Acta Informatica manuscript No.
(will be inserted by the editor)

A Tool for Deciding the Satisfiability of Continuous-time Metric Temporal Logic

Marcello M. Bersani · Matteo Rossi · Pierluigi San Pietro

Received: date / Revised version: date

Abstract Constraint LTL over clocks is a variant of CLTL, an extension of linear-time temporal logic allowing atomic assertions in a concrete constraint system. Satisfiability of CLTL over clocks is here shown to be decidable by means of a reduction to a decidable SMT (Satisfiability Modulo Theories) problem. The result is a complete Bounded Satisfiability Checking procedure, which has been implemented by using standard SMT solvers. The importance of this technique derives from the possibility of translating various continuous-time metric temporal logics, such as MITL and QTL, into CLTL over clocks itself. Although standard decision procedures of these logics do exist, they have never been realized in practice. Suitable translations into CLTL over clocks have instead allowed us the development of the first prototype tool for deciding MITL and QTL. The paper also reports preliminary, but encouraging, experiments on some significant examples of MITL and QTL formulae.

1 Introduction

Constraint LTL [18], called CLTL, is an extension of linear-time temporal logic allowing atomic assertions in a concrete constraint system. By carefully choosing the constraint system, CLTL may be decidable, as well as expressive and well-suited to define infinite-state systems and their properties.

In this paper, we define a variant of CLTL, called CLTL over clocks (CLTLoc), where arithmetic variables occurring in atomic assertions behave as *clocks*. At every (discrete) position in time, a clock measures the real time elapsed since the last position when the clock itself was "reset" (i.e., the variable was equal to 0); clocks can also be compared against an integer constant. By definition, in CLTLoc each position $i \in \mathbb{N}$ is associated with a real value (a "delay") corresponding to the "time elapsing" between i and the next position $i + 1$. This allows mixing of

This research was supported by the Programme IDEAS-ERC, Project 227977-SMScom and by PRIN Project 2010LYA9RH.

Politecnico di Milano – DEIB, Piazza Leonardo da Vinci, 32 – 20133 Milano, Italy

discrete events with continuous-time, a typical situation arising in many computer-controlled applications.

Satisfiability of CLTLoc is here shown to be decidable by means of a reduction to a decidable SMT (Satisfiability Modulo Theories) problem, resulting in a complete Bounded Satisfiability Checking procedure. Although other automata-based decision procedures are also suitable to show decidability of CLTLoc (e.g., [18]), the novelty of our reduction is that it can easily be implemented by using standard SMT solvers, such as [25]. In fact, the paper also reports on a new, publicly available, software tool to verify CLTLoc, allowing the application of CLTLoc to the specification and the verification of timed systems. However, a further advantage of our approach is that various continuous-time metric temporal logics, such as MITL (Metric Interval Temporal Logic) [4] and QTL (Quantified Temporal Logic) [22], may be translated into CLTLoc itself. These translations have allowed us the development of the first available tool for deciding both MITL and QTL. In this paper we report encouraging experiments on some significant verification examples, such as the timed lamp and its properties, in CLTLoc, MITL and QTL. Further evidence of the generality and effectiveness of our approach is provided by our translation of the extension of QTL with so-called Pnueli and counting modalities [29] into CLTLoc, thus providing its first concrete decision procedure.

In general, the existing level of support for verification of continuous-time temporal logics is not as well developed as for discrete-time models. Uppaal [7] is the de-facto standard tool for verification of Timed Automata, but it does not support continuous-time temporal logics: not only satisfiability checking is not available in Uppaal, but even the formalization of system properties in temporal logic is not allowed, aside from rather simple invariants and reachability properties. Satisfiability Modulo Theories is a promising but well-consolidated field, supported by efficient solvers that are able to decide problems of many disciplines. In particular, decidable SMT problems have been already considered in the recent past, for instance to solve reachability [26] and the bounded version of language inclusion [6] for Timed Automata. The idea is to give a direct representation of bounded runs of Timed Automata through an SMT formula, capturing a bounded unrolling of the transition relation. Similarly, also Bounded Model-Checking of Linear Temporal Logic on Timed Automata [5] can be tackled by reducing the problem to an instance of a SMT problem, by using a technique extending the traditional BMC procedure for LTL finite automata [17], but by restricting the set of valid runs to those that are periodic in the values of the clocks. Finite or periodic runs of Timed Automata can then be encoded in SMT formulae with explicit arithmetic. Nonetheless, also this approach has so far failed to produce a concrete decision procedure for logics such as MITL and QTL. This difficulty is caused by the gap of translating formulae into Timed Automata, a step which is avoided by our approach. Standard decision procedures of MITL and QTL logics were already defined some time ago (e.g., [4,24,30]), typically based on Timed Automata [3], but, to the best of our knowledge, they have never been realized in practice. This may suggest that these procedures are not easily implementable.

Temporal Logics such as TPTL (Timed Propositional Temporal Logic), MTL (Metric Temporal Logic), MITL and QTL, and operational model such as Timed Automata as well, may be interpreted over dense time domains in two ways: the "pointwise" semantics and the "continuous (or "interval-based") semantics [16]. In the pointwise semantics, an atomic formula is interpreted as an instantaneous event

with a timestamp. A behavior (or run) of the system is described by a timed word, which is a sequence $(a_0, t_0)(a_1, t_1)\ldots$, where each a_i is a symbol of the alphabet and each t_i is a real-valued timestamp. A timed word must be strictly increasing ($t_i < t_{i+1}$) and must verify the non-Zeno condition, i.e., it is finite or it diverges to infinity. The pointwise semantics is very natural when considering specifications of Timed Automata, with atomic formulae interpreted as *state transitions*. In the continuous semantics, atomic formulae are instead interpreted as *state predicates*, i.e., continuous flows or *signals*. A signal (also called a timed state sequence) is a mapping associating values in $\mathbb{R}_+$ with states. A finite variability condition (strictly related to the non-Zeno condition) is always assumed. There are various results of expressiveness and decidability concerning MTL over the two semantics. First, it is obvious that a MTL (and MITL as well) formula interpreted over the pointwise semantics can always be translated into an equivalent MTL (or MITL) formula in the continuous semantics. However, other results are less immediate. For instance, MTL is undecidable in the continuous semantics [4] (unless time-singular intervals are ruled out, thus obtaining the logic MITL), but it is decidable, although not primitive recursive, in the pointwise version over finite models [27]–paving the way to showing that MTL in the continuous semantics is strictly more expressive than MTL in the pointwise semantics [19, 16] (over both finite and infinite models). No similar expressiveness result is known for MITL.

CLTLoc is naturally defined over timed words and it as expressive as Timed Automata [15], i.e., it can define the same class of languages (the timed ω-regular). In this paper, we prove that CLTLoc is decidable, with an SMT-based procedure which has been implemented. We applied the resulting tool to a few examples of CLTLoc specifications, showing that their verification is feasible.

However, CLTLoc can also be used as a tool to interpret and verify other metric temporal logics. For instance, in [13] we provide a complete translation of MITL formulae over the continuous semantics into equisatisfiable CLTLoc formulae, thus allowing their verification with our CLTLoc tool. We have implemented this translation and we report on various experimental results on MITL specifications.

We also consider the case of MITL over the pointwise semantics, first by studying its expressiveness compared to CLTLoc. In this case, MITL is less expressive than CLTLoc, but we prove that CLTLoc is equally expressive with *projection-closed* MITL (pMITL), an extension of MITL allowing existential propositional quantifiers [21]. Clearly, MITL formulae on timed words may be verified by a simple conversion into equivalent MITL formulae on signals, which can then be translated into CLTLoc. However, the translation from pMITL to CLTLoc, defined in the equivalence proof, is much more compact than the one defined for the continuous case, since in general signals may be more complex than timed words. Therefore, it may be more efficient to apply the new translation to convert MITL pointwise formulae directly into CLTLoc.

The paper is organized as follows. The first part is devoted to the main definitions and to the proofs of decidability of CLTLoc and of equal expressiveness of CLTLoc and pMITL: Sect. 2 defines CLTLoc, and illustrates it by means of a running example (a timed lamp), while Sect. 3 briefly recalls the definition of MITL and some of its variants, both in the pointwise and in the continuous cases; Sect. 4 proves that CLTLoc is decidable; Sect. 5 shows that pMITL and CLTLoc are equally expressive over the pointwise semantics.

The remainder of the paper is devoted to illustrate the ideas and experimental results of our verification tool for CLTLoc: Sect. 6 outlines the SMT-based decision procedure of CLTloc, Sect. 7 recalls the general idea of [13] behind the translation of MITL over the continous semantics into CLTLoc, while Sect. 8 illustrates the tool, showing verification results for CLTLoc and MITL. Sect. 9 concludes.

2 Constraint LTL over clocks

Constraint LTL (CLTL [18,11]) is an extension of LTL allowing atomic formulae over a constraint system. Let AP be a finite set of atomic propositions, let V be a finite set of variables and let $\mathcal{D} = (D, \mathcal{R})$ be a *constraint system*, where D is a specific domain of interpretation for variables and constants and $\mathcal{R}$ is a finite family of relations on D (of various arities). The set AP coincides with the set $\mathcal{R}_0$ of 0-ary relations. *Temporal terms* are defined by the syntax: $\alpha := c \mid x \mid \mathrm{X}\alpha$, where c is a constant in D and x is a variable in V. Operator X is very similar to the LTL next operator $\mathbf{X}$, but it only applies to temporal terms, with the intended meaning that $\mathrm{X}\alpha$ is the *value* of α in the next position.

An *atomic constraint* is a term of the form $R(\alpha_1, \ldots, \alpha_n)$, for $n \geqslant 0$, where R is an n-ary relation of $\mathcal{R}$ and $\alpha_1, \ldots, \alpha_n$ are temporal terms.

Well-formed CLTL formulae are defined as follows:

$$\phi := p \mid R(\alpha_1, \ldots, \alpha_n) \mid \phi \wedge \phi \mid \neg\phi \mid \mathbf{X}(\phi) \mid \mathbf{Y}(\phi) \mid \phi \mathbf{U} \phi \mid \phi \mathbf{S} \phi$$

where $p \in AP$, every α_i is a temporal term, $R \in \mathcal{R}$, $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{U}$ and $\mathbf{S}$ are the usual "next", "previous", "until" and "since" operators of LTL, with the same meaning. The dual operators "release" $\mathbf{R}$, and "trigger" $\mathbf{T}$ may be defined as usual, i.e., $\phi \mathbf{R} \psi$ is $\neg(\neg\phi \mathbf{U} \neg\psi)$ and $\phi \mathbf{T} \psi$ is $\neg(\neg\phi \mathbf{S} \neg\psi)$. The semantics of CLTL is defined with respect to a constraint system $\mathcal{D}$ and the strict linear order $(\mathbb{N}, <)$ representing *positions* in time. An *interpretation* is a pair (π, σ), where $\sigma : \mathbb{N} \times V \to D$ is a mapping assigning for every variable $x \in V$ its value $\sigma(x, i)$ at each position $i \in \mathbb{N}$ and $\pi : \mathbb{N} \to \wp(AP)$ is a mapping associating a set of propositions with each position in $\mathbb{N}$. The semantics of CLTL at a position $i \in \mathbb{N}$ over an interpretation (π, σ) is defined in Table 1. The only case that needs to be explained is the evalutation σ of temporal terms:

$$\sigma(i, \alpha) = \sigma(i + |\alpha|, x_\alpha)$$

where x_α is the (unique) variable occurring in term α and $|\alpha|$ is the *depth* of a temporal term, namely the total amount of temporal shift needed in evaluating α: $|x| = 0$ when x is a variable, and $|\mathrm{X}\alpha| = |\alpha| + 1$. A formula $\phi \in$ CLTL is *satisfiable* if there exists a pair (π, σ) such that $(\pi, \sigma), 0 \models \phi$. In this case, we say that (π, σ) is a *model* of ϕ and we write simply $(\pi, \sigma) \models \phi$.

CLTLoc is a special case of CLTL, where the domain D is $\mathbb{R}_+$ (the set of nonnegative reals), the set $\mathcal{R} \backslash \mathcal{R}_0$ of relations is $\{<, =\}$ and the arithmetic variables behave as *clocks*.

Hence, the valuation of clocks is defined by a mapping $\sigma : \mathbb{N} \times V \to \mathbb{R}_+$, assigning, for every position $i \in \mathbb{N}$, a real value $\sigma(i, x)$ to each clock $x \in V$. Intuitively, a clock x measures the time elapsed since the last time when $x = 0$, i.e., the last "reset" of x. To ensure that time progresses at the same rate for every clock, σ

$$
\begin{aligned}
(\pi,\sigma), i \models p &\Leftrightarrow p \in \pi(i) \text{ for } p \in AP \\
(\pi,\sigma), i \models R(\alpha_1,\ldots,\alpha_n) &\Leftrightarrow (\sigma(i+|\alpha_1|, x_{\alpha_1}),\ldots,\sigma(i+|\alpha_n|, x_{\alpha_n})) \in R \\
(\pi,\sigma), i \models \neg\phi &\Leftrightarrow (\pi,\sigma), i \not\models \phi \\
(\pi,\sigma), i \models \phi \wedge \psi &\Leftrightarrow (\pi,\sigma), i \models \phi \,\text{and}\, (\pi,\sigma), i \models \psi \\
(\pi,\sigma), i \models \mathbf{X}(\phi) &\Leftrightarrow (\pi,\sigma), i+1 \models \phi \\
(\pi,\sigma), i \models \mathbf{Y}(\phi) &\Leftrightarrow (\pi,\sigma), i-1 \models \phi \wedge i > 0 \\
(\pi,\sigma), i \models \phi \mathbf{U} \psi &\Leftrightarrow \exists\, j \geqslant i : (\pi,\sigma), j \models \psi \ \wedge (\pi,\sigma), n \models \phi \ \forall\ i \leqslant n < j \\
(\pi,\sigma), i \models \phi \mathbf{S} \psi &\Leftrightarrow \exists\, 0 \leqslant j \leqslant i : (\pi,\sigma), j \models \psi \ \wedge (\pi,\sigma), n \models \phi \ \forall\ j < n \leqslant i
\end{aligned}
$$

Table 1 Semantics of CLTL.

must satisfy the following condition: for every position $i \in \mathbb{N}$, there exists a "time delay" $\delta_i > 0$ such that for every clock $x \in V$:

$$
\sigma(i+1, x) = \begin{cases} \sigma(i,x) + \delta_i, & \text{time progress} \\ 0 & \text{reset } x. \end{cases}
$$

In this case, σ is called a *clock valuation*. An interpretation for a CLTLoc formula ϕ is a pair (π, σ), where σ is a clock valuation and $\pi : \mathbb{N} \to \wp(\Sigma)$. The definition of the semantics of CLTLoc over (π, σ) is the same of CLTL.

It is often convenient to assume that at every position there is at least one clock which is not reset. To ensure that this is the case, just add a new clock *Now*, which is never reset, except possibly at position 0. Hence, the time delay δ_i is uniquely defined in each position i as $\sigma(i+1, Now) - \sigma(i, Now)$. The initial value of a clock, $\sigma(0, x)$, may be any non-negative value. If needed, one or more of the clocks may be initialized to 0 just by adding a constraint of the form $x = 0$.

To compare CLTLoc with other formalisms, we introduce the satisfiability of CLTLoc formulae over *timed ω-words*. A timed ω-word over $\wp(\Sigma)$ is a pair (π, τ) where $\pi : \mathbb{N} \to \wp(\Sigma)$ and τ is a monotonic function $\tau : \mathbb{N} \to \mathbb{R}$ such that $\forall i\ \tau(i) < \tau(i+1)$ (strong monotonicity). The value $\tau(i)$ is called the timestamp at position i, $i \in \mathbb{N}$. Given a CLTLoc interpretation (π, σ), let τ be such that $\tau(i) = \sigma(i, Now)$. Then, (π, τ) is called the timed ω-word associated with (π, σ) and it is denoted by $[(\pi, \sigma)]$.

A relation $\models$ can be defined for every timed ω-word (π, τ) as follows. Let $(\pi, \tau) \models \phi$ hold if there exists an interpretation (π, σ) such that $(\pi, \sigma) \models \phi$ and $(\pi, \tau) = [(\pi, \sigma)]$. A CLTLoc formula ϕ is satisfiable *over timed ω-words* if $(\pi, \tau) \models \phi$, for some (π, τ).

Before going further, to motivate our approach, we provide an example of a CLTLoc formula representing a simple yet realistic timed system.

Example 1

We consider the LTL specification of a timed lamp and its properties (studied in Sect. 8) from [28]. The lamp is controlled by two buttons, labeled ON and OFF respectively, which cannot be pressed simultaneously. The lamp itself can be either on or off. When ON is pressed the lamp is immediately turned on, regardless of its current state; similarly, if OFF is pushed then the lamp is immediately turned

off, also regardless of its current state. After ON is pressed, the lamp will not stay on forever, but, if no more buttons are pressed, it will automatically turn off with a delay Δ, a positive real constant. By pressing the ON button before the timeout expiration then the timeout is extended by a new delay Δ.

Our CLTLoc formula makes use of atomic propositions *on*, *off* and l representing, respectively, events "push button ON" and "push button OFF" and the state "light is on". Clocks may be used to measure the exact time elapsed since the last *on*; clearly some clock must be "reset" (i.e., set to 0, in analogy to Timed Automata) whenever ON is pressed, while when a clock is equal to Δ then the timeout *expires*. To simplify the introduction of clocks, we first define a few shorthands called *rst-c*, $test_{c=\Delta}$ and $test_{0<c\leqslant\Delta}$. They have the intuitive meaning (which will be formalized after the main specification) that they are true if, and only if, a clock c is reset or, respectively, $c = \Delta$, or $0 < c \leqslant \Delta$. The specification of the lamp, still lacking the precise clock specification, is defined by formula $\mathbf{G}\left(\bigwedge_{i=0}^{5}(i)\right)$, where $\mathbf{G}(\phi)$ is the usual globally operator defined by $\bot\mathbf{R}\phi$.

$$\neg(on \wedge off) \tag{1}$$

$$on \Leftrightarrow rst\text{-}c \tag{2}$$

$$\mathbf{Y}(l) \Rightarrow test_{0<c\leqslant\Delta} \tag{3}$$

$$turnoff \Leftrightarrow \mathbf{Y}(l) \wedge (off \vee test_{c=\Delta}) \tag{4}$$

$$l \Leftrightarrow \neg turnoff\ \mathbf{S}\ on. \tag{5}$$

Formula (1) ensures mutual exclusion; (2) states that the timeout must be (re)started whenever button ON is pressed; (3) constrains the time elapsed since the previous instant if the light was on at that moment (i.e., not more than Δ); (4) defines (for readability) an event *turnoff*, capturing the two cases when the lamp (supposed to be ON in the previous instant) must be turned off at the current instant (i.e., OFF being pressed or the timeout expiring); finally, (5) gives the specification of the light, as being on if, and only if, there was in the past an *on* event not followed by a *turnoff*. Initialization is implicit in the specification (at instant 0, the light is off).

To complete the specification, we must formalize also the behavior of clocks. In CLTLoc, "resetting a clock" c, e.g., following an *on* event, is as simple as stating that $on \Rightarrow c = 0$; testing a clock c against a constant Δ and causing say, a *turnoff* is a simple as stating that $c = \Delta \Rightarrow turnoff$. Unfortunately, the same clock cannot be tested and reset at the same time. When this is required, it is possible to introduce two clocks c_0 and c_1, rather than just one clock, so that they can be reset alternatively: only one of the two clocks is reset and a new reset of the same clock will eventually occur only after the occurrence of a reset of the other clock. The behavior of this clock pair is described by the axiom $\mathbf{G}((6) \wedge (7))$, where formulae (6) and (7) are:

$$\bigwedge_{i\in\{0,1\}} \left(c_i = 0 \Rightarrow \neg\mathbf{X}\left((c_{\overline{(i+1)}_2} > 0\,\mathbf{U}\,c_i = 0)\right)\right) \tag{6}$$

$$c_0 = 0 \Rightarrow \neg(c_1 = 0) \tag{7}$$

and $\overline{\cdot}_2$ stands for the modulo 2 operator (i.e., $\overline{1}_2 = 1$, $\overline{2}_2 = 0$). Finally, the above clock shorthands *rst-c*, $test_{c=\Delta}$ and $test_{0<c\leqslant\Delta}$ are defined as follows:

$$\textit{rst-c} \Leftrightarrow c_0 = 0 \vee c_1 = 0$$

$$test_{0<c\leqslant\Delta} \Leftrightarrow \bigvee_{i\in\{0,1\}} 0 < c_i \leqslant \Delta$$

$$test_{c=\Delta} \Leftrightarrow \bigvee_{i\in\{0,1\}} \Big(c_i = \Delta \wedge (c_{\overline{(i+1)}_2} > \Delta \vee c_{\overline{(i+1)}_2} = 0)\Big).$$

3 MITL, $\text{MITL}_{(0,\infty)}$, QTL, and pMITL

Let I be an interval of the form $\langle a, b\rangle$ or of the form $\langle a, +\infty)$, where $0 \leqslant a < b$ are integer constants, $\langle$ is either (or [, and $\rangle$ is) or]. Given a finite alphabet AP of atomic propositions, the syntax of (well-formed) formulae of MITL is defined as:

$$\phi := p \mid \phi \wedge \phi \mid \neg\phi \mid \phi \mathbf{U}_I \phi$$

where $p \in AP$. We often write, as customary, $\mathbf{U}$ instead of $\mathbf{U}_{(0,+\infty)}$.

Boolean operators $\vee, \top, \bot, \Rightarrow$ and the *globally* $\mathbf{G}_I$ and *eventually* $\mathbf{F}_I$ operators can be defined by the usual abbreviations, e.g. $\mathbf{F}_I\phi = \top\mathbf{U}_I\phi$ and $\mathbf{G}_I\phi = \neg\mathbf{F}_I(\neg\phi)$.

A *signal* is a function $M : \mathbb{R}_+ \to \wp(AP)$ which is assumed to be finitely variable, i.e., such that in every finite interval there is a finite number of change points to the value of the atomic propositions in AP.

The continuous semantics of MITL is defined in Table 2, for every $t \in \mathbb{R}_+$ and for every signal M. Notice that in the definition of the semantics an interval I is interpreted as an interval of real numbers.

$$\begin{aligned}
M, t \models p &\Leftrightarrow p \in M(t) \qquad p \in AP \\
M, t \models \neg\phi &\Leftrightarrow M, t \not\models \phi \\
M, t \models \phi \wedge \psi &\Leftrightarrow M, t \models \phi \text{ and } M, t \models \psi \\
M, t \models \phi \mathbf{U}_I \psi &\Leftrightarrow \exists t' > t\ t' - t \in I, M, t' \models \psi \text{ and } \forall\ t < t'' < t'\ M, t'' \models \phi
\end{aligned}$$

Table 2 Continuous semantics of MITL.

A MITL formula ϕ is *satisfiable in the continuous semantics* if there exists a signal M such that $M, 0 \models \phi$. In this case, M is called a *continuous model* of ϕ.

The pointwise semantics is defined by introducing a relation $\models$, defined in Table 3 for every timed ω-word (π, τ) and for every position $i \in \mathbb{N}$. A MITL formula ϕ is *satisfiable in the pointwise semantics* if there exists a timed ω-word (π, τ) such that $(\pi, \tau), 0 \models \phi$. In this case, (π, τ) is called a *pointwise model* of ϕ.

A useful operator on timed words is "next" $\mathbf{X}_I$, with the intuitive meaning that $\mathbf{X}_I\phi$ holds at position i if ϕ is true at position $i + 1$, and the difference of timestamps $\tau(i+1) - \tau(i)$ is in I. Since we adopted the strict version of $\mathbf{U}_I$, $\mathbf{X}$ can be defined as $\mathbf{X}_I\phi = \bot\mathbf{U}_I\phi$. It is also possible to define MITL with the non-strict version of $\mathbf{U}_I$, but in this case it is necessary to introduce also the (non-metric) next operator $\mathbf{X}$ as primitive.

$$(\pi,\tau), i \models p \Leftrightarrow p \in \pi(i) \text{ for } p \in AP$$
$$(\pi,\tau), i \models \neg\phi \Leftrightarrow (\pi,\tau), i \not\models \phi$$
$$(\pi,\tau), i \models \phi \wedge \psi \Leftrightarrow (\pi,\tau), i \models \phi$$
$$(\pi,\tau), i \models \phi \mathbf{U}_I \psi \Leftrightarrow \exists j > i : \tau(j) - \tau(i) \in I, (\pi,\tau), j \models \psi \text{ and } \forall\, i < k < j \;\; (\pi,\tau), k \models \phi$$

Table 3 Pointwise semantics of MITL.

It is sometimes useful to extend MITL with past-time operators, and in particular with the "since" temporal operator $\phi \mathbf{S}_I \psi$, whose semantics is the dual of the one of $\mathbf{U}_I$ ($\mathbf{P}_I$ is the past-time dual of $\mathbf{F}_I$, so $\mathbf{P}_I \phi = \top \mathbf{S}_I \phi$). MITL with past-time operators (MITL+Past) is strictly more expressive than MITL [16] over both the continuous and pointwise semantics. Nevertheless, our encoding for the continuous semantics, presented in Sect. 7, can also deal with past-time operators, so the examples of Sect. 8 will also include them.

From timed words to signals. MITL formulae may give different results when interpreted over the pointwise semantics and over the continuous semantics. To preserve their meaning, they must be translated into different MITL formulae. The translation is straighforward. Consider, for example, a timed word $\zeta_{4X} = (\pi_{4X}, \tau_{4X})$ such that for all $i \in \mathbb{N}$ it is $\pi_{4X}(i) = \{p\}$, and $\tau_{4X}(i) = 4i$; that is, p always holds along ζ_{4X}, and all timestamps are multiples of 4. Formula $\mathbf{F}_{(0,4)}(p)$, states that p holds in at least one position whose timestamp is in the open interval $(0, 4)$. It is easy to see that $\zeta_{4X}, 0 \models \mathbf{G}\Big(\neg \mathbf{F}_{(0,4)}(p)\Big)$, i.e., $\mathbf{F}_{(0,4)}(p)$ never holds along ζ_{4X} since for each position i there is no k such that $\tau(k) - \tau(i) \in (0, 4)$.

Consider now a signal $M_{4X} : \mathbb{R}_+ \to \wp(AP)$ such that $M_{4X}(i) = \{p\}$ for every $i \in \mathbb{N}$ which is a multiple of 4. The value of $M_{4X}(t)$ for every other $t \in \mathbb{R}_+$ can be defined by adopting the intuitive convention that the positions of a timed word are considered to represent instantaneous events, hence every atomic proposition is false in the instants that are not timestamps: $p \notin M_{4X}(t)$ if t is not a multiple of 4. However, it is immediate to see that, whatever convention is used, $M_{4X}, 0 \not\models \mathbf{G}\Big(\neg \mathbf{F}_{(0,4)}(p)\Big)$. In fact, $\mathbf{F}_{(0,4)}(p)$ must hold over M_{4X} at least in every instant t which is not an exact multiple of 4. The formula must be modified to be $\mathbf{G}\Big(\neg p \vee \neg \mathbf{F}_{(0,4)}(p)\Big)$. It is easy to see that, if p is the only atomic proposition, the new formula over the continuous semantics is equivalent to the original formula over the pointwise semantics.

$MITL_{(0,\infty)}$ and QTL. A syntactic restriction of MITL, called $\text{MITL}_{(0,\infty)}$, is one in which in intervals $I = \langle a, b\rangle$ it is either $a = 0$ or $b = \infty$. Its semantics can easily be derived from the semantics of MITL.

The logic QTL is $\text{MITL}_{(0,\infty)}$ in which intervals are only of the form $(0, 1)$. Despite their apparent simplicity, $\text{MITL}_{(0,\infty)}$ and QTL have the same expressive power of MITL [23].

Projection-closed MITL. Finally, we define an extension of MITL, here called *projection-closed MITL*, pMITL for short [21]. This logic, defined on timed words, is obtained by adding a set pAP of $n \geqslant 0$ propositional variables $q_1, \ldots, q_n$, which can

be existentially quantified. The logic is called "projection-closed", since the actual extension to MITL is its capacity of adding new propositional variables, which can then be eliminated ("projected" away) by an external existential quantification, hence without extending the alphabet. This allows the definition of timed ω-languages that are not counter-free [21] (e.g., "the number of occurrences of event a is even"), which cannot be defined in MITL.

The syntax of pMITL is defined by the clause: $\exists q_1 \dots q_n \phi$, where $pAP = \{q_1, \dots, q_n\}$ for some $n \geqslant 0$ and ϕ is a MITL formula on the alphabet $AP \cup pAP$. To follow our definitions of pointwise semantics, the semantics of pMITL may, e.g., be defined by extending mapping π also to set pAP: $\pi : \mathbb{N} \to \wp(AP \cup pAP)$. Then, we introduce the semantic clause:

$$(\pi, \tau), i \models \exists q_1 \dots q_n \phi \Leftrightarrow \text{ there exists } \pi' : \mathbb{N} \to \wp(AP \cup pAP) \mid (\pi', \tau), i \models \phi \text{ and } \forall j \in \mathbb{N}\ \pi'(j) - pAP = \pi(j) - pAP$$

The meaning is that $(\pi, \tau), i \models \exists q_1 \dots q_n \phi$ if there exists a mapping $\pi' : \mathbb{N} \to \wp(AP \cup pAP)$ such that $(\pi', \tau), i \models \phi$ and, at every position j, $\pi'(j)$ may differ from $\pi(j)$ only in the presence or in the absence of $q_1, \dots, q_n$.

Notice that the alphabet of pMITL is only apparently extended with the set pAP: every proposition in pAP must be existentially quantified, hence the actual alphabet is still set AP. This point is important when comparing the expressiveness of pMITL with CLTLoc.

4 Decidability of CLTL over clocks

We show that any CLTLoc formula ϕ over a set of clocks V can be translated into a suitable Büchi automaton $\mathcal{A}_\phi^{\mathcal{R}}$ that accepts words including both sequences of symbolic valuations [18,10] and constraints representing the clock regions induced by ϕ. Automaton $\mathcal{A}_\phi^{\mathcal{R}}$ is built using a slight variation of the construction of [18, 11], where instead variables are not restricted to behave as clocks.

The set $\mathcal{R}_\phi$ of clock regions induced by ϕ can be determined from constraints in ϕ, by applying arguments akin to those used in [3] to define the region graph for a timed automaton. Automaton $\mathcal{A}_\phi^{\mathcal{R}}$ is defined as the product of the Büchi automaton $\mathcal{A}_\phi$recognizing the symbolic models of ϕ [18] with the automaton $\mathcal{A}_{\mathcal{R}_\phi}$ recognizing the language of successive regions of $\mathcal{R}_\phi$. In other words, automaton $\mathcal{A}_\phi^{\mathcal{R}}$ accepts all the symbolic models belonging to $\mathscr{L}(\mathcal{A}_\phi)$ and such that the sequence of regions determined by clock constraints within symbolic valuations obeys the time-successor relation. The following construction only guarantees that time elapses and that all clocks progress of the same amount, but it does not enforce time progress. Since ϕ is a CLTL formula where atoms may be relations over clocks, $\mathcal{A}_\phi^{\mathcal{R}}$ does not force the value of the clocks, which is instead constrained only by the formula ϕ itself. Hence, the definition of the initial region does not follow the standard construction where all clocks are zero. Therefore, we simply consider each region as potentially initial.

To represent correctly the elapsing time as measured by clocks, the models of CLTLoc define sequences of clock regions $R_0 R_1, \dots$ such that, for all time positions i, R_{i+1} is a time-successor of region R_i [3], except for those clocks that are reset in

R_{i+1}. Let ϕ be a CLTLoc formula and $x, y \in V$. If $c(x)$ is the maximum constant in ϕ clock x is compared to, then

$$ac(x) := \{x = 0\} \cup \{x < c,\ c < x,\ x = c \mid \forall c \in \mathbb{N}^+, c \leqslant c(x)\}$$

is the set of all clock constraints between x and constant $c(x)$ and

$$ac(x, y) := \{x + c \sim y,\ y \sim x + c,\ y + d \sim x,\ x \sim y + d \mid \forall c, d \in \mathbb{N}^+, c < c(y), d < c(x)\},$$

where $\sim \in \{<, =\}$, is the set of all clock constraints between x and y. Finally, set:

$$ac(\phi) := \bigcup_{x,y \in V, x \neq y} ac(x) \cup ac(x, y)$$

is the set of all clock constraints induced by ϕ. Let $\mathcal{R}_\phi$ be the set of all clock regions over $ac(\phi)$ as defined in [3].

To enforce the time-successor relation between adjacent regions, let $\mathcal{A}_{\mathcal{R}_\phi} = (\mathcal{R}_\phi, \delta)$ be the automaton where δ is the transition relation containing all transitions $R \to R'$ such that $R' \in \mathcal{R}_\phi$ is a time-successor of $R \in \mathcal{R}_\phi$, except for the clocks whose value is 0 in R'.

We briefly recall some key elements of $\mathcal{A}_\phi$ (see [10] and [18] for further details). A *symbolic valuation* for a CLTL formula ϕ is a maximally consistent set of constraints of constraint system $\mathcal{D}$ over the terms appearing in ϕ, including those of the form $\mathrm{X}^i x$ and also every constant $0 \leqslant c \leqslant c(x)$, for each $x \in V$; for CLTLoc the constraint system is $\mathcal{D} = (\mathbb{R}, <, =)$. We indicate by $SV(\phi)$ the set of symbolic valuations associated with ϕ. Let $\models^{sym}$ denote the symbolic satisfaction relation: given a sequence of symbolic valuations $\rho \in SV(\phi)^\omega$, we write $\rho \models^{sym} \phi$ when ρ symbolically satisfies ϕ. We say that a symbolic model ρ admits an arithmetical model if there exists an assignment $\sigma : \mathbb{N} \times V \to D$ that satisfies ρ. The closure of ϕ, denoted $cl(\phi)$, is the smallest set containing all subformulae of ϕ and closed under negation. An *atom* $\Gamma \subseteq cl(\phi)$ is a maximally consistent subset of formulae of $cl(\phi)$, such that, for each subformula ξ in ϕ, either $\xi \in \Gamma$ or $\neg\xi \in \Gamma$. Let Q be the set of all the atoms Γ.

Automaton A_ϕ is the generalized Büchi automaton $(SV(\phi), Q \times SV(\phi), I, \eta, F)$, where $SV(\phi)$ is the input alphabet and $Q \times SV(\phi)$ is the state space. It is a specialized version of the Vardi-Wolper automaton for LTL formulae recognizing models for formula ϕ, i.e., sequences of symbolic valuations that admit an arithmetical model. The following lemma of [18] is key to prove the decidability of the satisfiability of CLTLoc formulae.

Lemma 1 ([18]) *Let ϕ be a* CLTL$((\mathbb{R}, <, =))$ *formula and $\rho \in SV(\phi)$. Then, $\rho \in \mathscr{L}(\mathcal{A}_\phi)$ if, and only if, $\rho \models^{sym} \phi$ and ρ admits an arithmetical model.*

Now, we define $\mathcal{A}_\phi^{\mathcal{R}}$ as the tuple $(SV(\phi) \times \mathcal{R}_\phi, Q \times SV(\phi) \times \mathcal{R}_\phi, I', \eta', F \times \mathcal{R}_\phi)$. Relation η' is defined as follows: $(\Gamma, sv, R) \xrightarrow{(sv,R)} (\Gamma', sv', R') \in \eta'$ if, and only if,

- $(\Gamma, sv) \xrightarrow{sv} (\Gamma', sv') \in \eta$ and
- $R \to R' \in \delta$ and
- $sv' \cup R'$ is satisfiable.

Set $I' \subseteq I \times \mathcal{R}_\phi$ consists of initial states (atoms) of A_ϕ, that are consistent with the regions in $\mathcal{R}_\phi$, i.e., if $(\Gamma, sv) \in I$ and $R \in \mathcal{R}_\phi$, then $sv \cup R$ is satisfiable. Satisfiability of quantifier-free formulae over $(\mathbb{R}, <, =)$ is well-known to be decidable [31,20].

A run ρ of $\mathcal{A}_\phi^{\mathcal{R}}$ is a sequence:

$$(\Gamma_0, sv_0, R_0) \xrightarrow{(sv_0, R_0)} \dots (\Gamma_i, sv_i, R_i) \xrightarrow{(sv_i, R_i)} (\Gamma_{i+1}, sv_{i+1}, R_{i+1}) \dots$$

where $(\Gamma_0, sv_0, R_0) \in I'$. Let (π, σ) be a sequence, with $\pi : \mathbb{N} \to \wp(AP)$ and $\sigma : \mathbb{N} \times V \to \mathbb{R}$. Then, (π, σ) *witnesses* ρ, i.e., $(\pi, \sigma) \models \rho$, when for all $i \geqslant 0$:

$$\pi(i) = \Gamma_i \cap AP \text{ and } \sigma, i \models sv_i \cup R_i.$$

Lemma 2 *Let ϕ be a CLTLoc formula. Then, $(\pi, \sigma) \models \phi$ if, and only if, there is an accepting run ρ of $\mathcal{A}_\phi^{\mathcal{R}}$ such that $(\pi, \sigma) \models \rho$.*

Proof We first show that if $(\pi, \sigma) \models \phi$ then there is an accepting run of automaton $\mathcal{A}_\phi^{\mathcal{R}}$. We have to show that when the variables of ϕ behave like clocks, then the sequence of valuations defines a sequence of successive clock regions, hence ρ is also a run of $\mathcal{A}_\phi^{\mathcal{R}}$. By Lemma 1, formula ϕ, being also a CLTL$((\mathbb{R}, <, =))$ formula, is satisfiable if, and only if, there is a run $\rho = sv_0 sv_1 \dots$ of automaton $\mathcal{A}_\phi$ recognizing symbolic models of the formula. In fact, we have that $\sigma, i \models sv_i$. Since the set of clock regions is a partition of the clocks space, there is only one region $R \in \mathcal{R}_\phi$ such that $\sigma, i \models R_i$, with $R_i = R$. Therefore, $\sigma, i \models sv_i \cup R_i$ and model (π, σ) induces a sequence of regions $R_0 R_1 \dots$. We now prove that each $R_i R_{i+1}$ in the sequence $R_0 R_1 \dots$ is a pair of successive regions. In fact, if $\sigma(i, x)$ and $\sigma(i+1, x)$ are two adjacent valuations for a clock x of ϕ, then either there is a $t > 0$ such that $x(i+1) = x(i) + t$ or $x(i+1) = 0$ (reset). Therefore, R_{i+1} is a time successor of R_i (except for the clocks whose value is 0 in R_{i+1}) and the sequence $R_0 R_1 \dots$ is a sequence of successive regions belonging to the language $\mathscr{L}(\mathcal{A}_{\mathcal{R}_\phi})$. By construction, each atom Γ_i is such that $\pi(i) = \Gamma_i \cap AP$. At each position i, we have that $\pi(i) = \Gamma_i \cap AP$ and $\sigma(i) \models sv_i \cup R_i$. Hence, $(\pi, \sigma) \models \rho$.

We now show that if automaton $\mathcal{A}_\phi^{\mathcal{R}}$ has an accepting run then ϕ is satisfiable. By Lemma 1, the sequence $sv_0 sv_1 \dots$ of symbolic valuations is a symbolic model of ϕ (interpreted as a pure CLTL$((\mathbb{R}, <, =))$ formula) that admits an arithmetical model; i.e., there is an infinite sequence of valuations of the variables occurring in ϕ and satisfying the constraints in sv_i, for all $i \geqslant 0$. We show that one can build a sequence of valuations that respects also the constraints in the sequence of clock regions $R_0 R_1 \dots$. The proof is by induction on $i \geqslant 0$. First, the initial state of the run is satisfiable by definition of $\mathcal{A}_\phi^{\mathcal{R}}$, hence there is a valuation that satisfies it. The inductive step is that if, for $i > 0$, $\sigma_0 \sigma_1 \dots \sigma_{i-1}$ is a sequence of valuations $V \to \mathbb{R}$ that satisfy both $sv_0 sv_1 \dots sv_{i-1}$ and $R_0 R_1 \dots R_{i-1}$, then there is $sigma_i$ such that $\sigma_0 \sigma_1 \dots \sigma_i$ satisfy both $sv_0 sv_1 \dots sv_i$ and $R_0 R_1 \dots R_i$, i.e, the sequence can be extended to i. By definition of $\mathcal{A}_\phi^{\mathcal{R}}$, the set of constraints $sv_i \cup R_i$ is satisfiable. Note that sv_i and R_i have some common constraints, and in particular those of the forms $x \sim c$, $c \sim x$, $x \sim y$ and $y \sim x$ (where $\sim \in \{<, =\}$ and $c \in \{0, \dots, c(x)\}$). In addition, R_i has constraints of the form $x + c \sim y$ and $y \sim x + c$. If all constraints are of the form $x = c$ for all $x \in V$, then sv_i and R_i define the same constraints on the clocks at position i, hence a valuation that satisfies sv_i (which exists by Lemma 1) also satisfies R_i.

If, instead, the constraints of some clocks are of the form $c_1 < x < c_2$ or of the form

$c(x) < x$, then the region defined by sv_i is open and dense. In this case, constraints of the form $x + c \sim y$ and $y \sim x + c$ appearing in R_i define a subset of the region defined by sv_i which is also open and dense. To extend sequence $\sigma_0\sigma_1 \dots \sigma_{i-1}$ with a valuation σ_i, the latter must obey constraints of the following forms (we write $\sigma_i(x)$ to indicate the value of clock x in valuation σ_i): (i) $\sigma_i(x) \sim c$ or $c \sim \sigma_i(x)$, (ii) $\sigma_i(x) \sim \sigma_{i-k}(y)$ for some $k \geqslant 0$ (these correspond to CLTL constraints such as $\mathrm{X}x > y$), (iii) $\sigma_i(x) + c \sim \sigma_i(y)$ or $\sigma_i(y) \sim \sigma_i(x) + c$. Since these constraints correspond to regions that are open and dense, σ_i does actually exist. □

As already noticed, time progress is not guaranteed by the construction of $\mathcal{A}^{\mathcal{R}}_{\phi}$. However, this requirement is easily achieved by the CLTLoc formula $\mathbf{GF}\,(x = 0) \vee \mathbf{FG}\,(x > c(x))$ (where $c(x)$ is the biggest constant clock x is compared to), for all clocks $x \in V$. The same condition is considered in [3] to guarantee time progression for Timed Automata.

Finally, the main result of this section is a direct consequence of Lemma 2 and of well-known properties of CLTL and Timed Automata:

Theorem 1 *Satisfiability for CLTLoc is decidable.*

Proof Let ϕ be a CLTLoc formula. By Lemma 2, we can build automaton $\mathcal{A}^{\mathcal{R}}_{\phi}$, recognizing symbolic models of ϕ and which has an accepting run (i.e., it accepts a non-empty language) if, and only if, ϕ is satisfiable. Checking the emptiness of the language $\mathscr{L}(\mathcal{A}_{\phi})$ can be done by standard techniques, which look for cycles in the graph of $\mathcal{A}^{\mathcal{R}}_{\phi}$ passing through (at least) one accepting control state. The language of $\mathcal{A}^{\mathcal{R}}_{\phi}$ is not empty if there exists a path of $\mathcal{A}^{\mathcal{R}}_{\phi}$ of the form

$$(\Gamma_0, R_0) \dots (\Gamma_{l-1}, R_{l-1})(\Gamma_l, R_l) \dots (\Gamma_k, R_k)$$

where $\Gamma_k = \Gamma_l$, $R_k = R_l$ and all atoms belonging to F occur at least once in $(\Gamma_{l-1}, R_{l-1})(\Gamma_l, R_l) \dots (\Gamma_{k-1}, R_{k-1})$. The word recognized by the run is ultimately periodic, over the alphabet $SV(\phi) \times \mathcal{R}_{\phi}$ of the form:

$$(sv_0, R_0) \dots (sv_{l-1}, R_{l-1}) \left((sv_l, R_l) \dots (sv_{k-1}, R_{k-1})\right)^{\omega}.$$

Complexity

The satisfiability problem for CLTLoc is PSPACE-hard, as every LTL formula (whose satisfiability problem is PSPACE-complete) is also a CLTLoc formula. PSPACE-completeness of CLTLoc can be roved by applying arguments similar to those used in [3] to show that the transition relation of the automaton to be checked for emptiness is computable in PSPACE. Consider a CLTLoc formula ϕ. Let $|\phi|$ be the number of subformulae of ϕ, let N be the number of clock variables in ϕ, and let K be the biggest constant against which the clock variables of ϕ are compared. Since the number of clock regions is $O(N! \cdot K^N)$ [3], the number of states of $\mathcal{A}^{\mathcal{R}}_{\phi}$ is $O(2^{|\phi|} \cdot N! \cdot K^N)$. However, to check $\mathscr{L}(\mathcal{A}^{\mathcal{R}}_{\phi})$ for emptiness, we do not need to build the whole state space, but we can work on-the-fly by considering only a constant number of vertices at a time. Since the space needed to store a vertex of $\mathcal{A}^{\mathcal{R}}_{\phi}$, when using a binary encoding for K, is polynomial in $|\phi| \log(K)$, the algorithm for checking the emptiness of $\mathcal{A}^{\mathcal{R}}_{\phi}$ is in PSPACE.

In Section 6, we provide a way to solve in practice the satisfiability of CLTLoc through the method used in [11] to solve satisfiability of CLTL. The technique relies on encoding CLTLoc formulae into formulae of a decidable fragment of first-order logic, which can then be solved by off-the-shelf SMT solvers. The decision procedure hinges on finding a finite sequence of assignments to the clocks appearing in the CLTLoc formula ϕ, which satisfies ϕ and it is a witness of an ultimately periodic sequence of successive clock regions. Since the clock regions define a partition of the space of clock assignments, each clock assignment uniquely identifies a region; thus, an exhaustive definition of all the regions is not needed.

5 Comparing CLTLoc and pMITL on the pointwise semantics

In this section we show that, over timed words, CLTLoc and pMITL have the same expressive power. To this end, we devise two semantics-preserving transformations, from pMITL formulae to CLTLoc ones and vice-versa.

Before presenting the transformations, we remark the following.

Remark 1 Propositional letters in CLTLoc formulae can be replaced by constraints on clocks that exactly capture their semantics. More precisely, every propositional letter p can be replaced by a new clock c_p, assuming that $p \Leftrightarrow c_p = 0$ (hence, $\neg p \Leftrightarrow c_p > 0$). For example, formula $p \Rightarrow \mathbf{X}(\neg p \wedge \mathbf{X}(p))$ becomes $c_p = 0 \Rightarrow \mathbf{X}(c_p > 0 \wedge \mathbf{X}(c_p = 0))$.

As a consequence of Remark 1, in the following we will freely introduce propositional letters in CLTLoc, as they can be easily eliminated.

5.1 From pMITL to CLTLoc

To transform pMITL formulae into CLTLoc ones, we first remark that the following equivalences hold for pMITL formulae, where $\mathbf{U}$ is an abbreviation for $\mathbf{U}_{(0,\infty)}$ – note that $\mathbf{U}_{(0,\infty)}\phi \equiv \mathbf{U}_{[0,\infty)}\phi$ because we adopted the strict version of the until operator.

Lemma 3 *Let (π,τ) be a timed word and $0 < a \leqslant b$. Then, for any $i \geqslant 0$,*

(1) $(\pi,\tau), i \models \phi \mathbf{U}_{[a,b\rangle}\psi \Leftrightarrow (\pi,\tau), i \models \phi\mathbf{U}\psi \wedge \mathbf{G}_{(0,a)}(\phi \wedge \phi\mathbf{U}\psi) \wedge \mathbf{F}_{[a,b\rangle}(\psi)$

(2) $(\pi,\tau), i \models \phi \mathbf{U}_{(a,b\rangle}\psi \Leftrightarrow (\pi,\tau), i \models \phi\mathbf{U}\psi \wedge \mathbf{G}_{(0,a]}(\phi \wedge \phi\mathbf{U}\psi) \wedge \mathbf{F}_{(a,b\rangle}(\psi)$

(3) $(\pi,\tau), i \models \phi \mathbf{U}_{(0,b\rangle}\psi \Leftrightarrow (\pi,\tau), i \models \phi\mathbf{U}\psi \wedge \mathbf{F}_{(0,b\rangle}(\psi)$

When b is ∞, equivalences (1), (2) can be simplified, respectively, in $\phi \mathbf{U}_{[a,\infty)}\psi \equiv \phi\mathbf{U}\psi \wedge \mathbf{G}_{(0,a)}(\phi \wedge \phi\mathbf{U}\psi)$ and $\phi \mathbf{U}_{(a,\infty)}\psi \equiv \phi\mathbf{U}\psi \wedge \mathbf{G}_{(0,a]}(\phi \wedge \phi\mathbf{U}\psi)$.

Thanks to Lemma 3, we can focus only on temporal operators $\mathbf{U}$ and $\mathbf{F}_I$.

We also have the following result, which shows that a formula $\mathbf{F}_{\langle a,b\rangle}(\psi)$ must stay true for at least $b - a$ time units.

Lemma 4 *Consider the MITL formula $\mathbf{F}_{\langle a,b\rangle}(\psi)$. For any timed word (π,τ) there cannot be two positions $i < j$ such that $\tau(j) - \tau(i) < b - a$, $(\pi,\tau), i \not\models \mathbf{F}_{\langle a,b\rangle}(\psi)$, $(\pi,\tau), j \not\models \mathbf{F}_{\langle a,b\rangle}(\psi)$, and there is $i < k < j$ such that $(\pi,\tau), k \models \mathbf{F}_{\langle a,b\rangle}(\psi)$.*

$\overset{\theta}{\downarrow}$	if j is the current position, then there are two timestamps t, t' such that: (i) $\tau(j) \leqslant t < t' \leqslant \tau(j+1)$; (ii) there is a position k such that $(\pi,\tau), k \models \psi$ and $\tau(k) - t \in \langle a, b\rangle$; and (iii) there is no position k' such that $(\pi,\tau), k' \models \psi$ and $\tau(k') - t' \in \langle a, b\rangle$
$\overset{\theta}{\uparrow}$	if j is the current position, then there are two timestamps t', t such that: (i) $\tau(j) \leqslant t' < t \leqslant \tau(j+1)$; (ii) there is no position k' such that $(\pi,\tau), k' \models \psi$ and $\tau(k') - t' \in \langle a, b\rangle$; and (iii) there is a position k such that $(\pi,\tau), k \models \psi$ and $\tau(k) - t \in \langle a, b\rangle$

Table 4 CLTLoc additional predicates introduced for subformulae θ of the form $\mathbf{F}_{\langle a,b\rangle}(\psi)$.

Proof Assume that there are three positions $i < k < j$ in (π,τ) that violate the property. Then, there is position $k' > k$ such that $\tau(k') - \tau(k) \in \langle a, b\rangle$ and $(\pi,\tau), k' \models \psi$; in addition, for any position k'' such that $\tau(k'') \in \langle a + \tau(i), b + \tau(i)\rangle$ or $\tau(k'') \in \langle a + \tau(j), b + \tau(j)\rangle$ it is $(\pi,\tau), k'' \not\models \psi$. Since $\tau(j) - \tau(i) < b - a$, then $a + \tau(j) < b + \tau(i)$, hence for any position k'' such that $\tau(k'') \in \langle a + \tau(i), b + \tau(j)\rangle$ it is $(\pi,\tau), k'' \not\models \psi$. But $a + \tau(i) < \tau(k') < b + \tau(j)$, which leads to a contradiction. □

Corollary 1 *For any timed word (π,τ) and a pair of positions i, j such that $\tau(j) - \tau(i) \leqslant b$, between i (included) and j (excluded) there cannot be more than $2\left\lceil \frac{b}{b-a} \right\rceil$ instants k such that the value of $\mathbf{F}_{\langle a,b\rangle}(\psi)$ differs in k and $k+1$.*

Let us consider a pMITL formula ϕ such that AP is the set of its non-quantified propositional letters, and pAP is the set of quantified ones. The corresponding CLTLoc formula is built upon the set AP, plus a set of fresh propositional letters that correspond to set pAP, and which are then transformed into constraints over a corresponding set of clocks V_{pAP} as shown in Remark 1. Moreover, we introduce additional clocks and propositional letters which capture the semantics of the subformulae of ϕ.

For each subformula θ we introduce

- a propositional letter, $\boldsymbol{\theta}$, which represents the truth of θ in the current position;
- two clocks, z^0_θ and z^1_θ, which are reset at each position i in which θ holds, and its value differs from the one in $i-1$ or the one in $i+1$ (or both).

We also introduce two clocks, z^0_δ and z^1_δ, which measure the distance between two consecutive elements of the timed word; that is, at each instant either one or the other is reset, and they are reset in an alternate way. In addition, if θ is of the form $\mathbf{F}_{\langle a,b\rangle}(\psi)$, we also introduce:

- two additional propositional letters, $\overset{\theta}{\downarrow}$ and $\overset{\theta}{\uparrow}$, which are formalized in Table 4, and which represent the condition where θ would become false (resp. true) between the current and the next positions, if suitable timestamps were added to the timed word;
- $2d$ auxiliary clocks, where $d = 2\lceil \frac{b}{b-a} \rceil + 1$, $x^0_\theta, \hat{x}^0_\theta, \ldots x^{d-1}_\theta, \hat{x}^{d-1}_\theta$, such that x^j_θ is reset at the current position if, and only if, $\hat{x}^j_\theta$ is reset at the next position; the clocks are reset whenever $\overset{\theta}{\uparrow}$ or $\overset{\theta}{\downarrow}$ hold in the current position.

Formula (8) enforces that the occurrence of a change in the truth of subformula θ entails the reset of one of z^0_θ, z^1_θ, and that clock z^0_θ is reset in the origin.

$$z^0_\theta = 0 \wedge \mathbf{XG}\Big(\boldsymbol{\theta} \wedge (\neg\mathbf{X}(\boldsymbol{\theta}) \vee \neg\mathbf{Y}(\boldsymbol{\theta})) \quad \Leftrightarrow \quad z^0_\theta = 0 \vee z^1_\theta = 0\Big) \tag{8}$$

Let $a \in \mathbb{N}$ and value $\overline{a}_k$ be $(a \bmod k)$. The clocks associated with a subformula θ are alternatively reset. Hence, between any two resets of clock z^0_θ there must be a reset of clock z^1_θ, and vice-versa:

$$\mathbf{G}\left(\bigwedge_{i\in\{0,1\}}\left(z^i_\theta = 0 \Rightarrow \mathbf{X}\left((z_\theta^{\overline{(i+1)}_2} = 0)\mathbf{R}(z^i_\theta \neq 0)\right)\right)\right). \tag{9}$$

Formula 10 defines that clock $z^0_\delta = 0$ (resp. $z^1_\delta = 0$) is reset in all even (resp. odd) positions.

$$z^0_\delta = 0 \wedge \mathbf{G}\left(\bigwedge_{i\in\{0,1\}}\left(z^i_\delta = 0 \Rightarrow z_\delta^{\overline{(i+1)}_2} > 0 \wedge \mathbf{X}\left(z_\delta^{\overline{(i+1)}_2} = 0\right)\right)\right) \tag{10}$$

We define $\texttt{clocks}_\theta = \bigwedge_{j=8}^{10} j$.

Let θ be $\mathbf{F}_{(a,b)}\psi$. Its $2d$ associated clocks $x^j_\theta, \hat{x}^j_\theta$ $(0 \leqslant j \leqslant d-1)$ are reset in pairs, i.e., a reset of x^j_θ is immediately followed by a reset of $\hat{x}^j_\theta$, as defined by Formula 11. In addition, clock $\hat{x}^{d-1}_\theta$ is reset in the origin (where no other clock $\hat{x}^j_\theta$ is reset).

$$\mathbf{G}\left(\bigwedge_{j=0}^{d-1}\left(x^j_\theta = 0 \Leftrightarrow \mathbf{X}\left(\hat{x}^j_\theta = 0\right)\right)\right) \wedge \hat{x}^{d-1}_\theta = 0 \wedge \bigwedge_{j=0}^{d-2}\hat{x}^j_\theta > 0 \tag{11}$$

Auxiliary clocks are reset according to Formulae (12) and (13). More precisely, Formula (12) states that every time one of propositions $\overset{\theta}{\lceil\!\rightarrow}, \overset{\theta}{\rightarrow\!\rfloor}$ holds, an auxiliary clock is reset, and that no two auxiliary clocks are reset at the same time. In addition, clock x^0_θ is reset in the origin.

$$x^0_\theta = 0 \wedge \mathbf{XG}\left(\left(\overset{\theta}{\lceil\!\rightarrow} \vee \overset{\theta}{\rightarrow\!\rfloor} \Leftrightarrow \bigvee_{j=0}^{d-1} x^j_\theta = 0\right) \wedge \left(\bigwedge_{i=0}^{d-1}\bigwedge_{j=0, i\neq j}^{d-1} \neg(x^i_\theta = 0 \wedge x^j_\theta = 0)\right)\right) \tag{12}$$

Formula (13), instead, states that the resets of clocks x^i_θ are circularly ordered. That is, if $x^i_\theta = 0$, then, from the next position, all clocks are strictly greater than 0 until $x_\theta^{\overline{i+1}_d} = 0$ occurs.

$$\mathbf{G}\left(\bigwedge_{i=0}^{d-1}\left(x^i_\theta = 0 \Rightarrow \mathbf{X}\left((x_\theta^{\overline{i+1}_d} = 0)\mathbf{R}\bigwedge_{j\in[0,d-1],\, j\neq\overline{i+1}_d}(x^j_\theta > 0)\right)\right)\right) \tag{13}$$

We define $\texttt{auxclocks}_\theta = \bigwedge_{j=11}^{13}(j)$.

We now define a mapping m associating a pMITL formula with an equivalent CLTLoc formula, thus capturing the semantics of pMITL in CLTLoc.

The cases for Boolean connectives and the non-metric **U** operator are straightforward.

- $\theta = p \in AP$: we simply write p.
- $\theta = p \in pAP$: we introduce clock c_p and write $c_p = 0$ wherever p appears.
- $\theta = \neg\psi$: in this case it is $m(\theta) = \boldsymbol{\theta} \Leftrightarrow \neg\boldsymbol{\psi}$.

- $\theta = \gamma \wedge \psi$: we have: $m(\theta) = \boldsymbol{\theta} \Leftrightarrow \boldsymbol{\gamma} \wedge \boldsymbol{\psi}$.
- $\theta = \gamma \mathbf{U}_{(0,\infty)} \psi$: we need to take into account that the $\mathbf{U}$ operator in MITL is strict, whereas it is not in CLTLoc, hence we have the following:

$$m(\theta) = \boldsymbol{\theta} \Leftrightarrow \mathbf{X}(\boldsymbol{\gamma} \mathbf{U} \boldsymbol{\psi}) \,.$$

- $\theta = \mathbf{F}_{(a,b)}(\psi)$: The semantics of the $\mathbf{F}_I$ operator is captured by the following formulae. We focus on the case in which $I = (a, b)$, the others being similar.

Formula (14) captures the condition when subformula θ is true at a position i. It identifies two conditions: either none of the d auxiliary clocks $\hat{x}^j_\theta$ is reset in i (which means that $i > 0$, as by Formula (11) $\hat{x}^{d-1}_\theta$ is reset in the origin), or one of them is. In the former case, there are no time instants between $\tau(i-1)$ and $\tau(i)$ in which θ can change value, so the subformula in i has the same value that it had in $i-1$, hence θ holds in i if it held in $i-1$ (i.e., if $\mathbf{Y}(\boldsymbol{\theta})$ is true). In the latter case (which includes the origin), the clock $\hat{x}^j_\theta$ that is reset is used to count the time and to check that there is a future position in which $\boldsymbol{\psi}$ holds and such that $\hat{x}^j_\theta$ is in (a, b) there.

$$\boldsymbol{\theta} \Leftrightarrow \left(\neg \bigvee_{j=0}^{d-1} \hat{x}^j_\theta = 0 \wedge \mathbf{Y}(\boldsymbol{\theta}) \right) \vee \bigvee_{j=0}^{d-1} \left(\hat{x}^j_\theta = 0 \wedge \mathbf{X} \left(\hat{x}^j_\theta > 0 \, \mathbf{U} \left(\boldsymbol{\psi} \ \wedge a < \hat{x}^j_\theta < b \right) \right) \right) \tag{14}$$

Formulae (15) and (16) capture the conditions that define proposition $\overset{\theta}{\Gamma\!\!\rightarrow}$. More precisely, $\overset{\theta}{\Gamma\!\!\rightarrow}$ holds at position i when a clock x^j_θ is reset at i (hence $\hat{x}^j_\theta$ is reset at $i+1$), and there is a time instant $\tau(i) \leqslant t < \tau(i+1)$ – i.e., one that might not correspond to a timestamp in timed word (π, τ) – such that there is a future position k, with $\tau(k) = t + b$, in which $\boldsymbol{\psi}$ holds, hence in k it is $x^j_\theta = \tau(k) - \tau(i) \geqslant b$ and $\hat{x}^j_\theta = \tau(k) - \tau(i+1) < b$. In addition, at time t subformula θ would become true, which can only happen if it does not hold in t, because the endpoints of interval (a, b) are excluded. For this to occur, either there are no positions in the timed word with timestamps in interval $(\tau(k) - (b-a), \tau(k))$, or $\boldsymbol{\psi}$ does not hold in any of the positions k' such that $\tau(k') \in (\tau(k) - (b-a), \tau(k))$. In the former case $\tau(k) - \tau(k-1) \geqslant b - a$, hence one of z^0_δ, z^1_δ is $\geqslant b - a$. In the latter case, $\boldsymbol{\psi}$ does not hold in $k-1$ (i.e., $\neg\mathbf{Y}(\boldsymbol{\psi})$ holds in k), and the last time $\boldsymbol{\psi}$ switched from true to false was at least $b - a$ time units before $\tau(k)$, that is, the clock between z^0_ψ and z^1_ψ that is not reset in k is $\geqslant b - a$. In the formulae of this section we write $z_\delta \sim c$ as an abbreviation for $\bigvee_{i \in \{0,1\}} z^i_\delta \sim c$ (similarly for $z_\psi \sim c$).

$$\overset{\theta}{\Gamma\!\!\rightarrow} \Leftrightarrow \bigvee_{j=0}^{d-1} \left(\begin{array}{l} x^j_\theta = 0 \quad \wedge \\ \mathbf{X} \left(x^j_\theta > 0 \, \mathbf{U} \left(\left(\begin{array}{l} \boldsymbol{\psi} \quad \wedge \\ x^j_\theta \geqslant b \ \wedge \hat{x}^j_\theta < b \end{array} \right) \wedge \left(\begin{array}{l} z_\delta \geqslant b - a \quad \vee \\ \neg\mathbf{Y}(\boldsymbol{\psi}) \wedge z_\psi \geqslant b - a \end{array} \right) \right) \right) \end{array} \right) \tag{15}$$

Formula (16) defines a sufficient condition for one of the d auxiliary clocks x^j_θ, with $j \in [0, d-1]$ to be reset. More precisely, if $\boldsymbol{\psi}$ holds at a position k and there is no

position $k' < k$ such that $\tau(k) - \tau(k') < b - a$ and $\boldsymbol{\psi}$ holds in k', then θ became true at time instant $\tau(k) - b$, so there is a clock x^j_θ that in k has value $\geqslant b$.

$$\boldsymbol{\psi} \wedge \left(z_\delta \geqslant b - a \quad \vee \quad \neg \mathbf{Y}(\psi) \wedge z_\psi \geqslant b - a\right) \Rightarrow \bigvee_{j=0}^{d-1} \left(x^j_\theta \geqslant b \wedge \hat{x}^j_\theta < b\right) \tag{16}$$

Formulae (17)-(18) are the analogous of (15)-(16) for the case in which θ is (and becomes) false. More precisely, Formula (17) states that $\overset{\theta}{\downarrow}$ holds at position i when a clock x^j_θ is reset at i (hence $\hat{x}^j_\theta$ is reset at $i+1$), and there is a time instant $\tau(i) < t \leqslant \tau(i+1)$ such that there is a future position k, with $\tau(k) = t + a$, in which $\boldsymbol{\psi}$ holds, hence in k it is $x^j_\theta = \tau(k) - \tau(i) > a$ and $\hat{x}^j_\theta = \tau(k) - \tau(i+1) \leqslant a$. Moreover, at time instant t subformula θ would become false, which can only happen if it does not hold in t (again, because the endpoints of the interval are excluded). For this to occur, either there are no positions in the timed word with timestamps in interval $(\tau(k), \tau(k) + (b-a))$, or $\boldsymbol{\psi}$ does not hold in any position k' such that $\tau(k') \in (\tau(k), \tau(k) + (b-a))$, hence including $k+1$. In the former case $\tau(k+1) - \tau(k) \geqslant b - a$, hence one of z^0_δ, z^1_δ is $\geqslant b - a$ in $k+1$. In the latter case, a clock z^j_ψ is reset in k, and there is no $k' > k$ such that $\boldsymbol{\psi}$ holds in k', and $a < \tau(k') - t < b$, that is, $z^j_\psi = \tau(k') - \tau(k) < b - a$, as $\tau(k) = t + a$.

$$\overset{\theta}{\downarrow} \Leftrightarrow \bigvee_{j=0}^{d-1} \left(\begin{array}{l} x^j_\theta = 0 \quad \wedge \\ \mathbf{X}\left(x^j_\theta > 0 \mathbf{U} \left(\begin{array}{l} \boldsymbol{\psi} \quad \wedge \\ x^j_\theta > a \ \wedge \\ \hat{x}^j_\theta \leqslant a \end{array} \right) \wedge \left(\begin{array}{l} \mathbf{X}(z_\delta \geqslant b - a) \quad \vee \\ \bigvee_{i=0}^{1} \left(\begin{array}{l} z^i_\psi = 0 \quad \wedge \\ \neg \mathbf{X}\left(z^i_\psi > 0 \mathbf{U} \left(\begin{array}{l} \boldsymbol{\psi} \wedge \\ z^i_\psi < b - a \end{array} \right) \right) \end{array} \right) \end{array} \right) \right) \end{array} \right) \tag{17}$$

Formula (18) states a sufficient condition for the reset of one of the clocks x^j_θ which is similar to Formula (16).

$$\boldsymbol{\psi} \wedge \left(\begin{array}{l} \mathbf{X}(z_\delta \geqslant (b-a)) \quad \vee \\ \bigvee_{i \in \{0,1\}} z^i_\psi = 0 \wedge \neg \mathbf{X}\left(z^i_\psi > 0 \mathbf{U} \left(\boldsymbol{\psi} \wedge z^i_\psi < b - a \right) \right) \end{array} \right) \Rightarrow \bigvee_{j=0}^{d-1} \left(\begin{array}{l} x^j_\theta > a \ \wedge \\ \hat{x}^j_\theta \leqslant a \end{array} \right) \tag{18}$$

Formula $m(\theta)$ in this case is $\bigwedge_{j=14}^{18} j$.

Theorem 2 *Let ϕ be a pMITL formula. A timed word (π, τ) is a model for ϕ (i.e., $(\pi, \tau) \models \phi$) if, and only if, it is also model for the following CLTLoc formula (where $sub(\phi)$ is the set of subformulae of ϕ):*

$$\bigwedge_{\theta \in sub(\phi)} \mathbf{G}(m(\theta)) \wedge \mathtt{clocks}_\theta \wedge \bigwedge_{\theta \in sub(\phi), \theta = \mathbf{F}_{\langle a,b \rangle}(\psi)} \mathtt{auxclocks}_\theta. \tag{19}$$

Proof First of all, for any timed word (π, τ) that is an interpretation for ϕ, there is a CLTLoc interpretation (π, σ) such that $(\pi, \tau) = [(\pi, \sigma)]$ and for each $\theta \in sub(\phi)$ it is $(\pi, \sigma) \models \mathtt{clocks}_\theta$ and also $(\pi, \sigma) \models \mathtt{auxclocks}_\theta$ if $\theta = \mathbf{F}_{\langle a,b \rangle}(\psi)$. In fact, formulae $\mathtt{clocks}_\theta$ and $\mathtt{auxclocks}_\theta$ only impose an ordering in the reset of clocks, and such

an ordering is compatible with any distribution of the propositions in π, including the fresh ones introduced by the encoding such as $\overset{\theta}{\Rsh}$ and $\overset{\theta}{\downarrow}$.

We now show by induction that, for all $\theta \in sub(\phi)$, it is $(\pi,\tau),i \models m(\theta)$; in addition, $(\pi,\tau),i \models \theta$ if, and only if, $(\pi,\tau),i \models \boldsymbol{\theta}$ (i.e., a subforumula θ holds at position i if, and only if, the corresponding proposition $\boldsymbol{\theta}$ introduced by the translation is true at i).

The cases of propositional letters and operators are trivial, and so is the case of the non-metric **U** temporal operator.

If $\theta = \mathbf{F}_{(a,b)}(\psi)$ we first show that formulae (15)-(18) correctly capture the intended meaning of $\overset{\theta}{\Rsh}$ and $\overset{\theta}{\downarrow}$. More precisely, $\overset{\theta}{\Rsh}$ and/or $\overset{\theta}{\downarrow}$ hold at position i if the value of θ would not stay constant between timestamps $\tau(i)$ and $\tau(i+1)$ if suitable positions were added to the timed word between i and $i+1$.

In fact, $\overset{\theta}{\Rsh}$ holds if there are at least two timestamps $t' < t$ in interval $[\tau(i),\tau(i+1)]$ such that the conditions for θ being true are not met in t', but they are in t. For this to occur, a sufficient and necessary condition is that there is $\tau(i) \leqslant t' < \tau(i+1)$ in which the conditions are met for θ to become true, i.e., it would be true right after t', but not in t' itself. This is equivalent to having a position k such that $\tau(k) = t' + b$ where ψ holds, and it does not hold in interval $(\tau(k)-(b-a),\tau(k))$. For ψ not to hold in $(\tau(k)-(b-a),\tau(k))$ there are two cases. In the first case, there are no positions k' such that $\tau(k') \in (\tau(k)-(b-a),\tau(k))$. This occurs when $\tau(k)-\tau(k-1) \geqslant b-a$, i.e., when $z_\delta \geqslant b-a$ in k. In the second case, ψ does not hold in $k-1$ (i.e., $\neg\mathbf{Y}(\psi)$ in k), and the last time ψ was true was at least $b-a$ instants ago, i.e., $z_\psi \geqslant b-a$ in k These two cases correspond to the antecedent of Formula (16), which entails that in k one of the x^j_θ clocks is $\geqslant b$, whereas $\hat{x}^j_\theta < b$. Hence, there is i' such that $\tau(k)-\tau(i') \geqslant b$, $\tau(k)-\tau(i'+1) < b$. Since $t'+b = \tau(k)$, it is $\tau(i) \leqslant t' < \tau(i+1)$, $\tau(k)-\tau(i) \geqslant b$, and $\tau(k)-\tau(i+1) < b$, so i is precisely i' captured by Formula (16), and Formula (15) holds in i. Finally, clock x^j_θ reset in i cannot be reset again before or in k, since, by Corollary 1, between i (included) and k (excluded) there can be at most $d-1$ positions in which the conditions for $\overset{\theta}{\Rsh}$ or for $\overset{\theta}{\downarrow}$ hold, and there are d clocks x^j_θ.
Conversely, if it does not exist a $t' \in [\tau(i),\tau(i+1))$ in which the conditions for θ to become true hold, then there is no position k such that $\tau(k) = t' + b$ and the conditions of the antecedent of Formula (16) are met. In addition, the right-hand side of Formula (15) does not hold in i, and $\overset{\theta}{\Rsh}$ is false there.

The case for $\overset{\theta}{\downarrow}$ is similar. It holds if there are at least two timestamps $t < t'$ in $[\tau(i),\tau(i+1)]$ such that the conditions for θ being true are met in t, but not in t'. For this to occur, a sufficient and necessary condition is that there is $\tau(i) < t' \leqslant \tau(i+1)$ in which the conditions are met for θ to become false, i.e., it would be true right before t', but not in t' itself. This is equivalent to having a position k such that $\tau(k) = t' + a$ where ψ holds, and it does not hold in interval $(\tau(k),\tau(k)+(b-a))$. For ψ not to hold in $(\tau(k),\tau(k)+(b-a))$ there are two cases. In the first case, there are no positions k' such that $\tau(k') \in (\tau(k),\tau(k)+(b-a))$. This occurs when $\tau(k+1)-\tau(k) \geqslant b-a$, i.e., when $\mathbf{X}(z_\delta \geqslant b-a)$ in k. In the second case, ψ becomes false (i.e., a clock z^i_ψ is reset in k), and it does not hold until z^i_ψ becomes $\geqslant b-a$. These two cases correspond to the antecedent of Formula (18), which entails that in k one of the x^j_θ clocks is $> a$, whereas $\hat{x}^j_\theta \leqslant a$. Hence, there is i' such that

$\tau(k) - \tau(i') > a$, $\tau(k) - \tau(i'+1) \leqslant a$. Since $t' + a = \tau(k)$, it is $\tau(i) < t' \leqslant \tau(i+1)$, $\tau(k) - \tau(i) > a$, and $\tau(k) - \tau(i+1) \leqslant a$, so i is precisely i' captured by Formula (18), and Formula (17) holds in i. Finally, clock x^j_θ reset in i cannot be reset again before or in k, by Corollary 1 as before.
Conversely, if it does not exist a $t' \in (\tau(i), \tau(i+1)]$ in which the conditions for θ to become false hold, then there is no position k such that $\tau(k) = t' + a$ and the conditions of the antecedent of Formula (18) are met. In addition, the right-hand side of Formula (17) does not hold in i, and $\overset{\theta}{\neg\!\downarrow}$ is false there.

Finally, we show that Formula (14) holds. For θ to be true in a position i, we have three cases: either $i = 0$, hence $\hat{x}^{d-1}_\theta$ is reset in i, by Formula (11); or $i > 1$ and nothing changes between positions $i-1$ and i; or there are two timestamps in $[\tau(i-1), \tau(i)]$ in which θ can change value, hence a clock $\hat{x}^j_\theta$ is reset in i. In the second case no clock $\hat{x}^j_\theta$ is reset in i (i.e., $\neg \bigvee_{j=0}^{d-1} \hat{x}^j_\theta = 0$, so i cannot be the origin), and θ holds in $i-1$, i.e., $\mathbf{Y}(\boldsymbol{\theta})$ in i. In the first and third cases a clock $\hat{x}^j_\theta$ is reset in i, and there is a position k such that $\boldsymbol{\psi}$ holds in k, and $\tau(k) - \tau(i) \in (a, b)$, i.e., $a < \hat{x}^j_\theta < b$ there. Again, $\hat{x}^j_\theta$ cannot be reset between i and k by Corollary 1. □

Before concluding this section we remark that the encoding presented above can also be used to realize a decision procedure for the satisfiability of MITL formulae. For this, it is enough, given a MITL formula ϕ, to build the corresponding CLTLoc formula (19), then solve it using the tool presented in Section 8.

5.2 From CLTLoc to pMITL

To show the equivalence between pMITL and CLTLoc, in this section we build a model-preserving transformation from CLTLoc to pMITL formulae. Since, as shown in [15], the future-only fragment of CLTLoc has the same expressiveness as the full language, including past operators, here we focus on the former.

Let us consider a future-only CLTLoc formula ϕ, with a set V of clocks appearing in it. Without loss of generality, we assume that all clocks of V are reset in the origin. To capture the behavior of the clocks of V, we introduce the following quantified propositions of set pAP:

- For each $x \in V$ we introduce a propositional letter r_x, which holds when clock x is reset (i.e., when condition $x = 0$ holds).
- For each constraint $x \sim c$ (with $c > 0$), independent of the nature of relation $\sim$, we introduce two propositional letters $p_{x \leqslant c}$ and $p_{x<c}$, which capture, respectively, the conditions $x \leqslant c$ and $x < c$.

The behavior of the new propositional letters is defined by Formula (20), which captures when $p_{x \leqslant c}$ and $p_{x<c}$ holds with respect to the truth of r_c. In the formula we introduce abbreviation $\mathbf{G}^i_I(\phi) = \phi \wedge \mathbf{G}_I(\phi)$ (resp. $\mathbf{F}^i_I(\phi) = \phi \vee \mathbf{F}_I(\phi)$), where i stands for "included", which requires (resp. allows) ϕ to hold in the current instant (hence $\mathbf{G}^i(\phi) = \mathbf{G}^i_{(0,\infty)}(\phi)$).

$$\mathbf{G}^i \left(r_x \Rightarrow \left(\begin{array}{l} \mathbf{G}^i_{(0,c]}(p_{x \leqslant c}) \wedge \mathbf{F}^i_{(0,c]} \left(\mathbf{G}_{(0,\infty)}(\neg p_{x \leqslant c}) \vee (\neg p_{x \leqslant c}) \mathbf{U}_{(0,\infty)} r_x \right) \ \wedge \\ \mathbf{G}^i_{(0,c)}(p_{x<c}) \wedge \mathbf{F}^i_{(0,c)} \left(\mathbf{G}_{(0,\infty)}(\neg p_{x<c}) \vee (\neg p_{x<c}) \mathbf{U}_{(0,\infty)} r_x \right) \end{array} \right) \right). \tag{20}$$

Temporal operators appearing in ϕ, instead, are transformed as follows:

$$\mathbf{X}(\phi) \longmapsto \bot \mathbf{U}_{(0,\infty)} \phi \tag{21}$$

$$\phi \mathbf{U} \psi \longmapsto \psi \vee (\phi \wedge \phi \mathbf{U}_{(0,\infty)} \psi). \tag{22}$$

Theorem 3 *For any CLTLoc formula ϕ there is an equivalent pMITL formula ϕ'.*

Proof Formula ϕ' is built according to the rules outlined above: temporal operators are replaced as in (21) and (22) and every constraint of the form $x \sim c$ is replaced by a suitable formula over propositions r_x, $p_{x \leqslant c}$ and $p_{x<c}$ (for example, $x = 0$ is replaced by r_x, and $x = c$ is replaced by $p_{x \leqslant c} \wedge \neg p_{x<c}$). In addition, the following conjunct is included in ϕ': $\bigwedge_{x \sim c \text{ in } \phi} (20)$.

To show that ϕ' has the same models as ϕ we prove that propositions r_x, $p_{x \leqslant c}$ and $p_{x<c}$ correctly capture constraints $x = 0$, $x \leqslant c$ and $x < c$ (the case for $\mathbf{X}$ and $\mathbf{U}$ is trivial). The case for $x = 0$ is trivial, as it need not obey any constraint, except formula ϕ itself. To show that $p_{x \leqslant c}$ holds if, and only if, $x \leqslant c$ does, consider the case where $x \leqslant c$ holds at position i of the timed word; then, there must be a $k \leqslant i$ such that $x = 0$ in k and $\tau(i) - \tau(k) \leqslant c$. As shown above, this corresponds to r_x holding in k, so the antecedent of Formula (20) holds there. As a consequence, $p_{x \leqslant c}$ holds in all positions $k' \geqslant k$ such that $\tau(k') - \tau(k) \leqslant c$, hence including i. Suppose instead that $x \leqslant c$ does not hold in i. Since we assume that all clocks are reset in the origin, there must be a $k \leqslant i$ such that $x = 0$ in k and $\tau(i) - \tau(k) > c$. Again, the antecedent of Formula (20) holds in k. By Formula (20) there must be a position $k \leqslant k' < i$ such that $\mathbf{G}_{(0,\infty)}(\neg p_{x \leqslant c}) \vee (\neg p_{x \leqslant c}) \mathbf{U}_{(0,\infty)} r_x$ holds there, and since there is no reset of clock x in $(\tau(k), \tau(i)]$, $\neg p_{x \leqslant c}$ must hold in i, i.e., $p_{x \leqslant c}$ does not hold there. The case for $p_{x<c}$ is similar. □

6 Solving CLTL-over-clocks satisfiability

In this section, we outline a decision procedure for the satisfiability problem of CLTLoc, by means of a SMT-based technique instead of automata. This approach is along the lines of previous works [8] and [9], where a complete procedure, called k-bounded satisfiability, was used to solve CLTL satisfiability by means of a polynomial reduction to a SMT problem. CLTL satisfiability can be decided by considering finite amount of k-bounded satisfiability tests, for increasing values of k. To deal with variables that behave like clocks, the method developed in [9] is here extended to represent time progress.

Given a CLTL formula ϕ, we say that ϕ is k-bounded satisfiable if there exists an ultimately periodic sequence of symbolic valuations $sv_0, \ldots, sv_{l-1}(sv_l \ldots sv_{k-1})^\omega$, which is a symbolic model of ϕ and such that there is a partial assignment of values to all the variables occurring in ϕ only for a finite number of positions in time, from 0 to $k+1$. In other words, in k-bounded satisfiability we look for a finite sequence of symbolic valuations $sv_0, \ldots, sv_{l-1}(sv_l \ldots sv_{k-1})sv_k$, where $sv_k = sv_l$, which admits a k-bounded arithmetical model and that is representative of an infinite symbolic model for ϕ of the form $sv_0, \ldots, sv_{l-1}(sv_l \ldots sv_{k-1})^\omega$. While PSPACE-complete, in practice k-bounded satisfiability can be quite efficient, at least when the value of k is small enough to perform the check: checking k-bounded satisfiability is then

equivalent to solve a few SMT problems in P. Obviously, the upper bound for k is in general exponential in the size of the formula.

In [9] we show how to solve k-bounded satisfiability for CLTL formulae over a class of arithmetical constraints that include the family of clock constraints used in Sect. 2. The k-bounded satisfiability problem is solved through a polynomial-time reduction to the satisfibility problem of a formula in the theory of Equality and Uninterpreted Functions combined with Linear Integers/Reals Arithmetic. The combination of the two theories is decidable and its decision procedure is implemented by many SMT-solvers. The reduction of [9] has been implemented in the `ae`^{2}`zot` plugin of the Zot tool [2]. Therefore, an instance of the k-bounded satisfiability problem for CLTL formulae has the complexity of the underlying SMT problem, which depends on the arithmetic theory required. In our case, since clocks are in $\mathbb{R}$, we solve SMT problems in QF-EUF $\cup$ LRA, whose complexity is P. A peculiarity of the SMT-based approach is that, if the set of symbolic valuations partitions the space D^λ (with D the domain of the variables in V and λ the size of a symbolic valuation, i.e., the number of temporal terms in it), then a sequence of valuations uniquely induces a sequence of symbolic valuations. By solving the k-bounded satisfiability problem for a formula ϕ we obtain, from the model of the QF-EUF formula, a finite prefix σ_k of some infinite model σ that satisfies the formula. Prefix σ_k is a sequence of valuations, complying with the constraints in the formula, that induces an ultimately periodic symbolic model for ϕ. Hence, unlike automata-based techniques, our approach does not require the explicit construction of the set $SV(\phi)$ because symbolic valuations of the model are deduced from σ_k. We exploit this to avoid building the set of clock regions induced by CLTLoc formulae.

To solve the satisfiability of CLTLoc we still use k-bounded satisfiability to look for ultimately periodic models, but we extend the method in order to represent clock regions and time progression. Representing clock regions is quite straightforward and exploits the fact that regions partition the space of all possible clock valuations. In other words, a clock valuation identifies a clock region, so it is not necessary to precompute the set of all clock regions from the formula. The only requirement to be enforced is the periodicity of the sequence of clock regions corresponding to clock valuations. If one is looking for a model of length k, the sequence of clock regions is of the form: $R_0 \dots R_{l-1}(R_l \dots R_{k-1})^\omega$, which is obtained from a finite sequence $R_0 \dots R_{l-1}(R_l \dots R_{k-1})R_k$ with the periodicity constraint $R_l = R_k$. The QF-EUF encoding of CLTL formulae is defined to enforce periodicity of all atomic formulae (atomic propositions and clock constraints) between positions k and l. For instance, given two clocks x, y, if $x = y$ holds at position k (i.e., $x(k) = y(k)$) then, by the periodicity constraints, it must also hold at position l: $x(k) = y(k) \Leftrightarrow x(l) = y(l)$. To obtain a periodic sequence of regions we provide the solver with all the clock constraints of set $ac(\phi)$ (see Section 4) which may occur in the definition of regions in $\mathcal{R}_\phi$ (but not all the regions). Observe that, for any variable x, $x(i)$ is the value $\sigma_k(i, x)$ of variable x at position i of the k-bounded model whereas, for instance, $\mathrm{X}x(i)$ is the value $\sigma_k(i + 1, x)$.

Let ϕ be a CLTL formula, V be the set of clocks appearing in ϕ, $x, y \in V$ and $c(x)$ be the maximum constant with which clock x is compared in ϕ. We recall the following definitions from Section 4. Set $ac(x)$ is the set of all clock constraints between x and constant $c(x)$ and set $ac(x, y)$ is the set of all clock constraints between x and y. Set $ac(\phi)$ is the set of all clock constraints induced by ϕ.

As in [9], we introduce a variable $\boldsymbol{loop} : \mathbb{N} \to \mathbb{N}$ to represent position l and to encode arithmetical terms α (with domain $\mathbb{R}$) appearing in formula ϕ, i.e., those of the set $terms(\phi)$, we introduce an *arithmetic formula function* $\boldsymbol{\alpha} : \mathbb{N} \to \mathbb{R}$; e.g., if x is a clock in ϕ then $\boldsymbol{x}$ is the function associated with it.

Let $\theta \in ac(\mathcal{R}_\phi)$. We write $\boldsymbol{\theta}(i)$ to indicate formula θ where all clocks are expressed by their associated function and with parameter i; e.g., if θ is $x \sim y$ then $\boldsymbol{\theta}(i)$ is $\boldsymbol{x}(i) \sim \boldsymbol{y}(i)$. We indicate with $Per(ac(\mathcal{R}_\phi))$ the following QF-EUF formula

$$\bigwedge_{\theta \in ac(\phi)} \boldsymbol{\theta}(k) \Leftrightarrow \boldsymbol{\theta}(\boldsymbol{loop})$$

that constrains regions at positions l ($\boldsymbol{loop}$) and k to be the same.

Time elapsing is represented by function $\boldsymbol{\delta} : \mathbb{N} \to \mathbb{R}$ that forces all clocks in ϕ to progress by the same amount of time between any two positions of the finite model. Strict monotonicity of time is guaranteed by letting function $\boldsymbol{\delta}$ to be strictly positive in $\{0, \ldots, k\}$. Finally, since variables $\boldsymbol{x}$ are defined over $\mathbb{R}$ but x are clocks, we enforce all clocks to be nonnegative at 0. We indicate with $Adv(C_\phi)$ the following QF-EUF formula:

$$\bigwedge_{i=0}^{k} \left(\boldsymbol{\delta}(i) > 0 \quad \wedge \bigwedge_{x \in C(\phi)} (\boldsymbol{x}(i+1) = \boldsymbol{x}(i) + \boldsymbol{\delta}(i) \vee \boldsymbol{x}(i+1) = 0) \right).$$

Formula $\bigwedge_{x \in C(\phi)} \boldsymbol{x}(0) \geqslant 0$ is added to define the initial value of clocks.

We solve the satisfiability problem of CLTLoc formula ϕ by feeding the SMT solver the set of constraints

$$|\phi'|_k \cup Per(ac(\mathcal{R}_\phi)) \cup Adv(C_{\phi'}),$$

where $|\phi'|_k$ is the bounded representation of ϕ' described in [9].

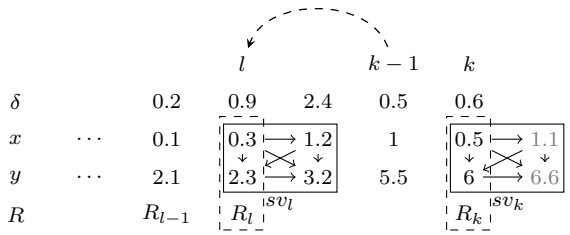

Fig. 1 A (portion of) bounded model satisfying infinitely often formula $(x < \mathrm{X}y)\mathbf{U}(x = 1)$. Numbers colored in grey are associated with terms $\mathrm{X}x$ and $\mathrm{X}y$ at position k.

Figure 1 shows a portion of a model satisfying infinitely often the CLTLoc formula $\phi = (x < \mathrm{X}y)\mathbf{U}(x = 1)$. The prefix would be trivial, then only the periodic part is depicted. The depth $|\phi|$ (i.e., the maximum depth of variables appearing in ϕ, see Section 2) is 1, because of term $\mathrm{X}y$, constant $c_{\max}$ is 1 and set $ac(\phi)$ of clock constraints induced by the formula (see Section 4) is $\{x = 0, 0 < x < 1, x = 1, 1 < x, y = 0, 0 < y < 1, y = 1, 1 < y, x < y, x = y, y < x\}$. Set $SV(\phi)$ is not defined here, for the sake of space. Solid rectangles in Figure 1 represent symbolic valuations

sv_l and sv_k where arrows $a \rightarrow b$ represent constraints of the form $a < b$. They are defined as $\{x < \mathrm{X}x, y < \mathrm{X}y, x < y, \mathrm{X}x < \mathrm{X}y, x < \mathrm{X}y, \mathrm{X} < y, x < 1, 1 < y, 1 < \mathrm{X}x, 1 < \mathrm{X}y\}$. Let us denote this set as S. $Per(ac(\phi))$ enforces regions R_l and R_k (dashed rectangles) to be equal through the following logical equivalences, for all θ in S: $\{x(l) = 0 \Leftrightarrow x(k) = 0, 0 < x(l) < 1 \Leftrightarrow 0 < x(k) < 1, x(l) = 1 \Leftrightarrow x(k) = 1, \ldots y(l) < x(l) \Leftrightarrow y(k) < x(k)\}$. In Figure 1, $R_l = \{x(l) < y(l), 0 < x(l) < 1, 1 < y(l)\}$ and $R_k = \{x(k) < y(k), 0 < x(k) < 1, 1 < y(k)\}$. The complete definition of $|\phi|_k$ can be found in [9].

For all formulae θ occurring in the definition of symbolic valuations in $SV(\phi)$, formula $\theta(l) \Leftrightarrow \theta(k)$ enforces sv_l to be equivalent to sv_k. Formula $1 \leqslant l \leqslant k-1$ imposes that position l is a valid position. The semantics of formula ϕ is achieved by the fixpoint definition of $\mathbf{U}$ over all positions between 0 and k. As in [9], formula ϕ is associated with a formula predicate over $\mathbb{N}$ and its translation is $\bigwedge_{i=0}^{k-1} \phi(i) \Leftrightarrow (x(i) = 1) \vee (\mathrm{X}y(i) > x(i) \wedge \phi(i+1))$. To represent correctly the semantics of temporal formulae, all the subformulae ϕ' of a CLTLoc formula must have the same truth value at positions k and l. This is enforced by imposing that $\phi'(l) \Leftrightarrow \phi'(k)$, for all ϕ' occurring in ϕ. The eventuality of $(x = 1)$, i.e., a positive occurrence of subformula $(x = 1)$ in the loop, is guaranteed by formula $\phi(k) \Rightarrow l \leqslant i_\phi \leqslant k \wedge (x_{i_\phi} = 1)$ that imposes subformula $(x = 1)$ to be satisfied at position i_ϕ, between l and k, when formula ϕ holds at k. In Figure 1, value $x_{k-1} = 1$ satisfies the eventuality for ϕ and for all the positions from l to k it holds that $x < \mathrm{X}y$. Hence, ϕ is satisfied infinitely often in the loop.

7 Encoding Metric Temporal Logics over the continuous semantics

We exploit the decision procedure for CLTLoc outlined in Sect. 6 to define mechanisms for deciding various metric temporal logics over continuous time. In [13], [14] and [12], we have defined several satisfiability-preserving reductions from metric temporal logics to CLTLoc; hence, satisfiability of formulae of these former logics can be determined by solving the corresponding problem for CLTLoc. In particular, MITL, $\mathrm{MITL}_{(0,\infty)}$, $\mathrm{MITL}_{(0,\infty)}$ with counting modalities [29], and their extensions with past operators are the logics we have targeted so far.

We now briefly show how to encode MITL and $\mathrm{MITL}_{(0,\infty)}$ (hence, QTL) formulae into CLTLoc ones, by providing some highlights of the reduction in a special case.

In general, in [4] it is shown that a signal can be seen as an infinite sequence of adjacent non-empty intervals starting from the origin. Each interval is a convex set of points over $\mathbb{R}$ that defines exactly the set of atomic propositions that are true in all the time instants in it. In our translation, we represent the truth of MITL (or $\mathrm{MITL}_{(0,\infty)}$) formulae, over the sequence of time intervals, by CLTLoc formulae that capture their semantics. We assume that signals are finitely variable. For these signals, time can be partitioned in a countable set of adjacent intervals such that the value of every subformula of ϕ is constant in each interval. In the case of MITL, we also restrict signals to intervals that are left-closed and right-open (l.c.r.o. signals, ). This allowed us to devise a simpler translation and, however, it can be relaxed. For instance, under the l.c.r.o assumption, a formula can not hold in isolated points but if it holds at time instant t then it

holds over a non-empty interval $[t, t+\varepsilon)$, for some $\varepsilon > 0$. Then, the case of isolated points is not considered for l.c.r.o signals.

For each subformula θ of ϕ, we introduce a CLTLoc predicate $\overleftarrow{\theta}$ that represents the value of θ in the intervals and the following abbreviations:

$$\underleftarrow{\xi} = \neg\, \overleftarrow{\xi} \qquad \lrcorner_\xi = \neg\mathbf{Y}(\overleftarrow{\xi}) \wedge\, \overleftarrow{\xi} \qquad \llcorner_\xi = \neg\mathbf{Y}(\underleftarrow{\xi}) \wedge\, \underleftarrow{\xi}$$

where $\lrcorner_\xi$, for example, captures the situation in which ξ changes value from false to true, with the formula being true in the current interval.

For simplicity, we focus our attention on temporal operators $\mathbf{F}_{(0,b]}(\psi)$ and $\mathbf{P}_{[0,b)}(\psi)$. We remark that it can be shown that, if ψ holds only in l.c.r.o. intervals, so do $\mathbf{F}_{(0,b]}(\psi)$ and $\mathbf{P}_{[0,b)}(\psi)$ (the same does not hold, for example, for $\mathbf{F}_{(0,b)}(\psi)$). For each subformula θ of ϕ, we introduce two clocks, z^0_θ and z^1_θ, which measure the time from the last change point (either $\lrcorner_\theta$ or $\llcorner_\theta$, so we have $\lrcorner_\theta \vee \llcorner_\theta \Leftrightarrow z^0_\theta = 0 \vee z^1_\theta = 0$), and whose resets alternate. Our translation defines the (sufficient and necessary) conditions causing events $\lrcorner_\theta$ and $\llcorner_\theta$ to occur, for all θ.

Formula (23), then, captures the condition in which formula $\theta = \mathbf{F}_{(0,b]}(\psi)$ becomes false: in this case, ψ must become false, and it cannot become true again for b instants (i.e., ψ cannot become true again before its associated clock that is reset when ψ becomes false hits b).

$$\llcorner_\theta \Leftrightarrow \llcorner_\psi \wedge \lrcorner_\psi \mathbf{R} \neg \left(\lrcorner_\psi \wedge \bigwedge_{i \in \{0,1\}} z^i_\psi \leqslant b \right) \tag{23}$$

The case for θ becoming true is not shown for brevity.

Consider the case $\theta = \mathbf{P}_{[0,b)}(\psi)$. Formula (24) captures the condition in which θ becomes true. This occurs when ψ becomes true and either the current instant is the origin (O is an abbreviation for $\neg\mathbf{Y}(\top)$), or ψ has never become true since the origin, or the last time ψ changed value (necessarily from false to true), this occurred more than b instants ago (i.e., the clock associated with ψ that is not reset now is $\geqslant b$).

$$\lrcorner_\theta \Leftrightarrow\ \lrcorner_\psi \wedge \left(O \vee \mathbf{Y}\left(\neg\lrcorner_\psi \mathbf{S}\ (O \wedge\, \underleftarrow{\psi}) \right) \vee \bigvee_{i \in \{0,1\}} z^i_\psi \geqslant b \right) \tag{24}$$

To conclude this section, we provide an example of MITL formula over two l.c.r.o. signals, whose model is intrinsically aperiodic in the values of the delays between changepoints. The existence of formulae admitting only aperiodic models shows that, in the decision procedure of Sect. 6, the periodicity must be enforced on the set of constraints defining regions, but not on the actual values of the clocks, nor on the time differences δ; therefore, the encoding of a CLTLoc formula ϕ' cannot include constraints of the form $\delta(k) = \delta(m)$ and $x(k) = x(m)$, for some $k > m \geqslant 0$. In other words, there are aperiodic models that do not admit a periodic sequence of time increments $\delta(0)\delta(1)\ldots(\delta_(m)\ldots\delta(k-1)^\omega$, even if the sequence of clock regions is periodic.

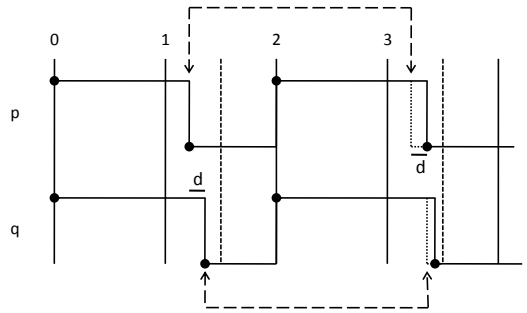

Fig. 2 Aperiodic model for MITL formula of Example 2

Example 2

Consider the behavior of two Boolean signals p, q, depicted in Figure 2. Signal p holds in $[2k, 2k+1+\varepsilon)$, for all k, and it is false elsewhere, as formalized by the following MITL formulae (recall that $\mathbf{G}^i_I(\phi) = \phi \wedge \mathbf{G}_I(\phi)$):

$$\mathbf{G}_{\langle 0,1]}p \wedge \mathbf{G}^i(\mathbf{G}_{\langle 0,1]}p \Rightarrow \mathbf{G}_{\langle 2,3]}p) \tag{25}$$

$$\mathbf{G}_{\langle 0,1]}q \wedge \mathbf{G}^i(\mathbf{G}_{\langle 0,1]}q \Rightarrow \mathbf{G}_{\langle 2,3]}q). \tag{26}$$

Both signals p and q hold over intervals longer than one time unit, because of the l.c.r.o. assumption. In addition, we require that q is at least as long as p by formula $\mathbf{G}^i(p \Rightarrow q)$. Formulae (25)-(26) above may, in general, admit periodic models; therefore, we have to restrict the set only to aperiodic models. This may be achieved by enforcing that, over intervals of the form $[2k+1, 2k+2]$, with $k \geqslant 0$, signal q is strictly longer than p, while over intervals $[2k+1.5, 2k+2)$ both p and q are false, as required by the following two formulae[1]:

$$\mathbf{G}^i(\mathbf{G}_{\langle 0,1]}(p \wedge q) \Rightarrow \mathbf{F}_{\langle 1,2]}((\neg p \wedge q)\mathbf{U}^i(\mathbf{G}_{\langle 0,0.5]}\neg q)) \tag{27}$$

$$\mathbf{G}^i(\neg p \wedge q \Rightarrow \mathbf{G}_{\langle 1,2]}p) \tag{28}$$

Signals p and q become false before each time instant $2k+1.5$ by requiring that $\neg p \wedge q$ occurs until both p and q are false over an interval of length 0.5. Let $t^p_k \in (2k+1, 2k+1.5)$ be the instant where p becomes false, and $t^q_k = t^p_k + \delta(k) \in (2k+1, 2k+1.5)$ the one where q becomes false; let $\delta_k = t^p_k - t^q_k$ be the length of the interval where p does not occur while q does. Formula (28) lengthens the duration of p of $\delta(k)$ time units over the next interval starting at $2k$: p holds in $[t^p_k+1, t^p_k+2+\delta(k))$. The series of values δ_k is strictly monotonic decreasing, because each value is arbitrarily strictly less than the previous one, i.e., $\delta(k) > \delta(k+1)$, for all $k \geqslant 0$. Therefore, the sequence of $\delta(i)$ is not periodic, although the sequence of clock regions induced by the clocks in the CLTLoc formula corresponding to the formulae above is.

[1] With slight abuse, we use rational bound 0.5; as customary, a formula with only integer bounds can be obtained by doubling all constants appearing in the formula.

8 Implementation and Experimental Results

The decision procedure of Sect. 6 for CLTLoc is implemented in a plugin, called `ae`^{2}`zot`, of our Zot toolkit [2], whereas the reductions outlined in Sect. 7 are implemented in the `qtlsolver` tool, available from [1]. The tool translates MITL and MITL$_{(0,\infty)}$ into CLTLoc, which can be checked for satisfiability by `ae`^{2}`zot`.

The resulting toolkit has a 3-layered structure, where CLTLoc is the intermediate layer between SMT-solvers and various temporal formalisms that can be reduced to CLTLoc. This not only supports (bounded) satisfiability verification of different languages, but it also allows the expression of different degrees of abstraction. For instance, MITL abstracts away the notion of clocks, inherently encompassed within temporal modalities, which are instead explicit in CLTLoc (as witnessed by the example of the timed lamp in Sect. 2) and available to a user, e.g., to express or verify properties where clocks are very convenient. In fact, preliminary experimental results point out that the time required to solve CLTLoc may be significantly smaller than the one needed for more abstract classes of languages, such as MITL. This gap is caused by the "effort" required to capture the semantics of temporal modalities, which, on the other hand, allow for more concise and manageable high-level specifications. One can then take advantage of the layered structure, which allows the resolution of a formula to be compliant also with constraints imposed at lower layers, for instance by adding at the CLTLoc layer some extra formula limiting the set of valid models (e.g., by discarding certain edges of some events or by adding particular timing requirements). Also the third layer (the SMT solver) may be used to add further constraints, e.g., to force the occurrence of a proposition or of a certain clock value at a specific discrete position of the finite model.

The current implementation of `qtlsolver` supports various reductions. More precisely, it realizes the MITL-to-CLTLoc translation tailored to l.c.r.o. signals, as highlighted in Sect. 7. It also implements a translation from MITL$_{(0,\infty)}$ to CLTLoc. This translation does not assume any special shape for signals, except that they be finitely variable; it natively supports operators $\mathbf{F}_{\langle 0,b\rangle}$ and $\mathbf{G}_{\langle 0,b\rangle}$ (and their past counterparts), where the bounds can be either included or excluded. These operators allow us to define concisely $\mathbf{F}_{\langle a,b\rangle}$ and $\mathbf{G}_{\langle a,b\rangle}$ as abbreviations. For instance, $\mathbf{G}_{(3,6)}(\phi)$ is equivalent to $\mathbf{G}_{(0,3)}\Big(\mathbf{F}_{(0,3)}\Big(\mathbf{G}_{(0,3)}(\phi)\Big)\Big)$; defining a similar equivalence using only the $\mathbf{F}_{(0,1)}$ and $\mathbf{G}_{(0,1)}$ modalities (see, e.g., [22]) involves the recursive expansions of each conjunct of $\mathbf{G}_{(3,4)}(\phi) \wedge \mathbf{G}_{[4,5)}(\phi) \wedge \mathbf{G}_{[5,6)}(\phi)$, where $\mathbf{G}_{(n,n+1)}(\phi)$ is equivalent to $\mathbf{G}_{(n-1,n)}\Big(\mathbf{F}_{(0,1)}\Big(\mathbf{G}_{(0,1)}(\phi)\Big)\Big)$.

The following two encodings are currently available (they both include past operators):

MITL: providing a direct definition of MITL operators, assuming l.c.r.o. intervals;
QTL: providing the definition of MITL$_{(0,\infty)}$ operators with unrestricted signals (other than they be finitely variable), and MITL operators through abbreviations.

We used the above two encodings and the CLTLoc decision procedure to carry out some verification experiments on the example of the Timed Lamp described in Sect. 2. More precisely, we have built several descriptions of the behavior of the lamp: (i) the CLTLoc model presented in Sect. 2; (ii) a MITL specification

assuming l.c.r.o. signals; (iii) a $\mathrm{MITL}_{(0,\infty)}$ specification in which predicates *on* and *off* are constrained to be true only in isolated instants. On each of these specifications we have carried out three experiments, assuming $\Delta = 5$: a check of the satisfiability of the specification, to show that it is consistent (*sat*); the (dis)proof of property "the light never stays on for more than Δ time units" (p_1); the proof of property "if at some point the light stays on for more than Δ time units, then there is an instant when *on* is pressed, and then it is pressed again before Δ time units" (p_2). Depending on the temporal logic and of the restrictions on the signals (l.c.r.o. or not) the formalization of the timed lamp and of the properties can change.

In the case of the CLTLoc specification of the timed lamp, in order to formalize properties p_1 and p_2 we introduce an auxiliary clock c_{aux}, which is reset every time the light is turned on, i.e., $c_{\mathrm{aux}} \Leftrightarrow l \wedge \mathbf{Y}(\neg l)$. Then, in CLTLoc property p_1 is captured by formula $\mathbf{G}(\mathbf{Y}(l) \Rightarrow c_{\mathrm{aux}} \leqslant \Delta)$. In addition, property p_2 is formalized by the following formula:

$$\mathbf{F}(l \wedge c_{\mathrm{aux}} \geqslant \Delta) \Rightarrow \mathbf{F}(on \wedge \mathbf{X}(\neg rst\text{-}c\mathbf{U}(on \wedge test_{0<c\leqslant\Delta}))) \tag{29}$$

The behavior of the timed lamp can be captured by the following MITL formula over l.c.r.o. signals (we write $\phi \mathbf{S}^i \psi$ for $\psi \vee (\phi \wedge \phi \mathbf{S}_{(0,\infty)} \psi)$ and $\mathbf{P}^i_I \phi$ for $\phi \vee \mathbf{P}_I \phi$):

$$\mathbf{G}^i\left(\left(l \Leftrightarrow (\ on \mathbf{S}^i \neg off\ ) \wedge \mathbf{P}^i_{(0,\Delta)}(on)\right) \wedge (on \Rightarrow \neg off)\right) \tag{30}$$

In MITL over l.c.r.o. signals, where predicates hold over non-null intervals, we limit the length of intervals in which *on* (and *off*) holds to be at most 1 by adding the following constraint:

$$\mathbf{G}^i\left(\neg \mathbf{G}_{(0,1]}(on) \wedge \neg \mathbf{G}_{(0,1]}(off)\right). \tag{31}$$

Over unrestricted signals, instead, we force *on* to hold only in isolated instants by adding the following $\mathrm{MITL}_{(0,\infty)}$ constraint (and similarly for *off*)

$$\mathbf{G}^i\left(\neg(on\ \mathbf{U}_{(0,+\infty)} \top) \wedge \neg(on\ \mathbf{S}_{(0,\infty)} \top)\right). \tag{32}$$

Properties p_1 and p_2 over unrestricted signals are captured by the following $\mathrm{MITL}_{(0,\infty)}$ formulae (where $\mathbf{F}^i$ stands for $\mathbf{F}^i_{(0,+\infty)}$):

$$\mathbf{G}^i\left(\mathbf{F}^i_{(0,\Delta]}(\neg l)\right) \tag{33}$$

$$\mathbf{F}^i\left(\mathbf{G}^i_{(0,\Delta]}(l)\right) \Rightarrow \mathbf{F}^i\left(on \wedge \mathbf{F}_{(0,\Delta]}(on)\right) \tag{34}$$

Over l.c.r.o. signals property p_1 is still captured by Formula (33); property p_2, instead, is more involved, and corresponds to the following formula:

$$\mathbf{F}^i\left(\mathbf{G}^i_{(0,\Delta]}(l)\right) \Rightarrow \mathbf{F}^i\left((\neg on \wedge \mathbf{P}^i_{(0,\Delta)}(on))\mathbf{U}^i on\right) \tag{35}$$

Table 8 reports the time and space required for the checks outlined above (all tests have been done using the Common Lisp compiler SBCL 1.1.2 on a 2.13GHz Core2 Duo MacBook Air with MacOS X 10.7 and 4GB of RAM; the solver was z3 4.0). All bounded satisfiability checks have been performed using a bound $k = 20$.

Table 5 Experimental results with the timed lamp, reporting Time (sec) and heap size (MB).

Problem	Satisfiable?	CLTL-o-c	MITL (l.c.r.o)	$\text{MITL}_{(0,\infty)}$ (unrest.)
sat	Yes	0.48/0.33 5.63	15.5/13.84 66.45	4.24/3.04 27.12
$\mathbf{p_1}$	Yes	0.52/0.35 6.22	36.74/33.16 102.47	17.2/14.86 63.5
$\mathbf{p_2}$	No	0.67/0.49 6.55	6.61/5.09 110.27	257.1/240.88 58.66

The first line of each row shows the total processing time (i.e., parsing and solving) and the time taken by the SMT-solver (both times in seconds). The second line reports the heap size (in Mbytes) required by Z3. In every case the specification is satisfiable, property p_1 does not hold (the tool returns a counterexample), while property p_2 holds ("unsat" is returned). In addition to the results shown in the table, a variant of Formula (29) where $test_{0<c<\Delta}$ is used instead of $test_{0<c\leqslant\Delta}$ (i.e., $\leqslant$ is replaced by $<$) is shown to not hold, and a counterexample is obtained in less than 1 second.

Finally, we present an interesting behavior over unrestricted signals. The behavior is captured by the following formulae, which state that p and q only occur in isolated instants, with p occurring exactly every 80 time units, and q occurring within 80 time units in the past from each p (origin excluded).

$$\mathbf{G}^i \left(\begin{array}{l} \mathbf{G}_{(0,80)}(\neg p) \Rightarrow \mathbf{G}_{(80,160)}(\neg p) \quad \wedge \\ (p \Rightarrow \mathbf{F}_{(0,160)} p) \ \wedge \ (q \Rightarrow (\neg q)\, \mathbf{U}\, \top) \end{array} \right) \wedge \\ p \ \wedge \ \mathbf{G}_{(0,80)}(\neg p) \ \wedge \ \mathbf{G}_{(0,\infty)}(p \Rightarrow \mathbf{P}_{(0,80)} q) \tag{36}$$

In this case, the bound $k = 10$ is enough to prove that the formula is satisfiable: a model is produced in about 40 secs. In aroud the same time, the solver shows that property $\mathbf{G}^i(p \Rightarrow \mathbf{F}_{(0,80)}(q))$ holds for model (36) (up to the considered bound), whereas property $\mathbf{G}^i(q \Rightarrow \mathbf{F}_{(0,80)}(q))$ does not hold. It is worth noticing that, in Formula (36), the constants 80 and 160 occurring in the temporal modalities are significantly greater than the above bound $k = 10$, since in principle any value is possible for the clock increments between two consecutive positions. Therefore, the length of the intervals described by a CLTLoc model is independent of the bound k, as long as k is large enough to capture all change points that are necessary to build a periodic sequence of regions.

9 Conclusions

This paper investigates a bounded approach to satisfiability checking of an extension of CLTL where variables behave like clocks (CLTLoc). The decidability of the logic (by means of an automata-based technique) is shown first, followed by an encoding into a decidable SMT problem. This encoding, implemented in our `ae2zot` tool, allows, both in principle and in practice, the use of SMT solvers to check the satisfiability of CLTLoc. We provide a short but non-trivial example of a CLTLoc specification describing a timed behavior over continuous time, which

should demonstrate the effectiveness of this approach, as we are able to (dis)prove various properties of the specification. The paper also outlines continuous time, metric temporal logics, namely MITL and $\text{MITL}_{(0,\infty)}$ (a generalization of QTL), showing that their extension pMITL, allowing existential propositional quantifiers, is as expressive as CLTLoc over the pointwise semantics. An encoding of MITL over the continuous semantics into CLTLoc is implemented in our `qtlsolver` tool. This shows that CLTLoc can be considered as a target language to reduce decision problems of various continuous-time formalisms, such as temporal logics, but in principle also Timed Automata or Timed Petri Nets.

To the best of our knowledge, our approach is the first allowing an effective implementation of a fully automated verification tool for continuous-time metric temporal logics such as MITL. The tool is still a non-optimized prototype, whose performance might also be substantially improved in future versions. Clearly, verification of formulae requiring many clocks may in general be infeasible, since satisfiability of MITL is EXPSPACE-complete (but we also support verification of an interesting, PSPACE-complete fragment of MITL). However, in practice a large number of clocks is not very frequent, and the examples of MITL formulae that we studied were verified in a fairly short time.

References

1. qtlsolver. available from `qtlsolver.googlecode.com`.
2. Zot: a bounded satisfiability checker. available from `zot.googlecode.com`.
3. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
4. R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.
5. G. Audemard, A. Cimatti, A. Kornilowicz, and R. Sebastiani. Bounded model checking for timed systems. In *Proc. of FORTE*, pages 243–259, 2002.
6. B. Badban and M. Lange. Exact incremental analysis of timed automata with an SMT-solver. In *FORMATS*, volume 6919 of *LNCS*, pages 177–192. 2011.
7. J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *Lect. on Concurrency and Petri Nets*, volume 3098 of *LNCS*, pages 87–124. 2004.
8. M. M. Bersani, A. Frigeri, A. Morzenti, M. Pradella, M. Rossi, and P. San Pietro. Bounded reachability for temporal logic over constraint systems. In *TIME 2010*, pages 43–50. IEEE Computer Society, 2010.
9. M. M. Bersani, A. Frigeri, A. Morzenti, M. Pradella, M. Rossi, and P. San Pietro. CLTL Satisfiability Checking without Automata. arXiv:1205.0946v1, 2012.
10. M. M. Bersani, A. Frigeri, M. Pradella, M. Rossi, A. Morzenti, and P. San Pietro. SMT-based Bounded Model Checking with Difference Logic Constraints. Technical report, Politecnico di Milano, 2010.
11. M. M. Bersani, A. Frigeri, M. Rossi, and P. San Pietro. Completeness of the bounded satisfiability problem for constraint LTL. In *Reachability Problems*, volume 6945 of *LNCS*, pages 58–71. 2011.
12. M. M. Bersani, M. Rossi, and P. San Pietro. Deciding continuous-time metric temporal logic with counting modalities. In *Reachability Problems*, volume 8169 of *LNCS*, pages 70–82. 2013.
13. M. M. Bersani, M. Rossi, and P. San Pietro. Deciding the satisfiability of MITL specifications. In *Proc. of the Int. Symp. on Games, Automata, Logics and Formal Verification (GandALF)*, pages 64–78, 2013.
14. M. M. Bersani, M. Rossi, and P. San Pietro. On the satisfiability of metric temporal logics over the reals. In *Proc. of the Int. Work. on Automated Verification of Critical Systems (AVOCS)*, pages 1–15, 2013.
15. M. M. Bersani, M. Rossi, and P. San Pietro. LTL+clocks is equivalent to timed automata. submitted for publication, available from `bit.ly/1lWTF5N`, 2014.

16. P. Bouyer, F. Chevalier, and N. Markey. On the expressiveness of TPTL and MTL. *Information and Computation*, 208(2):97–116, 2010.
17. E. M. Clarke, D. Kroening, J. Ouaknine, and O. Strichman. Completeness and complexity of bounded model checking. In *Verification Model Checking and Abstract Interpretation*, volume 2937 of *Lecture Notes in Computer Science*, pages 85–96. Springer, 2004.
18. S. Demri and D. D'Souza. An automata-theoretic approach to constraint LTL. *Information and Computation*, 205(3):380–415, 2007.
19. D. D'Souza and P. Prabhakar. On the expressiveness of mtl in the pointwise and continuous semantics. *International Journal on Software Tools for Technology Transfer (STTT)*, 9(1):1–4, 2007.
20. J. Ferrante and C. Rackoff. A decision procedure for the first order theory of real addition with order. *SIAM J. Comput.*, 4(1):69–76, 1975.
21. T. A. Henzinger, J.-F. Raskin, and P.-Y. Schobbens. The regular real-time languages. In *In Proc. 25th Int. Coll. Automata, Languages, and Programming (ICALP'98*, pages 580–591. Springer-Verlag, 1998.
22. Y. Hirshfeld and A. Rabinovich. Timer formulas and decidable metric temporal logic. *Information and Computation*, 198(2):148 – 178, 2005.
23. Y. Hirshfeld and A. M. Rabinovich. Logics for real time: Decidability and complexity. *Fundamenta Informaticae*, 62(1):1–28, 2004.
24. O. Maler, D. Nickovic, and A. Pnueli. From MITL to timed automata. In *Proc. of FORMATS*, volume 4202 of *LNCS*, pages 274–289. 2006.
25. Microsoft Research. Z3: An efficient SMT solver. http://research.microsoft.com/en-us/um/redmond/projects/z3/.
26. P. Niebert, M. Mahfoudh, E. Asarin, M. Bozga, O. Maler, and N. Jain. Verification of timed automata via satisfiability checking. In *FTRTFT*, volume 2469 of *LNCS*, pages 225–243. 2002.
27. J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *LICS*, pages 188–197, 2005.
28. M. Pradella, A. Morzenti, and P. San Pietro. Bounded satisfiability checking of metric temporal logic specifications. *ACM Trans. on Softw. Eng.g and Meth. (TOSEM)*, 22(3):20, 2013.
29. A. Rabinovich. Complexity of metric temporal logics with counting and the Pnueli modalities. *Th. Comp. Sci.*, 411:2331–2342, 2010.
30. P.-Y. Schobbens, J.-F. Raskin, and T. A. Henzinger. Axioms for real-time logics. *Theor. Comput. Sci.*, 274(1-2):151–182, 2002.
31. A. Tarski. A decision method for elementary algebra and geometry. Univ. of California Press, second ed., Berkeley, 1951.