Model Checking Timed Automata with One or Two Clocks

F. Laroussinie[1], N. Markey[1,2], and Ph. Schnoebelen[1]

[1] Lab. Spécification & Vérification
ENS de Cachan & CNRS UMR 8643
61, av. Pdt. Wilson, 94235 Cachan Cedex, France
email: {fl,markey,phs}@lsv.ens-cachan.fr

[2] Département d'Informatique – CP 212
Université Libre de Bruxelles
Bd du Triomphe, 1050 Bruxelles, Belgique
email: nmarkey@ulb.ac.be

Abstract. In this paper, we study model checking of timed automata (TAs), and more precisely we aim at finding efficient model checking for subclasses of TAs. For this, we consider model checking *TCTL* and $TCTL_{\leq,\geq}$ over TAs with one clock or two clocks.
First we show that the reachability problem is NLOGSPACE-complete for one clock TAs (*i.e.* as complex as reachability in classical graphs) and we give a polynomial time algorithm for model checking $TCTL_{\leq,\geq}$ over this class of TAs. Secondly we show that model checking becomes PSPACE-complete for full *TCTL* over one clock TAs. We also show that model checking *CTL* (without any timing constraint) over two clock TAs is PSPACE-complete and that reachability is NP-hard.

1 Introduction

Model checking is widely used for the design and debugging of critical reactive systems [Eme90,CGP99]. During the last decade, it has been extended to *real-time systems*, where quantitative information about time is required.

Timed models. Real-time model checking has been mostly studied and developed in the framework of Alur and Dill's *Timed Automata* (TAs) [ACD93,AD94], *i.e.* automata extended with *clocks* that progress synchronously with time. There now exists a large body of theoretical knowledge and practical experience for this class of systems. It is agreed that their main drawback is the complexity blowup induced by timing constraints: most verification problems are at least PSPACE-hard for Timed Automata [Alu91,CY92,ACD93,AL02].

Real-time automata are TAs with a unique clock which is reset after every transition. This subclass has been mostly studied from the language theory point of view [Dim00], but it is also considered in [HJ96] for modeling real-time systems. Clearly this subclass is less expressive than classical TAs with an arbitrary number of clocks but still it is natural and convenient for describing behavior of simply timed systems. For example, it may be useful to model systems where timing constraints are local, *i.e.* depend only of the time elapsed since the last transition. The use of a real valued clock offers a convenient and abstract concept of time. Moreover such kinds of restricted TAs are more natural and more expressive than models based on discrete Kripke Structures where some durations are associated with transitions (see for example Timed Transition Graphs [CC95] or Durational Kripke Structures [LMS02]).

Timed specifications. In order to express timing aspects of computations, we consider extensions of the classical temporal logic *CTL*. The idea is to use timing constraints tagging temporal modalities [AH92]. For example, the formula $\mathsf{EF}_{<10}\ A$ states that it is possible to reach a state verifying A ("$\mathsf{EF}\,A$") in less than 10 time units. Timing constraints can have three main forms: "$\leq c$" and "$\geq c$" set a lower or upper bound for durations, while "$= c$" requires a precise value. *TCTL* is the extension of *CTL* with all three types of constraints, while $TCTL_{\leq,\geq}$ is the fragment of *TCTL* where the "$=c$" constraints are forbidden.

Our contribution. In this paper, we aim at finding subclasses of Timed Automata that admit efficient model checking algorithms. For this purpose we consider one clock TAs (1C-TAs) which extend real-time automata because the clock is not required to be reset after each transition. First we show that reachability problem is NLOGSPACE-complete for 1C-TAs (*i.e.* as efficient as reachability in classical graphs) and we give a polynomial time algorithm for model checking $TCTL_{\leq,\geq}$ over 1C-TAs. These results are surprising because adding simple timing constraints induces generally a complexity blowup. Note efficient model checking $TCTL_{\leq,\geq}$ over 1C-TAs requires to use an *ad-hoc* algorithm: the classical region graph technique or the symbolic algorithms based on DBMs [Dil90] are not polynomial over this subclass.
Secondly we show that model checking becomes PSPACE-complete for full *TCTL* over 1C-TAs. Then we address the case of TAs with two clocks (2C-TAs), since it is well known that three clocks lead to PSPACE-hardness for reachability [CY92]. We show that model checking *CTL* (without any timing constraints) over 2C-TAs is already PSPACE-complete and that reachability is NP-hard.

These results emphasize the good properties of 1C-TAs and real-time automata, leading to efficient timed model checking.

Related work. Quantitative logics *for Timed Automata* are now well-known and many results are available regarding their expressive power, or the satisfiability and model checking problems [AH94,ACD93,AH93,AFH96,Hen98]. That exact durations may induce harder model checking complexity was already observed in the case of *LTL* and Timed Automata [AFH96]. Complexity of timed model checking is considered in [CY92] where it is shown that three clocks are sufficient to have PSPACE-hardness for the reachability problem. In [AL02], model checking is studied for several timed modal logics. In [ACH94] the expressive power of clocks in TAs is studied from the language theory point of view.

2 Timed automata

Let $\mathbb{N}$ and $\mathbb{R}$ denote the sets of natural and non-negative real numbers, respectively. Let $\mathcal{C}$ be a set of real valued clocks. We use $\mathcal{B}(\mathcal{C})$ to denote the set of boolean expressions over atomic formulae of the form [3] $x \sim k$ with $x \in \mathcal{C}$, $k \in \mathbb{N}$, and $\sim \in \{<, \leq, >, \geq, =\}$. Constraints of $\mathcal{B}(\mathcal{C})$ are interpreted over *valuations* for $\mathcal{C}$ clocks, that are functions from $\mathcal{C}$ to $\mathbb{R}$. The set of valuations is denoted by $\mathbb{R}^{\mathcal{C}}$. For every $v \in \mathbb{R}^{\mathcal{C}}$ and $d \in \mathbb{R}$, we use $v + d$ to denote the time assignment which maps each clock $x \in \mathcal{C}$ to the value $v(x) + d$. For every $r \subseteq \mathcal{C}$, we write $v[r \leftarrow 0]$ for the valuation which maps each clock in r to the value 0 and agrees with v over $\mathcal{C} \setminus r$. Let AP be a set of atomic propositions.

Definition 2.1. *A* timed automaton *(TA) is a 6-tuple* $A = \langle Q_A, \mathcal{C}, q_{init}, \rightarrow_A, \mathsf{Inv}_A, l_A\rangle$ *where* Q_A *is a finite set of* control states*,* $\mathcal{C}$ *is a finite set of clocks and* $q_{init} \in Q_A$ *is the* initial state*. The set* $\rightarrow_A \subseteq Q_A \times \mathcal{B}(\mathcal{C}) \times 2^{\mathcal{C}} \times Q_A$ *is a finite set of* action transitions*: for* $(q, g, r, q') \in \rightarrow_A$*,* g *is the enabling condition (or guard) of the transition and* r *is the set of clocks to be reset with the transition (we write* $q \xrightarrow{g,r}_A q'$*).* $\mathsf{Inv}_A : Q_A \rightarrow \mathcal{B}(\mathcal{C})$ *assigns a constraint, called an* invariant*, to any control state. Finally* $l_A : Q_A \rightarrow 2^{AP}$ *labels every control state with a subset of* AP*.*

A *state* (or *configuration*) of a timed automaton A is a pair (q, v), where $q \in Q_A$ is the current control state and $v \in \mathbb{R}^{\mathcal{C}}$ is the current clock valuation. The initial state of A is (q_{init}, v_0) where v_0 is the valuation mapping all clocks in $\mathcal{C}$ to 0.

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 elapsing, and reach $(q, v + t)$ for some $t \in \mathbb{R}$ whenever the invariant is satisfied. 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:

- $S = \{(q, v) \mid q \in Q_A$ and $v \in \mathbb{R}^{\mathcal{C}}$ 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 $q' = q$, $v' = v + t$ and $v + t' \models \mathsf{Inv}_A(q)$ for any $t' \leq t$ — we write $(q, v) \xrightarrow{\delta(t)} (q, v + t)$ —,
  - or $\exists q \xrightarrow{g,r}_A q'$ and $v \models g$, $v' = v[r \leftarrow 0]$ and $v' \models \mathsf{Inv}_A(q')$ — we write $(q, v) \rightarrow_a (q', v')$.
- $l : S \rightarrow 2^{AP}$ labels every state (q, v) with the subset of AP $l_A(q)$.

An execution of A is an infinite path in $\mathcal{T}_A$. Let $s = (q, v)$ be an A-configuration. An execution ρ from s can be described as an infinite sequence $s = s_0 \xrightarrow{\delta(t_0)}\rightarrow_a s_1 \xrightarrow{\delta(t_1)}\rightarrow_a \cdots$ for some $t_i \in \mathbb{R}$. Such an execution ρ goes through any configuration s' reachable from some s_i by a delay transition of duration $t \in [0; t_i]$ — we write $s' \in \rho$. Let $\mathrm{Exec}(s)$ be the set of all executions from s.

[3] Considering diagonal constraints $x - y \sim k$ does not matter for the complexity.

The standard notions of prefix, suffix and subrun apply for paths in TTS. Given $\rho \in \text{Exec}(s)$, any finite prefix σ leading to a configuration s' (denoted $s \overset{\sigma}{\mapsto} s'$) has a *duration*, $\mathsf{Time}(s \overset{\sigma}{\mapsto} s')$, defined as the sum of all delays along σ. Let $\text{Pref}(\rho)$ be the set of all prefixes of ρ.

Given $\rho \in \text{Exec}(s)$ and $s', s'' \in \rho$, we say that s' *precedes strictly* s'' along ρ (written $s' <_\rho s''$) iff there exists a finite subrun σ in ρ s.t. $s' \overset{\sigma}{\mapsto} s''$ and σ contains at least one non null delay transition or one action transition (*i.e.* σ is not reduced to $\xrightarrow{\delta(0)}$). Note that a configuration may have several occurrences along ρ and then it may be that $s <_\rho s$ or $s <_\rho s'$ and $s' <_\rho s$.

The size of a TA is $|Q_A| + |\mathcal{C}| + \sum_{(q,g,r,q') \in \to_A} |g| + \sum_q |\mathsf{Inv}_A(q)|$ where the size of a constraint is its length (constants are encoded in binary). We use $|{\to_A}|$ to denote the number of transitions in A.

3 Timed CTL

TCTL is the quantitative extension of *CTL* where temporal modalities are subscripted with constraints on duration [ACD93]. Formulae are interpreted over TTS states.

Definition 3.1 (Syntax of *TCTL*). *TCTL formulae are given by the following grammar:*

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

where $\sim$ can be any comparator in $\{<, \leq, =, \geq, >\}$, c any natural number and $P_i \in AP$.

Standard abbreviations include $\top, \bot, \varphi \vee \psi, \varphi \Rightarrow \psi, \ldots$ as well as $\mathsf{EF}_{\sim c}\,\varphi$ (for $\mathsf{E}\top\mathsf{U}_{\sim c}\,\varphi$), $\mathsf{AF}_{\sim c}\,\varphi$ (for $\mathsf{A}\top\mathsf{U}_{\sim c}\,\varphi$), $\mathsf{EG}_{\sim c}\,\varphi$ (for $\neg\mathsf{AF}_{\sim c}\,\neg\varphi$) and $\mathsf{AG}_{\sim c}\,\varphi$ (for $\neg\mathsf{EF}_{\sim c}\neg\varphi$). Further, the modalities U, F and G without subscripts are shorthand for $\mathsf{U}_{\geq 0}$, $\mathsf{F}_{\geq 0}$ and $\mathsf{G}_{\geq 0}$. The size $|\varphi|$ of a formula φ is defined in the standard way, with constants written in binary notation.

Definition 3.2 (Semantics of *TCTL*). *The following clauses define when a state s of some TTS $\mathcal{T} = \langle S, s_{\mathit{init}}, \to, l\rangle$ satisfies a TCTL formula φ, written $s \models \varphi$, by induction over the structure of φ (semantics of boolean operators is omitted).*

$$\begin{aligned}
s \models \mathsf{E}\varphi\mathsf{U}_{\sim c}\,\psi \;\; &\textit{iff} \;\; \exists\,\rho \in \mathit{Exec}(s) \;\mathit{with}\; \rho = \sigma \cdot \rho' \;\mathit{and}\; s \overset{\sigma}{\mapsto} s' \;\mathit{s.t.}\\
&\qquad \mathsf{Time}(s \overset{\sigma}{\mapsto} s') \sim c\,,\; s' \models \psi \;\mathit{and}\; \forall\, s'' <_\rho s',\; s'' \models \varphi\\
s \models \mathsf{A}\varphi\mathsf{U}_{\sim c}\,\psi \;\; &\textit{iff} \;\; \forall \rho \in \mathit{Exec}(s), \exists \sigma \in \mathit{Pref}(\rho),\; \mathit{s.t.}\; s \overset{\sigma}{\mapsto} s',\\
&\qquad \mathsf{Time}(s \overset{\sigma}{\mapsto} s') \sim c,\; s' \models \psi \;\mathit{and}\; \forall\, s'' <_\rho s',\; s'' \models \varphi
\end{aligned}$$

Thus, in $\mathsf{E}\varphi\mathsf{U}_{\sim c}\,\psi$, the classical until is extended by requiring that ψ be satisfied within a duration (from the current state) verifying the constraint "$\sim c$".

Given a TA $A = \langle Q, \mathcal{C}, q_{\text{init}}, \to_A, \mathsf{Inv}_A, l_A\rangle$ and a *TCTL* formula φ, we write $A \models \varphi$ when $s_{\text{init}} \models \varphi$.

4 Complexity of Timed Model checking

Given a TA A, the TTS $\mathcal{T}_{\mathcal{A}}$ may have an infinite number of states and then standard model checking techniques cannot be applied directly. Indeed the decidability of verification problems over TAs is based on the region graph technique: The infinite state space of configurations is partitioned in a finite number of *regions* (equivalence classes of a relation over valuations) which have the "same behavior" w.r.t. the property to be checked, then a standard model checking algorithm can be applied over this finite abstraction. The region graph mainly depends on the number of clocks and the constants occurring in the guard. One of the main drawbacks of timed model checking is that the size of the region graph is exponential in the number of clocks and the (encoding of) constants. Several data-structures have been proposed to verify non-trivial timed systems (for ex. DBM see [Dil90,Bou04]).

Reachability problem of timed automata is known to be PSPACE-complete [AH94]. In [CY92], reachability in TA is shown to be PSPACE-complete even when the number of clocks is 3 **or** when the constants occurring in the guard belong to $\{0, 1\}$.

For *TCTL*, model checking is PSPACE-complete [ACD93]. And it is EXPTIME-complete for many variants of timed μ-calculus [AL02]; Checking timed bisimilarity is also an EXPTIME-complete problem. Note that all these results hold for a $\mathbb{R}$ or $\mathbb{N}$ as time domain and these results still hold when considering a parallel composition of TAs instead of a single one [AL02].

In this paper, we consider two subclasses of TAs whose complexity for timed verification is not known: we will study TAs with one clock (1C-TAs) or two clocks (2C-TAs). Clearly these subclasses are more expressive than real-time automata where the unique clock is reset after any transition and than extensions of Kripke structures with integer durations.

We will assume that in 1C-TAs, the guards are given by two constants defining the minimal (resp. maximal) value for x to perform the transition: it is always possible to reduce, in polynomial time, any 1C-TA to an equivalent automaton verifying such a property.

5 Model checking one clock timed automata

For a 1C-TA, a valuation is just a real value: the time assignment associated with the automaton clock x. First we consider the reachability problem: "Given a TA and a control state q, is it possible to reach a configuration (q, v) from the initial state?"

Proposition 5.1. *Reachability in 1C-TAs is NLOGSPACE-complete.*

Proof. The NLOGSPACE-hardness comes from complexity of reachability in classical graphs. Now we give a NLGOSPACE algorithm. A 1C-TA configuration is a control state and a value for the clock x. It is sufficient to consider only the integer value of x and to know if the fractional part is zero or not, but the integer value cannot be stored directly in a logarithmic space algorithm and we have to use a more concise encoding.

Let A be a 1C-TA. Let $\mathbb{B}$ be the set of integer values used in the guards and zero. We use $b_0, b_1, \dots, b_k$ to range over $\mathbb{B}$ and assume $0 = b_0 < b_1 < \cdots$ and $|\mathbb{B}| = k+1$. The set $\mathbb{B}$ defines a set $\mathcal{I}_{\mathbb{B}}$ of $2(k+1)$ intervals $\lambda_0, \lambda_1, \dots$ with $\lambda_0 \stackrel{\text{def}}{=} [b_0; b_0], \lambda_1 \stackrel{\text{def}}{=} (b_0; b_1), \lambda_2 \stackrel{\text{def}}{=} [b_1; b_1], \cdots, \lambda_{2k+1} \stackrel{\text{def}}{=} (b_k, \infty)$. We will encode the configuration (q, x) by the pair $(q, n(x))$ s.t. $x \in \lambda_{n(x)}$. Since $k \leq 2 \cdot |{\rightarrow_A}|$, it is possible to store $n(x)$ in logarithmic space.

First the algorithm counts the number of different constants in guards of A: This is done by verifying that the constants occurring in the i-th transition are different from the constants used in the j-th transition with $j < i$ (this test is done by enumerating each bit of the constant c to be checked and verify the equivalence, it requires a space in $O(\log(\log(c))))$.

Then given a pair (q, n), the algorithm non-deterministically guesses another (q', n') and verifies that $(q', \lambda_{n'})$ is reachable from (q, λ_n), *i.e.* either $q = q'$ and $n' = n + 1$ (this is a delay transition), or there exists a transition $q \xrightarrow{g,r} q'$ s.t. g is satisfied by any value in λ_n and $n' = n$ (resp. $n' = 0$) if $r = \emptyset$ (resp. $r = \{x\}$). Assume $g = m_1 \leq x \leq m_2$, then checking $\lambda_n \models g$ can be done by counting the number n_1 of different constants less than m_1 and the number n_2 of those greater than m_2. Finally $\lambda_n \models g$ iff $\frac{n}{2} \geq n_1$ and $\frac{n}{2} \leq k - n_2$ (resp. $\frac{n-1}{2} \geq n_1$ and $\frac{n+1}{2} \leq k - n_2$) if n is even (resp. n is odd). These operations requires only a logarithmic space. $\square$

This result entails that analysing a 1C-TA is not more complex than analysing an untimed graph from the complexity theory. After this positive result, we now consider model checking for 1C-TA and $TCTL_{\leq,\geq}$:

Theorem 5.2. *Model checking $TCTL_{\leq,\geq}$ over 1C-TAs is P-complete.*

Proof. P-hardness follows from the case of CTL model checking. We present a polynomial algorithm to construct, for any state q and subformula ξ of Φ, an union of intervals $\mathsf{Sat}[q, \xi]$ over $\mathbb{R}$ containing the valuations for x s.t. $x \in \mathsf{Sat}[q, \xi]$ iff $(q, x) \models \xi$. Assume $\mathsf{Sat}[q, \xi] = \bigcup_{j=1,\dots,k} \langle \alpha_j, \beta_j \rangle$ with $\langle \in \{[, (\}$ and $\rangle \in \{],)\}$; We will see that it is sufficient to consider $\alpha_j, \beta_j \in \mathbb{N} \cup \{\infty\}$. We choose $\alpha_j < \beta_j$ and $\beta_j < \alpha_{j+1}$ if $j + 1 \leq k$ in order to keep its size (*i.e.* the number of intervals) small; Indeed we will show that $|\mathsf{Sat}[q, \xi]| \leq 2 \cdot |\xi| \cdot |{\rightarrow_A}|$. We denote by $\mathsf{Cst}_A \subseteq \mathbb{N} \cup \{\infty\}$ the set of all constants occurring in A (either in guards or in invariants) plus 0.

We only present here the labeling procedure for the modality $\mathsf{E}\varphi\mathsf{U}_{\leq c}\,\psi$: the case of boolean operators and atomic propositions is straightforward and the procedures for other modalities are given in Appendix A.

Assume $\xi = \mathsf{E}\varphi\mathsf{U}_{\leq c}\,\psi$. Assume also that $\mathsf{Sat}[q, \varphi]$ and $\mathsf{Sat}[q, \psi]$ have been already constructed. In order to compute $\mathsf{Sat}[q, \xi]$, we build a (finite) graph $G = (V_G, \rightarrow_G, l_G)$ where every node $v \in V_G$ corresponds to a set of configurations (q, λ) where λ is an interval over $\mathbb{R}$ s.t. (1) these configurations verify either ψ or $\varphi \wedge \neg\psi$, (2) for any guard g in an A-transition, $\lambda \models g$ or $\lambda \models \neg g$. This last requirement implies that the same sequences of action transitions are enabled from any configuration of (q, λ).

Every G-transition will correspond to an action transition of A or an abstract delay transition (leading to another node with different properties): G can be seen as a kind of region graph. The definition of intervals λ depends on $\mathsf{Sat}[q, \varphi]$ and $\mathsf{Sat}[q, \psi]$ and also on guards of A. Let $\mathbb{B}$ be the finite set $\mathsf{Cst}_A \cup \{\alpha_j, \beta_j \mid \exists q \in Q_A \text{ s.t. } \langle \alpha_j; \beta_j \rangle \in \mathsf{Sat}[q, \varphi] \cup \mathsf{Sat}[q, \psi]\}$. We enumerate $\mathbb{B}$ as b_0, b_1, ... with $b_i < b_{i+1}$. We define V_G as the pairs (q, λ) where (1) λ is of the form $[b_i; b_i]$ or $(b_i; b_{i+1})$, and (2) we have $\lambda \subseteq \mathsf{Sat}[q, \psi]$ or $\lambda \subseteq \mathsf{Sat}[q, \varphi] \cap \overline{\mathsf{Sat}[q, \psi]}$. The G-transitions are:

- actions: $(q, \lambda) \rightarrow_G (q', \lambda')$ if there exists $q \xrightarrow{g,r} q'$ in A such that $\lambda \models g$, $\lambda' = \lambda$ (resp. $\lambda' = [0; 0]$) if $r = \emptyset$ (resp. $r = \{x\}$), and $\lambda' \models \mathsf{Inv}_A(q')$.
- abstract delays: $(q, \lambda) \rightarrow_G (q, \mathsf{Succ}(\lambda))$ if $\mathsf{Succ}(\lambda) \models \mathsf{Inv}_A(q)$, where Succ is the function: $\mathsf{Succ}([b_i, b_i]) = (b_i; b_{i+1})$ and $\mathsf{Succ}((b_i; b_{i+1})) = [b_{i+1}; b_{i+1}]$ if $b_{i+1} < \infty$ and $\mathsf{Succ}((b_i; \infty)) = (b_i; \infty)$ otherwise.

Note that $|G| \leq (Q_A \cdot 2 \cdot |\mathbb{B}|) \cdot (2 + | \rightarrow_A |)$. We can now restrict G to the nodes satisfying $\mathsf{E}\varphi\mathsf{U}\psi$ by a standard algorithm and then clearly the nodes in V_G represent all A configurations satisfying $\mathsf{E}\varphi\mathsf{U}\psi$. We now have to see when there exists a path leading to a ψ-state and being short enough (*i.e.* $\leq c$) to witness ξ. For this we can compute for any node $(q, \lambda) \in V_G$ a *duration function* $\delta^{\psi}_{q,\lambda} : \lambda \rightarrow \mathbb{R}$ s.t. $\delta^{\psi}_{q,\lambda}(t)$ is the duration of a shortest path from (q, t) to some state verifying ψ (along a path satisfying φ). The crucial point is that such a duration function over λ has a special structure: it is first constant and then decreases with the slope -1. The constant part corresponds to configurations for which a shortest path starts by a sequence of action transitions where the clock is reset at least once before any time elapsing (and clearly this also holds for previous positions in λ), and the decreasing part corresponds to positions from which a delay transition occurs before reseting x along a shortest path. These functions can easily be encoded as pairs (c_1, c_2) with $c_1 \geq c_2$, with the following meaning:

$$
\begin{aligned}
\delta^{\psi}_{q,[b_i;b_i]}(t) &\stackrel{\text{def}}{=} c_1 \\
\delta^{\psi}_{q,(b_i;b_{i+1})}(t) &\stackrel{\text{def}}{=} \begin{cases} c_1 & \text{if } b_i < t \leq b_{i+1} - (c_1 - c_2) \\ c_2 - (t - b_{i+1}) & \text{if } b_{i+1} - (c_1 - c_2) < t < b_{i+1} \end{cases}
\end{aligned}
$$

Of course, it is also possible to have a pure constant function over λ (then $c_1 = c_2$) or a pure decreasing function (then $c_1 = c_2 + (b_{i+1} - b_i)$). See Figure 1 for more intuition.

The structure of the duration functions allows us to compute them by adapting the Bellman-Ford algorithm for single source shortest path over G. This algorithm is given in Appendix A. The idea is to compute the $\delta^{\psi}_{q,\lambda}$'s by successive approximations. Consider a shortest path (SP) π in $\mathcal{T}_{\mathcal{A}}$ starting from (q_0, x_0), leading to a state verifying ψ with intermediary states satisfying φ. The path π can be described as a sequence of $\xrightarrow{\delta(t_i)}\rightarrow_a$. Such a path in $\mathcal{T}_{\mathcal{A}}$ is associated with a path in G where the delay transitions $\xrightarrow{\delta(t_i)}$ are replaced by a sequence of abstract delay transitions. Clearly along a SP, a node (q, λ) occurs at most

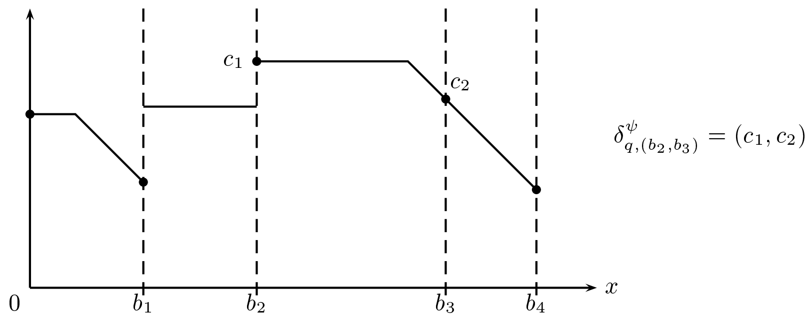

Fig. 1. Example of duration functions

once: given a configuration (q, x) with $x \in \lambda$, either a SP starts as the previous positions $x' < x$ in λ and it starts by action transitions that can be performed from (q, x), or the SP starts by delaying until $\mathsf{Succ}(\lambda)$ and in both cases it is not necessary to come back to (q, λ) later. Assume the size of a SP in G is k, then k is bounded by $|V_G| + 1$ and then it is discovered after the k-th step of the algorithm.

Once the $\delta^{\psi}_{q,\lambda}$'s have been computed, it remains to see which intervals or part of intervals contain positions whose distance to ψ is less than c. This step may lead to cut an interval in two parts (still at an integer point) and add new constants in $\mathbb{B}$; in Appendix A we show that the size of $\mathsf{Sat}[q, \xi]$ is bounded by $|\mathsf{Sat}[q, \psi]| + (|\mathsf{Sat}[q, \varphi]| + 2 \cdot |\rightarrow_A|)$ and the number of new constants in $\mathbb{B}$ is bounded by $|\rightarrow_A|$.

From the previous procedure and those in Appendix A, we can deduce that $|\mathsf{Sat}[q, \varphi]| \leq 2 \cdot |\varphi| \cdot |\rightarrow_A|$. This entails that the most complex procedure ($E\_\mathsf{U}_{<\_}$) runs in $O(|\xi|^2 \cdot |Q_A|^2 \cdot |\rightarrow_A|^3)$. This globally provides a complexity of $O(|\Phi|^3 \cdot |Q_A|^2 \cdot |\rightarrow_A|^3)$ for the full labeling procedure. More precisely we could show that the algorithm is in $O(|\Phi| \cdot |Q_A|^2 \cdot |\rightarrow_A| \cdot (\mathsf{Cst}_A + N^{\sim c}_{\Phi})^2)$ where $N^{\sim c}_{\Phi}$ is the number of Φ subformulae of the form $\mathsf{EU}_{\sim c}$ or $\mathsf{AU}_{\sim c}$. □

When considering exact durations in subscripts, model checking becomes PSPACE-hard, *i.e.* as hard as model checking TAs with several clocks:

Theorem 5.3. *Model checking TCTL on 1C-TAs is PSPACE-complete.*

Proof. Membership in PSPACE follows from the general result for TAs [ACD93]. PSPACE-hardness is shown by reducing QBF instance to a model checking problem over 1C-TA.

Consider a QBF instance $\Phi \stackrel{\text{def}}{=} Q_0 p_0 Q_1 p_1 \ldots Q_{n-1} p_{n-1} \cdot \varphi$: $Q_i \in \{\exists, \forall\}$, any p_i is a boolean variable for $i = 0, \ldots, n-1$, and φ is a propositional formula over the p_i's.

To reduce the QBF instance Φ to a model checking problem, we consider the 1C-TA A_Φ depicted in Figure 2 and the formulae $\overline{\Phi_i}$ with $i = 0, \ldots, n$ defined

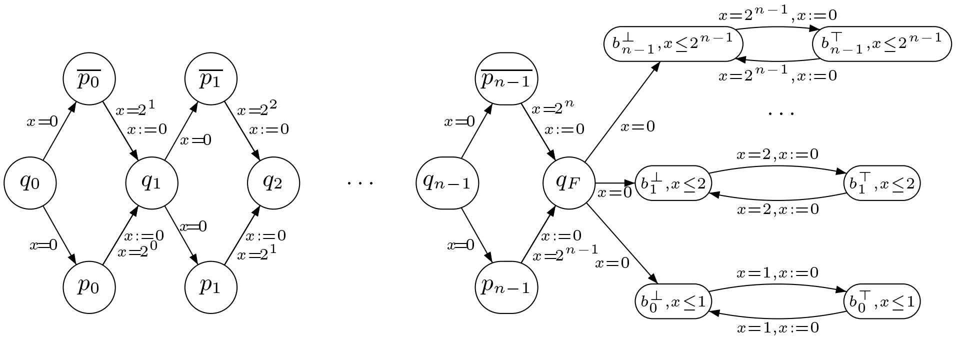

Fig. 2. 1C-TA A_Φ associated with QBF instance Φ

as:

$$0 \le i < n : \overline{\Phi_i} \stackrel{\text{def}}{=} \begin{cases} \mathsf{EF}_{=2^i}\Big((p_i \vee \overline{p_i}) \wedge \overline{\Phi_{i+1}}\Big) & \text{if } Q_i = \exists \\ \mathsf{AF}_{=2^i}\Big((p_i \vee \overline{p_i}) \wedge \overline{\Phi_{i+1}}\Big) & \text{if } Q_i = \forall \end{cases}$$

$$\overline{\Phi_n} \stackrel{\text{def}}{=} \varphi[p_i \leftarrow \mathsf{EF}_{=2^n-1}\, b_i^{\top}]$$

Now we show that Φ is valid iff $(q_0, 0) \models \overline{\Phi_0}$. Indeed, interpreting $\overline{\Phi_0}$ over $(q_0, 0)$ makes that every formula $\overline{\Phi_i}$ with $i = 1, \ldots, n$ is interpreted over some configurations in a set S_i located at duration $\sum_{j<i} 2^j$ from $(q_0, 0)$. More precisely S_i is composed by (p_{i-1}, l) and $(\overline{p_{i-1}}, l)$ with $l \in \{1, \ldots, 2^{i-1}\}$. A configuration in S_i can be seen as a boolean valuation for $p_0, \ldots, p_{i-1}$: The truth value of p_{i-1} is $\top$ iff the control state is p_{i-1} and the value of p_k ($k < i-1$) is given by the k-th bit of the binary encoding of $l - 1$. Moreover this valuation is preserved in the two possible successor configurations in S_{i+1} at duration 2^i from the current position. The alternation of existential EF and AF allows to simulate the alternation of quantifiers over the p_i's in Φ.

Finally Φ_n is interpreted over configurations of S_n which define valuations for $p_0, \ldots, p_{n-1}$. The configurations of the form (p_{n-1}, l) (resp. $(\overline{p_{n-1}}, l)$) with $l \in \{1, \ldots, 2^{n-1}\}$ are located at distance $0, \ldots, 2^{n-1}-1$ (resp. $2^{n-1}, \ldots, 2^n-1$) to q_F. Consider such a configuration (p_{n-1}, l) and assume $(p_{n-1}, l) \models \mathsf{EF}_{=2^n-1}\, b_k^{\top}$: Reaching $(q_F, 0)$ takes $2^{n-1} - l$, it remains to spend $2^{n-1} + l - 1$ in the loop $b_k^{\perp} b_k^{\top} \cdots$ and clearly $b_k^{\top}$ holds after this duration iff the k-th bit of $l-1$ is 1. $\square$

Note that the automaton depicted in Figure 2 is a real-time automaton (x is reset after every transition) and then we can deduce the following corollary:

Corollary 5.4.

- *Reachability in real-time automata is NLOGSPACE-complete.*
- *Model checking $TCTL_{\le,\ge}$ over real-time automata is P-complete.*
- *Model checking TCTL over real-time automata is PSPACE-complete.*

6 Model checking two clocks timed automata

When a timed automaton has two clocks, there is a complexity blow-up for model checking. First we have the following result for reachability:

Proposition 6.1. *Reachability problem in 2C-TAs is NP-hard.*

Proof. This follows from a simple encoding of the SUBSET-SUM problem [GJ79, p. 223]: assume we are given a set $\{a_1, \ldots, a_p\}$ of integers and a *goal* b, one asks whether there exists a subset $J \subseteq \{1, \ldots, p\}$ s.t. $\sum_{j \in J} a_j = b$. This problem is known to be NP-complete.

This problem is obviously equivalent to the reachability problem for state G in the automaton shown on figure 3.

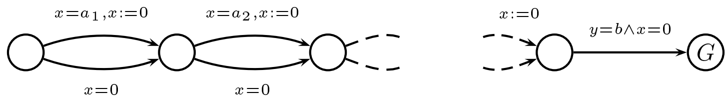

Fig. 3. Encoding of SUBSET-SUM in a 2C-TA

This complexity blow-up compared to the one clock case increases when considering model-checking:

Theorem 6.2. *The model checking problems for CTL, TCTL$_{\leq,\geq}$ or TCTL on 2C-TAs are PSPACE-complete*

Proof. The PSPACE-membership comes from PSPACE model checking algorithm for *TCTL* over classical TAs. It is sufficient to show PSPACE-hardness for the *CTL* case. Let $\Phi \stackrel{\text{def}}{=} O_0 p_0, O_1 p_1 \cdots O_{n-1} p_{n-1} \cdot \varphi$ be a QBF instance ($O_i \in \{\exists, \forall\}$ and φ is boolean formula over the p_i's). Consider the 2C-TA depicted in Figure 4.

Let $\overline{\Phi}$ be the following *CTL* formula:

$$\overline{\Phi} \stackrel{\text{def}}{=} \Big(O_0 q_0 \mathsf{U} \Big(q_1 \wedge (O_1 q_1 \mathsf{U} (q_2 \wedge \ldots \mathsf{U}(q_n \wedge \overline{\varphi}))) \Big) \Big)$$

with $\overline{\varphi} \stackrel{\text{def}}{=} \varphi[p_i \leftarrow \mathsf{EF} p_i]$. A path from q_0 to q_n defines a boolean valuation for the p_i's: performing the transition $q_i \xrightarrow{x=2^i, x:=0} q_{i+1}$ (resp. $q_i \xrightarrow{x=0} q_{i+1}$) assigns $\top$ (resp. $\perp$) to p_i. And in the configuration $(q_n, 0, v_y)$, the valuation is encoded in the value v_y (the total amount of time used to reach q_n). Then the branch $q_n \to s_i \to s_{i,1} \cdots$ allows us to check the value of the i-th bit of v_y, that is exactly the truth value of p_i. $\square$

Note that this last result is proved for a very simple subclass: the automaton used in proof of Theorem 6.2 has a clock which is reset after each transition. Despite this, model checking (untimed) *CTL* leads to PSPACE-hardness.

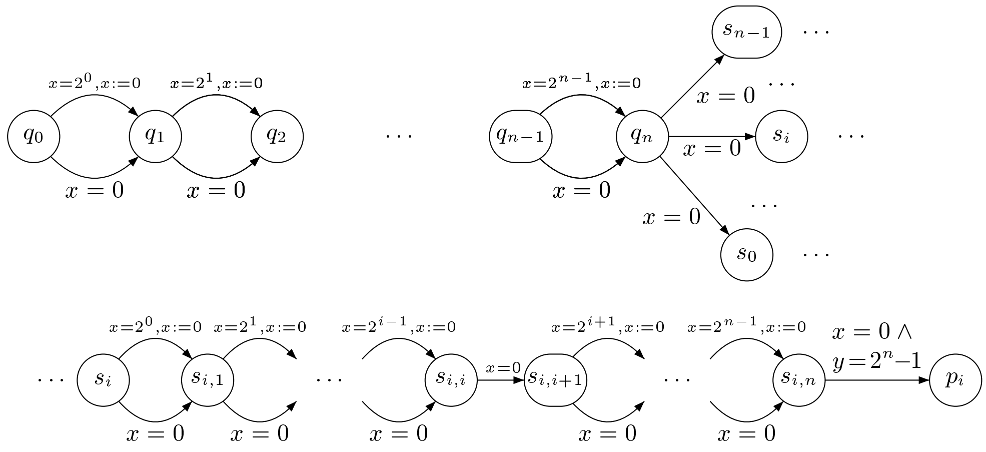

Fig. 4. The 2C-TA A_Φ associated with the QBF instance Φ

7 Conclusion

	1C-TAs real-time aut.	2C-TAs	TAs [ACD93,CY92]
Reachability	**NLOGSPACE-C**	**NP-hard**	PSPACE-C
$TCTL_{\leq,\geq}$ model checking	**P-complete**	**PSPACE-C**	PSPACE-C
$TCTL$ model checking	**PSPACE-complete**		PSPACE-C

Fig. 5. Summary of the results

Figure 5 gives an overview of the results presented in the paper and a comparison with the results for classical Timed Automata. The main results concern one-clock automata. First the reachability problem in 1C-TAs is as efficient as the reachability in classical graphs. Moreover model checking can be done efficiently if the property is expressed with $TCTL_{\leq,\geq}$ logic. This result is surprising because usually, in $TCTL$ model checking, the timing constraints are handled by adding a new clock in the system and we also have seen that any model checking problem, even for the untimed CTL, is PSPACE-hard over simple 2C-TAs. Moreover note that the efficiency requires an *ad hoc* algorithm to handle timing constraints.

In timed model checking, an important challenge consists in developing data structures enabling to manage complexity blow-up due to timing constraints and to parallel composition of components; indeed it would be very interesting to have the benefits of DBMs for the timing constraints and those of BDDs for the control state explosion, but today no convincing solution exists. Our results motivate research for algorithms and data structures for simply timed systems composed by a unique clock and a parallel composition of processes. Of course, analysing such systems is PSPACE-hard due to the composition, nevertheless efficient data structures for handling such systems could be more easily defined due to the simple timing constraints.

References

[ACD93] R. Alur, C. Courcoubetis, and D. L. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

[ACH94] R. Alur, C. Courcoubetis, and T. A. Henzinger. The observational power of clocks. In *Proc. 5th Int. Conf. Theory of Concurrency (CONCUR'94), Uppsala, Sweden, Aug. 1994*, volume 836 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 1994.

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AFH96] R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

[AH92] R. Alur and T. A. Henzinger. Logics and models of real time: A survey. In *Real-Time: Theory in Practice, Proc. REX Workshop, Mook, NL, June 1991*, volume 600 of *Lecture Notes in Computer Science*, pages 74–106. Springer, 1992.

[AH93] R. Alur and T. A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.

[AH94] R. Alur and T. A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–203, 1994.

[AL02] L. Aceto and F. Laroussinie. Is your model checker on time? On the complexity of model checking for timed modal logics. *Journal of Logic and Algebraic Programming*, 52–53:7–51, 2002.

[Alu91] R. Alur. *Techniques for Automatic Verification of Real-Time Systems*. PhD thesis, Stanford Univ., August 1991. Available as Tech. Report STAN-CS-91-1378.

[Bou04] P. Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24(3):281–320, 2004.

[CC95] S. Campos and E. M. Clarke. Real-time symbolic model checking for discrete time models. In T. Rus and C. Rattray, editors, *Theories and Experiences for Real-Time System Development*, volume 2 of *AMAST Series in Computing*, pages 129–145. World Scientific, 1995.

[CGP99] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.

[CY92] C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design*, 1(4):385–415, 1992.

[Dil90] D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Proc. Int. Workshop Automatic Verification Methods for Finite State Systems (CAV'89), Grenoble, June 1989*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer, 1990.

[Dim00] Catalin Dima. Real-time automata and the Kleene algebra of sets of real numbers. In *Proc. of STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Lille, France, February 2000*, volume 1770 of *Lecture Notes in Computer Science*, pages 279–289, 2000.

[Eme90] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 16, pages 995–1072. Elsevier Science, 1990.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[Hen98] T. A. Henzinger. It's about time: real-time logics reviewed. In *Proc. 9th Int. Conf. Concurrency Theory (CONCUR'98), Nice, France, Sep. 1998*, volume 1466 of *Lecture Notes in Computer Science*, pages 439–454. Springer, 1998.

[HJ96] Dang Van Hung and Wang Ji. On the design of hybrid control systems using automata models. In *Proc. 16th Conf. Found. of Software Technology and Theor. Comp. Sci. (FST&TCS'96), Hyderabad, India, Dec. 1996*, volume 1180 of *Lecture Notes in Computer Science*, pages 156–167. Springer, 1996.

[LMS02] F. Laroussinie, N. Markey, and Ph. Schnoebelen. On model checking durational Kripke structures (extended abstract). In *Proc. 5th Int. Conf. Foundations of Software Science and Computation Structures (FOSSACS'2002), Grenoble, France, Apr. 2002*, volume 2303 of *Lecture Notes in Computer Science*, pages 264–279. Springer, 2002.

A (End of) proof of Theorem 5.2

$\xi = \mathsf{E}\varphi\mathsf{U}_{\leq c}\,\psi$: We use a kind of Bellman-Ford algorithm to compute the δ^ψ, it uses the natural min operation over duration functions: $\min((c_1,c_2),(c_1',c_2')) \stackrel{\text{def}}{=} (\min(c_1,c_1'),\min(c_2,c_2'))$. The duration functions $d^\psi_{q,\lambda}$ are first initialized to 0 if $\lambda \subseteq \mathsf{Sat}[q,\psi]$ or to ∞ otherwise (no path leading to ψ states has been yet discovered) and then we use the following procedure:

For $i = 1$ to $|V_G| - 1$ do
  For any $(q,\lambda) \to_G (q',\lambda')$ do
    if $\Big(q = q' \wedge \lambda' = \mathsf{Succ}(\lambda) \wedge \lambda = (b_j; b_{j+1}) \wedge d^\psi_{q,\lambda'} = (c',c')\Big)$
      Then // delay transition - 1
        $d^\psi_{q,\lambda} = \min(d^\psi_{q,\lambda}, (b_{j+1} - b_j + c', c'))$
    else if $\Big(q = q' \wedge \lambda' = \mathsf{Succ}(\lambda) \wedge \lambda = [b_j; b_j] \wedge d^\psi_{q,\lambda'} = (c',c'')\Big)$
      Then // delay transition - 2
        $d^\psi_{q,\lambda} = \min(d^\psi_{q,\lambda}, (c',c'))$
    else // action transition
      if $(\lambda' = \lambda \vee \lambda' = [0;0])$
        Then $d^\psi_{q,\lambda} = \min(d^\psi_{q,\lambda}, d^\psi_{q,\lambda'})$

Then it remains to build $\mathsf{Sat}[q,\xi]$ from the duration functions δ^ψ_q and the threshold c: $x \in \mathsf{Sat}[q,\xi]$ iff $\delta^\psi_{q,\lambda}(x) \leq c$. This may lead to cut an interval in two parts. A crucial point of the algorithm is to merge as much as possible these (fragments of) G intervals, and we have to show that the size of $\mathsf{Sat}[q,\xi]$ can be bounded enough to ensure a polynomial algorithm. We are going to bound (1) the number of intervals of $\mathsf{Sat}[q,\xi]$ coming from a given interval of $\mathsf{Sat}[q,\varphi]$ and (2) the number of new constants (not present in $\mathbb{B}$) that can appear due to the cuts.

Consider an interval I of $\mathsf{Sat}[q,\varphi]$. This interval corresponds to a finite sequence of G-nodes $(q,\lambda_1),\cdots,(q,\lambda_k)$ s.t. $\lambda_{i+1} = \mathsf{Succ}(\lambda_i)$. The threshold "$\leq c$" may cut these intervals and provide non-adjacent intervals in $\mathsf{Sat}[q,\xi]$. We can distinguish two cases of cuts: (1) the cut is done between two intervals, or (2) the cut is done inside an interval. In both cases the cut is due to a unique constraint in a transition ($x <$ or $x >$) which can only cut this interval. Since a transition may contain at most two such constraints, the size of $\mathsf{Sat}[q,\xi]$ will be bounded by $|\mathsf{Sat}[q,\psi]|+(|\mathsf{Sat}[q,\varphi]|+2\cdot|{\rightarrow_A}|)$. Indeed:

1. Consider a cut between two intervals λ_j and λ_{j+1}. Assume $\lambda_j = [b_i;b_i]$ and $\lambda_{j+1} = (b_i,b_{i+1})$. Moreover assume $\delta^{\psi}_{q,\lambda_j} = (c_0,c_0)$ and $\delta^{\psi}_{q,\lambda_j} = (c_1,c_2)$. If there is a cut in b_i, then $c_0 < c < c_1$. The shortest paths enabled from b_i do not exist from (b_i,b_{i+1}) and then these SPs start by a sequence of action transitions and one of them (performed before any delay and reset) have a guard $x \leq b_i$. This transition can only cut in b_i the intervals of $\mathsf{Sat}[q,\varphi]$. The case $\lambda_j = (b_i;b_{i+1})$ and $\lambda_{j+1} = [b_{i+1},b_{i+1}]$ is similar.
2. Consider a cut inside an interval $\lambda_j = (b_i;b_{i+1})$. Then the cut occurs in the decreasing part of $\delta^{\psi}_{q,\lambda_j} = (c_1,c_2)$ and we have $c_1 > c > c_2$. The cut occurs in $b_{i+1}-(c-c_2)$, and introduces a new (integer) constant. For the valuations in the decreasing part, *i.e.* between $b_{i+1}-(c_1-c_2)$ and b_{i+1}, the shortest paths have delay transitions before any reset. This required delay is due to a guard of the form $x > k$ or $x \geq k$ along the SP. Such a constraint induces the cut and only this one (in configurations of the form $(\cdot,\lambda_j)$).

Therefore a guard $m < x < M$ may induce at most two cuts in $\mathsf{Sat}[q,\xi]$, then $|\mathsf{Sat}[q,\xi]| \leq |\mathsf{Sat}[q,\psi]| + |\mathsf{Sat}[q,\varphi]| + 2\cdot|{\rightarrow_A}|$. And it creates at most one new integer constant (this also holds for the other modalities) and this entails $\mathbb{B} \leq \mathsf{Cst}_A + |{\rightarrow_A}|\cdot|\xi|$. Finally the complexity of the procedure is in $O(|V_G|\cdot|{\rightarrow_G}|)$, with $|V_G| \leq |Q_A|\cdot 2\cdot|\mathbb{B}|$ and $|{\rightarrow_G}| \leq (|{\rightarrow_A}|+1)\cdot|V_G|$. This provides an algorithm in $O(|Q_A|^2\cdot|{\rightarrow_A}|^3\cdot|\xi|^2)$ for $\mathsf{E\_U}_{\leq c}$ _.

$\xi = \mathsf{E}\varphi\mathsf{U}_{\geq c}\,\psi$: For building $\mathsf{Sat}[q,\mathsf{E}\varphi\mathsf{U}_{\geq c}\psi]$, we use the same idea as for $\mathsf{E}\varphi\mathsf{U}_{\leq c}\psi$ formula based on the graph $G=(V_G,\rightarrow_G,l_G)$ but here we label nodes by $\varphi\wedge\neg\psi$, $\varphi\wedge\psi$ and $\neg\varphi\wedge\psi$. We restrict ourself to nodes satisfying $E\varphi\mathsf{U}\psi$ and we introduce a new atomic proposition $P_{\mathsf{SCC}^+(\varphi)}$ in order to label every node (q,λ) in G belonging to a strongly connected set of nodes satisfying φ and where at least one edge is an abstract delay transition. Labeling states for $P_{\mathsf{SCC}^+(\varphi)}$ can be done in time $O(|G|)$ once they are labeled for φ.

We can now solve the original problem. There are two ways a state can satisfy ξ:

- Either a path with loops is required so that a long enough duration is reached: such a state verifies the *CTL* formula $\mathsf{E}\,\varphi \mathsf{U} P_{\mathsf{scc+}(\varphi)}$ since any G state satisfies $\mathsf{E}\varphi\mathsf{U}\psi$. This can be done in $O(|V_G| + |{\rightarrow_G}|)$.
- Or a simple path is enough. Then we can use a (simple) variant of the earlier shortest paths method, this times geared towards *longest acyclic paths* (LAP). For this we just remove states labeled by $P_{\mathsf{scc+}(\varphi)}$, consider states satisfying $\neg\varphi \wedge \psi$ as final states and remove loops with null durations. The algorithm runs in $O(|V_G| + |{\rightarrow_G}|)$ and we keep (sub-)intervals whose LAP is above the threshold c.

Finally we build $\mathsf{Sat}[q, \xi]$ by merging the states of G satisfying $\mathsf{E}\varphi\mathsf{U}_{\geq c}\,\psi$. As for labeling $\mathsf{E}\varphi\mathsf{U}_{\leq c}\,\psi$, the procedure may add new constants (in the second case) and split intervals of $\mathsf{Sat}[q, \varphi]$ into several intervals in $\mathsf{Sat}[q, \xi]$ but we can argue as in the previous case and show that the size of $\mathsf{Sat}[q, \xi]$ is bounded by $|\mathsf{Sat}[q, \varphi]| + 2 \cdot |{\rightarrow_A}|$. The procedure runs in $O(|V_G| + |{\rightarrow_G}|)$, with $|V_G| \leq |Q_A| \cdot 2 \cdot |\mathbb{B}|$, $\mathbb{B} \leq \mathsf{Cst}_A + |{\rightarrow_A}| \cdot |\xi|$ and $|{\rightarrow_G}| \leq (|{\rightarrow_A}| + 1) \cdot |V_G|$. This gives an algorithm in $O(|Q_A| \cdot |{\rightarrow_A}|^2 \cdot |\xi|)$.

$\xi = \mathsf{A}\varphi\mathsf{U}_{\leq c}\,\psi$: We reduce to the previous cases using the following equivalences

$$\begin{aligned} \mathsf{A}\varphi\mathsf{U}_{\leq c}\,\psi &\equiv \mathsf{AF}_{\leq c}\,\psi \wedge \neg\mathsf{E}(\neg\psi)\mathsf{U}(\neg\psi \wedge \neg\varphi) \\ \mathsf{AF}_{\leq c}\,\psi &\equiv \mathsf{AF}\psi \wedge \neg\mathsf{E}\neg\psi\mathsf{U}_{>c}\,\top. \end{aligned}$$

$\xi = \mathsf{A}\varphi\mathsf{U}_{\geq c}\,\psi$: We use the equivalence $\mathsf{A}\varphi\mathsf{U}_{\geq c}\,\psi \equiv \mathsf{AG}_{<c}\,(\mathsf{A}\varphi\mathsf{U}_{>0}\,\psi)$. And it is easy to write a labeling procedure for $\mathsf{A}\varphi\mathsf{U}_{>0}\,\psi$ over the same G-graph used for $\mathsf{E}\varphi\mathsf{U}_{\geq c}\,\psi$: A node verifies $\mathsf{A}\varphi\mathsf{U}_{>0}\,\psi$ iff it verifies $\mathsf{AG}_{\leq 0}\,\varphi$, $\mathsf{A}\varphi\mathsf{U}\psi$ and after the first abstract delay transition $\mathsf{A}\varphi\mathsf{U}\psi$ has to hold.

Strict subscripts: The modalities with $< c$ or $> c$ are treated as the previous ones.