Timed Vacuity

Hana Chockler[1], Shibashis Guha[2(✉)], and Orna Kupferman[3]

[1] King's College London, London, UK
[2] Université Libre de Bruxelles, Brussels, Belgium
shibashis.guha@ulb.ac.be
[3] The Hebrew University, Jerusalem, Israel

Abstract. Vacuity is a leading sanity check in model-checking, applied when the system is found to satisfy the specification. The check detects situations where the specification passes in a trivial way, say when a specification that requires every request to be followed by a grant is satisfied in a system with no requests. Such situations typically reveal problems in the modelling of the system or the specification, and indeed vacuity detection is a part of most industrial model-checking tools.

Existing research and tools for vacuity concern discrete-time systems and specification formalisms. We introduce *real-time vacuity*, which aims to detect problems with real-time modelling. Real-time logics are used for the specification and verification of systems with a continuous-time behavior. We study vacuity for the branching real-time logic TCTL, and focus on vacuity with respect to the time constraints in the specification. Specifically, the logic TCTL includes the temporal operator U^J, which specifies real-time eventualities in real-time systems: the parameter $J \subseteq \mathbb{R}_{\geq 0}$ is an interval with integral boundaries that bounds the time in which the eventuality should hold. We define several tightenings for the U^J operator. These tightenings require the eventuality to hold within a strict subset of J. We prove that vacuity detection for TCTL is PSPACE-complete, thus it does not increase the complexity of model-checking of TCTL. Our contribution involves an extension, termed TCTL$^+$, of TCTL, which allows the interval J not to be continuous, and for which model checking stays in PSPACE. Finally, we describe a method for ranking vacuity results according to their significance.

1 Introduction

In temporal logic model-checking, we verify the correctness of a system with respect to a desired behavior by checking whether a mathematical model of the system satisfies a temporal-logic formula that specifies this behavior [12]. When the formula fails to hold in the model, the model checker returns a counterexample — some erroneous execution of the system [13]. In the last years there has been a growing awareness of the need of suspecting positive results of the model-checking process, as errors may hide in the modelling of the system or the behavior [22]. As an example, consider the property $G(req \rightarrow F\,grant)$ ("every request is eventually granted"). This property is clearly satisfied in a system in

which requests are never sent. It does so, however, in a *vacuous* (non-interesting) way, suggesting a suspicious behavior of the system.

In [6], Beer et al. suggested a first formal treatment of vacuity. As described there, vacuity is a serious problem: "our experience has shown that typically 20% of specifications pass vacuously during the first formal-verification runs of a new hardware design, and that vacuous passes always point to a real problem in either the design or its specification or environment" [6]. In the last decade, the challenge of detecting vacuous satisfaction has attracted significant attention (c.f., [5,7–9,11,18–20,23,25–27]).

Different definitions for vacuity exist in the literature and are used in practice. The most commonly used ones are based on the "mutation approach" of [6] and its generalization, as defined in [24]. Consider a model M satisfying a specification Φ. A subformula ψ of Φ *does not affect* (the satisfaction of) Φ in M if M satisfies also the (stronger) formula $\Phi[\psi \leftarrow \bot]$, obtained from Φ by changing ψ in the most challenging way. Thus, if ψ appears positively in Φ, the symbol $\bot$ stands for *false*, and if ψ is negative, then $\bot$ is *true*[1]. We say that M satisfies Φ vacuously if Φ has a subformula that does not affect Φ in M. Consider for example the formula $\Phi = G(req \rightarrow F grant)$ described above. In order to check whether the subformula *grant* affects the satisfaction of Φ, we model check $\Phi[grant \leftarrow false]$, which is equivalent to $G\neg req$. That is, a model with no requests satisfies Φ vacuously. In order to check whether the subformula *req* affects the satisfaction of Φ, we model check $\Phi[req \leftarrow true]$. This is equivalent to $GF grant$, thus a model with infinitely many *grant* signals satisfies Φ vacuously.

So far, research on vacuity has been chiefly limited to systems with a *discrete-time* behavior, modeled by means of labeled finite state-transition graphs. More and more systems nowadays have a *continuous-time* behavior. This includes embedded systems, mixed-signal circuits, general software-controlled physical systems, and cyber-physical systems. Such systems are modeled by *timed transition systems* [2], and their behaviors are specified by *real-time logics* [4]. Some preliminary study of vacuity for the linear real-time logic MITL [3] has been done in [16]. The framework there, however, considers only mutations that change literals in the formula to *true* or *false*. Thus, it adapts the propositional approach of [6,24] and does not involve mutations applied to the real-time aspects of the specifications.

In this paper, we extend the general notion of vacuity to the satisfaction of real-time properties. We focus on the temporal logic *Timed Computation Tree Logic* (TCTL, for short) [1]. The logic TCTL has a single temporal operator U^J, where $J \subseteq \mathbb{R}_{\geq 0}$ is an interval with integer bounds. The semantics of TCTL is defined over *Timed Transition Systems* (TTSs, for short). A TTS is typically generated by a timed automaton (TA) [2], which is an automaton equipped with a finite set of clocks, and whose transitions are guarded by clock constraints.

---

[1] The above definition assumes that ψ appears once in Φ, or at least that all its occurrences are of the same polarity; a definition that is independent of this assumption replaces ψ by a universally quantified proposition [5]. Alternatively, one could focus on a single occurrence of ψ.

The mutations we apply to TCTL formulas in order to examine real-time vacuity concern the U^J operator. Unlike the approach in [6,16,24], our mutations are applied to the real-time parameter, namely the interval J. The semantics of timed eventualities suggests two conceptually different types of strengthening for the U^J operator. First, we may tighten the upper bound for the satisfaction of the eventuality; that is, reduce the right boundary of J. Such a mutation corresponds to a check whether the specification could have actually required the eventuality to be satisfied more quickly. Second, we may reduce the span of J, namely replace it by a strict subset. Such a mutation corresponds to a check whether the specification could have been more precise about the time in which the eventuality has to be satisfied. From a technical point of view, when replacing the interval J by a strict subset J' (that is, $J' \subset J \subseteq \mathbb{R}_{\geq 0}$), we distinguish between cases where J' is continuous (that is, J' is an interval of the form $[m_1, m_2]$, $(m_1, m_2]$, $[m_1, m_2)$, or (m_1, m_2), for $m_1, m_2 \in \mathbb{N}$), and cases where J' is not continuous; that is, J' is a union of intervals. Dually, a specification may be weakened, again by two types of mutations, which replace J by an interval or a union of intervals J' such that $J \subset J'$.

Given a TTS M and a TCTL formula Φ, we say that a subformula ψ with a U^J operator (that is, ψ is $A\Phi_1 U^J \Phi_2$ or $E\Phi_1 U^J \Phi_2$) is not tight in Φ with respect to M if J can be strengthened to J' and still M satisfies Φ with the tighter eventuality. For example, if $\psi = A\Phi_1 U^J \Phi_2$, then M satisfies $\Phi[\psi \leftarrow A\Phi_1 U^{J'} \Phi_2]$. We say that Φ is *timed-vacuous* in M if $M \models \Phi$ and has a subformula ψ that is not tight in Φ. Note that timed vacuity is interesting only in cases ψ affects the satisfaction of Φ in M in the untimed case. In other words, if we could have mutated ψ to *true* or *false* without affecting the satisfaction of Φ in M, then clearly mutating J is not interesting. Thus, while we focus in this paper on timed vacuity, it is important to combine it with traditional vacuity checks. Consider for example the formula $\Phi = G(req \rightarrow F^{[0,4]} grant)$, asking each request to be satisfied within 4 time units. In order to check whether the subformula $F^{[0,4]} grant$ is tight, we can model check $\Phi[F^{[0,4]} grant \leftarrow F^{[0,1]} grant]$, where requests are asked to be satisfied within 1 time unit. A model that satisfies the stronger specification then satisfies Φ timed vacuously.

The need to consider mutations in which J' is not continuous results in formulas that are not in TCTL. Indeed, in the U^J operator in TCTL, the interval J is continuous, and we would like to examine mutations that replace, for example, the interval $[0, 4]$ by the union of the intervals $[0, 1] \cup [3, 4]$. We introduce an extension TCTL$^+$ of TCTL that allows to express eventualities that occurs in a union of a constant number of intervals with integral boundaries. We prove that the complexity of TCTL$^+$ model-checking is PSPACE-complete, thus it is not more complex than TCTL model checking. The PSPACE model-checking procedure for TCTL$^+$ leads to a PSPACE algorithm for timed vacuity, and we provide a matching lower bound.

In the case of traditional vacuity, it has been recognized that vacuity results differ in their significance. While in many cases vacuity results are valued as highly informative, there are also cases in which the results are viewed as

meaningless by users. Typically, a user gets a long list of mutations that are satisfied, each pointing to a different cause for vacuous satisfaction. In [15], the authors suggest a framework for *ranking of vacuity results* for LTL formulas. The framework is based on the *probability* of the mutated specification to hold in a random computation: the lower the probability of the mutation to hold is, the more alarming the vacuity information is. For example, the probability of $G\neg req$ to hold in a random computation is low, hence if the system satisfies it, this probably needs to be examined. We suggest an extension of the framework to TCTL. The extension involves two technical challenges. First, moving to a branching-time setting requires the development of a probabilistic space for trees (rather than computations). Second, the timed setting requires the probabilistic space to capture continuous time. We argue that once we move to an approximated reasoning about the probability of the mutations, the framework in [15] can be easily extended to TCTL, thus vacuity results can be ranked efficiently.

2 Preliminaries

2.1 TCTL, Timed Automata, and Timed Transition Systems

We assume that the reader is familiar with the branching time temporal logic CTL. We consider the logic Timed CTL (TCTL) [1], which is a real time extension of CTL. Formulas in TCTL are defined over a set AP of atomic propositions and use two path quantifiers A (for all paths) and E (exists a path), and one temporal operator U^J. A TCTL path formula is defined by the syntax $\varphi ::= \Phi_1 U^J \Phi_2$, where $J \subseteq \mathbb{R}_{\geq 0}$ is an interval whose bounds are natural numbers. Thus, the interval J is of the form $[m_1, m_2], (m_1, m_2], [m_1, m_2)$, or (m_1, m_2), for $m_1, m_2 \in \mathbb{N}$ and $m_1 \leq m_2$. For right-open intervals, we have $m_2 = \infty$. We refer to the quantity $m_2 - m_1$ as the *span* of J. Note that the next-step X operator of CTL is absent in TCTL, as time is considered to be continuous.

A timed automaton (TA, for short) is a non-deterministic finite state automaton that allows modelling of actions or events to take place at specific time instants or within a time interval. A TA expresses timed behaviours using a finite number of clock variables. All the clocks increase at the same rate. We use lower case letters x, y, z to denote clock variables and C to denote the set of clock variables. Clock variables take non-negative real values.

A *guard* is a conjunction of assertions of the form $x \sim k$ where $x \in C$, $k \in \mathbb{N}$ and $\sim \in \{\leq, <, =, >, \geq\}$. We use $\mathcal{B}(C)$ to denote the set of guards. A *clock valuation* or a valuation for short is a point $v \in \mathbb{R}_{\geq 0}^C$. For a clock $x \in C$, we use $v(x)$ to denote the value of clock x in v. We use $\lfloor v(x) \rfloor$ to denote the integer part of $v(x)$ while $frac(v(x))$ is used to denote the fractional part of $v(x)$. We define $\lceil v(x) \rceil$ as $\lfloor v(x) \rfloor + 1$ if $frac(v(x)) \neq 0$, else $\lceil v(x) \rceil = \lfloor v(x) \rfloor$. Along with other propositions, we will also use propositions of the form $v(x) \in J$, where $v(x)$ is the valuation of clock x and J is an interval with integer boundaries.

For a clock valuation v and $d \in \mathbb{R}_{\geq 0}$, we use $v + d$ to denote the clock valuation where every clock is being increased by d. Formally, for each $d \in \mathbb{R}_{\geq 0}$, the valuation $v + d$ is such that for every $x \in C$, we have $(v + d)(x) = v(x) + d$.

For a clock valuation v and a set $R \subseteq C$, we use $v_{[R \leftarrow \bar{0}]}$ to denote the clock valuation in which every clock in R is set to zero, while the value of the clocks in $C \backslash R$ remains the same as in v. Formally, for each $R \subseteq C$, the valuation $v_{[R \leftarrow \bar{0}]}$ is such that for every $x \in C$, we have

$$v_{[R \leftarrow \bar{0}]}(x) = \begin{cases} 0 & \text{if } x \in R \\ v(x) & \text{otherwise} \end{cases}$$

A timed automaton is defined by the tuple $\langle AP, \mathcal{L}, l_0, C, E, L \rangle$, where AP is a set of atomic propositions, $\mathcal{L}$ is a finite set of locations, $l_0 \in L$ is an initial location, C is a finite set of clocks, $E \subseteq \mathcal{L} \times \mathcal{B}(C) \times 2^C \times \mathcal{L}$ is a finite set of edges, and $L : \mathcal{L} \mapsto 2^{AP}$ is a labeling function.

A timed transition system, (TTS for short) [21], is $S = \langle AP, Q, q_0, \rightarrow, \hookrightarrow, L \rangle$, where AP is a set of atomic propositions, Q is a set of states, $q_0 \in Q$ is an initial state, $\rightarrow \subseteq Q \times \mathbb{R}_{\geq 0} \times Q$ is a set of delay transitions, $\hookrightarrow \subseteq Q \times Q$ is a set of discrete transitions, and $L : Q \mapsto 2^{AP}$ is a labelling function. We write $q \xrightarrow{d} q'$ if $(q, d, q') \in \rightarrow$ and write $q \hookrightarrow q'$ if $(q, q') \in \hookrightarrow$.

Let $\mathcal{A} = \langle AP, \mathcal{L}, l_0, C, E, L \rangle$ be a timed automaton. The semantics of a timed automaton is described by a TTS. The timed transition system $T(\mathcal{A})$ generated by $\mathcal{A}$ is defined as $T(\mathcal{A}) = (AP, Q, q_0, \rightarrow, \hookrightarrow, L)$, where

- $Q = \{(l, v) \mid l \in \mathcal{L}, v \in \mathbb{R}_{\geq 0}^C\}$. Intuitively, a state (l, v) corresponds to A being in location l and the clock valuation is v. Note that due to the real nature of time, this set is generally uncountable.
- Let v_{init} denote the valuation such that $v_{init}(x) = 0$ for all $x \in C$. Then $q_0 = (l_0, v_{init})$.
- $\rightarrow = \cup_{l \in \mathcal{L}} \cup_{v \in \mathbb{R}_{\geq 0}^C} \cup_{d \in \mathbb{R}_{\geq 0}} \{(l, v), d, (l, v+d)\}$.
- $\hookrightarrow = \{((l, v), (l', v')) \mid (l, v), (l', v') \in Q$ and there is an edge $e = (l, g, R, l') \in E$ and $v \models g$ and $v' = v_{[R \leftarrow \bar{0}]}\}$. That is, if there exists a discrete transition from l that is guarded by g and leads to l' while resetting the clocks in R, then if $v \models g$, the TTS can move from (l, v) to $(l, v_{[R \leftarrow \bar{0}]})$.

A run of a timed automaton is a sequence of the form $\pi = (l_0, v_0), (l_0, v'_0), (l_1, v_1),$ $(l_1, v'_1), (l_2, v_2), \dots$ where for all $i \geq 0$, we have $(l_i, v_i) \xrightarrow{d_i} (l_i, v'_i)$, i.e., $v'_i = v_i + d_i$ and $((l_i, v'_i), (l_i, v_{i+1})) \in \hookrightarrow$. Note that π is a continuous run in the sense that for a delay transition $(l_i, v_i) \xrightarrow{d_i} (l_i, v'_i)$, the run also includes all states $(l_i, v_i + d)$ for all $0 \leq d \leq d_i$, where $v'_i = v_i + d_i$. For a time $t_{\geq 0}$, we denote by $\pi[t]$ the state in π that is reached after elapsing time t.

Given a state $s \in Q$ of a TTS, we denote by $Paths(s)$, the set of all runs starting at s. Let AP be a set of action propositions. The satisfaction relation for TCTL formulas is as follows.

- $s \models p$, for $p \in AP$, iff $p \in L(s)$.
- $s \models \neg \Phi$ iff $s \not\models \Phi$.
- $s \models \Phi_1 \wedge \Phi_2$ iff $s \models \Phi_1$ and $s \models \Phi_2$.

  - $s \models A\Phi_1 U^J \Phi_2$ iff for all paths $\pi \in Paths(s)$, there exists a time $t \in J$ such that $\pi[t] \models \Phi_2$ and for all $t' < t$, we have $\pi[t'] \models \Phi_1$.
  - $s \models E\Phi_1 U^J \Phi_2$ iff there exists a path $\pi \in Paths(s)$ and a time $t \in J$ such that $\pi[t] \models \Phi_2$ and for all $t' < t$, we have $\pi[t'] \models \Phi_1$.

We say that a timed automaton $\mathcal{A}$ satisfies a TCTL formula Φ if the initial state (l_0, v_0) of $T(\mathcal{A})$ satisfies Φ.

Consider a timed automaton $A = \langle AP, \mathcal{L}, l_0, C, E, L \rangle$. For each clock $x \in C$, let M_x be the maximum constant that appears in A. We say that two valuations, v and v', are *region equivalent* with respect to A, denoted $v \equiv v'$, iff the following conditions are satisfied.

1. $\forall x \in C$, we have that $v(x) > M_x$ iff $v'(x) > M_x$, and
2. $\forall x \in C$, if $v(x) \leq M_x$, then $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ and $\lceil v(x) \rceil = \lceil v'(x) \rceil$, and
3. for each pair of clocks $x, y \in C$, if $v(x) \leq M_x$ and $v(y) \leq M_y$, we have $v(x) < v(y)$ iff $v'(x) < v'(y)$.

For a given valuation v, let $[v] = \{v' | v' \equiv v\}$. Note that the region equivalence is actually an equivalence relation. Every such equivalence class is called a *region*. Note that the number of regions is exponential in the number of clocks of a timed automaton. A tuple of the form $\langle l, [v] \rangle$, where $l \in \mathcal{L}$ is a location and $[v]$ is a region is called a *symbolic state* of A. Two valuations v and v' of a symbolic state $\langle l, [v] \rangle$ satisfy the same set of TCTL formulas [1].

A *region graph* is defined using the transition relation $\rightsquigarrow$ between symbolic states which is as follows:

  - $(l, [v]) \rightsquigarrow (l, [v'])$ if there exists a $d \in \mathbb{R}_{\geq 0}$ such that $(l, v) \xrightarrow{d} (l, v')$ and
  - $(l, [v]) \rightsquigarrow (l', [v'])$ if $(l, v) \hookrightarrow (l', v')$.

We note that the transition relation is finite since there are finitely many symbolic states. The *size of a region graph* equals the sum of the number of regions and the number of transitions in it. The region equivalence relation partitions the uncountably many states into a finite number of symbolic states.

2.2   Timed Vacuity

Consider a TCTL formulas Φ. Let ψ be a subformula of Φ, and let ξ be a TCTL formula. We use $\Phi[\psi \leftarrow \xi]$ to denote the TCTL formula obtained from Φ by replacing ψ by ξ.[2] Consider a TA $\mathcal{A}$ such that $\mathcal{A}$ satisfies Φ. We say that a subformula ψ of Φ *does not affect the satisfaction* of Φ in $\mathcal{A}$ (ψ does not affect Φ in $\mathcal{A}$, for short) if $\mathcal{A}$ satisfies $\Phi[\psi \leftarrow \xi]$ for every formula ξ. The above definition adapts the propositional approach of [6,24] to TCTL, and as we show in Lemma 2, checking this type of vacuity is easy and does not add challenges that are unique to the real-time setting. We thus focus on mutations applied to the real-time operator of TCTL.

---

[2] As discussed on Sect. 1, we assumes that ψ appears once in Φ or focus on a single occurrence of ψ in Φ.

Consider two intervals $J, J' \subseteq \mathbb{R}_{\geq 0}$ such that $J' \subset J$. Clearly, the TCTL formula $Q\Phi_1 U^J \Phi_2$, for $Q \in \{A, E\}$, is weaker than the formula $Q\Phi_1 U^{J'} \Phi_2$. In timed vacuity, we are interested in formulas of the form $Q\Phi_1 U^J \Phi_2$ in which the interval J can be mutated. Rather than considering all subsets of J, we restrict attention to subsets that are intervals or the union of two intervals where the interval boundaries are integers. Formally, we say that J' is a *strengthening* of J if $J' \subset J$ and either J' is an interval or $J' = J_1 \cup J_2$ for intervals J_1 and J_2. For example, $(2, 5]$ and $[2, 3) \cup (4, 5]$ are both strengthenings of $[2, 6)$. Dually, J' is a *weakening* of J if $J \subset J'$ and either J' is an interval or $J' = J_1 \cup J_2$ for continuous J_1 and J_2. We note here that dividing J into more than two intervals might not capture the user's intent.

A subformula ψ of Φ may have either *positive polarity* in Φ, namely be in the scope of an even number of negations, or a *negative polarity* in Φ, namely be in the scope of an odd number of negations (note that an antecedent of an implication is considered to be under negation).

Consider a TA $\mathcal{A}$ and a TCTL formula Φ such that $\mathcal{A} \models \Phi$. Let $\psi = Q\Phi_1 U^J \Phi_2$ be a subformula of Φ. We say that the ψ *is not tight in* Φ with respect to $\mathcal{A}$ if ψ is in a positive polarity and J can be strengthened to J' or ψ is in a negative polarity and J can be weakened to J' and we have that $\mathcal{A} \models \Phi[\psi \leftarrow Q\Phi_1 U^{J'} \Phi_2]$. We say that Φ is *timed-vacuous* in $\mathcal{A}$ if $\mathcal{A} \models \Phi$ and has a subformula ψ that is not tight in Φ.

3   TCTL$^+$ and Its Model Checking

Recall that in the definition of strengthening and weakening, we allowed the replacement of an interval J by the union of intervals. In order to handle such tightenings, we introduce an extension of TCTL, called TCTL$^+$, where path formulas may be a disjunction of formulas of the form $\Phi_1 U^J \Phi_2$. The semantics is extended in the expected way. That is, $s \models A \bigvee_{1 \leq i \leq k} \Phi_1^i U^{J_i} \Phi_2^i$ iff for all paths $\pi \in Paths(s)$ and for every $1 \leq i \leq k$, there exists a time $t_i \in J_i$ such that $\pi[t_i] \models \Phi_2^i$ and for all $t_i' < t_i$, we have $\pi[t_i'] \models \Phi_1^i$. The definition for the existential case is similar.

Remark 1. In CTL, allowing Boolean operations within the path formulas does not extend the expressive power [17]. In particular, the formula $A(Fp \vee Fq)$, for $p, q \in AP$, is equivalent to the CTL formula $AF(p \vee q)$. We conjecture that in the timed setting, Boolean operations within the path formulas do extend the expressive power. In particular, we conjecture that the TCTL$^+$ formula $A(F^{[1,2]}p \vee F^{[3,4]}q)$, does not have an equivalent TCTL formula. We also think that a technical proof for the above statement is highly non-trivial.

In this section we show that model-checking of TCTL$^+$ formulas is PSPACE-complete. First, recall that TCTL model-checking is PSPACE-complete [1]. TCTL model-checking is done by reducing it to CTL model-checking over region graph. Let $\mathcal{A} = \langle AP, \mathcal{L}, l_0, C, E, L \rangle$ be a TA and Φ be a TCTL formula and we want to check if $\mathcal{A} \models \Phi$. Let z be a clock that is not in the set C of clocks of the

TA $\mathcal{A}$. Consider the TTS $T(\mathcal{A})$ generated by $\mathcal{A}$. For a state $s = (l, v)$ in $T(\mathcal{A})$, we denote by $s[z = 0]$, the state (l, v') such that $v' \in \mathbb{R}_{\geq 0}^{C \cup \{z\}}$ and $v'(z) = 0$ and for all clocks $x \in C$, we have $v'(x) = v(x)$. We construct the region graph of $\mathcal{A}$ with this additional clock z such that z is never reset. Now for every state s in the TTS, we say that $s \models E(\Phi_1 U^J \Phi_2)$ iff $s[z = 0] \models E(\Phi_1 U(z \in J \wedge \Phi_2))$ and $s \models A(\Phi_1 U^J \Phi_2)$ iff $s[z = 0] \models A(\Phi_1 U(z \in J \wedge \Phi_2))$.

Algorithm 1. Computation of satisfaction set of TCTL$^+$ formula $\Phi = A(\Phi_1 U^{J_1} \Phi_2 \vee \Phi_3 U^{J_2} \Phi_4)$

Require: Region graph $R_\mathcal{A}$ of TA $\mathcal{A}$ extended with a fresh clock z that is never reset
    and a TCTL$^+$ formula Φ
Ensure: Compute $Sat(\Phi)$
1: $T_1 \leftarrow Sat(z \in J_1) \cap Sat(\Phi_2)$ ▷ T_1 is the set of vertices (regions) all whose
2:                                   successors are in $\Phi_1 U^{J_1} \Phi_2$
3: $T_2 \leftarrow Sat(z \in J_2) \cap Sat(\Phi_4)$ ▷ T_2 is the set of vertices (regions) all whose
4:                                   successors are in $\Phi_3 U^{J_2} \Phi_4$
5: $T_3 \leftarrow \emptyset$ ▷ T_3 is the set of vertices (regions) from which some paths satisfy
6:                  $\Phi_1 U^{J_1} \Phi_2$ while the other paths satisfy $\Phi_3 U^{J_2} \Phi_4$
7: $T \leftarrow T_1 \cup T_2$ ▷ The satisfaction set of Φ, finally also includes T_3
8: $T_4 \leftarrow \emptyset$ ▷ T_4 is the set of regions r for which all successor regions
9:                         are in T but r itself is not in the satisfaction set T
10: **while** $\{r \in Sat(\Phi_1) \cup Sat(\Phi_3) - (T \cup T_4) \mid Post(r) \subseteq T\} \neq \emptyset$ **do**
11:      Let $r \in Sat(\Phi_1) \cup Sat(\Phi_3) - (T \cup T_4) \mid Post(r) \subseteq T$
12:      **if** $r \in Sat(\Phi_1) - T$ **then**
13:          **if** $Post(r) \subseteq T_1$ **then**
14:              $T_1 \leftarrow T_1 \cup \{r\}$
15:              $T \leftarrow T \cup \{r\}$
16:          **end if**
17:      **end if**
18:      **if** $r \in Sat(\Phi_2) - T$ **then**
19:          **if** $Post(r) \subseteq T_2$ **then**
20:              $T_2 \leftarrow T_2 \cup \{r\}$
21:              $T \leftarrow T \cup \{r\}$
22:          **end if**
23:      **end if**
24:      **if** $r \in Sat(\Phi_1) \cap Sat(\Phi_3) - T$ and $Post(r) \subseteq T$ **then**
25:          **if** there exist two distinct successors r_1 and r_2 of r such that $r_1 \in T1 - T2$
     and $r2 \in T2 - T1$ or there exists a successor r' of r such that $r' \in T_3$ **then**
26:              $T_3 \leftarrow T_3 \cup \{r\}$
27:              $T \leftarrow T \cup \{r\}$
28:          **end if**
29:      **end if**
30:      **if** $r \notin T$ **then** $T_4 = T_4 \cup \{r\}$
31:      **end if**
32: **end while**
33: **return** T

Let $R_\mathcal{A}$ be the region graph of TA $\mathcal{A}$. The model-checking procedure involves CTL model-checking $(l_0, v_{init})[z = 0] \models \Phi'$, over a region graph over the set $C \cup \{z\}$ of clocks such that clock z is never reset. The formula Φ' is obtained from Φ by modifying the path formulas by introducing subformulas of the form $z \in J$ as mentioned above.

For a TCTL$^+$ formula Φ, the set of regions in the region graph of $\mathcal{A}$ that satisfy Φ is denoted by $Sat(\Phi)$. In Algorithm 1, we give a method to compute the satisfaction set for the TCTL$^+$ formula $\Phi = A(\Phi_1 U^{J_1} \Phi_2 \vee \Phi_3 U^{J_2} \Phi_4)$. Given a region r in a region graph, we denote by $Post(r)$, the set of regions that can be reached from r in a single step, i.e. using the transition $\rightsquigarrow$ once.

The running time of Algorithm 1 is proportional to the size of the region graph and the size of the TCTL$^+$ formula. It is easy to see that it runs in PSPACE. We note here that given a TA and a TCTL formula, model-checking of TCTL also takes time that is proportional to the size of the region graph of the TA and the size of the TCTL formula.

Lemma 1. *Algorithm 1 runs in PSPACE.*

In a TCTL$^+$ formula Φ, if we have arbitrary number of disjunctions of the form $\Phi_{1_1} U^J \Phi_{1_2}, \Phi_{2_1} U^J \Phi_{2_2}, \ldots, \Phi_{n_1} U^J \Phi_{n_2}$, then we need to maintain different subsets of the set $\mathcal{I} = \{J_1, \ldots, J_n\}$ of intervals. For each subset I of $\mathcal{I}$, we have the set of vertices (regions) from which for every member $J \in I$, there exists a path satisfying the disjunct $\Phi_1 U^J \Phi_2$. Hence with arbitrary number of intervals, the algorithm is also exponential in the size of the formula, but is still in PSPACE.

Showing PSPACE-hardness for TCTL$^+$ model-checking follows directly from the complexity of TCTL model-checking, since the syntax of TCTL is a proper subset of the syntax of TCTL$^+$. Thus we have the following theorem.

Theorem 1. *The model-checking problem for TCTL$^+$ is PSPACE-complete.*

4   Satisfying a TCTL Formula Timed Vacuously

In this section, we study the complexity of checking vacuity and timed vacuity in TCTL. We describe algorithms for strengthening an interval J in formulas of the form $Q\Phi_1 U^J \Phi_2$ and algorithms for checking timed vacuity.

4.1   Complexity Results

We start with the propositional approach, where a subformula ψ of Φ does not affect the satisfaction of Φ in a TA $\mathcal{A}$ iff $\mathcal{A}$ satisfies $\Phi[\psi \leftarrow \xi]$ for all formulas ξ. This definition is not effective, as it requires evaluating $\Phi[\psi \leftarrow \xi]$ for all formulas ξ. We first prove that as in the case of CTL [24], also in TCTL it is possible to check only the replacements of ψ by *true* and *false*.

Lemma 2. *For every subformula ψ of a TCTL formula Φ and for every TA $\mathcal{A}$ such that $\mathcal{A} \models \Phi$, if $\mathcal{A} \models \Phi[\psi \leftarrow \bot]$, then for every formula ξ, we have $\mathcal{A} \models \Phi[\psi \leftarrow \xi]$.*

Proof. We prove that for every formula ξ, the implications $\mathcal{A} \models \Phi[\psi \leftarrow \bot] \rightarrow \mathcal{A} \models \Phi[\psi \leftarrow \xi]$. This can be shown by proving $\Phi[\psi \leftarrow \bot] \rightarrow \Phi[\psi \leftarrow \xi]$. The proof is by structural induction on the syntax of TCTL formulas.

We only need to prove the induction step for the subformulas with a timed component, as the other cases are proved in [24].

Let $\Phi = E(\Phi_1 U^J \Phi_2)$. Thus we want to prove that for every formula ξ, we have $E(\Phi_1[\psi \leftarrow \bot] U^J \Phi_2[\psi \leftarrow \bot]) \rightarrow E(\Phi_1[\psi \leftarrow \xi] U^J \Phi_2[\psi \leftarrow \xi])$.

By induction hypothesis, we have that $E(\Phi_1[\psi \leftarrow \bot]) \rightarrow E(\Phi_1[\psi \leftarrow \xi])$ and $E(\Phi_2[\psi \leftarrow \bot]) \rightarrow E(\Phi_2[\psi \leftarrow \xi])$.

Suppose $\mathcal{A} \models E(\Phi[\psi \leftarrow \bot])$. Hence we have a time $t \in J$ and a run $\rho = (l_0, v_0) \xrightarrow{d_0} (l_0, v_0 + d) \xrightarrow{\tau} (l_1, v_1) \xrightarrow{d_1} \ldots \xrightarrow{d_i} (l_i, v_i) \ldots$ such that $t = \sum_{j=0}^{i} d_j$, and $s_i = (l_i, v_i) \models E(\Phi_2[\psi \leftarrow \bot])$. By the induction hypothesis, $(l_i, v_i) \models E(\Phi_2[\psi \leftarrow \xi])$. Also since $\mathcal{A} \models E(\Phi[\psi \leftarrow \bot])$, we have that over all the intermediate (possibly uncountably many) states s in ρ from (l_0, v_0) to s_i and possibly excluding s_i, we have $s \models E(\Phi_1[\psi \leftarrow \bot])$. Again, from the induction hypothesis, we have that $s \models E(\Phi_1[\psi \leftarrow \xi])$. Hence $\mathcal{A} \models E(\Phi_1[\psi \leftarrow \xi] U^J \Phi_2[\psi \leftarrow \xi])$, i.e., $\mathcal{A} \models \Phi[\psi \leftarrow \xi]$.

The proof for $A(\Phi_1 U^J \Phi_2)$ is similar. □

Theorem 2. *Given a TA $\mathcal{A}$, a TCTL formula Φ and a subformula ψ of Φ such that $\mathcal{A} \models \Phi$, the problem of checking whether ψ does not affect Φ in $\mathcal{A}$ is PSPACE-complete.*

Proof. We prove that the complementary problem, of deciding whether ψ affects Φ in $\mathcal{A}$ is PSPACE-complete. Since PSPACE is closed under complementation, the result follows. The membership follows from the fact that whether ψ affects the satisfaction of Φ in $\mathcal{A}$ can be decided by checking whether $\mathcal{A} \models \Phi[\psi \leftarrow \bot]$ and that TCTL model-checking is PSPACE-complete [1].

We prove PSPACE-hardness using a reduction from TCTL model checking. We construct a formula $\neg \Phi'$, a subformula ψ of $\neg \Phi'$ and a TA $\mathcal{A}'$ such that $\mathcal{A} \models \Phi$ iff ψ affects the satisfaction of $\neg \Phi'$ in $\mathcal{A}'$. Let $\mathcal{A} = \langle AP, \mathcal{L}, l_0, E, C, L \rangle$. We define $\mathcal{A}' = \langle AP', \mathcal{L}, l_0, E, C, L' \rangle$, such that $AP' = AP \cup \{q, r\}$ and $q, r \notin AP$ and $L'(l_0) = L(l_0) \cup \{r\}$ and $L'(l) = L(l)$ for $l \neq l_0 \in \mathcal{L}$. Let $\Phi' = q \wedge \Phi$. Clearly $\mathcal{A}' \not\models \Phi'$ and hence $\mathcal{A}' \models \neg \Phi'$. If $\mathcal{A} \models \Phi$, then considering $\psi = q$ and $\xi = r$, we have that $\mathcal{A}' \models \Phi'[\psi \leftarrow \xi]$, and hence $\mathcal{A}' \not\models \neg \Phi'[\psi \leftarrow \xi]$, i.e., q affects the satisfaction of Φ' in $\mathcal{A}'$. Now consider the case $\mathcal{A} \not\models \Phi$. Then also $\mathcal{A}' \models \Phi$ and hence $\mathcal{A}' \not\models q \wedge \Phi = \Phi'$. Thus if $\mathcal{A} \not\models \Phi$, we still have $\mathcal{A}' \models \neg \Phi$. For all ξ, we have $\mathcal{A}' \not\models \Phi'[\psi \leftarrow \xi]$, and hence $\mathcal{A}' \models \neg \Phi'[\psi \leftarrow \xi]$, i.e. q does not affect the satisfaction of Φ' in $\mathcal{A}'$. □

We now proceed to timed vacuity. In Theorem 3 below, we use a reduction from the set consisting of the true quantified boolean formulas (TQBF) which is a canonical PSPACE-complete problem.

Theorem 3. *Given a TA $\mathcal{A}$, a formula Φ such that $\mathcal{A} \models \Phi$ and a subformula $\psi = Q\Phi_1 U^J \Phi_2$ of Φ (where Q stands for a path quantifier A or E), the problem of checking whether ψ is not tight in Φ with respect to $\mathcal{A}$ is PSPACE-complete.*
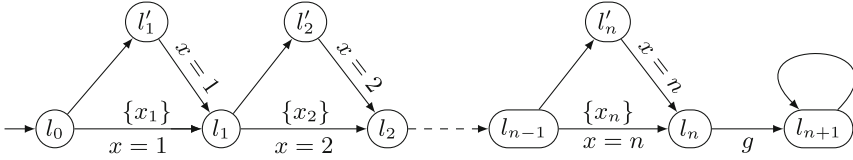
Fig. 1. Timed Automata encoding QBF in the proof of Theorem 3

Proof. We show that the complementary problem, of deciding whether ψ is tight in Φ with respect to $\mathcal{A}$ is PSPACE-complete. Since PSPACE is closed under complementation, the result follows. The membership follows from the fact that whether ψ is tight in Φ with respect to $\mathcal{A}$ can be decided by checking whether $\mathcal{A} \models \Phi[\psi \leftarrow Q\Phi_1 U^{J'}\Phi_2]$ where J' is a largest proper subset of J when ψ has a positive polarity and J' is a smallest proper superset of J when ψ has a negative polarity and this can be done in PSPACE. For example, if J is of the form $[m_1, m_2]$ and ψ has a positive polarity, then we consider replacing J with both $(m_1, m_2]$ and $[m_1, m_2)$, while if ψ has a negative polarity, then we consider replacing J with both $(m_1 - 1, m_2]$ and $[m_1, m_2 + 1)$.

For the lower bound, we show a reduction from TQBF[3], which is known to be PSPACE-complete [28]. Let $\alpha = Q_1 p_1.Q_2 p_2 \ldots Q_n p_n.\beta(p_1, \ldots, p_n)$ be a quantified boolean formula (QBF), such that β is a propositional formula over the propositions $p_1, \ldots, p_n$, and each $Q_i \in \{\exists, \forall\}$ is an existential or a universal quantifier.

Consider the timed automaton $\mathcal{A} = \langle \{p\}, \{l_0, \ldots, l_n\}, l_0, C, E, L \rangle$ shown in Fig. 1. The set of clocks C is $\{x, x_1, \ldots, x_n\}$, i.e., we have a clock x and for each proposition p_i, with $1 \leq i \leq n$, we have a clock x_i. The guard g is obtained from β by replacing each p_i in β with the atomic formula $x_i = n + 1$. For example, considering $n = 3$, if $\beta(p_1, p_2, p_3) = (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee p_3)$, then $g = (x_1 = 4 \vee x_2 \neq 4) \wedge (x_1 \neq 4 \vee x_3 = 4)$. We have $L(l_{n+1}) = \{p\}$, and for all $0 \leq i \leq n$, we have $L(l_i) = \emptyset$.

For every run of $\mathcal{A}$, for every $1 \leq i \leq n$, the location l_i is reached at time i. Let v be the clock valuation at time n when location l_n is reached. Note that $v(x) = n$, and for every $1 \leq i \leq n$, we have that $v(x_i)$ equals n if at location l_{i-1}, the edge (l_{i-1}, l'_i) is taken and equals $n - i$, if at location l_{i-1}, the edge (l_{i-1}, l_i) is taken. There are 2^n different paths from l_0 to l_n, and each can be viewed as encoding a different truth assignment for the n propositions $p_1, \ldots, p_n$. Location l_{n+1} is reached at time $n + 1$ with clock valuation $v + 1$. Let Φ be the TCTL formula $Q'_1 F^{[0,1)}(Q'_2 F^{[1,1]}(Q'_3 F^{[1,1]} \ldots Q'_n F^{[1,1]}(EF^{[1,1]}p)) \ldots)$ with $Q'_i = A$ if $Q_i = \forall$ and $Q'_i = E$ if $Q_i = \exists$ for all $1 \leq i \leq n$. Note that $\mathcal{A} \not\models \Phi$ since location l_1 is reached only after 1 unit of time elapses and we have $Q'_1 F^{[0,1)}$. Thus $\mathcal{A} \models \neg\Phi$. We claim that if α is satisfiable, then the subformula $\psi = \Phi$ of $\neg\Phi$ is tight in Φ with respect to $\mathcal{A}$. Note that ψ has a negative polarity in $\neg\Phi$,

---

[3] A similar construction has been used in [1] for showing the lower bound of model-checking TCTL formulas, but the proof we have here is different and more involved.

and hence we consider weakening the interval $[0, 1)$ in $Q'_1 F^{[0,1)}$. We consider the minimum possible weakening of $[0, 1)$ which gives us the interval $[0, 1]$. Let Φ' be the formula obtained from Φ by replacing $Q'_1 F^{[0,1)}$ with $Q'_1 F^{[0,1]}$. Note that $\mathcal{A} \models \Phi'$ and hence $\mathcal{A} \not\models \neg\Phi'$. Thus the formula $\psi = \Phi$ is tight in $\neg\Phi$ w.r.t. $\mathcal{A}$.

On the other hand, if α is not satisfiable, then also $\mathcal{A} \not\models \Phi$ and hence $\mathcal{A} \models \neg\Phi$, and no matter how we weaken the interval $F^{[0,1)}$ in $Q'_1 F^{[0,1)}$ in $\neg\Phi$, we have that $\mathcal{A}$ satisfies the resultant formula. Thus if α is not satisfiable, then ψ is not tight in Φ with respect to $\mathcal{A}$, and we are done. $\qquad\square$

Theorem 4. *Given a TA $\mathcal{A}$ and a TCTL formula Φ, checking whether Φ is timed vacuous in $\mathcal{A}$ is in PSPACE.*

Proof. By Theorem 3, for a given ψ, the problem of checking whether ψ is not tight is in PSPACE. Since there are there are $|\Phi|$ subformulas, we are done. $\quad\square$

4.2 Algorithms for Tightening TCTL Formulas

In this section we propose some algorithms for tightening TCTL formulas of the form $Q\Phi_1 U^J \Phi_2$, for $Q \in \{A, E\}$. We consider two types of tightening that are the most interesting ones from a user's perspective: One in which the interval J is strengthened to an interval that ends at the earliest possible time, and the other one in which J is strengthened to the smallest possible span.

Given a TA $\mathcal{A}$, a TCTL formula Φ and a subformula ψ of Φ, we propose an algorithm that strengthens J so that it ends at the earliest possible time. The algorithm proceeds as follows. Consider a path formula $\psi = \Phi_1^{\prec l, r \succ} \Phi_2$, where $\prec l \in \{[l, (l\}$ and $r \succ \in \{r), r]\}$, and assume that ψ appears in a positive polarity. Then, ψ is tightened to $\psi' = \Phi_1 U^{\prec l', r' \succ} \Phi_2$ if there is no $\psi'' = \Phi_1 U^{\prec l'', r'' \succ} \Phi_2$ such that $\mathcal{A} \models \Phi[\psi \leftarrow \psi'']$ and one of the following holds:

1. $r'' < r'$.
2. $r'' = r'$, $\psi'' = \Phi_1 U^{\prec l'', r')} \Phi_2$, and $\psi' = \Phi_1 U^{\prec l', r']} \Phi_2$.
3. $r'' \succ= r' \succ$ and $l' < l''$.
4. $r'' \succ= r' \succ$, $l' = l''$, $\psi'' = \Phi_1 U^{(l', r' \succ} \Phi_2$, and $\psi' = \Phi_1 U^{[l'', r' \succ} \Phi_2$.

Given a TA $\mathcal{A}$, a TCTL formula Φ such that $\mathcal{A} \models \Phi$, and a subformula $\psi = \Phi_1 U^{\prec l, r \succ} \Phi_2$ of Φ, we first fix the right boundary, i.e. change $r \succ$ to $r' \succ'$ and get a formula $\psi' = \Phi_1 U^{\prec l, r' \succ'} \Phi_2$. The bound is found by a binary search on the right boundary, while keeping $\prec l$ fixed and each time dividing the current interval in two. Once the right boundary is fixed in the subformula, the left boundary is tightened in a similar way.

The algorithm makes $\mathcal{O}(\log(n))$ calls to TCTL model-checking procedure where $n = r - l$.

With inputs from a user, we can further tighten TCTL formulas is to split the interval, resulting in a TCTL$^+$ formula. As the split might result in a pair of intervals such that their union is smaller than the original interval, the user might choose this tightening option. The split and tightening algorithm proceeds as follows. For a formula $\psi = \Phi_1 U^J \Phi_2$, let $J = \prec l, r \succ$. The user can specify

how to split the J, i.e. remove an interval I from J and check if $\mathcal{A}$ satisfies the formula $\Phi[\psi \leftarrow (\Phi_1 U^{\prec l, \inf(I)]}\Phi_2) \vee (\Phi_1 U^{[\sup(I), r\succ]}\Phi_2)]$. If $\mathcal{A}$ satisfies the formula, then the intervals in each of the disjunct can be strengthened subsequently. A user can actually use the algorithm to strengthen an interval J to J' and then split J' leading to a TCTL$^+$ formula.

The second interesting algorithm tightens a subformula $\Phi_1 U^J \Phi_2$ by strengthening J to the smallest possible single interval $J' \subseteq J$. This algorithm performs a binary search on the length of J' and makes $\mathcal{O}(n \log(n))$ calls to TCTL model-checking procedure (compared with the naive approach that tries all possibilities, hence making $\mathcal{O}(n^2)$ calls).

If ψ has a negative polarity then we weaken J to get the largest possible interval. We argue below that the weakening too can be done in PSPACE. If $\psi = \Phi_1 U^J \Phi_2$ has a negative polarity then we weaken J to J'. Suppose $J = \prec m_1, m_2 \succ$, where $\prec m_1 \in \{[, (\}$ and $m_2 \succ \in \{],)\}$. We reduce the left boundary of the interval, i.e. replace $\prec m_1$ with $[0, (0, [1, (1, \ldots, [m_1$ one after another when $\prec m_1 = (m_1$ and replace $\prec m_1$ with $[0, (0, [1, (1, \ldots, (m_1 - 1$ when $\prec m_1 = [m_1$ one after another and for each replacement check whether $\mathcal{A}$ still satisfies the formula obtained after the replacement.

Once we fix the left boundary to $\prec l$, we check how far the right boundary can be increased. Finding this maximum right boundary is tricky. We first replace $m_2 \succ$ with $\infty)$ and check if the formula obtained by replacing J with $J' = \prec l, \infty)$ in ψ is satisfied by $\mathcal{A}$. If $\mathcal{A}$ satisfies the formula obtained by replacing J with J', we are done. Otherwise, Let R be the number of regions in the region graph of $\mathcal{A}$. All valuations of a region satisfy the same set of TCTL formulas [1] and the amount by which the right boundary of J can be increased is related to the number of regions in the region graph of $\mathcal{A}$. We note that from any region r, a given region r' can be reached within a maximum time of R time units. If $\mathcal{A}$ does not satisfy the formula obtained by replacing J with J', then we find the maximal weakening of the right interval by replacing $m_2 \succ$ with $m_2 + R]$, $m_2 + R)$, $m_2 + R - 1]$, $\ldots$, $m_2]$ one after another when $m_2 \succ$ is $m_2)$ and by replacing $m_2 \succ$ with $m_2 + R]$, $m_2 + R)$, $m_2 + R - 1]$, $\ldots$, $m_2 + 1)$ one after another when $m_2 \succ$ is $m_2]$ and checking if $\mathcal{A}$ satisfies the formula obtained after the replacement.

5   Ranking Vacuity Results

In [15], the authors suggest to rank vacuity results for LTL according to their significance, where significance is defined using probability. The probabilistic model in [15] is that for each atomic proposition p and for each state in a random computation π, the probability of p to hold in the state is $\frac{1}{2}$. Then, $pr(\Psi)$, namely the probability of an LTL formula Ψ, is defined as the probability of Ψ to hold in a random infinite computation. To see the idea behind the framework, consider the LTL specification $G(req \rightarrow F\,grant)$ and its mutations $G(\neg req)$ and $GF\,grant$. It not hard to see that in the probabilistic model above, the probability of $G(\neg req)$ to hold in a random infinite computation is 0, whereas the probability

of $GF\,grant$ to hold is 1. It is argued in [15] that the lower is the probability of the mutation to hold in a random computation, the higher the vacuity rank should be. In particular, vacuities in which the probability of the mutation is 0, as is the case with $G(\neg req)$, should get the highest rank and vacuities in which the probability is 1, as is the case with $GF(grant)$, should get the lowest rank. Intuitively, when a mutation with a low probability holds, essentially against all chances, then the user should be more alarmed than when a mutation with a high probability holds, essentially as expected.

Since the problem of calculating $pr(\Psi)$ is PSPACE-complete [14,15], an efficient way to obtain an *estimated* probability of satisfaction in random computations has been proposed in [15]. Rather than a probability in $[0,1]$, the estimation is *three valued*, returning 0, 1, or $\frac{1}{2}$, with $\frac{1}{2}$ indicating that the estimated probability is in $(0,1)$. Extending the framework to TCTL involves two technical challenges: a transition to a branching-time setting, and a transition to a timed setting. As we show below, once we compensate on an estimated reasoning, the transitions do not require new techniques.

Let us start with the transition to the branching setting. Recall that the probabilistic model in [15] defines $pr(\Psi)$ as the probability of Ψ to hold in a random infinite computation. Thus, [15] ignores the structure of the analyzed system, in particular the fact that infinite computations are generated by finitely many states. This makes a difference, as, for example, the probability of Gp to hold in a computation generated by n states is $\frac{1}{2^n}$, whereas $pr(Gp) = 0$. In the branching setting, ignoring the structure of the analyzed system plays an additional role, as it abstracts the branching degree. For example, the probability of a CTL formula AXp to hold in a state with n successors is $\frac{1}{2^n}$ (see also [10]). Note, however, that once we move to a three-valued approximation, the approximated probability of $AX\Phi$ to hold in a state agrees with the approximated probability of Φ to hold in a state, and is independent of the number of successors! Moreover, the same holds for existential path quantification: the approximation probability of $EX\Phi$ agrees with that of Φ. It follows that the calculation of the estimated probability of a CTL formula can ignore path quantification and proceeds as the one for LTL described in [15].

We continue to the timed setting and TCTL formulas. Our probabilistic model is based on random region graphs. Indeed, as the truth value of a TCTL formula in a TTS is defined with respect to the induced region graph, we define the probability of a TCTL formula as its probability to hold in a random region graph. It is easy to see that for TCTL formulas of the form *true*, *false*, p, $\neg\Phi$, and $\Phi\vee\Psi$, the estimated probability defined for CTL is valid also for TCTL. We continue to formulas of the form $A\Phi U^J\Psi$ and $E\Phi U^J\Psi$. Here too, we can ignore path quantification and observe that if the estimated probability of Ψ is 0, then so is the estimated probability of $\Phi U^J\Psi$, and similarly for 1. Another way for $\Phi U^J\Psi$ to have estimated probability 1 is when the estimated probabilities of Φ is 1, that of Ψ is in $(0,1)$, and $J = [0,\infty)$. In all other cases, the estimated probability of $\Phi U^J\Psi$ is in $(0,1)$.

By the above, the three-valued estimated probability of a TCTL formula Φ, denoted $Epr(\Phi)$, is defined by induction on the structure of the formula as follows (with $Q \in \{A, E\}$).

- $Epr(false) = 0$.
- $Epr(true) = 1$.
- $Epr(p) = \frac{1}{2}$.
- $Epr(\neg\Phi) = 1 - Epr(\Phi)$.
- $Epr(\Phi \wedge \Psi) = \begin{cases} 1 & \text{if } Epr(\Phi) = 1 \text{ and } Epr(\Psi) = 1 \\ 04 & \text{if } Epr(\Phi) = 0 \text{ or } Epr(\Psi) = 0 \\ \frac{1}{2} & \text{otherwise.} \end{cases}$
- $Epr(Q\Phi U^J \Psi) = \begin{cases} 0 & \text{if } Epr(\Psi) = 0 \\ 1 & \text{if } (Epr(\Psi) = 1) \text{ or} \\ & (Epr(\Phi) = 1, Epr(\Psi) = \frac{1}{2}, \text{ and } J = [0, \infty)) \\ \frac{1}{2} & \text{otherwise.} \end{cases}$

Recall that we calculate the three-valued estimated probability for the mutations of a given TCTL specification. Thus, the calculation may also be applied for TCTL$^+$ formulas. Fortunately, the estimated probability of a disjunction $\bigvee_{1 \leq i \leq k} \Phi^i U^{J_i} \Psi^i$ follows the same lines as these in which $k = 1$. In particular, for the purpose of calculating the estimated probability of mutations, we know that the formula at hand is obtained by strengthening $Q\Phi U^J \Psi$ by splitting J to intervals that form a strict subset of it. Hence, we can assume that the formula is of the form $Q \bigvee_{1 \leq i \leq k} \Phi U^{J_i} \Psi$ (that is, same Φ^i and Ψ^i in all disjuncts), and the union of the intervals J_i is a strict subset of $[0, \infty)$. Accordingly, we have the following.

- $Epr(Q \bigvee_{1 \leq i \leq k} \Phi U^{J_i} \Psi) = \begin{cases} 0 & \text{if } Epr(\Psi) = 0 \\ 1 & \text{if } Epr(\Psi) = 1 \\ \frac{1}{2} & \text{otherwise.} \end{cases}$

Note that the estimation not only loses preciseness when the probability is in $(0, 1)$ but also ignores semantic relations among subformulas. For example, $Epr(p \wedge \neg p) = \frac{1}{2}$, whereas $pr(p \wedge \neg p) = 0$. Such relations, however, are the reasons to the PSPACE-hardness of calculating $pr(\varphi)$ precisely, and the estimation in Epr is satisfactory, in the following sense:

Theorem 5. *For every TCTL$^+$ formula Φ, the following hold.*

- *If $pr(\Phi) = 1$, then $Epr(\Phi) \in \{1, \frac{1}{2}\}$, if $pr(\Phi) = 0$, then $Epr(\Phi) \in \{0, \frac{1}{2}\}$, and if $pr(\Phi) \in (0, 1)$, then $Epr(\Phi) = \frac{1}{2}$.*
- *$Epr(\Phi)$ be calculated in linear time.*

By Theorem 5, ranking of mutations for TCTL formulas by estimated probability of their mutations can be done in linear time. Now, one can ask how helpful the estimation is. As demonstrated in [15], the estimation agrees with the intuition of designers about the importance of vacuity information. In fact, when $Epr(\Phi)$ does not agree with $pr(\Phi)$, the reason is often *inherent vacuity* in the specification [18], as in the example of $p \wedge \neg p$ above, where we want the formula to be ranked as alarming.

6   Conclusions

Vacuity detection is a widely researched problem, with most commercial model-checking tools including an automated vacuity check. In this paper, we extended the definition of vacuity to the timed logic TCTL and demonstrated that vacuous satisfaction can indicate problems in the timing aspects of the modelling or the specification. We considered strengthening of TCTL properties resulting from tightening the interval J in the operator U^J. While we can tighten the interval in many different ways, we considered only the tightenings that preserve the user's intent: tightening the right bound (forcing the eventuality to happen as early as possible), shortening the interval (forcing it to be tighter), and replacing the interval J with a strictly smaller union of its two sub-intervals J_1 and J_2 (allowing the tightening be more precise). We note that, in principle, it is possible to examine a replacement of J by a union of a larger number of sub-intervals, incurring only a polynomial increase in runtime. Replacing an interval J with $J_1 \cup J_2$ results in a formula that is not in TCTL. We introduced an extension TCTL$^+$ of TCTL, which includes eventualities occurring in a union of a constant number of intervals and proved that TCTL$^+$ model-checking is PSPACE-complete, thus it is not higher than that of TCTL. We also proved that the vacuity problem for TCTL is in PSPACE, hence it is not harder than model checking. Finally, as extending vacuity to consider real-time leads to a high number of vacuity results, we observed that the framework for ranking of LTL vacuity results by their approximated importance can be applied to TCTL as well.

References

1. Alur, R., Courcoubetis, C., Dill, D.: Model-checking in dense real-time. Inf. Comput. **104**(1), 2–34 (1993)
2. Alur, R., Dill, D.: A theory of timed automata. Theoret. Comput. Sci. **126**(2), 183–236 (1994)
3. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. J. ACM **43**(1), 116–146 (1996)
4. Alur, R., Henzinger, T.A.: Logics and models of real time: a survey. In: de Bakker, J.W., Huizing, C., de Roever, W.P., Rozenberg, G. (eds.) REX 1991. LNCS, vol. 600, pp. 74–106. Springer, Heidelberg (1992). https://doi.org/10.1007/BFb0031988
5. Armoni, R., et al.: Enhanced vacuity detection in linear temporal logic. In: Hunt, W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 368–380. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45069-6_35
6. Beer, I., Ben-David, S., Eisner, C., Rodeh, Y.: Efficient detection of vacuity in ACTL formulas. Formal Methods Syst. Des. **18**(2), 141–162 (2001)
7. Bustan, D., Flaisher, A., Grumberg, O., Kupferman, O., Vardi, M.Y.: Regular vacuity. In: Borrione, D., Paul, W. (eds.) CHARME 2005. LNCS, vol. 3725, pp. 191–206. Springer, Heidelberg (2005). https://doi.org/10.1007/11560548_16
8. Chechik, M., Gheorghiu, M., Gurfinkel, A.: Finding environment guarantees. In: Dwyer, M.B., Lopes, A. (eds.) FASE 2007. LNCS, vol. 4422, pp. 352–367. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71289-3_27

9. Chockler, H., Gurfinkel, A., Strichman, O.: Beyond vacuity: towards the strongest passing formula. In: Proceedings of the 8th International Conference on Formal Methods in Computer-Aided Design, pp. 1–8 (2008)

10. Chockler, H., Halpern, J.Y.: Responsibility and blame: a structural-model approach. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence, pp. 147–153 (2003)

11. Chockler, H., Strichman, O.: Before and after vacuity. Formal Methods Syst. Des. **34**(1), 37–58 (2009)

12. Clarke, E., Grumberg, O., Long, D.: Verification tools for finite-state concurrent systems. In: de Bakker, J.W., de Roever, W.-P., Rozenberg, G. (eds.) REX 1993. LNCS, vol. 803, pp. 124–175. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58043-3_19

13. Clarke, E.M., Grumberg, O., McMillan, K.L., Zhao, X.: Efficient generation of counterexamples and witnesses in symbolic model checking. In: Proceedings of the 32st Design Automation Conference, pp. 427–432. IEEE Computer Society (1995)

14. Courcoubetis, C., Yannakakis, M.: The complexity of probabilistic verification. J. ACM **42**, 857–907 (1995)

15. Ben-David, S., Kupferman, O.: A framework for ranking vacuity results. In: Van Hung, D., Ogawa, M. (eds.) ATVA 2013. LNCS, vol. 8172, pp. 148–162. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-02444-8_12

16. Dokhanchi, A., Hoxha, B., Fainekos, G.E.: Formal requirement elicitation and debugging for testing and verification of cyber-physical systems. CoRR, abs/1607.02549 (2016)

17. Emerson, E.A., Halpern, J.Y.: Sometimes and not never revisited: on branching versus linear time. J. ACM **33**(1), 151–178 (1986)

18. Fisman, D., Kupferman, O., Sheinvald-Faragy, S., Vardi, M.Y.: A framework for inherent vacuity. In: Chockler, H., Hu, A.J. (eds.) HVC 2008. LNCS, vol. 5394, pp. 7–22. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01702-5_7

19. Gurfinkel, A., Chechik, M.: Extending extended vacuity. In: Hu, A.J., Martin, A.K. (eds.) FMCAD 2004. LNCS, vol. 3312, pp. 306–321. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30494-4_22

20. Gurfinkel, A., Chechik, M.: How vacuous is vacuous? In: Jensen, K., Podelski, A. (eds.) TACAS 2004. LNCS, vol. 2988, pp. 451–466. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24730-2_34

21. Henzinger, T.A., Manna, Z., Pnueli, A.: Timed transition systems. In: de Bakker, J.W., Huizing, C., de Roever, W.P., Rozenberg, G. (eds.) REX 1991. LNCS, vol. 600, pp. 226–251. Springer, Heidelberg (1992). https://doi.org/10.1007/BFb0031995

22. Kupferman, O.: Sanity checks in formal verification. In: Baier, C., Hermanns, H. (eds.) CONCUR 2006. LNCS, vol. 4137, pp. 37–51. Springer, Heidelberg (2006). https://doi.org/10.1007/11817949_3

23. Kupferman, O., Li, W., Seshia, S.A.: A theory of mutations with applications to vacuity, coverage, and fault tolerance. In: Proceedings of the 8th International Conference on Formal Methods in Computer-Aided Design, pp. 1–9 (2008)

24. Kupferman, O., Vardi, M.Y.: Vacuity detection in temporal model checking. Softw. Tools Technol. Transf. **4**(2), 224–233 (2003)

25. Namjoshi, K.S.: An efficiently checkable, proof-based formulation of vacuity in model checking. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 57–69. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27813-9_5

26. Purandare, M., Somenzi, F.: Vacuum cleaning CTL formulae. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 485–499. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45657-0_39
27. Purandare, M., Wahl, T., Kroening, D.: Strengthening properties using abstraction refinement. In: Proceedings of Design, Automation and Test in Europe (DATE), pp. 1692–1697. IEEE (2009)
28. Stockmeyer, L.J.: On the combinational complexity of certain symmetric boolean functions. Math. Syst. Theory **10**, 323–336 (1977)