Deciding Safety and Liveness in TPTL

David Basin\textsuperscript{a}, Carlos Cotrini Jiménez\textsuperscript{a,}\textsuperscript{∗}, Felix Klaedtke\textsuperscript{b,1}, Eugen Zălinescu\textsuperscript{a}

\textsuperscript{a}Institute of Information Security, ETH Zurich, Switzerland

\textsuperscript{b}NEC Europe Ltd., Heidelberg, Germany

Abstract

We show that deciding whether a TPTL formula describes a safety property is EXPSPACE-complete. Moreover, deciding whether a TPTL formula describes a liveness property is in 2-EXPSPACE. Our algorithms for deciding these problems extend those presented by Sistla\textsuperscript{[1]} to decide the corresponding problems for LTL.

Keywords: temporal logic, safety and liveness, verification, complexity

1. Introduction

Safety and liveness\textsuperscript{[2, 3]} are two important classes of system properties. A safety property claims that something “bad” never happens and a liveness property claims that something “good” can eventually happen. Identifying a system property as a safety or liveness property helps in finding a suitable method for its verification. For example, model checking can be improved when the system specification is known to be a safety property\textsuperscript{[4]}. Also, when a property is known to be a safety property\textsuperscript{[4]}, checking can be improved when the system specification is a suitable method for its verification. For example, model checking can be improved when the system specification is a suitable method for its verification.

Sistla\textsuperscript{[1]} proved that deciding whether an LTL formula describes a safety property is PSPACE-complete and that for liveness properties the problem is in EXPSPACE. However, analogous results for TPTL have not, until now, been given. In this article, we build upon Sistla’s ideas to decide the corresponding problems for TPTL. We prove that deciding whether a TPTL formula describes a safety property is EXPSPACE-complete and that for liveness properties the problem is in 2-EXPSPACE. To the best of our knowledge, establishing tight lower bounds for deciding liveness in TPTL and LTL are open problems.

The remainder of this article is organized as follows. In Section 2 we give background and, in particular, we recall TPTL’s syntax and semantics. In Section 3 we introduce quasimodels and quasicounterexamples for TPTL. These notions, suitably adapted from\textsuperscript{[5, 6]}, facilitate the proof of correctness of our decision algorithms. In Sections 4 and 5, we prove our complexity results and in Section 6 we draw conclusions.

2. Preliminaries

An infinite sequence over a set \(S\) is a function from \(\mathbb{N}\) to \(\{0,1,\ldots,\ell-1\}\) to \(S\). For a finite sequence \(\alpha\) and a sequence \(\beta\), let \(\alpha\beta\) denote their concatenation and let \(|\alpha|\) denote \(\alpha\)’s length. The prefix of length \(i\) in \(\mathbb{N}\) of a sequence \(\alpha\) is the sequence \(\alpha_{\leq i} := \alpha(0)\alpha(1)\ldots\alpha(i-1)\), where we assume that \(|\alpha| > i\). The sequences \(\alpha_{< i}, \alpha_{> i}, \) and \(\alpha_{\geq i}\) are defined similarly. For a finite nonempty sequence \(\alpha\), let \(\alpha^\omega\) be the infinite sequence \(\alpha\alpha\ldots\). For a sequence \(\alpha \in \mathbb{N}\), let \(\bar{\alpha}\) be the sequence defined by \(\bar{\alpha}(i) := \sum_{0 \leq k \leq i} \alpha(k)\) and \(\bar{\alpha}(i,j) := \sum_{i \leq k \leq j} \alpha(k)\), for \(i,j \in \mathbb{N}\) with \(i \leq j\).

2.1. TPTL

Syntax. Let \(P\) be a finite set of atomic propositions and \(V\) a countable set of variables, with \(V \cap P = \emptyset\). The terms \(\pi\) and formulas \(\varphi\) of TPTL are defined by the grammar

\[
\pi ::= x + c \mid c \\
\varphi ::= \text{false} \mid p \mid \pi_1 \leq \pi_2 \mid \pi_1 \equiv_m \pi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid \circ \varphi \mid \varphi_1 \cup \varphi_2 \mid x.\varphi,
\]

where \(x, c, p, m\) and \(\pi\) range over \(V, \mathbb{N}, P, \) and \(\mathbb{N} \setminus \{0\}\), respectively. We abbreviate \(x + 0\) by \(x\). For a formula \(\varphi\), we write \(\neg \varphi\) for \(\varphi \rightarrow \text{false}\) and true for \(\neg \text{false}\). The syntactic sugar for the Boolean connectives \(\land\) and \(\lor\) is as expected. We let \(\Diamond \psi := \text{true} \psi\) and \(\Box \psi := \neg \Diamond \neg \psi\). All occurrences of a variable \(x\) in a formula of the form \(x.\psi\) are said to be \emph{bound} by \(x.\psi\). An occurrence of \(x\) in \(\varphi\) that is not bound by any subformula \(x.\psi\) of \(\varphi\) is \emph{free}. We denote with

\textsuperscript{∗}Corresponding author.

\textsuperscript{1}This work was partly done when the author was at ETH Zurich.

\textsuperscript{Preprint submitted to Elsevier}
\( \varphi[x \mapsto z] \) the formula obtained from \( \varphi \) by replacing all free occurrences of \( x \in V \) with \( z \in V \). Finally, we write \( \pi_1 \sim \pi_2 \) to denote any formula of the form \( \pi_1 \leq \pi_2 \) or \( \pi_1 \equiv_m \pi_2 \), with \( \pi_1 \) and \( \pi_2 \) terms and \( m \geq 1 \). We call \( \pi_1 \sim \pi_2 \) a time constraint.

Let \( n_\varphi \) be the number of connectives in \( \varphi \). Also, let

\[
k_\varphi := 2 \cdot (\prod_i (1 + c)) \cdot (\prod_m m),
\]

where \( c \) ranges over the constants occurring in formulas of the form \( \pi_1 \leq \pi_2 \) in \( \varphi \), with \( \pi_1 \) and \( \pi_2 \) terms, and \( m \) ranges over the constants such that \( \equiv_m \) occurs in \( \varphi \). When there are no constants in \( \varphi \), we define \( k_\varphi := 2 \). We define the length of a formula as the number of symbols needed to write the formula, assuming that a binary encoding is used to represent constants and to enumerate variables. Note that the length of a formula \( \varphi \) is linear in \( n_\varphi \log n_\varphi + \log k_\varphi \).

**Semantics.** Let \( \Sigma = 2^P \times \mathbb{N} \) and let \( \Sigma^* \) and \( \Sigma^\omega \) be the sets of all finite and infinite sequences over \( \Sigma \) respectively. We usually write a sequence

\[
(\sigma(0), \delta(0)) (\sigma(1), \delta(1)) \ldots \in \Sigma^* \cup \Sigma^\omega
\]
as \( \sigma \circ \delta \), where \( \sigma \) and \( \delta \) are sequences over \( 2^P \) and \( \mathbb{N} \), respectively. TPTL formulas are interpreted over timed words. A timed word is an infinite sequence \( \sigma \circ \delta \in \Sigma^\omega \) such that \( \delta(i) > 0 \), for infinitely many \( i \). A timed word \( \sigma \circ \delta \) is \( k \)-bounded, for \( k \in \mathbb{N} \), if \( \delta(i) \leq k \), for all \( i \in \mathbb{N} \).

Note that Alur and Henzinger [8] define timed words differently. There, a timed word is an infinite sequence \( \sigma \circ \delta \in \Sigma^\omega \) such that \( \tau \) is non-decreasing and for all \( i \in \mathbb{N} \), there is \( j > i \) such that \( \tau(j) > \tau(i) \). However, the sets of timed words of both definitions are essentially the same: We can map a timed word \( \sigma \circ \delta \) under our definition to the timed word \( \sigma \circ \delta \) under their definition. Intuitively, for \( i > \delta(i) \), \( \delta(i) \) indicates the time elapsed between the events \( \sigma(i - 1) \) and \( \sigma(i) \) and \( \delta(i) \) indicates the time when the event \( \sigma(i) \) takes place.

A valuation is a mapping from \( V \) to \( \mathbb{N} \). We extend valuations to terms in the usual way. For a timed word \( \sigma \circ \delta \), a formula \( \psi \), a valuation \( v \), and \( i \in \mathbb{N} \), we define satisfaction, written \( \sigma \circ \delta \circ v, i \models \psi \), by induction on the structure of \( \psi \).

\[
\begin{align*}
\sigma \circ \delta \circ v, i \not\models \varphi &\quad \text{false} \\
\sigma \circ \delta \circ v, i \models p &\quad \text{iff } p \in \sigma(i) \\
\sigma \circ \delta \circ v, i \models \pi_1 \leq \pi_2 &\quad \text{iff } v(\pi_1) \leq v(\pi_2) \\
\sigma \circ \delta \circ v, i \models \pi_1 \equiv_m \pi_2 &\quad \text{iff } v(\pi_1) \equiv_m v(\pi_2) \\
\sigma \circ \delta \circ v, i \not\models \psi_1 \rightarrow \psi_2 &\quad \text{iff } \sigma \circ \delta \circ v, i \not\models \psi_1 \text{ or } \sigma \circ \delta \circ v, i \models \psi_2 \\
\sigma \circ \delta \circ v, i \not\models \psi &\quad \text{iff } \sigma \circ \delta \circ v, i + 1 \not\models \psi \\
\sigma \circ \delta \circ v, i \not\models \psi_1 \cup \psi_2 &\quad \text{iff there is } j \geq i \text{ with } \sigma \circ \delta \circ v, j \models \psi_2 \text{ and } \sigma \circ \delta \circ v, k \models \psi_1, \text{ for all } k \text{ with } i \leq k < j \\
\sigma \circ \delta \circ v, i \models \varphi(x) &\quad \text{iff } \sigma \circ \delta \circ v[x \mapsto \delta(i)], i \models \psi
\end{align*}
\]

Here \( v[x \mapsto \delta(i)] \) is the valuation obtained from \( v \) by setting \( v(x) \) to \( \delta(i) \). We say that \( \sigma \circ \delta \) satisfies a sentence \( \varphi \) (i.e., a formula without free variables) if \( \sigma \circ \delta, v, 0 \models \varphi \), for any valuation \( v \).

### 2.2. Safety and liveness

A timed word \( \tau \) refutes the safety of a sentence \( \varphi \) if \( \tau \) does not satisfy \( \varphi \) and for every \( i \in \mathbb{N} \), there is a sequence \( \tau' \in \Sigma^* \) such that \( \tau' \leq \tau \) satisfies \( \varphi \). The sentence \( \varphi \) is safe—or describes a safety property—if there is no timed word refuting \( \varphi \)’s safety [2, 3].

A sequence \( \tau \) in \( \Sigma^* \) is a good prefix for \( \varphi \) if there is \( \tau' \in \Sigma^\omega \) such that \( \tau' \leq \tau \) is a timed word that satisfies \( \varphi \). The sentence \( \varphi \) describes a liveness property if every sequence in \( \Sigma^* \) is a good prefix for \( \varphi \) [2, 3].

### 2.3. Additional notions and machinery

**Time-constraint normal form.** Following [5], we show that we can restrict our attention to sentences of a certain form. Let \( \varphi \) be a sentence and a variable not occurring in \( \varphi \). The sentence \( \tilde{\varphi} \) is obtained from \( \varphi \) by replacing every variable-free term \( c \) with \( z + c \) and then performing the necessary arithmetic manipulations to leave any time constraint in the form \( z + c \approx y \) or \( z \approx y + c \) with \( x, y \in V \) and \( c \in \mathbb{N} \).

The following lemma follows from the observation that a timed word \( \sigma \circ \delta \) satisfies \( \varphi \) iff \( \exists \sigma \circ \delta \) satisfies \( \sigma \circ \tilde{\varphi} \).

**Lemma 1.** A sentence \( \varphi \) describes a safety property if \( \exists \sigma \circ \tilde{\varphi} \) does and \( \varphi \) describes a liveness property if \( \forall \sigma \circ \tilde{\varphi} \) does.

For the rest of the article, we assume without loss of generality that \( z.\varphi \) is a sentence where every time constraint in \( \varphi \) is of the form \( x + c \approx y \) or \( x \approx y + c \), with \( x, y \in V \) and \( c \in \mathbb{N} \).

**Updating time constraints.** A key observation underlying the algorithm for deciding satisfiability in TPTL presented by Alur and Henzinger [5] is that every formula can be split into a present and a future condition. Note that

\[
\sigma \circ \delta, v, i \models \Diamond q \iff \sigma \circ \delta, v, i \not\models q \text{ or } \sigma \circ \delta, v, i + 1 \not\models \Diamond \neg q
\]

One must be careful when time constraints occur in the formula. For example, consider the expression \( \sigma \circ \delta, v, i \models z.\neg \psi \not\models z.\psi \lim (y \leq z + 5 \land q) \). Note that \( \neg \psi \) refers to the current time. This expression can be satisfied by having \( \sigma \circ \delta, v, i \not\models z.\neg \psi \lim (z \leq z + 5 \land q) \) in the current state or \( \sigma \circ \delta, v, i + 1 \not\models z.\neg \psi \lim (y \leq (z - \delta(i + 1)) + 5 \land q) \) in the next state. Note that we updated the time constraint as the current time has changed by \( \delta(i + 1) \).

We recall some notation from [5] for updating time constraints. For a formula of the form \( z.\psi \) and \( d \in \mathbb{N} \), let \( z.\psi^d \) be the formula obtained by replacing every occurrence of \( z \) in \( \psi \) with \( z - d \). Formally, \( z.\psi^d \) is defined inductively as follows.

\[
\begin{align*}
- z.\psi^0 &\equiv z.\psi \\
- z.\psi^{d+1} &\equiv z.\psi^d \text{ by replacing every term of the form } z + (c + 1) \text{ with } z + c, \text{ and every subformula of the form } z \leq y + c, y + c \leq z, \text{ and } z \equiv_m y + c \text{ with true, false, and } z \equiv_m y + ((c + 1) \mod m), \text{ respectively.}
\end{align*}
\]
For example, let \( z \cdot \varphi = z \cdot \Diamond y. (y \leq z + 5 \land q) \). Then \( z \varphi^2, z \varphi^3 \) and \( z \varphi \) are \( z \cdot \Diamond y. (y \leq z + 3 \land q), z \cdot \Diamond y. (y \leq z \land q) \), and \( z \cdot \Diamond (false \land q) \), respectively.

The next lemma, from \([3]\), shows that \( z \varphi \) correctly denotes the formula \( z \cdot \varphi \) after replacing every free occurrence of \( z \) with \( d \). It is proved by induction on \( \psi \)'s structure.

**Lemma 2.** For every formula \( z \cdot \varphi \) and every \( d \leq \delta(i) \), we have \( \sigma \otimes \delta, v, i \models z \varphi \) iff \( \sigma \otimes \delta, v[z \mapsto d], i \models \varphi \).

**The closure of a formula.** The algorithm of Alur and Henzinger follows the tableau method. A tableau for a formula \( z \varphi \) is built from a set \( Cl(z \varphi) \) of sentences called the **closure** of \( z \varphi \) \([3]\). The closure of \( z \varphi \) is the smallest set that contains \( z \varphi \) and is closed under the operation \( Sub \), which is defined as:

- \( Sub(z \cdot \varphi) := \{ z \cdot \varphi \} \), if \( \psi \) is an atomic formula,
- \( Sub(z (\psi_1 \to \psi_2)) := \{ z \cdot \psi_1, z \cdot \psi_2 \} \),
- \( Sub(z \cdot \varphi := \{ z \cdot \varphi \mid d \in N \} \),
- \( Sub(z \cdot \psi_1 U \varphi_2) := \{ z \cdot \psi_1, z \cdot \psi_2, z \cdot \psi_1 \varphi_2 \} \), and
- \( Sub(z \cdot x) := \{ z \cdot \varphi[x \mapsto \alpha] \} \).

For example, for \( z \cdot \varphi = z \cdot (p U y. (y \leq z + 5)) \), \( Cl(z \cdot \varphi) \) contains: \( z \cdot \varphi \), \( z \cdot true, z \cdot false \), \( z \cdot (p U y \cdot false) \), \( z \cdot (p U y \cdot false) \) \( z \cdot (\leq z + 5) \), \( z \cdot (\leq z + 5) \), \( z \cdot (\leq z + i) \), \( z \cdot (\leq z + i) \), \( z \cdot (\leq z + i) \), and \( z \cdot (\leq z + i) \), for \( i \leq 5 \).

Note that for any \( z \cdot \varphi \), \( Cl(z \cdot \varphi) \) only contains sentences. In particular, \( z \cdot \varphi \) is the only variable that occurs in any formula of the form \( z \cdot (\pi_1 \sim \pi_2) \in Cl(z \cdot \varphi) \).

**Avoiding valuations.** The following lemma shows that when evaluating a sentence \( z \cdot \varphi \) in \( Cl(z \cdot \varphi) \) at a position in a timed word, we only need not consider valuations.

**Lemma 3.** Let \( z \cdot \varphi \) be a sentence in \( Cl(z \cdot \varphi) \). For a timed word \( \sigma \otimes \delta, v \) and \( i \in N \), we have the following according to the form of \( z \cdot \varphi \):

1. \( \sigma \otimes \delta, v, i \not\models z \cdot \varphi \) false.
2. \( \sigma \otimes \delta, v, i \not\models z \cdot p \) if \( p \in \sigma(i) \), for \( p \in P \).
3. \( \sigma \otimes \delta, v, i \models z \cdot z \sim z + c \) if \( 0 \sim c \) and \( \sigma \otimes \delta, v, i \not\models z \cdot z + c \sim z \) if \( c \sim 0 \), for \( i \in N \).
4. \( \sigma \otimes \delta, v, i \models z \cdot (\psi_1 \to \psi_2) \) if \( \sigma \otimes \delta, v, i \not\models z \cdot \psi_1 \) or \( \sigma \otimes \delta, v, i \models z \cdot \psi_2 \).
5. \( \sigma \otimes \delta, v, i \not\models z \cdot \psi \) if \( \sigma \otimes \delta, v, i \not\models z \cdot \psi \).
6. \( \sigma \otimes \delta, v, i \models z \cdot (\psi_1 U \psi_2) \) if \( (a) \sigma \otimes \delta, v, i \not\models z \cdot \psi_2 \) or \( (b) \sigma \otimes \delta, v, i \models z \cdot \psi_1 \) and \( \sigma \otimes \delta, v, i \models z \cdot \psi_1 U \psi_2 \).
7. \( \sigma \otimes \delta, v, i \not\models z \cdot x \cdot \psi \) if \( \sigma \otimes \delta, v, i \not\models z \cdot \psi \).
- $y + c \leq z$ or $y \leq z + c$. Here both formulas become false.
- $z + c \equiv_m y$ or $z \equiv_m y + c$. The two cases are similar, so we consider only the second one. Here $\psi^d$ equals
  \[
  z \equiv_m y + ((d + c) \mod m)
  \]
  and $\psi^d$ is
  \[
  z \equiv_m y + ((c \cdot m \cdot \varphi + (d \mod m \cdot \varphi) + c) \mod m).
  \]
  If we simplify the last expression, we obtain
  \[
  (c \cdot m \cdot \varphi + (d \mod m \cdot \varphi) + c) \mod m = (d + c) \mod m,
  \]
  where the last equality follows from $(d \mod m \cdot \varphi) \mod m = d \mod m$.

**Lemma 6.** Let $d_i \in \mathbb{N}$, for $1 \leq i \leq k$. Let $\Delta_k$ and $\Delta_k$ be $d_1 + d_2 + \ldots + d_k$ and $d_1 + d_2 + \ldots + d_k$, respectively. Then $z.\varphi^{d_1} \equiv z.\varphi^{d_1}$, for any subformula $z.\varphi \in \mathcal{C}(z.\varphi)$.

**Proof.** By induction on $k$. Note that $z.\varphi^{d_1} \equiv z.\varphi^{d_1}$ can be obtained by first computing $z.\varphi^{d_1}$ and then computing from that $(z.\varphi^d)^{d_1}$, for any subformula $z.\varphi^d \in \mathcal{C}(z.\varphi^d)$.

**Lemma 7.** Let $\sigma \otimes \delta$ be a timed word and let $\hat{\delta}$ be the sequence defined by $\hat{\delta}(i) := \delta(i)$, for $i \in \mathbb{N}$. Then $\sigma \otimes \delta$ is a $k_\varphi$-bounded timed word that satisfies $z.\varphi$ iff $\sigma \otimes \delta$ satisfies $z.\varphi$.

**Proof.** Prove that $\sigma \otimes \hat{\delta}, 0, \varphi \equiv z.\varphi$ if $\sigma \otimes \delta, 0, \varphi \equiv z.\varphi$, for all $z.\varphi \in \mathcal{C}(z.\varphi)$. For this, use the well-founded induction schema presented in the proof of Lemma 3.

3. Quasimodels and quasicounterexamples

Our algorithm for deciding whether a TPTL sentence is safe is inspired by the algorithm presented in [1] for LTL, which, in turn, is based on an algorithm for deciding satisfiability in LTL [10]. We recall briefly how they work.

LTL models are infinite sequences over the alphabet $2^P$. A model is regular if it has the form $a \beta^\omega$, for some finite nonempty sets $a \cdot \varphi$ and $\beta$ over $2^P$. An LTL formula $\psi$ is satisfiable if there is a regular model that satisfies $\psi$. To decide whether an LTL formula $\psi$ is satisfiable, the algorithm non-deterministically guesses two finite sequences $f_1$ and $f_2$ of sets of subformulas of $\psi$. The formula $\psi$ is satisfiable if there is a regular model $\alpha \beta^\omega$ such that the sequences $f_1$ and $f_2$ satisfy the following: for $i < |f_1|$, the set $f_1(i)$ contains exactly all the subformulas of $\psi$ satisfied by $\alpha^i \beta^\omega$ and for $j < |f_2|$, the set $f_2(j)$ contains exactly all the subformulas of $\psi$ satisfied by $\beta^j \omega$. In particular, $f_1(i)$ contains $\alpha(i)$ and $f_2(j)$ contains $\beta(j)$, for all $i < |f_1|$ and $j < |f_2|$. The sequence $f_1.f_2$ provides all the information needed to build $\alpha$ and $\beta$. Moreover, it contains evidence that $\alpha \beta^\omega$ satisfies $\psi$. The sequence $f_1.f_2$ is called a quasimodel for $\psi$. In general, a quasimodel for an LTL formula $\psi$ is a sequence $f$ of sets of subformulas of $\psi$ for which there is a model $\gamma$ that satisfies $\psi$ and such that $f(i)$ contains all the subformulas of $\psi$ satisfied by $\gamma^i$. The elements of a quasimodel are called quasistates for $\psi$, which are maximal consistent sets of subformulas of $\psi$.

The algorithm for checking whether an LTL formula describes a safety property is similar but more involved. It non-deterministically guesses a representation of a quasicounterexample, which consist of quasimodels $f, g_0, g_1, \ldots$, witnessing that the formula $\varphi$ is not safe. In particular, $f$ is a quasimodel for $\neg \varphi$ and $f <^i g_i$ is a quasimodel for $\varphi$, for every $i \in \mathbb{N}$.

These observations carry over from LTL to TPTL, with some modifications. The algorithms for satisfiability and safety work in the same way and analogous regularity properties hold for TPTL. We adapt the notions of quasistate, quasimodel, and quasicounterexample for TPTL in the Sections 3.1.3.2 and 3.3, respectively. Quasistates and quasimodels were already adapted to TPTL in [8]—with different names though—and we recall them for the sake of completeness. Note that these notions are implicit in [10, 1] for LTL. Quasistates and quasimodels were introduced in [9] to simplify the correctness proofs for decision algorithms of some fragments of first-order temporal logic.

3.1. Quasistates

**Definition 1.** A quasistate for $z.\varphi$ is a pair $(\Phi, d)$, where $d \in \mathbb{N}$ and $\Phi$ is a maximally consistent subset of $\mathcal{C}(z.\varphi)$, that is, $\Phi$ must satisfy the following conditions.

- $z.\varphi \notin \Phi$.
- $z.(z \sim z + c) \in \Phi$ iff $0 \sim c$, for every $z.(z \sim z + c) \in \mathcal{C}(z.\varphi)$, and $z.(z + c \sim z) \in \Phi$ iff $c \sim 0$, for every $z.(z + c \sim z) \in \mathcal{C}(z.\varphi)$.
- $z.(\psi_1 \rightarrow \psi_2) \in \Phi$ iff $z.\psi_1 \notin \Phi$ or $z.\psi_2 \in \Phi$, for every $z.(\psi_1 \rightarrow \psi_2) \in \mathcal{C}(z.\varphi)$.
- $z.(\psi_1 \cup \psi_2) \in \Phi$ iff (i) $z.\psi_2 \notin \Phi$ or (ii) $z.\psi_1 \in \Phi$ and $z.(\psi_1 \cup \psi_2) \in \Phi$, for every $z.(\psi_1 \cup \psi_2) \in \mathcal{C}(z.\varphi)$.
- $z.\psi \in \Phi$ iff $z.\psi[x \mapsto z] \in \Phi$, for every $z.\psi \in \mathcal{C}(z.\varphi)$.

For $k \in \mathbb{N}$, we say a quasistate $(\Phi, d)$ is $k$-bounded if $d \leq k$ and we denote with $z.\varphi$ the number of $k_\varphi$-bounded quasistates for $z.\varphi$.

By Lemma 3 the set $\text{Sub}(z.\varphi)$ is finite, which implies that $\mathcal{C}(z.\varphi)$ is finite. In particular, the size of $\mathcal{C}(z.\varphi)$ is at most $n_{k_\varphi}$ [8]. Hence we have that

$$3(z.\varphi) \leq 2^{n_{k_\varphi}} \cdot k_\varphi < 2^{(n_{k_\varphi} + 1)k_\varphi}.$$
3.2. Quasimodels

Let $f$ be a sequence of quasimodels for $z \varphi$ with $f(i) = (\Phi_i, d_i)$, for $i \in \mathbb{N}$, and let $\delta$ be the sequence defined by $\delta(i) := d_i$, for $i \in \mathbb{N}$. Recall that $\delta(i, j) := \sum_{i < k \leq j} \delta(k)$. Suppose that $z.(\psi_1 \cup \psi_2)$ occurs in $\Phi_i$. Then we say that $f$ realizes the occurrence of $z.(\psi_1 \cup \psi_2)$ in $\Phi_i$, if there is $j \geq i$ such that $z.(\delta(i, j))$ occurs in $\Phi_i$. When the set $\Phi_i$ is clear from the context, we say instead that $f$ realizes $z.(\psi_1 \cup \psi_2)$.

For $(\Phi', d)$ and $(\Phi', d')$ two quasimodels for $z \varphi$, we say that $(\Phi', d')$ is a successor of $(\Phi, d)$ if, for any $z \circ \psi \in Cl(z, \varphi)$, it holds that $z \circ \psi \in \Phi$ if and only if $z \circ \psi \in \Phi'$.

Definition 2. A quasimodel for $z \varphi$ is an infinite sequence $f$ of quasimodels for $z \varphi$ with $f(i) = (\Phi_i, d_i)$ such that:

(QM-1) $d_i > 0$, for infinitely many $i$,

(QM-2) $z \varphi \in f(0)$,

(QM-3) $f(i + 1)$ is a successor of $f(i)$, for all $i \in \mathbb{N}$, and

(QM-4) any occurrence of the form $z.(\psi_1 \cup \psi_2)$ in $f$ is realized by $f$.

The quasimodel is $k$-bounded if $d_i \leq k$, for all $i \in \mathbb{N}$.

The proofs of the following two results are simple extensions of those presented in [11]. They show a one-to-one correspondence between timed words satisfying $z \varphi$ and quasimodels for $z \varphi$.

Theorem 1.

1. Let $\sigma \otimes \delta$ be a timed word that satisfies $z \varphi$ and let $f_{\sigma \otimes \delta}$ be the sequence defined by $f_{\sigma \otimes \delta}(i) := (\Phi_i, \delta(i))$ with $\Phi_i := \{z \psi \in Cl(z, \varphi) \mid \sigma \otimes \delta, i \models z \psi\}$.

The sequence $f_{\sigma \otimes \delta}$ is a quasimodel for $z \varphi$.

2. Let $f$ be a quasimodel for $z \varphi$ with $f(i) = (\Phi_i, d_i)$ and let $\sigma_f \otimes \delta_f$ be the pair of sequences defined by $\sigma_f(i) := \{p \in P \mid z.p \in \Phi_i\}$ and $\delta_f(i) := d_i$, for any $i \in \mathbb{N}$. The pair $\sigma_f \otimes \delta_f$ is a timed word that satisfies $z \varphi$.

Recall that for $d \in \mathbb{N}$, $d$ is defined as $c_{\varphi} \cdot m_{\varphi} + (d \mod m_{\varphi})$ if $d \geq c_{\varphi} \cdot m_{\varphi}$ and $d = d$, otherwise.

Theorem 2. Let $f$ be an infinite sequence of quasimodels for $z \varphi$ with $f(i) = (\Phi_i, d_i)$ and let $\hat{f}$ be the infinite sequence defined as $\hat{f}(i) = (\Phi_i, d_i)$. Then $f$ is a quasimodel for $z \varphi$ if and only if $\hat{f}$ is a $k_{z \varphi}$-bounded quasimodel for $z \varphi$.

Proof. We prove just the “only if” direction. The “if” direction is proved similarly. Requirements (QM-1) and (QM-2) are clear. For (QM-3) and (QM-4), use Lemmas 5 and 6. Finally, recall that $d_i < k_{z \varphi}$. Hence $\hat{f}$ is a $k_{z \varphi}$-bounded quasimodel for $z \varphi$.

Lemma 8. Let $f$ be a quasimodel for $z \varphi$. If there are $i, j \in \mathbb{N}$ such that $i \leq j$ and $f(i) = f(j)$, then $f' = f^S f'^S$ is also a quasimodel for $z \varphi$.

Proof. We adapt the proof in [3] to TPTL. Let $f(i) = (\Phi_i, d_i)$, for $i \in \mathbb{N}$ and let $\delta$ be the sequence defined by $\delta(i) := d_i$, for all $i \in \mathbb{N}$. (QM-1) and (QM-2) clearly hold for $f'$. To check (QM-3), note that $z \circ \psi \in f(i)$ iff $z \circ \psi \in f(j)$ iff $z.(\delta(i)+1) \in f(j+1)$. We check (QM-4) as follows. Let $z.(\psi_1 \cup \psi_2) \in f(m)$ for some $m$. If $m > j$ then clearly $z.(\psi_1 \cup \psi_2)$ is realized by $f'$. Suppose then $m \leq i$. If $z.(\psi_1 \cup \psi_2) \in f(\ell)$ for some $\ell \leq i$, then we are done; otherwise, $z.(\psi_1 \cup \psi_2) \in f(m)$ must occur in $f(i)$. It follows that $z.(\psi_1 \cup \psi_2) \in f(j)$, and since $f$ is a quasimodel for $z \varphi$, the occurrence of $z.(\psi_1 \cup \psi_2)$ in $f(j)$ is realized by $f^S f'^S$. Hence $z.(\psi_1 \cup \psi_2)$ is realized by $f'$.

To decide whether there is a quasimodel for $z \varphi$, the following lemma from [8] shows that we only need to find two particular finite sequences of quasimodels.

Lemma 9. There is a quasimodel for $z \varphi$ iff there are sequences $f_1$ and $f_2$ of $k_{z \varphi}$-bounded quasimodels for $z \varphi$ such that:

1. $|f_1| \leq \tau(z, \varphi)$ and $|f_2| \leq (|Cl(z, \varphi)| + 2) \cdot \tau(z, \varphi)$,

2. $z \varphi \in f_1(0)$, $z \varphi \in f_2(0)$,

3. $d > 0$ for some $(\Phi, d)$ in $f_2$,

4. $f_j(i + 1)$ is a successor of $f_j(i)$ for $i < |f_j| - 1$ and $j \in \{1, 2\}$,

5. $f_2(0)$ is a successor of the last quasimodels of $f_1$ and $f_2$, and

6. every occurrence of a formula of the form $z.(\psi_1 \cup \psi_2)$ in $f_2(0)$ is realized by $f_2$.

Proof. The proof of an analogous lemma in [9] applies here as well. For the “if” direction, note that $f_1 f_2$ is a quasimodel for $z \varphi$. We prove the “only if” direction, where we assume that $f$ is a quasimodel for $z \varphi$ with $f(i) = (\Phi_i, d_i)$. By Theorem 2 we assume $f$ is $k_{z \varphi}$-bounded. Take $s$ such that $f(s) = f(i)$, for infinitely many $i > s$. Apply Lemma 8 whenever $i_1 < i_2 < s$ and $f(i_1) = f(i_2)$. This yields a quasimodel $f_1 f_2^S$ with $|f_1| \leq \tau(z, \varphi)$.

We now explain how to get $f_2$. Suppose there is a formula of the form $z.(\psi_1 \cup \psi_2)$ in $f_2^S(0)$. Take $k \geq 0$ such that $z.(\psi_2^{(s+k)}) \notin f_2^S(k)$, where $\delta$ is the sequence defined by $\delta(i) := d_i$, for $i \in \mathbb{N}$. Apply Lemma 8 whenever $i_1 < i_2 < k$ and $f_2^S(i_1) = f_2^S(i_2)$. This yields the quasimodel $f_1 f_2^S(0) f_2^S f'^S$, where $s' := s + k$. Note that $f_2^S(0) f'$ has length at most $\tau(z, \varphi)$ and realizes the occurrence of $z.(\psi_1 \cup \psi_2)$ in $f_2(0)$. Suppose there is another formula in $f_2^S(0)$ of the form $z.(\psi'_1 \cup \psi'_2)$. If $f'$ realizes $z.(\psi'_1 \cup \psi'_2)$ then do nothing; otherwise, take $k'$ such that
A quasicounterexample is a model that satisfies some requirements. For TPTL, we define quasicounterexamples as follows. Let \( z.\varphi \) be a TPTL sentence. A model \( M \) is a quasicounterexample for \( z.\varphi \) if for every \( t \), there is a timed word \( w_t \) such that

\[
M \models z.\varphi(w_t).
\]

To construct a quasicounterexample, we proceed inductively. Suppose \( z.\varphi \) is not safe. Then there is a timed word \( \tau \) satisfying \( z.\varphi \) such that for any \( j \), the prefix \( \tau \upharpoonright j \) can be extended to a timed word \( \tau_j \) satisfying \( z.\varphi \). For \( j \in \mathbb{N} \), let \( M_f \) and \( M_h \) be the quasicountermodels for \( z.\varphi \) and \( z.\varphi \), respectively, defined by \( \tau \) and \( \tau_j \) according to Theorem 2, respectively. By Theorem 2, assume \( f \) and \( h \) are \( k \)-bounded, for \( j \in \mathbb{N} \).

Let \( H = \{ h_0, h_1, \ldots \} \). We may regard \( H \) as a set of infinite words from the alphabet consisting of all \( k \)-bounded quasistates for \( z.\varphi \), which is finite. We build inductively a sequence \( \alpha \) of quasistates as follows. Let \( \alpha(0) \) be the first quasistate obtained by concatenating both grids \( M_f \) and \( M_h \). For the inductive step, suppose we have already built the first \( i+1 \) quasistates \( \alpha^i \). We define \( \alpha^{i+1} \) as follows. Let \( \alpha^i(\varphi) = f(\varphi) \) with \( f \) and \( g \) as above. We define \( \alpha^{i+1}(\varphi) = f(\varphi) \) with \( f \) and \( g \) as above.

3.3. Quasicounterexamples

We now define quasicounterexamples for TPTL. Suppose \( z.\varphi \) is not safe. Then there is a timed word \( \tau \) that does not satisfy \( z.\varphi \) and such that for every \( i \in \mathbb{N} \), the sequence \( \tau^i \) can be extended to a timed word \( \tau_j \) that satisfies \( z.\varphi \). We cannot build such a family \( \tau \) in a finite set of timed words when \( z.\varphi \) is not safe.

Two quasistates \( (\varphi_1, d_1) \) and \( (\varphi_2, d_2) \) are compatible if \( d_1 = d_2 \) and \( z.\varphi \in \varphi_1 \) if \( z.\varphi \in \varphi_2 \) for every \( p \in P \). Two sequences of quasistates are compatible if they are element-wise compatible.

We now give the definition of quasicounterexample.

**Definition 3.** Let \( f \) be an infinite sequence of quasistates and \( g \) a mapping from \( \mathbb{N} \times \mathbb{N} \) to quasistates. The pair \( (f, g) \) is a quasicounterexample for \( z.\varphi \) if

- \( f \) is a quasimodel for \( z.\varphi \),
- \( g(i) \) is a quasimodel for \( z.\varphi \) for all \( i \in \mathbb{N} \), where \( g(i) \) is the sequence \( g(0,0), g(1,0), \ldots, g(i-1,1), g(i,1), g(i,2), \ldots \), and
- \( f \) and \( g(0,0), g(1,0) \) are compatible.

A quasicounterexample is \( k \)-bounded if \( f \) and \( g(i) \), for all \( i \in \mathbb{N} \), are all \( k \)-bounded.

**Lemma 10.** The sentence \( z.\varphi \) is not safe if it has a \( k \)-bounded quasicounterexample.

**Proof.** \((\Leftarrow)\) Let \( (f, g) \) be a \( k \)-bounded quasicounterexample for \( z.\varphi \). According to Theorem 1, let \( \sigma_1 \otimes \delta_1 \) and \( \sigma_{i,j} \otimes \delta_{i,j} \) be the timed words defined by \( f \) and \( g(i,j) \), for each \( j \in \mathbb{N} \), respectively. Note that \( \sigma_f \otimes \delta_f \) satisfies \( z.\varphi \) and \( \sigma_{i,j} \otimes \delta_{i,j} \) satisfies \( z.\varphi \) for all \( j \). Now, since \( f \) and \( g(0,0), g(1,0), \ldots \) are compatible, the finite prefix of \( \sigma_f \otimes \delta_f \) of length \( t \in \mathbb{N} \) is the same finite prefix of \( \sigma_{i,j} \otimes \delta_{i,j} \) of length \( t \). Hence, every finite prefix of \( \sigma_f \otimes \delta_f \) can be extended to a timed word that satisfies \( z.\varphi \). Therefore, \( z.\varphi \) is not safe.

\((\Rightarrow)\) Suppose \( z.\varphi \) is not safe. Then there is a timed word \( \tau \) satisfying \( z.\varphi \) such that for any \( j \) in \( \mathbb{N} \) the prefix \( \tau^j \) can be extended to a timed word \( \tau_j \) satisfying \( z.\varphi \). For \( j \in \mathbb{N} \), let \( f \) and \( h \) be the quasicountermodels for \( z.\varphi \) and \( z.\varphi \), respectively, defined by \( \tau \) and \( \tau_j \) according to Theorem 2, respectively. By Theorem 2, assume \( f \) and \( h \) are \( k \)-bounded, for \( j \in \mathbb{N} \).

Let \( H = \{ h_0, h_1, \ldots \} \). We may regard \( H \) as a set of infinite words from the alphabet consisting of all \( k \)-bounded quasistates for \( z.\varphi \), which is finite. We build inductively a sequence \( \alpha \) of quasistates as follows. Let \( \alpha(0) \) be the first quasistate obtained by concatenating both grids \( M_f \) and \( M_h \). For the inductive step, suppose we have already built the first \( i+1 \) quasistates \( \alpha^i \). We define \( \alpha^{i+1} \) as follows. Let \( \alpha^i(\varphi) = f(\varphi) \) with \( f \) and \( g \) as above. We define \( \alpha^{i+1}(\varphi) = f(\varphi) \) with \( f \) and \( g \) as above.

For a function \( g \) mapping \( \mathbb{N} \times \mathbb{N} \) into quasistates, we define \( g^\leq \) as the restriction of \( g \) over \( \{0, 1, \ldots, i\} \times \mathbb{N} \). Other functions such as \( g^{<}, g^{\geq}, g^{>} \) are defined analogously.

Suppose \( g_1 \) and \( g_2 \) are mappings from \( \{0, 1, \ldots, k\} \times \mathbb{N} \) and \( \mathbb{N} \times \mathbb{N} \) into quasistates respectively. Let \( g_1, g_2 \) be the mapping obtained by concatenating both grids \( \{0, 1, \ldots, k\} \times \mathbb{N} \) and \( \mathbb{N} \times \mathbb{N} \) along the first dimension.

**Lemma 11.** Let \( (f, g) \) be a quasicounterexample for \( z.\varphi \) such that \( f(i) = f(j) \) and \( g(i,0) = g(j,0) \) for some \( i < j \). Then \( (f^{<\leq} f^{\geq}, g^{<\leq} g^{\geq}) \) is a quasicounterexample for \( z.\varphi \).

**Proof.** Let \( g' = g^{<\leq} g^{\geq} \). Note \( f^{<\leq} f^{\geq} \) is a quasimodel for \( z.\varphi \) and each \( g' \) is a quasimodel for \( z.\varphi \), by Lemma 8. Clearly, \( f^{<\leq} f^{\geq} \) and \( g'(0,0)g'(1,0) \) are compatible.

**4. Deciding safety in TPTL**

The following theorem gives a computable criterion for deciding whether a TPTL sentence is safe. This theorem
naturally extends the criterion for deciding whether an LTL formula is safe \([1]\).

**Theorem 3.** The sentence \(z \varphi\) is not safe iff there are finite sequences \(f_1, f_2, h_1, h_2, h_3, h_4\) meeting the following requirements.

1. \(f_1 f_2^2\) is a quasimodel for \(z \neg \varphi\) and \(h_1 h_2 h_3 h_4^2\) is a quasimodel for \(z \varphi\).
2. \(|f_1| = |h_1| \leq 2(z.\varphi)^2\), \(|f_2| = |h_2| \leq (|Cz(\varphi)| + 2) \cdot 2(z.\varphi)^2\), \(|h_3| \leq 2(z.\varphi)\), and \(|h_4| \leq (|Cz(\varphi)| + 2) \cdot 2(z.\varphi)\).
3. \(f_1 f_2^2\) and \(h_1 h_2\) are compatible, and
4. the first quasistate of \(h_2\) is a successor of the last quasistate of \(h_2\).

**Proof.** (\(\Rightarrow\)) Suppose \(z \varphi\) is not safe. Then \(z \varphi\) has a \(k\)-bounded quasicounterexample \((f_1, g)\). Suppose \(f(i) = (\Phi_i, d_i)\), for \(i \in \mathbb{N}\) and let \(d\) be the sequence defined by \(d(i) = d_i\), for \(i \in \mathbb{N}\). We construct \(f_1, f_2, h_1, h_2, h_3, h_4\) and \(f(1)\) using ideas similar to those used in Lemma \([11]\).

We start with \(f_1\) and \(h_1\). Take \(s\) such that \(g(s, 0) = g(i, 0)\) and \((f(s) = f(i))\), for infinitely many \(i > s\). Apply Lemma \([11]\) whenever \(i < i_2 < s\), \(g(i_1, 0) = g(i_2, 0)\), and \((f(i_1) = f(i_2))\). This yields the quasicounterexample \((f_1 f_2^2, g_1)\) with \(|f_1| \geq 2(z.\varphi)^2\). Take \(h_1\) as the sequence \(g_1(0, 0) g_1(1, 0) \ldots g_1(|f_1| - 1, 0)\).

We now explain how to get \(f_2\) and \(h_2\). Suppose there is a formula in \(f_2^2(0)|\) of the form \(z. (\psi_1 \cup \psi_2)\). Take any \(k\) such that \(z. \psi_2^{\delta(x, s+k)} \in f_2^2(k)\). Apply Lemma \([11]\) whenever \(i_1 < i_2 < k\), \(f_2^2(i_1) = f_2^2(i_2)\), and \(g_{2^2}(i_1, 0) = g_{2^2}(i_2, 0)\). This yields the quasicounterexample \((f_1 f_2^2, g_1)\) with \(|f_2| \geq 2(z.\varphi)^2\).

With \(f_2\) and \(h_2\) known, \(f_1 f_2^2\) is a quasimodel for \(z \neg \varphi\) by construction of Lemma \([11]\). Replacing \(f_3\) by \(f_3^2\) and \(h_3\) by \(h_3^2\) yields a quasimodel for \(z \varphi\) by applying Lemma \([11]\) whenever \(i_1 < i_2 < k\), \(f_3^2(i_1) = f_3^2(i_2)\), and \(g_{3^2}(i_1, 0) = g_{3^2}(i_2, 0)\). This yields the quasicounterexample \((f_1 f_2^2 f_3^2, g_1)\) with \(|f_3| \geq 2(z.\varphi)^2\). Finally, \(h_4\) follows from Lemma \([11]\).

Theorem 4. Deciding whether a TPTL sentence is safe is EXPSPACE-complete.

**Proof.** EXPSPACE-hardness follows from the fact that deciding whether a TPTL formula is valid is EXPSPACE-complete \([5]\). The sentence \(z \varphi\) is valid iff \(z \varphi \lor \square q\) is safe, where \(q\) is an atomic proposition not occurring in \(z \varphi\).

We now present a non-deterministic algorithm that decides whether a TPTL sentence is safe by guessing finite
sequences $f_1, f_2, h_1, h_2, h_3, h_4$ of quasistates that satisfy the requirements of Theorem 3. This algorithm uses an amount of memory exponential in the length of $z. \varphi$. By Savitch’s theorem, it follows that deciding whether a TPTL sentence is safe is in EXPSPACE.

First, guess a number $\ell_1 \leq \sharp(z. \varphi)^2$. Now guess two compatible $k_\varphi$-bounded quasistates $(\Phi_0, d_0)$ and $(\Psi_0, e_0)$, with $z. \varphi \notin \Phi_0$ and $z. \varphi \in \Psi_0$. They are the first quasistates for $f_1$ and $h_1$ respectively. Next, for $i$ from 1 to $\ell_1 - 1$, guess two compatible $k_\varphi$-bounded quasistates $(\Phi_i, d_i)$ and $(\Psi_i, e_i)$ that are successors of $(\Phi_{i-1}, d_{i-1})$ and $(\Psi_{i-1}, e_{i-1})$ respectively. This gives rise to the two sequences $f_1$ and $h_1$. Similarly, guess the sequences $f_2$ and $h_2$. Guess a number $\ell_2 \leq \sharp(\text{Cl}(z. \varphi)) + 2 \cdot \sharp(z. \varphi)^2$ and guess two compatible $k_\varphi$-bounded quasistates $(\Phi_0', d_0')$ and $(\Psi_0', e_0')$. These quasistates must be successors of $(\Phi_{\ell_1-1}, d_{\ell_1-1})$ and $(\Psi_{\ell_1-1}, e_{\ell_1-1})$. To check conditions 1 and 4 of Theorem 3 set a variable $n$.

Proof. ($\Rightarrow$) Let $\sigma = (a_0, d_0)(a_1, d_1) \ldots (a_k, d_k)$ be a sequence in $\Sigma_\varphi^* \subseteq \Sigma^*$. By assumption, there is $\sigma' \in \Sigma^\omega$ of the form $\sigma' = (a_{k+1}, d_{k+1})(a_{k+2}, d_{k+2}) \ldots$ such that $\sigma \sigma'$ is a timed word that satisfies $z. \varphi$. Let

$$\sigma' := (a_{k+1}, d_{k+1})(a_{k+2}, d_{k+2}) \ldots$$

By Lemma 4, $\sigma \sigma'$ satisfies $z. \varphi$. So $\sigma$ is a $k_\varphi$-good prefix for $z. \varphi$. ($\Leftarrow$) For $\sigma = (a_0, d_0)(a_1, d_1) \ldots (a_k, d_k) \in \Sigma^*$, let $\hat{\sigma} := (a_0, d_0)(a_1, d_1) \ldots (a_k, d_k)$, which is in $\Sigma_\varphi^*$. By assumption, $\hat{\sigma}$ is a $k_\varphi$-good prefix for $z. \varphi$. So, there is $\sigma' \in \Sigma^\omega$ such that $\sigma \sigma'$ is a $k_\varphi$-bounded timed word that satisfies $z. \varphi$. By Lemma 2, $\sigma \sigma'$ satisfies $z. \varphi$, so $\sigma$ is a good prefix for $z. \varphi$.

Definition 4. An infinite sequence of quasistates for a formula $z. \varphi$ is called a fulfilling path for $z. \varphi$ if it meets the conditions (QM1), (QM2), and (QM3) from Definition 2.

The following lemma is proved similarly to Lemma 3.

Lemma 13. There is a fulfilling path for $z. \varphi$ iff there are sequences $f_1$ and $f_2$ of $k_\varphi$-bounded quasistates for $z. \varphi$ such that:

1. $|f_1| \leq \sharp(z. \varphi)$ and $|f_2| \leq (|\text{Cl}(z. \varphi) + 2| \cdot \sharp(z. \varphi)$,
2. $d > 0$ for some $(\Phi, d)$ in $f_2$,
3. $f_j(i + 1)$ is a successor of $f_j(i)$ for $i < |f_j| - 1$ and $j \in \{1, 2\}$,
4. $f_2(0)$ is a successor of the last quasistates of $f_1$ and $f_2$, and
5. every occurrence of the form $z. (\psi_1 \cup \psi_2)$ in $f_2(0)$ is realized by $f_2$.

Lemma 14. There is an algorithm that, given a quasistate $(\Phi_0, d_0)$ for a formula $z. \varphi$, decides whether there is a fulfilling path $f$ for $z. \varphi$ such that $f(0) = (\Phi_0, d_0)$. The algorithm uses space exponential in the length of $z. \varphi$.

Proof. By Savitch’s theorem, it suffices to give a non-deterministic algorithm that uses space exponential in $n_\varphi$. The algorithm guesses a fulfilling path of the form described in Lemma 3. First, guess the lengths of $f_1$ and $f_2$, namely $\ell_1$ and $\ell_2$ with $\ell_1 \leq \sharp(z. \varphi)$ and $\ell_2 \leq (|\text{Cl}(z. \varphi) + 2| \cdot \sharp(z. \varphi)$. Then for $i$ from 1 to $\ell_1 - 1$, guess a $k_\varphi$-bounded quasistate $(\Phi_i, d_i)$ that is a successor of $(\Phi_{i-1}, d_{i-1})$. After guessing $f_1$, let $b = 0$ and let $T$ be the set of all formulas of the form $z. (\psi_1 \cup \psi_2) \in (\Phi_{\ell_1-1}$. Guess $f_2$ as follows. First, guess a $k_\varphi$-bounded quasistate $(\Phi_0, d_0')$ that is a successor of $(\Phi_{\ell_1-1}, d_{\ell_1-1})$. Then for $i$ from 1 to $\ell_2 - 1$, guess a $k_\varphi$-bounded quasistate $(\Phi'_i, d'_i)$ that is a successor of $(\Phi'_{i-1}, d'_{i-1})$. Every time the next quasistate $(\Phi'_i, d'_i)$ for $f_2$ is guessed, set $b$ to 1 if $d'_i > 0$ and remove from $T$ all formulas $z. (\psi_1 \cup \psi_2)$ such that $z. (\psi_1 \cup \psi_2)$ is empty, and $(\Phi_0, d_0')$
is a successor of \((\Phi'_{ℓ−1}, d'_{ℓ−1})\). If these checks succeed, then by Lemma \ref{lem:successor}, there is a fulfilling path for \(z.ϕ\). The complexity result follows from the proof of Theorem \ref{thm:algorithm}.

Recall that an automaton is a tuple \((Q, Γ, δ, q_0, F)\), where \(Q\) is a finite nonempty set of states, \(Γ\) is a finite nonempty alphabet, \(q_0 \in Q\) is the initial state, \(δ \subseteq Q × Γ × Q\) is the transition relation, and \(F \subseteq Q\) is the set of accepting states. We can see \(δ\) as a set of directed edges between states that are labeled with elements of \(Γ\).

**Theorem 5.** There is an algorithm that decides whether a formula \(z.ϕ\) describes a liveness property. The algorithm uses space doubly exponential in the length of \(z.ϕ\).

**Proof.** The algorithm has two parts. First, build an automaton \(A\) over the alphabet \(Σ_ϕ\) that accepts \(σ \in Σ_ϕ^∗\) if \(σ\) is a \(k_ϕ\)-good prefix for \(z.ϕ\). Second, check if \(A\) accepts all the words in \(Σ_ϕ^∗\). By Lemma \ref{lem:acceptance}, \(A\) accepts all the words in \(Σ_ϕ^∗\) if \(z.ϕ\) describes a liveness property.

First, we define \(A\). \(A\)'s set of states is the set of all \(k_ϕ\)-bounded quasistates for \(z.ϕ\) together with a distinguished initial state called \(init\). For two states \(s_1, s_2\) and \((a, d) \in Σ_ϕ\), there is an edge from \(s_1\) to \(s_2\) labeled with \((a, d)\) iff
1. \(s_2 = (Φ, d')\) with \(d' = d\),
2. \(p \in a\) iff \(p \in Φ\) for all \(p \in P\),
3. if \(s_1 = init\) then \(z.ϕ \in Φ\), and
4. if \(s_1 \neq init\) then \(s_2\) is a successor of \(s_1\).

Finally, for every state \(s\) different from \(init\), apply Lemma \ref{lem:successor} and make \(s\) accepting iff there is a fulfilling path \(f\) for \(z.ϕ\) such that \(f(0) = s\).

Next, we prove that \(A\) accepts \(σ \in Σ_ϕ^∗\) iff \(σ\) is a \(k_ϕ\)-good prefix for \(z.ϕ\). If \(σ\) is accepted by \(A\), then there is a path \(init, s_0, s_1, \ldots, s_{ℓ−1}\) in \(A\) such that \(s_0\) is an accepting state and the concatenation of the labels of the edges in the path reads \(σ\). Let \(f\) be a fulfilling path for \(z.ϕ\) such that \(f(0) = s_0\). It is easy to prove that \(s_0, s_1, \ldots, s_{ℓ−1}\) is a \(k_ϕ\)-bounded quasimodel for \(z.ϕ\) and that the timed word defined by this quasimodel according to Theorem \ref{thm:quasimodel} is an extension of \(σ\). Therefore, \(σ\) is a \(k_ϕ\)-good prefix for \(z.ϕ\).

Suppose now that \(σ = (a_0, d_0)(a_1, d_1)\ldots(a_{ℓ−1}, d_{ℓ−1})\) is a \(k_ϕ\)-good prefix for \(z.ϕ\). Let \(σ' = (a_0, d_0)(a_{ℓ−1}, d_{ℓ−1})\ldots(Φ_{ℓ−1}, d_{ℓ−1})\) be such that \(σ'\) is a \(k_ϕ\)-bounded timed word that satisfies \(z.ϕ\). Let \(f_1 = (Φ_0, d_0)(Φ_1, d_1)\ldots(Φ_{ℓ−1}, d_{ℓ−1})\) and \(f_2 = (Φ_{ℓ−1}, d_{ℓ−1})(Φ_{ℓ−1}, d_{ℓ−1})\ldots\) be the sequences of quasistates for \(z.ϕ\), where \(Φ_i := \{z, ψ \in Cl(z.ϕ) | σ' i \models z.ϕ\}\), for \(i \in N\). By Theorem \ref{thm:quasimodel}, \(f_1, f_2\) is a quasimodel for \(z.ϕ\). It follows that \(f_2\) is a fulfilling path for \(z.ϕ\) and hence \(f_2(0)\) is an accepting state in \(A\). Also, \(init, f_1, f_2(0)\) is a path in \(A\) such that the concatenation of the edges in the path reads \(σ\). Therefore, \(A\) accepts \(σ\).

We now analyze the complexity of building \(A\) and checking if \(A\) accepts all the words in \(Σ_ϕ^∗\). The number of states of \(A\) is \(1 + 2^{|z.ϕ|} \leq 2^{(n_ϕ+1)k_ϕ} = 2^{O(n_ϕk_ϕ)}\). The size of the alphabet \(Σ_ϕ\) is \(|2^{|Γ|}\cdot k_ϕ = O(k_ϕ)|\). Therefore, building \(A\) takes \(2^{O(n_ϕk_ϕ)}\) space. Checking if \(A\) accepts \(Σ_ϕ^∗\) takes space polynomial in the number of \(A\)'s states, which is \(2^{O(n_ϕk_ϕ)}\) space. The values of \(n_ϕ\) and \(k_ϕ\) are respectively linear and exponential in the length of \(z.ϕ\), and therefore our algorithm uses space doubly exponential in the length of \(z.ϕ\).

\(\square\)

6. Conclusion

Sistla \cite{Sistla19} proved that deciding safety and liveness for LTL are PSPACE-complete and in EXPSPACE, respectively. We have carried over his proofs to TPTL and proved that the corresponding problems for TPTL are EXPSPACE-complete and in 2-EXPSPACE, respectively. Concerning liveness, we have the following lower bounds. Checking liveness is PSPACE-hard for LTL and EXPSPACE-hard for TPTL. This is because \(ϕ\) is satisfiable iff \(\Diamond ϕ\) describes a liveness property. Tighter lower bounds for deciding liveness remain unknown for both LTL and TPTL. Note that we considered a discrete time domain for TPTL. In the case of dense time, satisfiability for TPTL is undecidable \cite{Barber01}, and thus checking safety and liveness are both undecidable.

References