Tangier, Morrocco

***Abstract*—In previous works, the timed logic TCTL was extended with importants modalities, in order to abstract *transient* states that last for less than k time units. For all modalities of this extension, called TCTL$^{\Delta}$, the decidability of the model-checking problem has been proved with an appropriate extension of Alur and Dill's region graph. But this theoretical result does not support a natural implementation due to its state-space explosion problem. This is not surprising since, even for TCTL timed logics, the model checking algorithm that is implemented in tools like UPPAAL or KRONOS is based on a so-called zone algorithm and data structures like DBMs, rather than on explicit sets of regions.**

In this paper, we propose a symbolic model-checking algorithm which computes the characteristic sets of some TCTL$^{\Delta}$ formulae and checks their truth values. This algorithm generalizes the zone algorithm for TCTL timed logics. We also present a complete correctness proof of this algorithm, and we describe its implementation using the DBM data structure.

Keywords: *Timed automata, symbolic model checking, back-ward analysis algorithm, correctness, data structures.*

I. Introduction

Timed verification. Temporal logic is a convenient formalism for specifying systems and reasoning about them. Furthermore, model-checking techniques lead to the automatic verification that a model of a system satisfies some temporal logic specification. These methods have been extended to real-time verification: systems are modeled with timed automata [6], [7] and timed logics like TCTL [3] are used to express timed specification like "any problem is followed by an alarm within 3 seconds". Analysis tools have been developped [22], [25], [30] and successfully applied to numerous case studies.

Timed temporal logics and duration properties. Along with the study of timed automata, various timed logics have been defined to extend the classical temporal logics with quantitative modalities. For example, this was done with MTL [29], [8], [31], an extension of LTL, and TCTL [9], [3], [26], where CTL modalities are augmented with time comparisons of the form $\sim c$, where $\sim$ is a comparison operator. Another related logic is the Parametrized TCTL [18] where TCTL and the timed model are in turn extended with parameters.

In another direction, since the introduction of the *duration calculus* [19] in order to express duration properties, numerous works have been devoted to the algorithmic computation of such properties for timed systems. Since *clocks*, which evolve at the rate of time (as in timed automata), are sometimes not expressive enough, hybrid variables (with multiple slopes) have been considered. The resulting model of hybrid automata has been largely studied in the subsequent years [27]. However, while some decidability results could be obtained [5], [28], using stopwatches (*i.e.* variables with slopes 0 and 1) already leads to undecidability for the reachability problem [4].

Further research has thus been devoted to weaker models where hybrid variables are only used as *observers*, *i.e.* are not tested in the automaton and thus play no role during a computation. These variables, sometimes called costs or prices in this context can be used in an optimization criterium [5], [10], [11], [16] or as constraints in temporal logic formulas. For instance, the logic WCTL [17], [15], interpreted over timed automata extended with costs, adds cost contraints on modalities: it is possible to express that a given state is reachable within a fixed cost bound.

Abstracting transient states. When practical examples are considered, the need for abstracting transient states often happens. This is the case for systems which handle variables, subject to instantaneous changes of value. This motivated the work in [12], [13], where events that do not last continuously for at least k time units could be abstracted by introducing an extension of TCTL called TCTL$^{\Delta}$. The theoretical decidability result of TCTL$^{\Delta}$ model-checking problem rely on an extension of the region graph proposed in [13]. However, the region graph is not used for implementation, but tools like UPPAAL or KRONOS use a so-called "zone algorithm". This algorithm computes on-the-fly the set of reachable symbolic states, that is pairs (q, Z) where q is a control state and Z a zone. One of the major advantage of zones is that they can be easily implemented using data structures like DBMs [24].

Contribution. The aim of this paper is to provide an implementable algorithm for TCTL$^{\Delta}$ model-checking. The algorithm we propose is an extension of the zone algorithm used for TCTL timed logics in tools like UPPAAL and KRONOS. We also provide a possible implementation of this algorithm using the DBM data structure. The main result of this paper is the proof of correctness of our algorithm. This proof uses several techniques, from properties of zones and symbolic model-checking to properties of fixed point theory.

Outline. The structure of the paper is the following: we first recall the main features of timed automata model and give definitions for the syntax and semantics of TCTL$^{\Delta}$ timed logic (Section 2); we present after some known decidability results of the TCTL$^{\Delta}$ model-checking (Section 3); we then describe the classical zone algorithm for TCTL timed logics (Section 4); we present thereafter our algorithm, we give a complete proof of its correctness (Section 5) and the following section

is devoted to explain how to implement it using the DBMs (Section 6); we end this paper with some concluding remarks (Section 7).

II. Basic Notions

Let $\mathbb{N}$ and $\mathbb{R}$ denote the sets of natural and non-negative real numbers, respectively. Let X be a set of real valued clocks. We write $\mathcal{C}(X)$ for the set of boolean expressions over atomic formulae of the form $x \sim k$ with $x \in X$, $k \in \mathbb{N}$, and $\sim \in \{<, \leq, =, \geq, >\}$. Constraints of $\mathcal{C}(X)$ are interpreted over *valuations* for clocks, i.e. mappings from X to $\mathbb{R}$. The set of valuations is denoted by $\mathbb{R}^X$. For every $v \in \mathbb{R}^X$ and $d \in \mathbb{R}$, we use $v + d$ to denote the time assignment which maps each clock $x \in X$ to the value $v(x) + d$. For every $r \subseteq X$, we write $v[r \leftarrow 0]$ for the valuation which maps each clock in r to the value 0 and agrees with v over $X \setminus r$. Let AP be a set of atomic propositions.

A. Timed Automata

Definition 1. *A* timed automaton *(TA) is a tuple* $A = \langle X, Q_A, q_{init}, \rightarrow_A, \mathsf{Inv}_A, l_A \rangle$ *where* X *is a finite set of clocks,* Q_A *is a finite set of* locations or control states *and* $q_{init} \in Q_A$ *is the* initial location. *The set* $\rightarrow_A \subseteq Q_A \times \mathcal{C}(X) \times 2^X \times Q_A$ *is a finite set of* action transitions*: for* $(q, g, r, q') \in \rightarrow_A$*,* g *is the enabling condition and* r *is a set of clocks to be reset with the transition (we write* $q \xrightarrow{g,r}_A q'$*).* $\mathsf{Inv}_A : Q_A \rightarrow \mathcal{C}(X)$ *assigns an* invariant *to each control state. Finally* $l_A : Q_A \rightarrow 2^{\mathsf{AP}}$ *labels every location with a subset of* AP.

A *state* (or *configuration*) of a TA A is a pair (q, v), where $q \in Q_A$ is the current location and $v \in \mathbb{R}^X$ is the current clock valuation. The initial state of A is (q_{init}, v_0) with $v_0(x) = 0$ for any x in X. There are two kinds of transition. From (q, v), it is possible to perform the *action transition* $q \xrightarrow{g,r}_A q'$ if $v \models g$ and $v[r \leftarrow 0] \models \mathsf{Inv}_A(q')$ and then the new configuration is $(q', v[r \leftarrow 0])$. It is also possible to let time elapse, and reach $(q, v + d)$ for some $d \in \mathbb{R}$ whenever the invariant is satisfied along the delay. Formally the semantics of a TA A is given by a Timed Transition System (TTS) $\mathcal{T}_A = (S, s_{\text{init}}, \rightarrow_{\mathcal{T}_A}, l)$ where:

- $S = \{(q, v) \mid q \in Q_A \text{ and } v \in \mathbb{R}^X \text{ s.t. } v \models \mathsf{Inv}_A(q)\}$ and $s_{\text{init}} = (q_{\text{init}}, v_0)$.
- $\rightarrow_{\mathcal{T}_A} \subseteq S \times S$ and we have $(q, v) \rightarrow_{\mathcal{T}_A} (q', v')$ iff
  - either $q' = q$, $v' = v + d$ and $v + d' \models \mathsf{Inv}_A(q)$ for any $d' \leq d$. This is a delay transition — we write $(q, v) \xrightarrow{d} (q, v + d)$ —,
  - or $\exists q \xrightarrow{g,r}_A q'$ and $v \models g$, $v' = v[r \leftarrow 0]$ and $v' \models \mathsf{Inv}_A(q')$. This is an action transition — we write $(q, v) \rightarrow_a (q', v')$.
- $l : S \rightarrow 2^{\mathsf{AP}}$ labels every state (q, v) with the subset $l_A(q)$ of AP .

An execution (or run) of A is an infinite path $s_0 \rightarrow_{\mathcal{T}_A} s_1 \rightarrow_{\mathcal{T}_A} s_2 \ldots$ in $\mathcal{T}_A$ such that (1) time diverges and (2) there are infinitely many action transitions. Note that an execution can be described as an alternating infinite sequence $s_0 \xrightarrow{d_1} \rightarrow_a s_1 \xrightarrow{d_2} \rightarrow_a \cdots$ for some $d_i \in \mathbb{R}$. Such an execution ρ goes through any configuration s' reachable from some s_i by a delay transition of duration $d \in [0, d_i]$. Let $\text{Exec}(s)$ be the set of all executions from s. With a run $\rho : (q_0, v_0) \xrightarrow{d_1} \rightarrow_a (q_1, v_1) \xrightarrow{d_2} \rightarrow_a \ldots$ of A, we associate the sequence of absolute dates defined by $t_0 = 0$ and $t_i = \sum_{j \leq i} d_j$ for $i \geq 1$, and in the sequel, we often write ρ as the sequence $((q_i, v_i, t_i))_{i \geq 0}$.

Example 1. An example of timed automaton is given below (Fig. 1), where P is an atomic proposition and x, y are clocks.

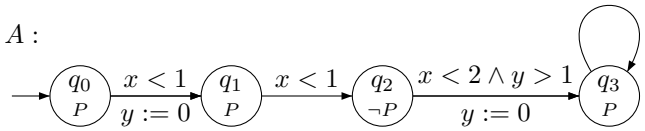

Figure 1: Example of timed automaton.

An example of run is depicted below,

$$\rho : (q_0, (0, 0)) \xrightarrow{0.1} \rightarrow_a (q_1, (0.1, 0)) \xrightarrow{0.8} \rightarrow_a (q_2, (0.9, 0.8)) \ldots$$

□

A state (q, v) can occur several times along a run ρ, the notion of *position* [1] allows us to distinguish them: every occurrence of a state is associated with a unique position. Given a position p, the corresponding state is denoted by s_p. The standard notions of prefix, suffix and subrun apply to paths in TTS: given a position $p \in \rho$, $\rho^{\leq p}$ is the prefix leading to p, $\rho^{\geq p}$ is the suffix issued from p. Finally a subrun σ from p to p' is denoted by $p \overset{\sigma}{\mapsto} p'$.

Note that the set of positions along ρ is totally ordered by $<_\rho$. Given two positions p and p', we say that p *precedes strictly* p' along ρ (written $p <_\rho p'$) iff there exists a finite subrun σ of ρ s.t. $p \overset{\sigma}{\mapsto} p'$ and σ contains at least one non null delay transition **or** one action transition (*i.e.* σ is not reduced to $\xrightarrow{0}$). We write $\sigma <_\rho p$ when for any position p' in the subrun σ, we have $p' <_\rho p$.

Given a position $p \in \rho$, the prefix $\rho^{\leq p}$ has a *duration*, $\mathsf{Time}(\rho^{\leq p})$, defined as the sum of all delays along $\rho^{\leq p}$. Since time diverges along an execution, we have: for any $t \in \mathbb{R}$, there exists $p \in \rho$ such that $\mathsf{Time}(\rho^{\leq p}) > t$.

For a subset $P \subseteq \rho$ of positions in ρ, we define a natural measure $\hat{\mu}(P) = \mu\{\mathsf{Time}(\rho^{\leq p}) \mid p \in P\}$, where μ is Lebesgue measure on the set of real numbers. In the sequel, we only use this measure when P is a subrun of ρ: in this case, for a subrun σ such that $p \overset{\sigma}{\mapsto} p'$, we simply have $\hat{\mu}(\sigma) = \mathsf{Time}(\rho^{\leq p'}) - \mathsf{Time}(\rho^{\leq p})$.

B. Definition of TCTL$^\Delta$.

The syntax of TCTL was extended in [13] to express that a formula holds everywhere except on subruns with duration a parameter $k \in \mathbb{N}$: TCTL$^\Delta$ is obtained by adding to TCTL the modalities $\mathsf{E\_U}^k_{\sim c}\mathsf{\_}$ and $\mathsf{A\_U}^k_{\sim c}\mathsf{\_}$, where $k \in \mathbb{N}$.

Definition 2 (Syntax of TCTL$^\Delta$)**.** TCTL$^\Delta$ *formulae are given by the following grammar:*

$$\varphi, \psi ::= P_1 \mid P_2 \mid \ldots \mid \neg\varphi \mid \varphi \wedge \psi \mid \mathsf{E}\varphi\mathsf{U}_{\sim c}\psi \mid \mathsf{A}\varphi\mathsf{U}_{\sim c}\psi \mid \mathsf{E}\varphi\mathsf{U}^k{}_{\sim c}\psi \mid \mathsf{A}\varphi\mathsf{U}^k_{\sim c}\psi$$

where $P_i \in \mathsf{AP}$*,* $\sim$ *belongs to the set* $\{<, >, \leq, \geq, =\}$ *and* $c, k \in \mathbb{N}$.

[1] Note that as it is possible to perform a sequence of action transitions in 0 t.u., we cannot replace the notion of positions by a function from f_ρ from $\mathbb{R}$ to S.

Standard abbreviations include $\top, \varphi \vee \psi, \varphi \Rightarrow \psi, \ldots$ as well as :

$\mathsf{EF}^k_{\sim c}\,\varphi \stackrel{def}{=} \mathsf{E}(\top\,\mathsf{U}^k_{\sim c}\,\varphi)$ $\quad$ $\mathsf{AF}^k_{\sim c}\,\varphi \stackrel{def}{=} \mathsf{A}(\top\,\mathsf{U}^k_{\sim c}\,\varphi)$
$\mathsf{EG}^k_{\sim c}\,\varphi \stackrel{def}{=} \neg\mathsf{AF}^k_{\sim c}\neg\varphi$ $\quad$ $\mathsf{AG}^k_{\sim c}\,\varphi \stackrel{def}{=} \neg\mathsf{EF}^k_{\sim c}\neg\varphi$

Moreover U^k stands for $\mathsf{U}^k_{\geq 0}$.

Definition 3 (Semantics of TCTL^Δ)**.** *The following clauses define when a state s of some TTS $\mathcal{T} = \langle S, s_{init}, \rightarrow, l\rangle$ satisfies a TCTL^Δ formula φ, written $s \models \varphi$, by induction over the structure of φ.*

$s \models \neg\varphi$ *iff* $s \not\models \varphi$
$s \models \varphi \wedge \psi$ *iff* $s \models \varphi$ *and* $s \models \psi$
$s \models \mathsf{E}\varphi\mathsf{U}_{\sim c}\psi$ *iff* $\exists\, \rho \in Exec(s)$ *s.t.* $\rho \models \varphi\mathsf{U}_{\sim c}\psi$
$s \models \mathsf{A}\varphi\mathsf{U}_{\sim c}\psi$ *iff* $\forall\, \rho \in Exec(s)$ *we have* $\rho \models \varphi\mathsf{U}_{\sim c}\psi$
$s \models \mathsf{E}\varphi\mathsf{U}^k_{\sim c}\psi$ *iff* $\exists\, \rho \in Exec(s)$ *s.t.* $\rho \models \varphi\mathsf{U}^k_{\sim c}\psi$
$s \models \mathsf{A}\varphi\mathsf{U}^k_{\sim c}\psi$ *iff* $\forall\, \rho \in Exec(s)$ *we have* $\rho \models \varphi\mathsf{U}^k_{\sim c}\psi$

$\rho \models \varphi\mathsf{U}_{\sim c}\psi$ *iff* $\exists p \in \rho$ *s.t.* $\mathsf{Time}(\rho^{\leq p}) \sim c \wedge s_p \models \psi \wedge \forall p' <_\rho p,\ s_{p'} \models \varphi$
$\rho \models \varphi\mathsf{U}^k_{\sim c}\psi$ *iff there exists a subrun σ along ρ, a position $p \in \sigma$ s.t.* $\mathsf{Time}(\rho^{\leq p}) \sim c \wedge \hat{\mu}(\sigma) > k \wedge \forall p' \in \sigma,\ s_{p'} \models \psi$ *and for all subrun σ' s.t. $\sigma' <_\rho p \wedge \forall p' \in \sigma',\ s_{p'} \models \neg\varphi$ we have $\hat{\mu}(\sigma') \leq k$*

Modality $\mathsf{E}\varphi\mathsf{U}^k_{\sim c}\psi$ means that it is possible to reach a sufficiently long interval ($> k$) where ψ is true, around a position at a distance $\sim c$ and, before this position, φ is everywhere true except along negligible duration subpaths ($\leq k$). Whereas modality $\mathsf{A}\varphi\mathsf{U}^k_{\sim c}\psi$ means that along any path, ψ lasts long enough ($> k$) around a position at a distance $\sim c$ and, before this position, φ is everywhere true except along negligible duration subpaths ($\leq k$).

III. Decidability result for TCTL^Δ

In this section we recall the decidability result for the TCTL^Δ model checking [13]. First, we remind that the classical notion of region proposed by Alur, Courcoubetis and Dill [3] for TCTL is also correct for TCTL^Δ. Nevertheless it needs a stronger notion of equivalence for the runs in order to preserve the truth value of TCTL^Δ formulae [13]. Then we recall that adding the modalities U^k does not increase the complexity of the verification.

A. Region graph

Given a set X of clocks and $M \in \mathbb{N}$, two valuations $v, v' \in \mathbb{R}^X$ are *M-equivalent* [3] (written $v \cong_M v'$) if:

1) for any $x \in X$ $\lfloor v(x)\rfloor = \lfloor v'(x)\rfloor$ or $(v(x) > M \wedge v'(x) > M)$,
2) for any $x, y \in X$ s.t. $v(x) \leq M$ and $v(y) \leq M$, we have: $\mathsf{frac}(v(x)) \leq \mathsf{frac}(v(y)) \Leftrightarrow \mathsf{frac}(v'(x)) \leq \mathsf{frac}(v'(y))$ and $\mathsf{frac}(v(x)) = 0 \Leftrightarrow \mathsf{frac}(v'(x)) = 0$.

An equivalence class of $\cong$ is called a *region*; and a region is called a *boundary region* if it contains valuations v s.t. the fractional part of $v(x)$ is 0, for some clock x. Given a TA A, we use M_A to denote the maximal constant occurring in A (in its guards or invariants). We write simply $\cong$ instead of $\cong_M$ when M is clear from the context.

Example 2. Consider a automaton with two clocks x and y and the constant M equal to 2. The set of regions associated with this automaton can be described by the figure beside (Fig. 2). The region drawn in gray corresponds to the valuations satisfying the following constraints:

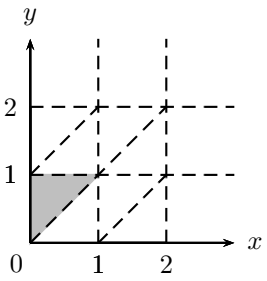

Figure 2: Example of Region.

$$0 < x < 1 \ \wedge\ 0 < y < 1 \ \wedge\ frac(y) < frac(x).$$

□

Moreover, the equivalence $\cong_{M_A}$ is consistent w.r.t. TCTL^Δ formulae [13], *i.e.* for all $\Phi \in \mathsf{TCTL}^\Delta$ and $v, v' \in \mathbb{R}^X$ s.t. $v \cong_{M_A} v'$, we have: $(q, v) \models \Phi \ \Leftrightarrow\ (q, v') \models \Phi$.

To illustrate this result, consider the formula $\Phi = \mathsf{E}\varphi\mathsf{U}^k_{\sim c}\psi$ and assume that $(q, v) \models \Phi$, *i.e.* there exists a run $\rho = ((q_i, v_i, t_i))_{i\geq 0}$ from (q, v) satisfying $\varphi\mathsf{U}^k_{\sim c}\psi$. The consistency of $\cong$ for TCTL^Δ timed logics, means that there exists an *equivalent* run ρ' from (q, v') which also satisfies $\varphi\mathsf{U}^k_{\sim c}\psi$, with v, v' are in the same region.

For this, the equivalence over runs is defined as follows [13]: Given a TA A, two runs $\rho = ((q_i, v_i, t_i))_{i\geq 0}$ and $\rho' = ((q'_i, v'_i, t'_i))_{i\geq 0}$ are equivalent (written $\rho \cong^* \rho'$) if

1) for all $i \geq 0$, $q_i = q'_i$,
2) for all $i \geq 0$, $(v_i, t_i) \cong_{M_A} (v'_i, t'_i)$,
3) for all $0 \leq j < i$, (i) $\mathsf{frac}(t_j) < \mathsf{frac}(t_i)$ iff $\mathsf{frac}(t'_j) < \mathsf{frac}(t'_i)$ and (ii) $\mathsf{frac}(t_j) = \mathsf{frac}(t_i)$ iff $\mathsf{frac}(t'_j) = \mathsf{frac}(t'_i)$.

Such that the equivalence $\cong$ is extended to pairs (v_i, t_i) as follows: $(v_i, t_i) \cong (v'_i, t'_i)$ iff (1) $v_i \cong v'_i$, (2) $\lfloor t_i\rfloor = \lfloor t'_i\rfloor$ and $\mathsf{frac}(t_i) = 0$ iff $\mathsf{frac}(t'_i) = 0$ and (3) for each clock $x \in X$, (i) $\mathsf{frac}(v_i(x)) < \mathsf{frac}(t_i)$ iff $\mathsf{frac}(v'_i(x)) < \mathsf{frac}(t'_i)$ and (ii) $\mathsf{frac}(v_i(x)) = \mathsf{frac}(t_i)$ iff $\mathsf{frac}(v'_i(x)) = \mathsf{frac}(t'_i)$.

The equivalence on runs used in [3] to prove that regions are compatible with TCTL formulae only requires conditions *(ER 1)* and *(ER 2)*. This is however not sufficient for proving the compatibility of regions with TCTL^Δ formulae. Indeed, back to the *Example 1* and consider the two following runs (Fig. 3), which are equivalent in [3]:

$\rho : (q_0, (0, 0)) \stackrel{0.1}{\rightarrow}_a (q_1, (0.1, 0)) \stackrel{0.8}{\rightarrow}_a (q_2, (0.9, 0.8)) \stackrel{0.3}{\rightarrow}_a (q_3, (1.2, 0)) \ldots$

$\rho' : (q_0, (0, 0)) \stackrel{0.8}{\rightarrow}_a (q_1, (0.8, 0)) \stackrel{0.1}{\rightarrow}_a (q_2, (0.9, 0.1)) \stackrel{1.05}{\rightarrow}_a (q_3, (1.95, 0)) \ldots$

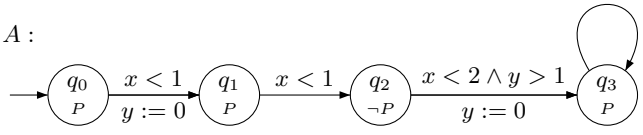

Figure 3: Example of equivalence over runs.

□

The runs ρ and ρ' satisfy conditions *(ER 1)* and *(ER 2)* but the delays spent in state q_2 where P does not hold are respectively 0.3 and 1.05, so that $\rho \models G^1 P$ whereas $\rho' \not\models G^1 P$.

This is why we need the stronger equivalence above which also requires condition *(ER 3)*. Note that the proof of the equivalence

$\cong_{M_A}$ consistency for TCTL^{Δ} timed logics is given in [13].

B. Labeling algorithm

The main result of the labeling algorithm is reducing the model-checking problem $A \models \Phi$ with a TA $A = \langle X, Q_A, q_{\text{init}}, \rightarrow_A, \mathsf{Inv}_A, l_A \rangle$ and $\Phi \in \mathsf{TCTL}^{\Delta}$, to a model-checking problem $A' \models \Phi'$ where A' is a *region graph* (*i.e.* a finite Kripke structure) and Φ' is a CTL-like formula [13].

Let X^* be the set of clocks $X \cup \{z, z_r, z_{\bar{l}}\}$. The three extra clocks are used to verify timing constraints in the formula: z is used to handle subscripts $\sim c$ in U modalities (as in TCTL model checking) and the clock $z_{\bar{l}}$ (resp z_r) is used to measure time elapsing when the left (resp. right) part in U^k modalities is false (resp true).

Let M_Φ be the maximal constant occurring in the timing constraints in Φ and k_m be the maximal k occurring in a modality U^k in Φ. Let M be $\max(M_A, M_\Phi + k_m)$.

The region graph $\mathcal{R}_{A,\Phi} = (V, \rightarrow, l, F)$ for A and Φ is defined as usual over X^* and M [3]: its set of states V is $\{(q,\gamma) | q \in Q_A \text{and} \gamma \in \mathbb{R}^{X^*}/\cong_M\}$, the transitions correspond to action transitions ($\rightarrow_a$) in A or delay transitions ($\rightarrow_t$, leading to the *successor region* denoted by $succ(\gamma)$). The states are labeled with atomic propositions AP and we also use additional propositions for the extra clocks: a state (q,γ) is labeled with the proposition $(\!|y \sim a|\!)$ with $y \in \{z, z_{\bar{l}}, z_r\}$ and $0 \leq a \leq M$, when $\gamma \models y \sim a$ (see [3], [12] for the detailed construction of $\mathcal{R}_{A,\Phi}$).

Labeling algorithm.: The algorithm consists in labeling the vertices of $\mathcal{R}_{A,\Phi}$ with the subformulae of Φ they satisfy, starting from the subformulae of length 1 and length 2 and so on.

Consider a formula Ψ of the form $\mathsf{E}\varphi_l \mathsf{U}^k_{\sim c} \varphi_r$ or $\mathsf{A}\varphi_l \mathsf{U}^k_{\sim c} \varphi_r$. At this step we know for every state (q,γ) of $\mathcal{R}_{A,\Phi}$ whether it satisfies (or not) φ_l and φ_r (*i.e.* whether any (q,v) with $v \in \gamma$ satisfies φ_l or/and φ_r). First we define a variant of $\mathcal{R}_{A,\Phi}$, called $\mathcal{R}^{\varphi_l,\varphi_r}_{A,\Phi}$, where some transitions are modified according to the truth value of φ_l and φ_r:

1) we replace the transitions $(q,\gamma) \rightarrow_t (q, succ(\gamma))$ by $(q,\gamma) \rightarrow_a (q, \gamma[z_{\bar{l}} \leftarrow 0])$ when $(q,\gamma) \models \varphi_l$, $(q, succ(\gamma)) \models \neg\varphi_l$ and $\gamma \not\models z_{\bar{l}} = 0$.
2) we replace the transitions $(q,\gamma) \rightarrow_a (q',\gamma')$ by $(q,\gamma) \rightarrow_a (q',\gamma'[z_{\bar{l}} \leftarrow 0])$ when $(q,\gamma) \models \varphi_l$, $(q',\gamma') \models \neg\varphi_l$.
3) we replace the transitions $(q,\gamma) \rightarrow_t (q, succ(\gamma))$ by $(q,\gamma) \rightarrow_a (q, \gamma[z_r \leftarrow 0])$ when $(q,\gamma) \models \neg\varphi_r$, $(q, succ(\gamma)) \models \varphi_r$ and $\gamma \not\models z_r = 0$.
4) we replace the transitions $(q,\gamma) \rightarrow_a (q',\gamma')$ by $(q,\gamma) \rightarrow_a (q',\gamma'[z_r \leftarrow 0])$ when $(q,\gamma) \models \neg\varphi_r$, $(q,\gamma') \models \varphi_r$.

Due to these changes, in $\mathcal{R}^{\varphi_l,\varphi_r}_{A,\Phi}$, the clock $z_{\bar{l}}$ (resp. z_r) measures the time elapsed since $\neg\varphi_l$ (resp. φ_r) is true : they are reset when the truth value of the corresponding formula changes. In the following we will use two abbreviations:

$$\overleftarrow{\varphi_l}^{|} \stackrel{\text{def}}{=} \varphi_l \vee (\!|z_{\bar{l}} \leq k|\!) \qquad \overleftarrow{\varphi_r} \stackrel{\text{def}}{=} \varphi_r \wedge (\!|z_r > k|\!)$$

The first one states that φ_l holds or did hold less than k t.u. ago. And the second one states that φ_r lasts for more than k t.u. We will also use the abbreviation $\overleftarrow{\neg\varphi_l}$ to denote $\neg\varphi_l \wedge (\!|z_{\bar{l}} > k|\!)$: the formula $\neg\varphi_l$ has held for more than k t.u. And we use $\overleftarrow{\neg\varphi_r}^{|}$ for $\neg\varphi_r \vee (\!|z_r \leq k|\!)$.

Therefore, the construction of the region graph $\mathcal{R}^{\varphi_l,\varphi_r}_{A,\Phi}$ allows us to decide the values of $\overleftarrow{\varphi_l}^{|}$ and $\overleftarrow{(\neg\varphi_l)}$, for any formula Ψ of the form $\mathsf{E}\varphi_l \mathsf{U}^k_{\sim c} \varphi_r$ or $\mathsf{A}\varphi_l \mathsf{U}^k_{\sim c} \varphi_r$. Furthermore, for all TA A and TCTL^{Δ} formula Ψ the labeling algorithm labels (q,γ) with Ψ in $\mathcal{R}_{A,\Phi}$ iff $(q,v) \models \Psi$ for any $v \in \gamma$ [13].

The proof of this decidability result is based on a generalization of the construction of the region graph for TCTL timed logics (as presented in [6], [7]). Instead of it, and for reasons of efficiency to avoid the state-space explosion problem, model-checkers like UPPAAL or KRONOS use a symbolic analysis algorithm to explore finitely the reachable state-space (this algorithm is called the "zone algorithm"). The implementation of this algorithm uses a data structure initially proposed by [24], the Difference Bounded Matrices, DBMs for short.

The aim of this paper is precisely to propose such an algorithm for decidable TCTL^{Δ} model-checking. The algorithm we propose is an extension of the algorithm used in UPPAAL and KRONOS. Hence, we will first recall the zone algorithm for TCTL timed logics. After this brief presentation of a so much used algorithm, we will come back to TCTL^{Δ} timed logic and present our algorithm for its symbolic model-checking. The remainder of the paper is devoted to present a complete correctness proof of our algorithm and we describe its implementation using the DBM data structure.

IV. Classical zone algorithm, State of the Art

In this section, we describe the on-the-fly analysis algorithm, which is implemented in some tools for the verification of classical timed logics [14], [21], [33], [23], [2].

A. Zones

For timed automata, the set of configurations is infinite. To check this model, it is therefore necessary to manipulate sets of configurations, and therefore to provide a symbolic representation, called zone. A zone is a set of valuations defined by a conjunction of simple constraints $x \sim c$ or $x - y \sim c$ where x and y are clocks, $\sim$ is a comparison sign, and c is a integer constant. In forward and backward analysis, the objects that will be handled are pairs (q, Z) where q is a control state of the automaton and Z a zone.

$$(q, Z) = \{(q, v) \mid v \in Z\}$$

On zones, multiple operations can be performed:

- **Future** of Z, defined by $\overrightarrow{Z} = \{v + t \mid v \in Z \wedge t \in \mathbb{R}_{\geq 0}\}$;
- **Past** of Z, defined by $\overleftarrow{Z} = \{v - t \mid v \in Z \wedge t \in \mathbb{R}_{\geq 0}\}$;
- **Intersection** of two zones, defined by $Z \cap Z' = \{v | v \in Z \wedge v \in Z'\}$;
- **Clock reset** in a zone, defined by $[Y \leftarrow 0]Z = \{[Y \leftarrow 0]v | v \in Z\}$;
- **Inverse clock reset** of a zone, defined by $[Y \leftarrow 0]^{-1}Z = \{v \mid [Y \leftarrow 0]v \in Z\}$.

These operations, defined through the first order formulas on the zones, preserve zones [32].

Example 3. Consider the zone Z drawn in (dark) gray on the figure beside (Fig. 4): Z is defined by the clock constraint

$$1 < x < 4 \wedge 2 < y < 4 \wedge x - y < 1 \wedge y - x < 2.$$

Taking the operation Future of Z, $\overrightarrow{Z}$ is drawn in light gray and in dark gray; it is defined by the clock constraint

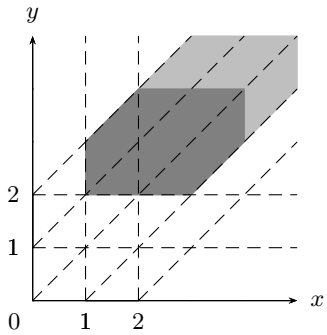

Figure 4: Example of Zone.

$$x > 1 \wedge y > 2 \wedge x - y < 1 \wedge y - x < 2.$$

□

B. The Algorithm

We give now an idea about how it is possible to check the TCTL properties [33]. The construction to be described avoids building

region graph, because such an approach would not be very effective, and there's no data structures really adapted to the regions in terms of complexity. The idea of the algorithm [33] is to calculate for each formula, its characteristic set defined as set of pairs (q, Z) where q is a control state of the automaton and Z a zone, *i.e.*

$$[\![\psi]\!] = \bigcup_{(q,Z)\models\psi} (q, Z) = \{((q, v), w) \mid (q, v) \models \psi\}$$

Where w is a valuation on clocks corresponding to the *Until* operators in the TCTL formula.

The construction is by induction on the structure of the formula:

$$\begin{array}{l}
[\![p]\!] = \{((q, v), w) \mid q \text{ labeled by } p\} \\
[\![True]\!] = \{((q, v), w) \mid q \text{ is a state }\} \\
[\![\neg\varphi]\!] = [\![True]\!] \setminus [\![\varphi]\!] \\
[\![\varphi_1 \vee \varphi_2]\!] = [\![\varphi_1]\!] \cup [\![\varphi_2]\!] \\
[\![\varphi_1 \wedge \varphi_2]\!] = [\![\varphi_1]\!] \cap [\![\varphi_2]\!]
\end{array}$$

It remains to describe the characteristic sets of formulas that have the *Until* operator. For the formula $\mathsf{E}\varphi_1\mathsf{U}_{\sim c}\varphi_2$, the characteristic set is given by the following recurrent sequence [33]:

$$[\![\mathsf{E}\varphi_1\mathsf{U}_{\sim c}\varphi_2]\!] = EU([z \leftarrow 0][\![\varphi_1]\!], [\![\varphi_2]\!] \cap [\![z \sim c]\!])$$

Where z is the clock corresponding to the operator U and $EU(R_1, R_2) = \bigcup_{i\geq 0} E_i$ with

$$\begin{cases} E_0 = R_2 \\ E_{i+1} = Pre[R_1](E_i) \cup Pre(E_i) \end{cases}$$

$Pre[R_1](E_i)$ represents the set of configurations that allow to reach E_i by letting time pass while staying in R1, while $Pre(E_i)$ represents the configurations that allow to reach E_i by taking an action transition.

A clock is attached to each U operator in the formula. it's used to handle subscripts $\sim c$ in *Until* modalities. We note that the above analysis is in fact a backward analysis. We do not describe the algorithm of $\mathsf{A}\varphi_1\mathsf{U}_{\sim c}\varphi_2$ which also uses a backward analysis, but slightly more complicated, it is described for example in [26].

V. Back to TCTL$^\Delta$ timed logic: Symbolic Model-Checking Algorithm

In this section, we propose a symbolic model-checking algorithm which computes the characteristic sets of some TCTL$^\Delta$ formulae and checks their truth values using a backward analysis. This algorithm extends the zone algorithm for TCTL timed logics. We also present a complete correctness proof of this algorithm, and we describe its implementation using the DBM data structure in the next section.

A. Modality $\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2$

For this modality, the approach we have opted is to split the semantics of formula $\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2$ in two parts, the right and the left part (as depicted in Fig. 5). The left part represents the subrun where φ_1 is true everywhere except along negligible duration subpaths ($\leq k$), until reaching the right part which represents the subrun where φ_2 lasts long enough around a position ($z \sim c$), and before this position φ_1 is true except along negligible duration subpaths.

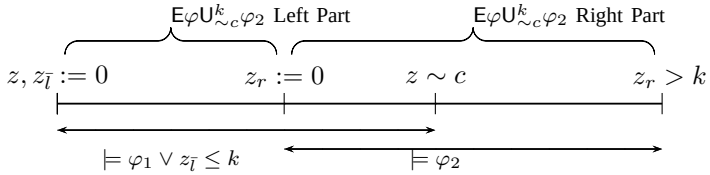

Figure 5: Illustration of $\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2$ Modality.

1) $\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2$ *Right part:*

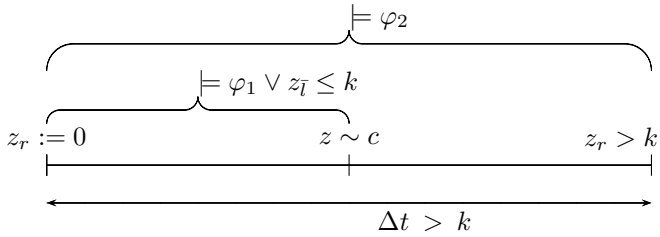

Figure 6: Illustration of $\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2$ Right part.

First case: $\sim \in \{<, \leq\}$
In this case, it is necessary and sufficient that constraint $z \sim c$ be verified at the beginning of the subrun where φ_2 lasts long enough ($> k$). Thus all the right part of $\mathsf{E}\varphi\mathsf{U}^k_{\sim c}\varphi_2$, as depicted in the figure above (Fig. 6), can be reduced andx expressed using the following TCTL formula :

$$(z \sim c) \wedge [z_r \leftarrow 0](\mathsf{E}\,\varphi_2\,\mathsf{U}\,(\varphi_2 \wedge z_r > k))$$

Second case: $\sim \in \{>, \geq, =\}$
In this case, we will split the subrun where φ_2 is true for more than k t.u into two parts, one satisfying $(\varphi_1 \vee z_{\bar{l}} \leq k)$, followed by the other which satisfying $z \sim c$ at its first position as depicted in the figure above (Fig. 6). Thus, the semantic of the $\mathsf{E}\varphi\mathsf{U}^k_{\sim c}\varphi_2$ Right part is deduced from *Definition 3* as follows:

Let s be a state of some TTS $\mathcal{T} = \langle S, s_{\text{init}}, \rightarrow, l\rangle$ which satisfies the $\mathsf{E}\varphi\mathsf{U}^k_{\sim c}\varphi_2$ Right part, written $s \models \mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)$, we have :

$$\begin{array}{ll}
s \models \mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2) & \textit{iff} \;\; \exists\, \sigma \in \text{Exec}(s) \text{ s.t. } \hat{\mu}(\sigma) > k \\
& \wedge\, \sigma \models \varphi_2 \wedge (\exists p \in \sigma \mid s_p \models z \sim c) \\
& \wedge\, \forall p' <_\sigma p,\; s_{p'} \models \varphi_1 \vee z_{\bar{l}} \leq k
\end{array}$$

Now, we propose and prove that the following sequence is increasing by inclusion, stationary, and its least upper bound represents the set of all symbolic states (*i.e.*, characteristic sets defined in section 4.2, as set of pairs (q, Z) where q is a control state of the automaton and Z a zone) that satisfying $\mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)$:

$$\begin{cases}
Y_0 & = [\![(z \sim c) \wedge (\mathsf{E}\,\varphi_2\,\mathsf{U}\,(\varphi_2 \wedge z_r > k))]\!] \\
Y_{n+1} & = Y_n \vee \Big(\Big([\![\varphi_2 \wedge \varphi_1]\!] \triangleright [z_{\bar{l}} \leftarrow 0](Y_n \wedge \neg\varphi_1) \Big) \\
& \qquad \vee \Big([\![\varphi_2 \wedge \varphi_1]\!] \triangleright (Y_n \wedge \varphi_1) \Big) \\
& \qquad \vee \Big([\![\varphi_2 \wedge (\neg\varphi_1 \wedge z_{\bar{l}} \leq k)]\!] \triangleright Y_n \Big) \Big)
\end{cases}$$

Note that z_r is reset when the stationary value of the sequence Y_n is reached, *i.e.* after that the set of symbolic states satisfying $\mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)$ is computed.

The recurrent sequence Y_n computes for each iteration the predecessors of current states represented by Y_n. As we said in section 3.2, the clock $z_{\bar{l}}$ measures time elapsing when φ_1 is false, so it will be reset at each transition from set of states satisfying φ_1 to another satisfying not $\neg\varphi_1$. Without losing information about clock $z_{\bar{l}}$, and in order to further optimize our sequence, $z_{\bar{l}}$ can also be reset when transition from set of states satisfying φ_1 to set of states satisfying φ_1, and therefore the sequence Y_n becomes as follows:

$$\begin{cases}
Y_0 & = [\![(z \sim c) \wedge (\mathsf{E}\,\varphi_2\,\mathsf{U}\,(\varphi_2 \wedge z_r > k))]\!] \\
Y_{n+1} & = Y_n \vee \Big(\Big([\![\varphi_2 \wedge \varphi_1]\!] \triangleright [z_{\bar{l}} \leftarrow 0]Y_n \Big) \\
& \qquad \vee \Big([\![\varphi_2 \wedge (\neg\varphi_1 \wedge z_{\bar{l}} \leq k)]\!] \triangleright Y_n \Big) \Big)
\end{cases}$$

Now we define the operator $\triangleright$ as follows:

Definition 4 (Predecessor operator $\triangleright$). *Given a TA A, a TTS $\mathcal{T} = \langle S, s_{init}, \rightarrow, l\rangle$, an alphabet Σ which denotes a finite set of actions and two characteristic sets Q_1 and Q_2. Calculate $Q_1 \triangleright Q_2$ is to determine:*

- $Q_1 \triangleright Q_2$:
  - All the instantaneous predecessors of Q_2 states that verify Q_1, *i.e.* the states satisfying Q_1 and can reach Q_2 by an action transition denoted $Q_1 \triangleright^a Q_2$.
  - Union, all temporal predecessors of Q_2 that verify Q_1, *i.e.* all states that can reach a state of Q_2 by a delay transition, such that all intermediates states are in Q_1.
    $q \in Q_1 \triangleright^t Q_2 \;\Leftrightarrow\; q \in Q_1 \wedge \exists t > 0$ s.t. $q + t \in Q_2$ and $\forall t' < t \;\; q + t' \in Q_1$

Back to our sequence Y_n, it can be written as follows :

$$\begin{cases} Y_0 &= [\![(z \sim c) \wedge (\mathsf{E}\,\varphi_2\,\mathsf{U}\,(\varphi_2 \wedge z_r > k))]\!] \\ Y_{n+1} &= g(Y_n) \end{cases}$$

Such that: $g(Y) = Y \vee \Big(\Big([\![\varphi_2 \wedge \varphi_1]\!] \triangleright [z_{\bar{l}} \leftarrow 0]Y \Big) \vee \Big([\![\varphi_2 \wedge (\neg\varphi_1 \wedge z_{\bar{l}} \leq k)]\!] \triangleright Y \Big) \Big)$

The particularity of the backward analysis is that the iterative calculating described by the sequence Y_n terminates, the reason is quite simple; it is fairly easy to show that if Z' is a zone and that this zone is an union of regions, then the zone $Z = g(Z')$ is not only a zone, but also is an union of regions [1]. As there's a finite number of regions, the number of pairs (q, Z) that can be computed in a backward analysis is finite.

Thus we show that the sequence Y_n is increasing by inclusion, stationary, and its least upper bound represents the characteristic set of $\mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)$:

$$[\![\mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)]\!] = [z_r \leftarrow 0]Sup\ Y_n$$

Proof: (sketch.). In order to prove this result we show at first that the least upper bound of the sequence Y_n is the least fixpoint of g.

Let be E the set of symbolic states defined as :

$$E = \{(q, Z) \mid q \in Q_A \text{ and Z is a Zone}\}$$

We know that E is a finite set, hence the power set $P(E)$ is also finite.

Furthermore, the first term of Y_n is given as the characteristic set of a TCTL formula, then we have $Y_0 \in P(E)$. As all operations in the function g preserve zones [32], so $\forall n \in \mathbb{N}\ Y_n = g^n(Y_0) \in P(E)$.

The sequence Y_n is monotonic by inclusion, because $Y_n \subseteq Y_{n+1}$ $\forall n \in \mathbb{N}$. Thus Y_n is monotonic in the finite set $P(E)$, so Y_n is stationary, *i.e.* $\exists\ r \in \mathbb{N}$ such that $\forall\ n \geq r,\ Y_n = Y_r$.

Moreover, since $(P(E), \subseteq)$ is a complete partially ordered set, then its finite subset $(W, \subseteq)$ defined as $W = \{Y_0, Y_1, ..., Y_n, ...\}$ is also a complete totally ordered set.

Also, $g : W \longmapsto W$ is a monotonic (order-preserving) function, because $\forall\ Y,\ Y' \in W$, if $Y \subseteq Y'$ we have: $g(Y) \subseteq g(Y')$ (immediate using the definition of the operator $\triangleright$).

Since in finite sets, monotonic function is always Scott-continuous, so using *Kleene's fixed-point theorem*, the least fixpoint of g is the least upper bound of the sequece $g^n(Y_0) = Y_n$, such that Y_0 is the least element of W (intersection of its elements).

$$Sup\ Y_n = \mu.Y.g(Y)$$

On another side, if we prove that $[\![\mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)]\!]$ $= [z_r \leftarrow 0]\mu.Y.g(Y)$, we have then the result we're looking for.

Let be $Q = [\![\mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)]\!]$.

1. First of all we show that Q is a fixpoint of $g : Y \rightarrow g(Y)$. We prove that:

$$\begin{cases} Q &= g(Q),\ \textit{i.e.:} \\ \mathrm{Q} &= Q \vee \Big([\![\varphi_2 \wedge \varphi_1]\!] \triangleright [z_{\bar{l}} \leftarrow 0]Q \Big) \\ & \quad \vee \Big([\![\varphi_2 \wedge (\neg\varphi_1 \wedge z_{\bar{l}} \leq k)]\!] \triangleright Q \Big) \end{cases}$$

$\subseteq$ / We have obviously $Q \subseteq g(Q)$

$\supseteq$ / Let $q \in Q \vee \Big([\![\varphi_2 \wedge \varphi_1]\!] \triangleright [z_{\bar{l}} \leftarrow 0]Q \Big) \vee \Big([\![\varphi_2 \wedge (\neg\varphi_1 \wedge z_{\bar{l}} \leq k)]\!] \triangleright Q \Big)$, we prove that $q \in Q = [\![\mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)]\!]$

$\square$ Suppose that $q \in \Big([\![\varphi_2 \wedge \varphi_1]\!] \triangleright [z_{\bar{l}} \leftarrow 0]Q \Big)$, then $q \in [\![\varphi_2 \wedge \varphi_1]\!]$ and $\exists q' \in [z_{\bar{l}} \leftarrow 0]Q$, $\exists \alpha \in \mathbb{R}^*_+ \cup \Sigma$ s.t. :
$q \triangleright^\alpha q'$, and if $\alpha = t > 0$, we have $\forall t' < t \;\; q + t' \in [\![\varphi_2 \wedge \varphi_1]\!]$

Moreover, $q' \in [z_{\bar{l}} \leftarrow 0]Q = [z_{\bar{l}} \leftarrow 0][\![\mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)]\!]$, *i.e.:*
$\exists\, \sigma \in \text{Exec}(q')$ s.t. $\hat{\mu}(\sigma) > k \wedge \sigma \models \varphi_2$
$\wedge\, (\exists p \in \sigma \mid s_p \models z \sim c) \wedge \forall p' <_\sigma p,\ s_{p'} \models \varphi_1 \vee z_{\bar{l}} \leq k$

So whatever the type of the transition α, action or delay, we can verify that the subrun σ' defined as: $\sigma' = (q \triangleright^\alpha q').\sigma$ also verify :
$\hat{\mu}(\sigma') > k \wedge \sigma' \models \varphi_2$
$\wedge\, (\exists p \in \sigma' \mid s_p \models z \sim c) \wedge \forall p' <_{\sigma'} p,\ s_{p'} \models \varphi_1 \vee z_{\bar{l}} \leq k$

Given that $\sigma' \in Exec(q)$, so we have $q \in [\![\mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)]\!]$, *i.e.:* $q \in Q$.

$\square$ Suppose that $q \in \Big([\![\varphi_2 \wedge (\neg\varphi_1 \wedge z_{\bar{l}} \leq k)]\!] \triangleright Q \Big)$, in the same manner as the previous proof, we show taht $q \in Q$.

Consequently it follows that: $\forall q \in g(Q)$ we have $q \in Q$, *i.e.:* $g(Q) \subseteq Q$. Hence the result $Q = g(Q)$, which means that $Q = [\![\mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)]\!]$ is a fixpoint of $g : Y \rightarrow g(Y)$.

2. Now we prove that $Q = [z_r \leftarrow 0]\mu.Y.g(Y)$:

$\square$ Let $q \in Q$, so: $\exists\, \sigma \in \text{Exec}(q)$ s.t. $\hat{\mu}(\sigma) > k \wedge \sigma \models \varphi_2$
$\wedge\, (\exists p \in \sigma \mid s_p \models z \sim c) \wedge \forall p' <_\sigma p,\ s_{p'} \models \varphi_1 \vee z_{\bar{l}} \leq k$

In other words, $\exists\, \sigma \in \text{Exec}(q) : \sigma = q \triangleright^{\alpha_0} q_1 \triangleright^{\alpha_1} ... \triangleright^{\alpha_{i-1}} q_i \triangleright ...$, with $\alpha_i \in \mathbb{R}^*_+ \cup \Sigma$, and z_r is reset at the beginning of σ, such that $q_i \in Y_0 = [\![(z \sim c) \wedge (\mathsf{E}\,\varphi_2\,\mathsf{U}\,(\varphi_2 \wedge z_r > k))]\!]$, and $\forall j < i$, we have:

$$\begin{cases} \mathrm{q}_j &\models \varphi_2 \wedge (\varphi_1 \vee z_{\bar{l}} \leq k\,), & if\ \alpha_j \in \Sigma \\ \mathrm{q}_j + t' &\models \varphi_2 \wedge (\varphi_1 \vee z_{\bar{l}} \leq k\,),\ \forall t' < t & if\ \alpha_j = t \in \mathbb{R}^*_+ \end{cases}$$

Let be q_{i-1} from the subrun σ, we have $q_{i-1} \triangleright^{\alpha_{i-1}} q_i$. So q_{i-1} is a predecessor of $q_i \in Y_0$, that verifies $\varphi_2 \wedge (\varphi_1 \vee z_{\bar{l}} \leq k\,)$. Then, according to the definition of the function g, $q_{i-1} \in g(Y_0) = Y_1$.

By the same reasoning we deduce that $q_{i-2} \in g^2(Y_0) = Y_2$. This is repeated until reaching $q \in [z_r \leftarrow 0]g^i(Y_0) = [z_r \leftarrow 0]Y_i$, *i.e.* $Q \subseteq [z_r \leftarrow 0]Y_i$. As $Y_i \subseteq Sup\ Y_n = \mu.Y.g(Y)$, then $Q \subseteq [z_r \leftarrow 0]\mu.Y.g(Y)$.

Moreover Q is already a fixpoint of g, *i.e.* $\mu.Y.g(Y) \subseteq Q$, which means that $[z_r \leftarrow 0]\mu.Y.g(Y) \subseteq [z_r \leftarrow 0]Q = Q$.

Hence, $Q = [\![\mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)]\!]$ is the least fixpoint of $g : Y \to g(Y)$:

$$[\![\mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)]\!] = [z_r \leftarrow 0]\mu.Y.g(Y)$$

Since we proved that the least fixpoint of g is the least upper bound of the sequence Y_n, we have finally:

$$[\![\mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)]\!] = [z_r \leftarrow 0]Sup\ Y_n$$

■

2) $\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2$ *Left part:*

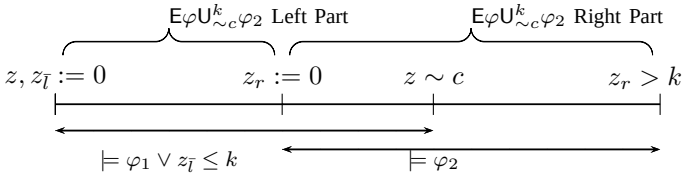

Figure 7: Illustration of $\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2$ Left part.

Now we propose and prove that the characteristic set of $\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2$ (deduced from the left part modality Fig. 7) is given by least upper bound of the following stationary and increasing (by inclusion) sequence:

$$\begin{cases} X_0 & = [\![\mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)]\!] \\ X_{n+1} & = X_n \vee \Big(\Big([\![\varphi_1]\!] \triangleright [z_{\bar{l}} \leftarrow 0]X_n \Big) \\ & \qquad \vee \Big([\![(\neg\varphi_1 \wedge z_{\bar{l}} \leq k)]\!] \triangleright X_n \Big) \Big) \end{cases}$$

Where $z,\ z_{\bar{l}}$ are reset when the stationary value of the sequence X_n is reached, *i.e.* after that the set of symbolic states satisfying $\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2$ is computed.

Proof: (sketch.). We show in the same way as the previous proof that: $[\![\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2]\!] = \mu.X.f(X)$, s.t. :

$$\begin{cases} X_0 & = & [\![\mathsf{RP}(\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2)]\!] \\ X_{n+1} & = & f(X_n) \end{cases}$$

Such that:

$$f(X) = X \vee \Big(\Big([\![\varphi_1]\!] \triangleright [z_{\bar{l}} \leftarrow 0]X \Big) \vee \Big([\![(\neg\varphi_1 \wedge z_{\bar{l}} \leq k)]\!] \triangleright X \Big) \Big).$$

Therefore we have the following result:

$$[\![\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2]\!] = [z \leftarrow 0][z_{\bar{l}} \leftarrow 0](Sup\ X_n)$$

Note that when computing iterations of the sequence X_n (resp Y_n), the stop condition is given by convergence to the fixed point of f (resp g), *i.e.* $X_{n+1} = X_n$ (resp $Y_{n+1} = Y_n$).

■

VI. Implementation of the Algorithm using DBMs

To prove that the DBMs are appropriate to implement algorithms proposed in the previous section, we will show how to compute using the DBMs the new operations on zones appearing in the TCTL^Δ model-checking algorithm. Indeed, we first recall the main features of the DBM data structure, then we give an effective method for computing the operation $Q_1 \triangleright Q_2$. We present after pseudocode for $\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2$ Model-Checking algorithm.

A. The Implementation: the DBM Data Structure

In order to implement the TCTL model-checking algorithm, we need a data structure to represent the zones and this data structure must allow to test for inclusion of zones and to compute easily the different operations used in the algorithm, that is the intersection of two zones, the past of a zone, the image of a zone by a reset and the normalization of a zone. Tools like UPPAAL or KRONOS use the data structure proposed by Dill in [24], the DBM data structure. A detailed presentation of this data structure can be found in [20].

A *difference bounded matrice* (say DBM for short) for n clocks is an $(n+1)$-square matrice of pairs:

$$(m; \prec) \in \mathbb{V} = (\mathbb{Z} \times \{<, \leq\}) \cup \{(\infty; <)\}.$$

A DBM $M = (m_{i,j}; \prec_{i,j})_{i,j=1\ldots n}$ defines the following subset of $\mathbb{R}^n$ (the clock x_0 is supposed to be always equal to zero, i.e. for each valuation v, $v(x_0) = 0$):

$$\{v : \{x_1, ..., x_n\} \longrightarrow \mathbb{R} \,|\, \forall\ 0 \leq i, j \leq n,\ v(x_i) - v(x_j) \prec_{i,j} m_{i,j}\}$$

where $\gamma < \infty$ means that γ is some real (there is no bound on it). This subset of $\mathbb{R}^n$ is a zone and will be denoted, in what follows, by $[\![M]\!]$. Each DBM on n clocks represents a zone of $\mathbb{R}^n$. Note that several DBMs can define the same zone.

Example 4. The zone defined by the equations $x_1 > 3 \wedge x_2 \leq 5 \wedge x_1 - x_2 < 4$ can be represented by the two DBMs

$$\begin{pmatrix} (0,\leq) & (-1,\leq) & (0,\leq) \\ (4,\leq) & (0,\leq) & (2,\leq) \\ (5,\leq) & (0,\leq) & (0,\leq) \end{pmatrix} \text{ and } \begin{pmatrix} (0,\leq) & (-1,\leq) & (0,\leq) \\ (4,\leq) & (0,\leq) & (2,\leq) \\ (5,\leq) & (0,\leq) & (0,\leq) \end{pmatrix}$$

Thus the DBMs are not a canonical representation of zones. Moreover, it isn't possible to test syntactically whether $[\![M]\!] = \emptyset$ or $[\![M_1]\!] = [\![M_2]\!]$. A normal form has thus been defined for representing zones. Its computation uses the Floyd-Warshall algorithm and some syntactic rewritings (see [24], [20] for a description of this procedure). In what follows, we denote by $\Phi(M)$ the normal form of M. Before stating some very important properties of the normal form, we define a total order on V in the following way: if $(m; \prec), (m'; \prec') \in \mathbb{V}$, then $(m; \prec) \leq (m'; \prec') \iff$

$$\begin{cases} m < m' \\ \text{or} \\ m = m' \text{ and either } \prec = \prec' \text{ or } \prec' = \leq \end{cases}$$

Of course, for each $m \in \mathbb{Z}$, it holds that $m < \infty$. We define $>, \geq$ and $<$ in a natural way. These orders are extended to the DBMs in the following way: let $M = (m_{i,j}; \prec_{i,j})_{i,j=0\ldots n}$ and $M' = (m'_{i,j}; \prec'_{i,j})_{i,j=0\ldots n}$ be two DBMs, then

$M \leq M' \iff$ for evry $i, j = 0...n$, $(m_{i,j}; \prec_{i,j}) \leq (m'_{i,j}; \prec'_{i,j})$.

We can now state some (very useful) properties of normal forms. If M and M' are DBMs, then:

(i) $[\![M]\!] = [\![\phi(M)]\!]$ and $\phi(M) \leq M$,
(ii) $[\![M]\!] \subseteq [\![\phi(M)]\!] \iff \phi(M) \leq M' \iff \phi(M) \leq \phi(M')$

The last point expresses the fact that the test for inclusion of zones can be checked syntactically on the normal forms of the DBMs (representing the zones).

Normal forms of DBMs can be characterized in a natural way. if $M = (m_{i,j}; \prec_{i,j})_{i,j=0\ldots n}$ is a DBM such that $[\![M]\!] \neq \emptyset$, then the two following properties are equivalent:

1) M is in normal form,
2) for every $i, j = 0...n$, for every real $-m_{j,i} \prec_{j,i} r \prec_{i,j} m_{i,j}$, there exists a valuation $v \in [\![M]\!]$ such that $v(x_j) - v(x_i) = r$ (still assuming that $v(x_0) = 0$).

This property expresses the fact that if a DBM is in normal form, then no constraint of this DBM can be tightened using the Floyd-Warshall algorithm.

Computation of Some Operations on DBMs. As we argued at the beginning of the section, the data structure used to represent zones must also be appropriate to compute all the operations on zones that are used by the TCTL model-chacking algorithm, namely future, past, intersection, image by resets and normalization. These operations on DBMs are described nicely in [20], here we recall them quickly.

Intersection. Assume that $M = (m_{i,j}; \prec_{i,j})_{i,j=1...n}$ and $M' = (m'_{i,j}; \prec'_{i,j})_{i,j=1...n}$ are two DBMs in normal form. Then, defining $M'' = (m''_{i,j}; \prec''_{i,j})_{i,j=1...n}$ by

$(m''_{i,j}; \prec''_{i,j}) = min\big((m_{i,j}; \prec_{i,j}), (m'_{i,j}; \prec'_{i,j})\big)$ for every indexes $i, j = 0...n$.

We get that $[\![M'']\!] = [\![M]\!] \cap [\![M']\!]$. Note that it can be the case that M'' is not in normal form.

Future. Assume that $M = (m_{i,j}; \prec_{i,j})_{i,j=1...n}$ is a DBM in normal form. We define the DBM $\overrightarrow{M} = (m'_{i,j}; \prec'_{i,j})_{i,j=1...n}$ by:

$$\begin{cases} (m'_{i,j}; \prec'_{i,j}) = (m_{i,j}; \prec_{i,j}) \text{ if } j \neq 0 \\ (m'_{i,0}; \prec'_{i,0}) = (\infty; <) \end{cases}$$

We get that $[\![\overrightarrow{M}]\!] = \overrightarrow{[\![M]\!]}$ and that the DBM $\overrightarrow{M}$ is in normal form.

Past. Assume that $M = (m_{i,j}; \prec_{i,j})_{i,j=1...n}$ is a DBM in normal form. We define the DBM $\overleftarrow{M} = (m'_{i,j}; \prec'_{i,j})_{i,j=1...n}$ by:

$$\begin{cases} (m'_{i,j}; \prec'_{i,j}) = (m_{i,j}; \prec_{i,j}) \text{ if } i \neq 0 \\ (m'_{0,j}; \prec'_{0,j}) = (0; \leq) \end{cases}$$

We get that $[\![\overleftarrow{M}]\!] = \overleftarrow{[\![M]\!]}$ and note that it can be the case that $\overleftarrow{M}$ is not in normal form.

Image by resets. Assume that $M = (m_{i,j}; \prec_{i,j})_{i,j=1...n}$ is a DBM in normal form. We define the DBM $M_{x_k:=0} = (m'_{i,j}; \prec'_{i,j})_{i,j=1...n}$ by:

$$\begin{cases} (m'_{i,j}; \prec'_{i,j}) = (m_{i,j}; \prec_{i,j}) & \text{if } i, j \neq k \\ (m'_{k,k}; \prec'_{k,k}) = (m'_{0,k}; \prec'_{0,k}) = (m'_{k,0}; \prec'_{k,0}) = (0; \leq) & \\ (m'_{i,k}; \prec'_{i,k}) = (m_{i,0}; \prec_{i,0}) & \text{if } i \neq k \\ (m'_{k,i}; \prec'_{k,i}) = (m_{0,i}; \prec_{0,i}) & \text{if } i \neq k \end{cases}$$

We get that $[\![M_{x_k:=0}]\!] = [x_k \leftarrow 0][\![M]\!]$ and that the DBM $M_{x_k:=0}$ is in normal form.

DBM Normal form (Zone Normalization). The DBM normal form can be computed using a shortest path algorithm. Floyd-Warshall algorithm is often used to transform DBM to canonical form. We define the DBM $\Phi(M) = (m'_{i,j}; \prec'_{i,j})_{i,j=1...n}$ by:

$(m''_{i,j}; \prec''_{i,j}) = min\big((m_{i,j}; \prec_{i,j}), (m'_{i,k}; \prec'_{i,k}) + (m'_{k,j}; \prec'_{k,j})\big)$ for every index $k = 0...n$.

We get that $\Phi(M)$ the normal form of M.
Note that to manipulate DBMs efficiently we need two operations on bounds: comparison and addition. We define that $(m_1; \prec_1) < (m_2; \prec_2)$ if $m_1 < m_2$ and $(m; <) < (m; \leq)$. Further we define addition as $(m_1; \leq) + (m_2; \leq) = (m_1 + m_2; \leq)$ and $(m_1; <) + (m_2; \prec_2) = (m_1 + m_2; <)$.

B. Computing the operator $\triangleright$

We present now an effective method for computing the operation $Q_1 \triangleright Q_2$, where Q_1 and Q_2 are characteristic sets represented by sets of symbolic states of the form $\bigcup_q (q, {Z_q}^i)$, for $i = 1, 2$. The method consists in determining all instantaneous and temporal predecessors as follows:

1) Instantaneous predecessors:

Let $q_1, q_2 \in Q_A$ be two control states, $e = q_1 \xrightarrow{g,r}_A q_2$ an edge from q_1 to q_2. Let (q_2, Z_2) be a symbolic state. We have [33] :

$$pre_e(q_2,\ Z_2) = \Big(q_1,\ g \wedge [r \leftarrow 0](Z_2)\Big)$$

i.e., $pre_e(q_2,\ Z_2)$ is the symbolic state representing predecessors, through instantaneous transition via the edge e, of states characterized by $(q_2,\ Z_2)$.

Example 5. Let A be the timed automaton depicted below (Fig. 8), with two clocks x and y.

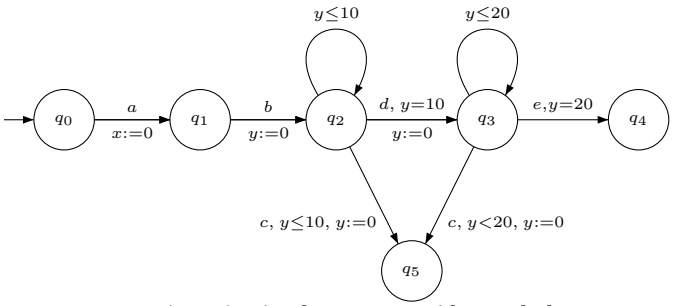

Figure 8: Timed automaton with two clocks.

The symbolic state that characterizes instantaneous predecessor of the symbolic state $(q, Z) = (q_5,\ y = 0 \wedge z > 35)$ through the edge from q_2 to q_5, is calculated as follows:

$$\begin{aligned} \text{pre}_{q_2,q_5}(q, Z) &= (q_2,\ y \leq 10 \ \wedge\ [y \leftarrow 0](y = 0 \wedge z > 35)) \\ &= (q_2,\ y \leq 10 \ \wedge\ (0 = 0 \wedge z > 35)) \\ &= (q_2,\ y \leq 10 \ \wedge\ z > 35) \end{aligned}$$

□

Therefore, for a symbolic state (q, Z), we have :

$$pre(q, Z) = \bigcup_{e \in E} pre_e(q, Z)$$

i.e. $pre(q, Z)$ is the set of symbolic states that characterizes all instantaneous predecessors of the symbolic state (q, Z).

2) Temporal predecessors:

Let (q, Z_1) and (q, Z_2) be two symbolic states. We define:

$$(q, Z_1) \triangleright_t (q, Z_2) = (q, \exists t > 0 \text{ s.t. } (Z_2 + t \ \wedge\ \forall t' < t \ \ Z_1 + t'))$$

i.e. $(q, Z_1) \triangleright_t (q, Z_2)$ represents temporal predecessors of states characterized by (q, Z_2), such that all intermediates valuations satisfy Z_1.

Furthermore, Yovine proved in [33] that $(q, Z_1) \triangleright_t (q, Z_2)$ is also a symbolic state. Indeed he demonstrated that quantifiers can be eliminated and then the result is a timing constraint.

Example 6. Let Z_1 and Z_2 be zones defined as:

$$\begin{aligned} \text{Z}_1 &= (0 < x \ \vee\ 0 < y) \\ \text{Z}_2 &= (y \leq 10 \ \wedge\ 35 < z) \end{aligned}$$

Back to the timed automaton depicted in *Example 5*, the symbolic state $(q_2, Z_1) \triangleright_t (q_2, Z_2)$ is calculated as follows :

$$\exists t > 0 \text{ s.t. } \Big((y + t \leq 10 \ \wedge \ 35 < z + t) \ \wedge \ \forall t' < t \ \ (0 < x + t' \ \vee \ 0 < y + t')\Big)$$

The universal quantifier can be written in existential quantifier. We obtain:

$$\exists t > 0 \text{ s.t. } \Big((y + t \leq 10 \ \wedge \ 35 < z + t) \ \wedge \ \neg\exists t' \geq 0 \ \ (t' < t \ \wedge \ \neg(0 < x + t' \ \vee \ 0 < y + t'))\Big)$$

Which is equivalent to

$$\exists t > 0 \text{ s.t. } \Big((y + t \leq 10 \ \wedge \ 35 < z + t) \ \wedge \ \neg\exists t' \geq 0 \ \ (t' < t \ \wedge \ x + t' \leq 0 \ \wedge \ y + t' \leq 0)\Big)$$

It is possible to eliminate the quantifier [33]

$$\exists t > 0 \text{ s.t. } \Big((y + t \leq 10 \ \wedge \ 35 < z + t) \ \wedge \ \neg(0 \leq t' < t \ \wedge \ x + t' \leq 0 \ \wedge \ y + t' \leq 0)\Big)$$

From which we deduce

$$\exists t > 0 \text{ s.t. } \Big((y + t \leq 10 \ \wedge \ 35 < z + t) \ \wedge \neg(x \leq 0 \ \wedge \ y \leq 0)\Big)$$

Whose negation is

$$\exists t > 0 \text{ s.t. } \Big((y + t \leq 10 \ \wedge \ 35 < z + t) \ \wedge (0 < x \ \vee \ 0 < y)\Big)$$

The quantifier is removed by the same procedure. The result is then:

$$(q_2, Z_1) \triangleright_t (q_2, Z_2) = \Big(q_2, \ (y \leq 10 \ \wedge \ y - z < -25)\Big) \qquad \square$$

Finally, using operators pre and $\triangleright_t$ it is possible to compute the operation $Q_1 \triangleright Q_2$. Therefore, we reduce all operations appearing in the TCTL^{Δ} model checking algorithm to known operations on zones, which are obviously implemented through the DBM data structure.

C. Pseudo-Code for $\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2$ Model-Checking algorithm

We now give the pseudo-code for Model-Checking algorithm of $\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2$ modality.

Algorithm 1 Model-Checking of $\mathsf{E}\varphi_1\mathsf{U}^k_{\sim c}\varphi_2$ Modality

```
1: function RIGHT PART(Eφ1U^k_{~c}φ2 : TCTL^Δ)
2:     // TCTL formula
3:     TargetSet := [[(z ~ c) ∧ (Eφ2U(φ2 ∧ zr > k))]];
4:
5:     repeat
6:         CurrentSet := TargetSet;
7:         TargetSet := TargetSet ∪ CurrentSet;
8: TargetSet := TargetSet ∪( [[φ2 ∧ φ1]] ▷ [z_l ← 0] CurrentSet );
9: TargetSet := TargetSet ∪( [[φ2 ∧ (¬φ1 ∧ z_l ≤ k)]]▷ CurrentSet );
10:    until TargetSet = CurrentSet
11:    return [zr ← 0]TargetSet;
12: end function
13:
14: function CHARACTERISTIC SET(Eφ1U^k_{~c}φ2 : TCTL^Δ)
15:    TargetSet := RIGHT PART(Eφ1U^k_{~c}φ2);
16:
17:    repeat
18:        CurrentSet := TargetSet;
19:        TargetSet := TargetSet ∪ CurrentSet;
20:        TargetSet := TargetSet ∪( [[φ1]] ▷ [z_l ← 0] CurrentSet );
21:        TargetSet := TargetSet ∪( [[¬φ1 ∧ z_l ≤ k]]▷ CurrentSet );
22:    until TargetSet = CurrentSet
23:    return [z ← 0][z_l ← 0]TargetSet;
24: end function
```

VII. Conclusion

In this paper, we proposed a symbolic model-checking algorithm that computes the characteristic sets of some TCTL^{Δ} formulae and checks their truth values. Moreover, we gave an accurate description of an implementation of our algorithm using zones and DBMs, the same approach as the one used in model-checkers like UPPAAL or KRONOS, in order to avoid the state-space explosion problem caused by the explicit construction of region graphs. Indeed, to get a tool from this algorithm, no much work is now necessary : the computation of each step of the algorithm is precisely described in this paper. Moreover, our algorithm appears really as an extension of the zone algorithm for TCTL timed logic, and its complexity is not more important.

Thus, this work is the link that was missing between the theoretical work did by (Houda Bel Mokadem et al.) to abstract transient events in [13] (namely decidability and expressiveness) and a tool that would deal with TCTL^{Δ} timed logic.

References

[1] L. Aceto, P. Bouyer, A. Burgueño, and K. G. Larsen. The power of reachability testing for timed automata. *Theoretical Computer Science*, 300(1–3):411–475, 2003.

[2] R. ALUR. Timed automata. In *In Proc. 11th Int. Conf. Computer Aided Verification (CAVâ99), Trento, Italy, July 1999, vol. 1633 of Lecture Notes in Computer Science, pp. 8â22*. Springer-Verlag, 1999.

[3] R. Alur, C. Courcoubetis, and D. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

[4] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer science*, 138(1):3–34, 1995.

[5] R. Alur, C. Courcoubetis, and T. A. Henzinger. Computing accumulated delays in real-time systems. *Formal Methods in System Design*, 11(2):137–156, 1997.

[6] R. Alur and D. Dill. Automata for modeling real-time systems. In *Proc. 17th International Colloquium on Automata, Languages and Programming (ICALP'90)*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, 1990.

[7] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science (TCS)*, 126(2):183–235, 1994.

[8] R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *Journal of the Association for Computing Machinery (JACM)*, 43(1):116–146, 1996.

[9] R. Alur and T. A. Henzinger. Logics and models of real-time: a survey. In *Real-Time: Theory in Practice, Proc. REX Workshop 1991*, volume 600 of *Lecture Notes in Computer Science*, pages 74–106. Springer, 1992.

[10] R. Alur, S. La Torre, and G. J. Pappas. Optimal paths in weighted timed automata. In *Proc. 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, pages 49–62. Springer, 2001.

[11] G. Behrmann, A. Fehnker, Th. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata. In *Proc. 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2001.

[12] H. Belmokadem, B. Bérard, P. Bouyer, and F. Laroussinie. A new modality for almost everywhere propeties in timed automata. In *Proc. 16th International Conference on Concurrency Theory (CONCUR05)*, volume LNCS 3653, pages 110–124. Springer, 2005.

[13] H. Belmokadem, B. Bérard, P. Bouyer, and F. Laroussinie. Timed temporal logics for abstracting transient states. In *Proc. 4th International Symposium on Automated Technology for Verification and Analysis*. Springer., 2006.

[14] J. BENGTSSON and F. LARSSON. Uppaal, a tool for automatic verification of real-time systems. In *Masterˆas thesis, Department of Computer Science, Uppsala University (Sweden)*. ISSN 0283-0574, 1996

[15] P. Bouyer, Th. Brihaye, and N. Markey. Improved undecidability results on weighted timed automata. *Information Processing Letters*, 2006. To appear.

[16] P. Bouyer, E. Brinksma, and K. G. Larsen. Staying alive as cheaply as possible. In *Proc. 7th International Workshop on Hybrid Systems: Computation and Control (HSCC'04)*, volume 2993 of *Lecture Notes in Computer Science*, pages 203–218. Springer, 2004.

[17] Th. Brihaye, V. Bruyère, and J.-F. Raskin. Model-checking for weighted timed automata. In *Proc. Joint Conference on Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System (FORMATS+FTRTFT'04)*, volume 3253 of *Lecture Notes in Computer Science*, pages 277–292. Springer, 2004.

[18] V. Bruyère, E. Dall'Olio, and J.-F. Raskin. Durations, parametric model-checking in timed automata with presburger arithmetic. In *Proc. 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS'03)*, volume 2607 of *Lecture Notes in Computer Science*, pages 687–698. Springer, 2003.

[19] Z. Chaochen, C. Hoare, and A. Ravn. A calculus of duration. *Information Processing Letters*, 40(5):269–276, 1991.

[20] E. CLARKE, O. GRUMBERG, and D. PELED. Model-checking. In *The MIT Press, Cambridge, Massachusetts.* Springer-Verlag, 1999.

[21] C. DAWS. Analyse par simulation symbolique des systÃ¨mes temporisÃ©s avec kronos. In *Research report*. Verimag, 1997.

[22] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Proc. Hybrid Systems III: Verification and Control (1995)*, volume 1066 of *Lecture Notes in Computer Science*, pages 208–219. Springer, 1996.

[23] C. Daws and S. Tripakis. Model-checking of real-time reachability properties using abstractions. In *In Proc. 4th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems, vol. 1384 of Lecture Notes in Computer Science, pp. 313â329*. Springer-Verlag., 1998.

[24] D. DILL. Timing assumptions and verification of finite-state concurrent systems. In *In Proc. the Workshop Automatic Verification Methods for Finite State Systems, vol. 407 of Lecture Notes in Computer Science, pp. 197â212*. Springer-Verlag., 1989.

[25] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: the next generation. In *Proc. 16th IEEE Real-Time Systems Symposium (RTSS'95)*, pages 56–65. IEEE Computer Society Press, 1995.

[26] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model-checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[27] Th. A. Henzinger. The theory of hybrid automata. In *Proc. 11th Annual Symposim on Logic in Computer Science (LICS'96)*, pages 278–292. IEEE Computer Society Press, 1996.

[28] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Decidable integration graphs. *Information and Computation*, 150(2):209–243, 1999.

[29] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[30] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *Journal of Software Tools for Technology Transfer (STTT)*, 1(1–2):134–152, 1997.

[31] J. Ouaknine and J. Worrell. On the decidability of Metric Temporal Logic. In *Proc. 20th IEEE Symposium on Logic in Computer Science (LICS'05)*, 2005.

[32] A. SCHRIJVER. Theory of linear and integer programming. In *Interscience Series in Discrete Mathematics and Optimization*. Wiley, 1998.

[33] S. YOVINE. Model checking timed automata. In *In School on Embedded Systems, vol. 1494 of Lecture Notes in Computer Science*. Springer-Verlag, 1998.