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 Appliquées de Tanger,
BP 1818 Tanger Principal, Tangier, Morocco

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, backward 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 developed [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 criterion [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 constraints 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 \( C(X) \) 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 \( C(X) \) 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 \( r \leq X \), 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 \( \text{AP} \) be a set of atomic propositions.

A. Timed Automata

**Definition 1.** A timed automaton (TA) is a tuple \( A = \langle X, Q, q_{\text{init}}, \text{Inv}, l_A \rangle \) where \( X \) is a finite set of clocks, \( Q \) is a finite set of locations or control states and \( q_{\text{init}} \in Q \) is the initial location. The set \( \rightarrow A \subseteq X \times C(X) \times 2^X \times Q \) 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 \( q \xrightarrow{g,r} A \)). \( \text{Inv}: Q \rightarrow C(X) \) assigns an invariant to each control state. Finally \( l_A: Q \rightarrow \text{AP} \) labels every location with a subset of \( \text{AP} \).

A state (or configuration) of a TA \( A \) is a pair \((q, v)\), where \( q \in Q \) is the current location and \( v \in \mathbb{R}^X \) is the current clock valuation. The initial state of \( A \) is \((q_{\text{init}}, v_0)\) 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 \( q \xrightarrow{g,r} A \) if \( v \models g \) and \( v[r \leftarrow 0] \models \text{Inv}(q') \) 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) \( T_A = (S, s_{\text{init}}, \rightarrow_{T_A}, l) \) where:

- \( S = \{(q, v) \mid q \in Q \text{ and } v \in \mathbb{R}^X \text{ s.t. } v \models \text{Inv}(q) \} \)
- \( s_{\text{init}} = (q_{\text{init}}, v_0) \)
- \( \rightarrow_{T_A} \subseteq S \times S \) and we have \((q, v) \rightarrow_{T_A} (q', v')\) iff
  - either \( q' = q \), \( v' = v+d \) and \( v[d' \leq d] \models \text{Inv}(q) \) 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 \( v'[r \leftarrow 0] \models \text{Inv}(q') \). This is an enabling transition — we write \((q, v) \xrightarrow{g,r} (q', v') \)
- \( l: S \rightarrow \text{AP} \) labels every state \((q, v)\) with the subset \( l_A(q) \) of \( \text{AP} \).

An execution (or run) of \( A \) is an infinite path \( s_0 \rightarrow_{T_A} s_1 \rightarrow_{T_A} s_2 \ldots \) in \( T_A \) 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 \( s_0 \xrightarrow{d_1} s_1 \xrightarrow{d_2} \ldots \) for some \( d_i \in \mathbb{R} \). Such an execution \( \rho \) goes through any configuration \( s_i \) 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 \( \rho : (q_0, v_0) \xrightarrow{d} a \) \((q_1, v_1) \xrightarrow{d} a \ldots \) 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 \( \rho \) 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.

\[
\begin{array}{c}
q_0 \xrightarrow{y < 1} q_1 \xrightarrow{y < 2 \land y > 1} q_3
\end{array}
\]

*Figure 1: Example of timed automaton.*

An example of run is depicted below,

\[
\rho : (q_0, (0,0)) \xrightarrow{y < 1} (q_1, (0,1,0)) \xrightarrow{y < 2 \land y > 1} (q_2, (0.9,0.8)) \ldots
\]

A state \((q, v)\) can occur several times along a run \( \rho \), 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 \( p \in \rho \), \( p{\rightarrow_p} \) is the prefix leading to \( p \), \( p{\rightarrow_p} \) is the suffix issued from \( p \). Finally a subrun \( \sigma \) from \( p \) to \( p' \) is denoted by \( p \xrightarrow{\sigma} p' \).

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

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

For a subset \( P \subseteq \rho \) of positions in \( \rho \), we define a natural measure \( \mu(P) = \mu(\text{Time}(p{\rightarrow_p}) \mid p \in P) \), where \( \mu \) is Lebesgue measure on the set of real numbers. In the sequel, we only use this measure when \( P \) is a subrun of \( \rho \); in this case, for a subrun \( \sigma \) such that \( p \xrightarrow{\sigma} p' \), we simply have \( \mu(\sigma) = \text{Time}(p{\rightarrow_p}) - \text{Time}(p{\rightarrow_p}) \).

B. Definition of TCTL\(^\Delta\).

The syntax of TCTL\(^\Delta\) 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 \( E_{\leq k,\sim} \) and \( A_{\leq k,\sim} \) where \( k \in \mathbb{N} \).

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

\[
\varphi, \psi ::= P_1 \mid P_2 \mid \ldots \mid \neg \varphi \mid \varphi \land \psi \mid E_{\varphi} U_{\sim} \psi \mid A_{\varphi} U_{\sim} \psi \mid E_{\varphi} U_{\leq} \psi \mid A_{\varphi} U_{\leq} \psi
\]

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

---

1Note 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 \( j_p \) from \( \mathbb{R} \) to \( S \).
Standard abbreviations include $\top, \varphi \lor \psi, \varphi \Rightarrow \psi, \ldots$ as well as:

$$EF^k_{\varphi} \triangleq E(\top U^k_{\varphi}) \quad AF^k_{\varphi} \triangleq A(\top U^k_{\varphi})$$

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

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

$$s \models \neg \varphi \iff s \not\models \varphi$$
$$s \models \varphi \land \psi \iff s \models \varphi \text{ and } s \models \psi$$
$$s \models E_{\varphi}U_{\leq c}\psi \iff \exists \rho \in \text{Exec}(s) \text{ s.t. } \rho \models U_{\leq c}\varphi \land \forall \rho' \prec \rho, s_{\rho'} \models \psi$$
$$s \models A_{\varphi}U_{\leq c}\psi \iff \forall \rho \in \text{Exec}(s) \text{ we have } \rho \models U_{\leq c}\varphi$$

$$s \models A_{\varphi}U_{\leq c}\psi \iff \forall \rho \in \text{Exec}(s) \text{ we have } \rho \models U_{\leq c}\varphi$$

Density $E_{\varphi}U_{\leq c}\psi$ means that it is possible to reach a sufficiently long interval ($> k$) where $\psi$ is true, around a position at a distance $\sim c$ and, before this position, $\varphi$ is everywhere true except along negligible duration subpaths ($\leq k$). Whereas modality $A_{\varphi}U_{\leq c}\psi$ means that along any path, $\psi$ lasts long enough ($> k$) around a position at a distance $\sim c$ and, before this position, $\varphi$ is everywhere true except along negligible duration subpaths ($\leq k$).

**III. DECIDABILITY RESULT FOR TCTL$^A$**

In this section we recall the decidability result for the TCTL$^A$ 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$^A$. Nevertheless it needs a stronger notion of equivalence for the runs in order to preserve the truth value of TCTL$^A$ 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 $x \in X$ \([v(x)] = [v'(x)] \text{ or } (v(x) > M \land v'(x) > M)$$
2) for any $x, y \in X$ s.t. $v(x) \leq M$ and $v(y) \leq M$, we have:
\[ \text{frac}(v(x)) \leq \text{frac}(v(y)) \iff \text{frac}(v'(x)) \leq \text{frac}(v'(y)) \text{ and } \text{frac}(v(x)) = 0 \iff \text{frac}(v'(x)) = 0 \]

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 $\cong$ 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:

$$0 < x < 1 \land 0 < y < 1 \land \frac{\text{frac}(x)}{\text{frac}(y)}$$

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

To illustrate this result, consider the formula $\Phi = E_{\varphi}U_{\leq c}\psi$ 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 $v_{\leq c}\varphi$. The consistency of $\cong$ for TCTL$^A$ timed logics, means that there exists an equivalent run $\rho'$ from $(q, v')$ which also satisfies $\varphi_{\leq c}\psi$, 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 $0 \leq j < i$, \((i) \text{ frac}(t_j) < \text{frac}(t_i) \text{ iff } \text{frac}(t'_j) < \text{frac}(t'_i)\)
and \((ii) \text{ frac}(t_j) = \text{frac}(t_i) \text{ iff } \text{frac}(t'_j) = \text{frac}(t'_i)\)

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

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$^A$ formulae. Indeed, back to the Example 1 and consider the two following runs (Fig. 3), which are equivalent in [3]:

$$\rho : (q_0, (0, 0)) \overset{0.1}{\longrightarrow} (q_1, (0.1, 0)) \overset{0.8}{\longrightarrow} (q_2, (0.9, 0.8)) \overset{0.3}{\longrightarrow} (q_3, (1.2, 0)) \ldots$$
$$\rho' : (q_0, (0, 0)) \overset{0.8}{\longrightarrow} (q_1, (0.8, 0)) \overset{0.1}{\longrightarrow} (q_2, (0.9, 0.1)) \overset{1.05}{\longrightarrow} (q_3, (1.95, 0)) \ldots$$

**Fig. 3: Example of equivalence over runs.**

The runs $\rho$ and $\rho'$ 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\(^A\) 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 \(A = (X, Q_A, q_{0A}, \delta_A, L, \text{In} \cdot A, L_A)\) and \(\Phi \in \text{TCTL}^A\), to a model-checking problem \(A' \models \Phi'\) where \(A'\) is a region graph (i.e. a finite Kripke structure) and \(\Phi'\) is a CTL-like formula [13].

Let \(X^*\) be the set of clocks \(X \cup \{z, z', z''\}\). 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 \(z_1\) (resp. \(z_2\)) is used to measure time elapsing when the left (resp. right) part in \(U^c\) modalities is false (resp. true).

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

The region graph \(R_{A,\Phi} = (Q, \delta, F)\) for \(A\) and \(\Phi\) is defined as usual over \(X^*\) and \(M\) [3]: its set of states \(Q\) is \(\{(q, \gamma)| q \in Q_A\text{ and }\gamma \in \mathbb{R}^{X^*}/_{\cong_M}\}\), the transitions correspond to action transitions \((\cdot, a)\) in \(A\) or delay transitions \((\cdot, \tau)\), leading to the successor region denoted by \(\text{succ}(\gamma)\). The states are labeled with atomic propositions \(AP\) and we also use additional propositions for the extra clocks: a state \((q, \gamma)\) is labeled with the proposition \(\langle y \sim a\rangle\) with \(y \in \{z, z', z''\}\) and \(0 \leq a \leq M\), when \(\gamma = y \sim a\) (see [3], [12] for the detailed construction of \(R_{A,\Phi}\)).

### Labeling algorithm:

The algorithm consists in labeling the vertices of \(R_{A,\Phi}\) with the subformulae of \(\Phi\) they satisfy, starting from the subformulae of length 1 and length 2 and so on.

Consider a formula \(\Psi\) of the form \(E_{\varphi_1} U_{\varphi_2} \varphi_3\) or \(A_{\varphi_1} U_{\varphi_2} \varphi_3\). At this step we know for every state \((q, \gamma)\) of \(R_{A,\Phi}\) whether it satisfies (or not) \(\varphi_1\) and \(\varphi_3\) (i.e. whether any \(q'\) with \(v \in \gamma\) satisfies \(\varphi_1\) or/and \(\varphi_3\)). First we define a variant of \(R_{A,\Phi}\), called \(R_{A,\Phi}^\sim\), where some transitions are modified according to the truth value of \(\varphi_1\) and \(\varphi_3\):

1. we replace the transitions \((q, \gamma) \rightarrow \langle q, \sigma, (\gamma)\rangle\) by \((q, \gamma) \rightarrow \langle q, \sigma, (\gamma)\rangle\) when \((q, \gamma) \models \varphi_1\) and \(\gamma \not\models \varphi_3\) and
2. we replace the transitions \((q, \gamma) \rightarrow \langle q', \gamma'\rangle\) by \((q, \gamma) \rightarrow \langle q', \gamma'\rangle\) when \((q, \gamma) \models \varphi_1\) and \(\gamma \not\models \varphi_3\).

3. we replace the transitions \((q, \gamma) \rightarrow \langle q, \sigma, (\gamma)\rangle\) by \((q, \gamma) \rightarrow \langle q, \sigma, (\gamma)\rangle\) when \((q, \gamma) \models \varphi_1\) and \(\gamma \not\models \varphi_3\).
4. we replace the transitions \((q, \gamma) \rightarrow \langle q', \gamma'\rangle\) by \((q, \gamma) \rightarrow \langle q', \gamma'\rangle\) when \((q, \gamma) \models \varphi_1\) and \(\gamma \not\models \varphi_3\).

Due to these changes, in \(R_{A,\Phi}^\sim\), the clock \(z_1\) (resp. \(z_2\)) measures the time elapsed since \(\varphi_1\) (resp. \(\varphi_3\)) is true: they are reset when the truth value of the corresponding formula changes. In the following we will use two abbreviations:

\[\varphi_1 \overset{\sim}{\models} \varphi_1 \lor (\langle z_1 \leq k \rangle)\]
\[\varphi_3 \overset{\sim}{\models} \varphi_3 \lor (\langle z_2 > k \rangle)\]

The first one states that \(\varphi_1\) holds or did hold less than \(k\) t.u. ago. And the second one states that \(\varphi_3\) lasts for more than \(k\) t.u. We will also use the abbreviation \(\varphi_1 \overset{\sim}{\models}\) to denote \(\varphi_1 \lor (\langle z_1 \geq k \rangle)\): the formula \(\varphi_1\) has held for more than \(k\) t.u. And we use \(\varphi_3 \overset{\sim}{\models}\) for \(\varphi_3 \lor (\langle z_2 < k \rangle)\).

Therefore, the construction of the region graph \(R_{A,\Phi}^\sim\) allows us to decide the values of \(\varphi_1\) and \(\varphi_3\), for any formula \(\Psi\) of the form \(E_{\varphi_1} U_{\varphi_2} \varphi_3\) or \(A_{\varphi_1} U_{\varphi_2} \varphi_3\). Furthermore, for all TA \(A\) and TCTL\(^A\) formula \(\Phi\) the labeling algorithm labels \((q, \gamma)\) with \(\Psi\) in \(R_{A,\Phi}\) iff \((q, \gamma) \models \Psi\) 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\(^A\) 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\(^A\) 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.

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

On zones, multiple operations can be performed:

- **Future of Z**, defined by \(\bar{Z} = \{v + t \mid v \in Z \land t \in \mathbb{R}_{\geq 0}\}\);
- **Past of Z**, defined by \(\bar{Z} = \{v - t \mid v \in Z \land t \in \mathbb{R}_{\geq 0}\}\);
- **Intersection** of two zones, defined by \(Z \cap Z' = \{v \mid v \in Z \land v \in Z'\}\);
- **Clock reset** in a zone, defined by \(Y \leftarrow 0\)\(Z = \{v \mid v \in Z\}\);
- **Inverse clock reset** of a zone, defined by \(Y \leftarrow 0^{-1}\)\(Z = \{v \mid v \in Z\}\).

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

\[1 < x < 4 \land 2 < y < 4 \land x - y < 1 \land y - x < 2\]

Taking the operation Future of \(Z, \bar{Z}\) is drawn in light gray and in dark gray; it is defined by the clock constraint.

\[x > 1 \land y > 2 \land x - y < 1 \land y - x < 2\]

Figure 4: Example of Zone.

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.

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

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:

\[
[p] = \{(q, v), w) \mid q \text{ labeled by } p\}
\]

\[
[\text{true}] = \{(q, v), w) \mid q \text{ is a state}\}
\]

\[
[\neg \varphi_1] = \text{true} \setminus [\varphi_1]
\]

\[
[\varphi_1 \lor \varphi_2] = [\varphi_1] \cup [\varphi_2]
\]

\[
[\varphi_1 \land \varphi_2] = [\varphi_1] \cap [\varphi_2]
\]

It remains to describe the characteristic sets of formulas that have the Until operator. For the formula \(E_\varphi U_\varphi \varphi_2\), the characteristic set is given by the following recurrent sequence [33]:

\[
[E_\varphi U_\varphi \varphi_2] = E(U [(z \preceq 0) \left[ \varphi_1 \right] \land [z \sim c])
\]

Where \(z\) is the clock corresponding to the operator \(U\) and \(E(U_{R1, R2}) = \bigcup_{i \geq 0} E_i\) with

\[
E_0 = R_2
\]

\[
E_{i+1} = \text{Pre}[R_i](E_i) \cup \text{Pre}(E_i)
\]

\(\text{Pre}[R_i](E_i)\) represents the set of configurations that allow to reach \(E_i\) by letting time pass while staying in \(R_i\), while \(\text{Pre}(E_i)\) 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 \(A_\varphi U_\varphi \varphi_2\) which also uses a backward analysis, but slightly more complicated, it is described for example in [26].

V. BACK TO TCTL\(^A\) TIMED LOGIC: SYMBOLIC MODEL-CHECKING ALGORITHM

In this section, we propose a symbolic model-checking algorithm which computes the characteristic sets of some TCTL\(^A\) 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 \(E \varphi \varphi_1 U^k_{\prec c} \varphi_2\)

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

\[
\text{Left Part}
\]

\[
\text{Right Part}
\]

\[
\varphi_1 \land \varphi_2 \implies \varphi_2
\]

\[
\varphi_1 \lor z_2 \leq k
\]

\[
z_{r} = 0 \quad z_1 \sim c \quad z_{r} > k
\]

\[
\Delta t > k
\]

**Figure 6: Illustration of \(E \varphi_1 U^k_{\prec c} \varphi_2\) 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 \(\varphi_2\) lasts long enough \((> k)\). Thus all the right part of \(E \varphi_1 U^k_{\prec c} \varphi_2\), as depicted in the figure above (Fig. 6), can be reduced and expressed using the following TCTL formula:

\[
(z \sim c) \land [z_r \leftarrow 0](E \varphi_2 U \varphi_1 U \varphi_2)
\]

**Second case:** \(\sim \in \{>, \geq\}\)

In this case, we will split the subrun where \(\varphi_2\) is true for more than \(k\) time into two parts, one satisfying \((\varphi_1 \land z \sim c)\), 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 \(E \varphi_1 U^k_{\prec c} \varphi_2\) Right part is deduced from Definition 3 as follows:

Let \(s\) be a state of some TTS \(T = (\langle S, \delta, \rightarrow, I \rangle)\) which satisfies the \(E \varphi_1 U^k_{\prec c} \varphi_2\) Right part, written \(s \models \text{RP}(E \varphi_1 U^k_{\prec c} \varphi_2)\), we have:

\[
\exists \sigma \in \text{Exec}(s) \text{ s.t. } \bar{\mu}(\sigma) > k
\]

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 \(\text{RP}(E \varphi_1 U^k_{\prec c} \varphi_2)\):

\[
Y_0 = [z \sim c) \land (E \varphi_2 \varphi_1 \varphi_2)]
\]

\[
Y_{n+1} = Y_n \lor \bigg( [\varphi_2 \land \varphi_1] \lor [z_r \leftarrow 0](Y_n \land \neg \varphi_1) \bigg)
\]

\[
\lor \bigg( [\varphi_2 \land \varphi_1] \lor [Y_n \land \varphi_1] \bigg)
\]

\[
\lor \bigg( [\varphi_2 \land \neg \varphi_1 \land z \leq k] \lor Y_n \bigg)
\]

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 \(\text{RP}(E \varphi_1 U^k_{\prec c} \varphi_2)\) 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 \(z\) measures time elapsing when \(\varphi_2\) is false, so it will be reset at each transition from set of states satisfying \(\varphi_2\) to another satisfying \(\neg \varphi_1\). Without losing information about clock \(z\), and in order to further optimize our sequence, \(z\) can also be reset when transition from set of states satisfying \(\varphi_2\) to set of states satisfying \(\varphi_1\), and therefore the sequence \(Y_n\) becomes as follows:

\[
Y_0 = [z \sim c) \land (E \varphi_2 \varphi_1 \varphi_2)]
\]

\[
Y_{n+1} = Y_n \lor \bigg( [\varphi_2 \land \varphi_1] \lor [z_r \leftarrow 0](Y_n) \bigg)
\]

\[
\lor \bigg( [\varphi_2 \land \neg \varphi_1 \land z \leq k] \lor Y_n \bigg)
\]
Now we define the operator $\triangleright$ as follows:

**Definition 4 (Predecessor operator $\triangleright$).** Given a TA $A$, a TTS $T = \langle S, \Sigma, \rightarrow, I \rangle$, an alphabet $\Sigma$ which denotes a finite set of actions and two characteristic sets $Q_1$ and $Q_2$. Calculate $Q_1 \triangleright Q_2$ to is determined:

- $Q_1 \triangleright Q_2$:
  - 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 $Q_1 \triangleright Q_2$.
  - 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 intermediate states are in $Q_1$.

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

\[
\begin{align*}
Y_0 & = \{ (z \sim c) \land (E \varphi_2 U (\varphi_2 \land z > k)) \} \\
Y_{n+1} & = g(Y_n)
\end{align*}
\]

Such that: $g(Y) = Y \lor \left( (\varphi_2 \land \varphi_1) \triangleright [z_t \leftarrow 0]Y \lor (\varphi_2 \land (\neg \varphi_1 \land z \leq k)) \triangleright Y \right)

The particularity of the backward analysis is that the iterative calculation 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 $\text{RP}(E \varphi_2 U_{\varphi_2} \varphi_2)$:

$$\text{RP}(E \varphi_2 U_{\varphi_2} \varphi_2) = [z_t \leftarrow 0] \text{Sup } Y_n$$

**Proof:** (sketch). In order to prove this result we first at shows that the first 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:

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

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 $\forall n \in N \ Y_n = g^n(Y_0) \in P(E)$.

The sequence $Y_n$ is monotonic by inclusion, because $Y_n \subseteq Y_{n+1}$ $\forall n \in N$. Thus $Y_n$ is monotonic in the finite set $P(E)$, so $Y_n$ is stationary, i.e. $\exists r \in N$ 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, $g : W \rightarrow W$ 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 sequence $g^n(Y_0) = Y_n$, such that $Y_0$ is the least element of $W$ (intersection of its elements).

**Proof:** (sketch). In order to prove this result we first at shows that the first 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:

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

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 $\forall n \in N \ Y_n = g^n(Y_0) \in P(E)$.

The sequence $Y_n$ is monotonic by inclusion, because $Y_n \subseteq Y_{n+1}$ $\forall n \in N$. Thus $Y_n$ is monotonic in the finite set $P(E)$, so $Y_n$ is stationary, i.e. $\exists r \in N$ 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, $g : W \rightarrow W$ 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 sequence $g^n(Y_0) = Y_n$, such that $Y_0$ is the least element of $W$ (intersection of its elements).
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, $Q = [\text{RP} (E \varphi_1 U^k_{\leq c} \varphi_2)]$ is the least fixpoint of $g: Y \to g(Y)$:

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

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

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

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

$$\begin{align*}
E \varphi U^k_{\leq c} \varphi_2 & \quad \text{Right Part} \\
\left[ \begin{array}{l}
z, z_1 := 0 \\
z_r := 0 \\
z \sim c \\
z_r > k
\end{array} \right] & \quad \left[ \begin{array}{l}
z_1 \\
ϕ \leftarrow z_1 \leq k
\end{array} \right]
\end{align*}$$

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

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

$$\begin{align*}
X_0 &= [\text{RP} (E \varphi_1 U^k_{\leq c} \varphi_2)] \\
X_{n+1} &= X_n \lor \left( [\varphi_1] \triangleright [z_1 \leftarrow 0] X_n \right) \\
& \quad \lor \left( [\lnot \varphi_1 \land z_1 \leq k] \triangleright X_n \right)
\end{align*}$$

Where $z, z_1$ are reset when the stationary value of the sequence $X_n$ is reached, i.e. after that the set of symbolic states satisfying $E \varphi_1 U^k_{\leq c} \varphi_2$ is computed.

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

$$\begin{align*}
X_0 &= [\text{RP} (E \varphi_1 U^k_{\leq c} \varphi_2)] \\
X_{n+1} &= f(X_n)
\end{align*}$$

Such that:

$$f(X) = X \lor \left( [\varphi_1] \triangleright [z_1 \leftarrow 0] X_n \right) \lor \left( [\lnot \varphi_1 \land z_1 \leq k] \triangleright X_n \right)$$

Therefore we have the following result:

$$[E \varphi_1 U^k_{\leq c} \varphi_2] = [z \leftarrow 0] [z_1 \leftarrow 0] (\text{Sup} X_n)$$

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 TCTLA 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 $E \varphi_1 U^k_{\leq c} \varphi_2$ 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 matrix** (say DBM for short) for $n$ clocks is an $(n + 1)$-square matrix of pairs:

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

A DBM $\mathcal{M} = (m_{i,j}; \prec_{i,j}), i,j=0,...,n$ 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 $v$, $v(x_0) = 0$):

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

where $\gamma < \infty$ means that $\gamma$ 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 $[\mathcal{M}]$. 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 $x_1 > 3$ and $x_2 \leq 5$ is characterized by the two DBMs $$(\begin{array}{c}
0 \leq x_1 \\
(0, \leq) (0, \leq)
\end{array})$$ and $$(\begin{array}{c}
0 \leq x_2 \\
(0, \leq) (0, \leq)
\end{array})$$

Thus the DBMs are not a canonical representation of zones. Moreover, it is not possible to test syntactically whether $[\mathcal{M}] = 0$ or $[\mathcal{M}] = [\mathcal{M}]$. 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 $\Phi (\mathcal{M})$ the normal form of $\mathcal{M}$. Before stating some very important properties of the normal form, we define a total order on $\mathbb{V}$ in the following way: if $(m; \prec),(m'; \prec') \in \mathbb{V}$, then

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

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

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

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

(i) $[\mathcal{M}] = [\Phi (\mathcal{M})]$ and $\Phi (\mathcal{M}) \leq [\mathcal{M}]$.

(ii) $[\mathcal{M}] \subseteq [\Phi (\mathcal{M})] \iff \Phi (\mathcal{M}) \leq [\mathcal{M}'] \iff [\mathcal{M}] \subseteq \Phi (\mathcal{M}')$.

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 $\mathcal{M} = (m_{i,j}; \prec_{i,j}), i,j=0,...,n$ is a DBM such that $[\mathcal{M}] \neq 0$, then the following two properties are equivalent:

1) $\mathcal{M}$ is in normal form.
2) for every $i,j = 0,...,n$, for every real $-m_{i,j} < r \prec_{i,j} m_{i,j}$, there exists a valuation $v \in [\mathcal{M}]$ such that $v(x_i) - v(x_i) = r$ (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-checking 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 $M = (m_{i,j}; \preceq_{i,j})_{i,j=1...n}$ and $M' = (m_{i,j}; \preceq'_{i,j})_{i,j=1...n}$ are two DBMs in normal form. Then, defining $M'' = (m''_{i,j}; \preceq''_{i,j})_{i,j=1...n}$ by

$$(m''_{i,j}; \preceq''_{i,j}) = \min \left((m_{i,j}; \preceq_{i,j}), (m'_{i,j}; \preceq'_{i,j})\right)$$

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 $M = (m_{i,j}; \preceq_{i,j})_{i,j=1...n}$ is a DBM in normal form. We define the DBM $M = (m_{i,j}; \preceq'_{i,j})_{i,j=1...n}$ by:

$$m'_{i,j} = m_{i,j}; \preceq_{i,j}$$

We get that $[M] = [M]$ and that the DBM $M$ is in normal form.

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

$$m'_{i,j} = \min (m_{i,j}; \preceq_{i,j})$$

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

**Image by resets.** Assume that $M = (m_{i,j}; \preceq_{i,j})_{i,j=1...n}$ is a DBM in normal form. We define the DBM $M' = (m_{i,j}; \preceq'_{i,j})_{i,j=1...n}$ by:

$$(m''_{i,j}; \preceq''_{i,j}) = \min \left((m_{i,j}; \preceq_{i,j}), (m'_{i,j}; \preceq'_{i,j})\right)$$

for every index $k = 0...n$.

We get that $[M''] = [M]$ and that the DBM $M' = \Phi(M)$ 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 $\Phi = (m''_{i,j}; \preceq''_{i,j})_{i,j=1...n}$ by:

$$(m''_{i,j}; \preceq''_{i,j}) = \min \left((m_{i,j}; \preceq_{i,j}), (m'_{i,j}; \preceq'_{i,j})\right)$$

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 $(m_1; \preceq_1) < (m_2; \preceq_2)$ if $m_1 < m_2$ and $(m_1; \preceq) < (m_2; \preceq_2)$. Further we define addition as $(m_1; \preceq) + (m_2; \preceq) = (m_1 + m_2; \preceq)$ and $(m_1; \preceq) + (m_2; \preceq_2) = (m_1 + m_2; \preceq)$.

### B. Computing the operator $	riangleright$

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 $\bigcup_{q \in Q} (q, Z_q)$, 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, $e = q_1 \xrightarrow{g} q_2$ an edge from $q_1$ to $q_2$. Let $(q_1, Z_2)$ be a symbolic state. We have [33]:

$$\text{pre}_e(q_2, Z_2) = \left(q_1, g \land (r \leftarrow 0)(Z_2)\right)$$

i.e., $\text{pre}_e(q_2, Z_2)$ 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 $y$.

![Timed automaton with two clocks](image)

The symbolic state that characterizes instantaneous predecessor of the symbolic state $(q, Z) = (q_5, y = 0 \land z > 35)$ through the edge from $q_2$ to $q_3$, is calculated as follows:

$$\text{pre}_e(q_2, Z_5) = (q_2, y \leq 10 \land (y \leftarrow 0)(y = 0 \land z > 35))$$

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

$$\text{pre}(q, Z) = \bigcup_{e \in \mathcal{E}} \text{pre}_e(q, Z)$$

i.e. $\text{pre}(q, Z)$ 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:

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

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

Furthermore, Yovine proved in [33] that $(q, Z_1) \triangleright (q, Z_2)$ 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:

$$Z_1 = (0 < x \lor 0 < y)$$

$$Z_2 = (y \leq 10 \land 35 < z)$$
Back to the timed automaton depicted in Example 5, the symbolic state \((q_2, Z_1) \Rightarrow (q_2, Z_2)\) is calculated as follows:

\[
\forall t > 0 \text{ s.t. } \left( (y + t \leq 10 \land 35 < z + t) \land \\
\forall t' < t \leq (0 < x + t' \lor 0 < y + t') \right)
\]

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

\[
\exists t > 0 \text{ s.t. } \left( (y + t \leq 10 \land 35 < z + t) \land \\
\exists t' \geq 0 \ (t' < t \land \neg((0 < x + t' \lor 0 < y + t')) \right)
\]

Which is equivalent to

\[
\exists t > 0 \text{ s.t. } \left( (y + t \leq 10 \land 35 < z + t) \land \\
\exists t' \geq 0 \ (t' < t \land x + t' \leq 0 \land y + t' \leq 0) \right)
\]

It is possible to eliminate the quantifier [33]

\[
\exists t > 0 \text{ s.t. } \left( (y + t \leq 10 \land 35 < z + t) \land \\
\exists (0 \leq t' < t \land x + t' \leq 0 \land y + t' \leq 0) \right)
\]

From which we deduce

\[
\exists t > 0 \text{ s.t. } \left( (y + t \leq 10 \land 35 < z + t) \land \neg(x \leq 0 \land y \leq 0) \right)
\]

Whose negation is

\[
\forall t > 0 \text{ s.t. } \left( (y + t \leq 10 \land 35 < z + t) \land (0 < x \lor 0 < y) \right)
\]

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

\[
(q_2, Z_1) \Rightarrow (q_2, Z_2) = \left( q_2, (y \leq 10 \land y - z < -25) \right)
\]

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

### C. Pseudo-Code for \(\varepsilon \varphi_1 U^k,\varphi_2\) Model-Checking algorithm

We now give the pseudo-code for Model-Checking algorithm of \(\varepsilon \varphi_1 U^k,\varphi_2\) modality.

#### Algorithm 1 Model-Checking of \(\varepsilon \varphi_1 U^k,\varphi_2\) Modality

1. function RightPart(\(\varepsilon \varphi_1 U^k,\varphi_2\) : TCTL\(^A\))
2. // TCTL formula
3. TargetSet := \{ \[z \sim c\] \land (\(\varepsilon \varphi_2 U (\varphi_2 \land z_r > k)\)) \};
4.
5. repeat
6. CurrentSet := TargetSet;
7. TargetSet := TargetSet \cup CurrentSet;
8. TargetSet := TargetSet \cup \{[\varphi_2 \land \neg \varphi_1] [z_l \leftarrow \emptyset] \} TargetSet \};
9. TargetSet := TargetSet \cup \{[\neg \varphi_1 \land z_l \leq k] [\varphi_2] \} TargetSet \};
10. until TargetSet = CurrentSet
11. return \([z_r \leftarrow \emptyset] TargetSet;
12. end function
13.
14. function CharacteristicSet(\(\varepsilon \varphi_1 U^k,\varphi_2\) : TCTL\(^A\))
15. TargetSet := RightPart(\(\varepsilon \varphi_1 U^k,\varphi_2\));
16.
17. repeat
18. CurrentSet := TargetSet;
19. TargetSet := TargetSet \cup CurrentSet;
20. TargetSet := TargetSet \cup \{[\varphi_1] [z_l \leftarrow \emptyset] \} TargetSet \};
21. TargetSet := TargetSet \cup \{[\neg \varphi_1 \land z_l \leq k] [\varphi_2] \} TargetSet \};
22. until TargetSet = CurrentSet
23. return \([z_l \leftarrow \emptyset] TargetSet;
24. end function

### VII. Conclusion

In this paper, we proposed a symbolic model-checking algorithm that computes the characteristic sets of some TCTL\(^A\) 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 UPATEL 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\(^A\) timed logic.

### REFERENCES


