A Regular Metric Temporal Logic

Shankara Narayanan Krishna, Khushraj Madnani, Paritosh Pandya

A Regular Metric Temporal Logic

Shankara Narayanan Krishna[1], Khushraj Madnani[1], and Paritosh Pandya[1]

1 **Dept of CSE, IIT Bombay, India**
krishnas,khushraj@cse.iitb.ac.in
2 **TIFR**
pandya@tifr.res.in

Abstract

We study an extension of MTL in pointwise time with regular expression guarded modality $\mathsf{Reg}_I(\mathsf{re})$ where re is a regular expression over subformulae. We study the decidability and expressiveness of this extension, called RegMTL, as well as its fragment SfrMTL where only star-free extended regular expressions are allowed. Using the technique of temporal projections, we show that RegMTL has decidable satisfiability by giving an equisatisfiable reduction to MTL. Moreover, we identify a subset MITL[UReg] for which our (polynomial time computable) equi-satisfiable reduction gives rise to formulae of MITL. Thus, MITL[UReg] has elementary decidability. As our second main result, we show that SfrMTL is equivalent to partially ordered (or very weak) 1-clock alternating timed automata. We also identify natural fragments of logic TPTL which correspond to our logics.

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

Temporal logics provide constructs to specify qualitative ordering between events in time. Real time logics are quantitative extensions of temporal logics with the ability to specify real time constraints amongst events. The main modality in Metric Temporal Logic (MTL) is the until modality $a\,\mathsf{U}_I b$ which, when asserted at a point specifies that there is a future within a time distance in I where b holds, and a holds continuously till then. Two notions of MTL have been studied in the literature : continuous and pointwise. It is known [1] that satisfiability checking of MTL is undecidable in the continuous semantics even for finite words, while for the pointwise case, this is decidable [10]. The complexity of the satisfiability problem for MTL over finite timed words is known to be non-primitive recursive (NPR) in the pointwise semantics, while if the intervals I allowed in the until modalities are non-punctual, then the complexity drops to EXPSPACE in both the pointwise and continuous semantics. The fragment of MTL with only non-punctual intervals is denoted MITL, and was introduced in [1]. A non-punctual interval has the form $\langle x, y\rangle$ where $x < y$, $x \in \mathbb{N}, y \in \mathbb{N} \cup \{\infty\}$.

There are various natural extensions of temporal logics have been studied both in classical and timed logic domain. Wolper extended LTL with certain grammar operators to achieve MSO completeness. Baziramwabo, McKenzie and Thérien extended LTL with modular and group modalities, and showed that the latter is as expressive as regular languages [2]. Counting LTL is an extension of LTL with threshold counting. It has been shown that this extension does not increase the expressive power of LTL [6]. As another extension, LTL with just modulo counting modalities has been studied by [7]. In timed logics, Raskin's Ph.D thesis studied various extensions of MITL with the ability to count over the entire model. Rabinovich et. al. extended continuous MITL with counting (called the C modality) and Pnueli modalities [13] and showed that these extensions are more expressive than MITL. The counting modalities $\mathsf{C}_n(\phi)$, specify that the number of points that satisfy ϕ within

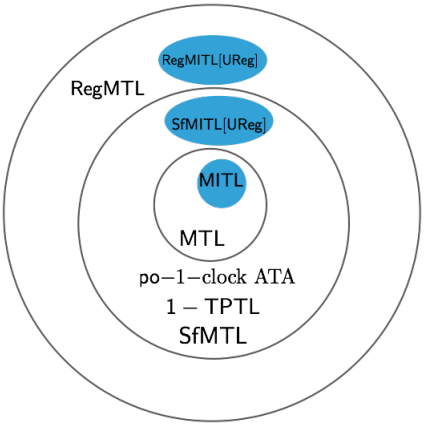

Figure 1 Big picture of the paper. The interval logic star-free MTL denoted SfrMTL is equivalent to the freeze logic $1-$ TPTL, which is equivalent to po−1-clock-ATA. All the logics in blue have an elementary complexity, while SfMITL[UReg] is strictly more expressive than MITL, and RegMITL[UReg] is more expressive than its star-free counterpart SfMITL[UReg].

the next unit interval is at least n. The Pnueli modality, P_{n_k} is a generalization of the threshold counting modality : $\mathsf{P}_{n_k}(\phi_1, \ldots, \phi_k)$ specifies that there is an increasing sequence of timestamps $t_1, \ldots, t_k$ in the next unit interval such that ϕ_i is true at t_i.

Contributions This paper is on extensions of MTL in the point-wise semantics. Contributions of this paper are as follows:

- **Generalizations**: We generalize some of these extended modalities(Pnueli, modulo counting) that has been studied in the literature with a Reg_I and UReg_I modality which allows us to specify a regular expression over subformulae within some time interval in the future. Let $\mathsf{re}(\phi_1, \ldots, \phi_k)$ be a regular expression over formulae $\phi_1, \ldots, \phi_k$. The $\mathsf{Reg}_I(\mathsf{re}(\phi_1, \ldots, \phi_k))$ modality specifies that the pattern of the behaviour of the subformulae, $\phi_1, \ldots, \phi_k$, in the time segment within interval I in the future is in accordance with $\mathsf{re}(\phi_1, \ldots, \phi_k)$, while the $\psi_1 \mathsf{UReg}_{I,\mathsf{re}(\phi_1,\ldots,\phi_k)} \psi_2$ modality asserts that there exist a point j in the future within interval I where ψ_2 is true, and at all the points strictly between the present point and j, ψ_1 is true and the behaviour of $\phi_1, \ldots, \phi_k$ in this region is in accordance with $\mathsf{re}(\phi_1, \ldots, \phi_k)$. This extension of MTL is denoted as RegMTL.
- **Satisfiability Checking**: We show that RegMTL is decidable over finite timed words with non primitive recursive complexity using the technique of oversampled temporal projections. The check $\mathsf{Reg}_I(\mathsf{re})$ at each point in the timed word is taken care of by annotating the timed word with an encoding of the runs of the DFA corresponding to the re. We show that the runs of the automaton can be captured in a way requiring only bounded amount of information, and that this can be captured in MTL, giving rise to an equisatisfiable MTL formula.
- **Automata-Logic Connection and Expressiveness**: We show that SfrMTL, the subclass of RegMTL where the regular expressions are star free, characterize exactly 1 clock partially ordered alternating timed automata. If K is the maximum constant used in the automaton, we show that the behaviour of each location of the automaton over time can be asserted using LTL formulae over timed regions $[0,0], (0,1), \ldots, [K,K], (K,\infty)$. This enables us to assert the behaviour of the automaton starting at any location as a $\mathsf{Reg}_I(\mathsf{re})$ formula where the re is captured by an LTL formula. This also implies that SfrMTL is exactly equivalent to 1-TPTL (the most expressive decidable fragment of TPTL in

pointwise semantics). This is the first such equivalence of logics with interval constraints (SfrMTL) and freeze quantifications (1−TPTL) in pointwise semantics to the best of our knowledge.

- **Complexity**: We focus on non punctual fragments of RegMTL, and show that satisfiability with only UReg modality has a 2EXPSPACE upper bound, while, surprisingly, if one considers a special case of the Reg_I modality which only specifies the parity of a proposition in the next unit interval (the iseven modality), the complexity is $\mathbf{F}_{\omega^\omega}$-hard. Finally we also explore the complexity with UM, a restricted form of Ureg that allows to specify only modulo counting constraints, and show its satisfiability to be EXPSPACE-complete. It is important to note that in spite of being a special case, UM is exponentially more succinct then UReg.
- **Novel Proof Techniques**: The logic RegMTL uses modalities that can assert the truth of a regular expression within a time interval. The satisfiability of RegMTL requires one to check the truth of these regular expressions at arbitrary points of the model; we do this by encoding the runs of the automaton corresponding to the regular expression starting at each point in the model. We show that the information pertaining to the potentially unbounded number of runs originating from the unboundedly many points of the model can be stored using bounded memory, by merging the runs when they reach the same state. This idea of merging the runs and encoding them in the model is new, to the best of our knowledge. The other novelty in terms of proof techniques used is while proving that RegMTL is at least as expressive as partially ordered 1-clock alternating timed automata. The timed behaviours enforced by any state of the automaton is captured by writing LTL formulae over the clock regions, and putting them together as RegMTL formulae $\mathsf{Reg}_I(\mathsf{re})$ where the re is a star-free expression obtained corresponding to the LTL formula asserted over clock region I.

2 Preliminaries

2.1 Timed Temporal Logics

We first describe the syntax and semantics of the timed temporal logics needed in this paper : MTL and TPTL. Let Σ be a finite set of propositions. A finite timed word over Σ is a tuple $\rho = (\sigma, \tau)$. σ and τ are sequences $\sigma_1\sigma_2\ldots\sigma_n$ and $t_1t_2\ldots t_n$ respectively, with $\sigma_i \in 2^\Sigma - \emptyset$, and $t_i \in \mathbb{R}_{\geq 0}$ for $1 \leq i \leq n$ and $\forall i \in dom(\rho)$, $t_i \leq t_{i+1}$, where $dom(\rho)$ is the set of positions $\{1, 2, \ldots, n\}$ in the timed word. Given $\Sigma = \{a, b\}$, $\rho = (\{a, b\}, 0.8)(\{a\}, 0.99)(\{b\}, 1.1)$ is a timed word. ρ is strictly monotonic iff $t_i < t_{i+1}$ for all $i, i+1 \in dom(\rho)$. Otherwise, it is weakly monotonic. The set of finite timed words over Σ is denoted $T\Sigma^*$.

Metric Temporal Logic (MTL) extends linear temporal logic (LTL) by adding timing constraints to the "until" modality of LTL. MTL is parameterized by using a permitted set of open, half-open or closed time intervals, denoted by $I\nu$. The end points of these intervals are in $\mathbb{N} \cup \{0, \infty\}$. Such an interval is denoted $\langle a, b\rangle$. For example, $[1, 6), [3, \infty)$. For $t \in \mathbb{R}_{\geq 0}$ and interval $\langle a, b\rangle$, $t + \langle a, b\rangle$ stands for the interval $\langle t + a, t + b\rangle$.

Metric Temporal Logic

Given a finite alphabet Σ, the formulae of MTL are built from Σ using boolean connectives and time constrained version of the modality U as follows:
$\varphi ::= a(\in \Sigma) \mid true \mid \varphi \wedge \varphi \mid \neg\varphi \mid \varphi\, \mathsf{U}_I \varphi$, where $I \in I\nu$. For a timed word $\rho = (\sigma, \tau) \in T\Sigma^*$, a position $i \in dom(\rho) \cup \{0\}$, and an MTL formula φ, the satisfaction of φ at a position i of ρ

is denoted $(\rho, i) \models \varphi$, and is defined as follows:
$\rho, i \models a \leftrightarrow a \in \sigma_i$, $\rho, i \models \neg\varphi \leftrightarrow \rho, i \not\models \varphi$
$\rho, i \models \varphi_1 \wedge \varphi_2 \leftrightarrow \rho, i \models \varphi_1$ and $\rho, i \models \varphi_2$
$\rho, i \models \varphi_1 \mathsf{U}_I \varphi_2 \leftrightarrow \exists j > i$, $\rho, j \models \varphi_2, t_j - t_i \in I$, and $\rho, k \models \varphi_1 \; \forall \; i < k < j$

We assume the existence of a special point called 0, outside $dom(\rho)$. The time stamp of this point is 0 ($t_0 = 0$).[1] ρ satisfies φ denoted $\rho \models \varphi$ iff $\rho, 1 \models \varphi$. The language of a MTL formula φ is $L(\varphi) = \{\rho \mid \rho, 0 \models \varphi\}$. Two formulae φ and ϕ are said to be equivalent denoted as $\varphi \equiv \phi$ iff $L(\varphi) = L(\phi)$. Additional temporal connectives are defined in the standard way: we have the constrained future eventuality operator $\Diamond_I a \equiv true \, \mathsf{U}_I a$ and its dual $\Box_I a \equiv \neg\Diamond_I \neg a$. We also define the next operator as $\mathsf{O}_I \phi \equiv \bot \mathsf{U}_I \phi$. Weak versions of operators are defined as $\Diamond_I^{\mathsf{ns}} a = a \vee \Diamond_I a, \Box_I^{\mathsf{ns}} a \equiv a \wedge \Box_I a$, $a \, \mathsf{U}_I^{\mathsf{ns}} b \equiv b \vee [a \wedge (a \, \mathsf{U}_I b)]$ if $0 \in I$, and $[a \wedge (a \, \mathsf{U}_I b)]$ if $0 \notin I$. Also, $a \, \mathsf{W} b$ is a shorthand for $\Box a \vee (a \, \mathsf{U} b)$. The subclass of MTL obtained by restricting the intervals I in the until modality to non-punctual intervals is denoted MITL.

▶ **Theorem 1** ([10]). *Satisfiability checking of* MTL *is decidable over finite timed words and is non-primitive recursive.*

Timed Propositional Temporal Logic (TPTL)

In this section, we recall the syntax and semantics of TPTL. A prominent real time extension of linear temporal logic is TPTL, where timing constraints are specified with the help of freeze clocks. The set of TPTL formulas are defined inductively as:
$\varphi ::= a (\in \Sigma) \mid true \mid \varphi \wedge \varphi \mid \neg\varphi \mid \varphi \, \mathsf{U} \varphi \mid y.\varphi \mid y \in I$
There is a set $\mathcal{C}$ of clock variables progressing at the same rate, $y \in \mathcal{C}$, and I is an interval of the form $<a, b>$ $a, b \in \mathbb{N}$ with $< \in \{(, [\}$ and $> \in \{],)\}$.

TPTL is interpreted over finite timed words over Σ. The truth of a formula is interpreted at a position $i \in \mathbb{N}$ along the word. For a timed word $\rho = (\sigma_1, t_1) \ldots (\sigma_n, t_n)$, we define the satisfiability relation, $\rho, i, \nu \models \phi$ saying that the formula ϕ is true at position i of the timed word ρ with valuation ν of all the clock variables.

1. $\rho, i, \nu \models a \leftrightarrow a \in \sigma_i$
2. $\rho, i, \nu \models \neg\varphi \leftrightarrow \rho, i, \nu \not\models \varphi$
3. $\rho, i, \nu \models \varphi_1 \wedge \varphi_2 \leftrightarrow \rho, i, \nu \models \varphi_1$ and $\rho, i, \nu \models \varphi_2$
4. $\rho, i, \nu \models x.\varphi \leftrightarrow \rho, i, \nu[x \leftarrow t_i] \models \varphi$
5. $\rho, i, \nu \models x \in I \leftrightarrow t_i - \nu(x) \in I$
6. $\rho, i, \nu \models \varphi_1 \, \mathsf{U} \varphi_2 \leftrightarrow \exists j > i$, $\rho, j, \nu \models \varphi_2$, and $\rho, k, \nu \models \varphi_1 \; \forall \; i < k < j$

ρ satisfies ϕ denoted $\rho \models \phi$ iff $\rho, 1, \bar{0} \models \phi$. Here $\bar{0}$ is the valuation obtained by setting all clock variables to 0. We denote by $k-$TPTL the fragment of TPTL using at most k clock variables. The fragment of TPTL with k clock variables is denoted $k-$TPTL.

MTL with Regular Expressions (RegMTL)

In this section, we introduce the extension of MTL with regular expressions, that forms the core of the paper. These modalities can assert the truth of a regular expression within a particular time interval with respect to the present point. For example, $\mathsf{Reg}_{(0,1)}(\varphi_1.\varphi_2)^*$

[1] MTL cannot check the time stamp of the first action point

when evaluated at a point i, asserts that either $\tau_{i+1} \geq \tau_i + 1$ (corresponds to ϵ) or, there exist $2k$ points $\tau_i < \tau_{i_1} < \tau_{i_2} < \cdots < \tau_{i_{2k}} < \tau_{i+1}$, $k > 0$, $0 < \tau_{i+1} - \tau_i < 1$, such that φ_1 evaluates to true at $\tau_{i_{2j+1}}$, and φ_2 evaluates to true at $\tau_{i_{2j+2}}$, for all $j \geq 0$.

RegMTL Syntax: Formulae of RegMTL are built from Σ (atomic propositions) as follows:
$\varphi ::= a(\in \Sigma) \mid true \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathsf{Reg}_I \mathsf{re} \mid \varphi \mathsf{UReg}_{I,\mathsf{re}} \varphi$
$\mathsf{re} ::= \varphi \mid \mathsf{re}.\mathsf{re} \mid \mathsf{re} + \mathsf{re} \mid \mathsf{re}^*$ where $I \in I\nu$.

An *atomic* regular expression re is any well-formed formula $\varphi \in$ RegMTL. For a regular expression re, let Γ be the set of all subformulae and their negations appearing in re. For example, if $\mathsf{re} = a\mathsf{UReg}_{(0,1),\mathsf{Reg}_{(1,2)}[\mathsf{Reg}_{(0,1)}b]}b$, then Γ consists of $\mathsf{Reg}_{(1,2)}[\mathsf{Reg}_{(0,1)}b], \mathsf{Reg}_{(0,1)}b, b$ and their negations.

Let $\mathsf{Cl}(\Gamma)$ denote consistent sets[2] in $\mathcal{P}(\Gamma)$. $L(\mathsf{re})$ is the set of strings over $\mathsf{Cl}(\Gamma)$ defined as follows. Let $S \in \mathsf{Cl}(\Gamma)$.

$$L(\mathsf{re}) = \begin{cases} \{S \mid a \in S\} & \text{if } \mathsf{re} = a, \\ \{S \mid \varphi_1, \varphi_2 \in S\} & \text{if } \mathsf{re} = \varphi_1 \wedge \varphi_2, \\ \{S \mid \varphi \notin S\} & \text{if } \mathsf{re} = \neg\varphi, \\ L(\mathsf{re}_1).L(\mathsf{re}_2) & \text{if } \mathsf{re} = \mathsf{re}_1.\mathsf{re}_2, \\ L(\mathsf{re}_1) \cup L(\mathsf{re}_2) & \text{if } \mathsf{re} = \mathsf{re}_1 + \mathsf{re}_2, \\ [L(\mathsf{re}_1)]^* & \text{if } \mathsf{re} = (\mathsf{re}_1)^*. \end{cases}$$

If re is not an atomic regular expression, but has the form $\mathsf{re}_1 + \mathsf{re}_2$ or $\mathsf{re}_1.\mathsf{re}_2$ or $(\mathsf{re}_1)^*$, then we use the standard definition of $L(\mathsf{re})$ as $L(\mathsf{re}_1) \cup L(\mathsf{re}_2)$, $L(\mathsf{re}_1).L(\mathsf{re}_2)$ and $[L(\mathsf{re}_1)]^*$ respectively.

RegMTL Semantics: For a timed word $\rho = (\sigma, \tau) \in T\Sigma^*$, a position $i \in dom(\rho) \cup \{0\}$, and a RegMTL formula φ, we define the satisfaction of φ at a position i as follows.

1. $\varphi = \varphi_1 \mathsf{UReg}_{I,\mathsf{re}} \varphi_2$.
   - Consider first the case when re is any atomic regular expression. Let Γ be the set of all subformulae appearing in re. For positions $i < j \in dom(\rho)$, let $\mathsf{Seg}(\Gamma, i, j)$ denote the untimed word over $\mathsf{Cl}(\Gamma)$ obtained by marking the positions $k \in \{i+1, \ldots, j-1\}$ of ρ with $\psi \in \Gamma$ iff $\rho, k \models \psi$. Then $\rho, i \models \varphi_1 \mathsf{UReg}_{I,\mathsf{re}} \varphi_2 \leftrightarrow \exists j{>}i$, $\rho, j \models \varphi_2, t_j - t_i \in I$, $\rho, k \models \varphi_1$ $\forall i{<}k{<}j$ and, $\mathsf{Seg}(\Gamma, i, j) \in L(\mathsf{re})$, where $L(\mathsf{re})$ is the language of the regular expression re.
2. $\varphi = \mathsf{Reg}_I \mathsf{re}$. As above, let Γ be the set of all subformulae appearing in re. Then for a position $i \in dom(\rho)$ and an interval I, let $\mathsf{TSeg}(\Gamma, I, i)$ denote the untimed word over $\mathsf{Cl}(\Gamma)$ obtained by marking all the positions k such that $\tau_k - \tau_i \in I$ of ρ with $\psi \in \Gamma$ iff $\rho, k \models \psi$. Then $\rho, i \models \mathsf{Reg}_I \mathsf{re} \leftrightarrow \mathsf{TSeg}(\Gamma, I, i) \in L(\mathsf{re})$.

Example 1. Consider the formula $\varphi = a\mathsf{UReg}_{(0,1),ab^*}b$. Then $\mathsf{re}{=}ab^*$ and $\Gamma{=}\{a, b, \neg a, \neg b\}$. For $\rho{=}(\{a\}, 0.1)(\{a\}, 0.3)(\{a, b\}, 1.01)$, $\rho, 1 \models \varphi$, since $a{\in}\sigma_2, b{\in}\sigma_3$, $\tau_3{-}\tau_1{\in}(0, 1)$ and the untimed word obtained at position 2 is a which is in $L(ab^*)$.

For $\rho = (\{a\}, 0.1)(\{a\}, 0.3)(\{a\}, 0.5)(\{a\}, 0.9)(\{b\}, 1.01)$, we know that $\rho, 1 \not\models \varphi$, since the untimed word obtained is $aaa \notin L(ab^*)$. *Example 2.* Consider the formula $\varphi = \mathsf{Reg}_{(0,1)}[\neg\mathsf{Reg}_{(0,1)}a]$. Then $\Gamma = \{\neg\mathsf{Reg}_{(0,1)}a, \mathsf{Reg}_{(0,1)}a, a, \neg a\}$.

1. For the word $\rho = (\{a, b\}, 0.1)(\{a, b\}, 1.01)(\{a\}, 1.2)$, $\mathsf{TSeg}(\Gamma, (0, 1), 1) = \{a, \mathsf{Reg}_{(0,1)}a\}$ is the marking of position 2. $\rho, 2 \models \mathsf{Reg}_{(0,1)}a$ since $\rho, 3 \models a$. Hence, $\rho, 1 \not\models \varphi$.
2. For $\rho = (\{a, b\}, 0.1)(\{b\}, 0.7)(\{a, b\}, 1.01)(\{a\}, 1.2)$, $\mathsf{TSeg}(\Gamma, (0, 1), 1){=}\{b\}.\{a, b, \mathsf{Reg}_{(0,1)}a\}$. $\rho, 1 \not\models \varphi$.

[2] a set S is consistent iff $\varphi \in S \leftrightarrow \neg\varphi \notin S$

3. Lastly, for $\rho = (\{a,b\},0.1)(\{a,b\},1.01)(\{b\},1.2)$, we obtain $\rho,1 \models \varphi$, since $\rho,3 \not\models a$, and hence position 2 is not marked $\mathsf{Reg}_{(0,1)}a$.

Example 3. Consider the formula $\varphi = \mathsf{Reg}_{(0,1)}[\mathsf{Reg}_{(0,1)}a]^*$.
For $\rho = (\{a,b\},0.1)(\{a,b\},0.8)(\{b\},0.99)(\{a,b\},1.5)$, we have $\rho,1 \not\models \mathsf{Reg}_{(0,1)}[\mathsf{Reg}_{(0,1)}a]^*$, since point 2 is not marked $\mathsf{Reg}_{(0,1)}a$, even though point 3 is.

The language accepted by a RegMTL formula φ is given by $L(\varphi) = \{\rho \mid \rho,0 \models \varphi\}$.

Subclasses of RegMTL

As a special subclass of RegMTL, we consider the case when the regular expressions do only mod counting. With this restriction, the $\varphi\mathsf{URReg}_{I,\mathsf{re}}\varphi$ modality is written as $\varphi\mathsf{UM}_{I,\theta}\varphi$ where θ has the form $\#\psi = k\%n$, while the Reg_I modality is written as $\mathsf{MC}_I^{k\%n}$. In both cases, $k,n \in \mathbb{N}$ and $0 \le k \le n-1$. This restriction of RegMTL, written $\mathsf{MTL}_{\mathsf{mod}}$ has the form $\varphi ::= a(\in \Sigma) \mid true \mid \varphi \wedge \varphi \mid \neg\varphi \mid \varphi \mid \mathsf{MC}_I^{k\%n}\varphi \mid \varphi\mathsf{UM}_{I,\theta}\varphi$. The obvious semantics of $\rho,i \models \mathsf{MC}_I^{k\%n}\varphi$ checks if the number of times φ is true in $\tau_i + I$ is $M(n)+k$, where $M(n)$ denotes a non-negative integer multiple of n, and $0 \le k \le n-1$. $\rho,i \models \varphi_1\mathsf{UM}_{I,\#\psi=k\%n}\varphi_2$ checks the existence of $j > i$ such that $\tau_j - \tau_i \in I$, and the number of times ψ is true in between i,j is $M(n)+k$, $0 \le k \le n-1$.

Example 4. The formula $\varphi = \Box^{\mathsf{ns}}(a \to \mathsf{MC}_{(0,1)}^{0\%2}b)$ says that whenever there is an a at a time point t, the number of b's in the interval $(t,t+1)$ is even. The formula $\psi = (a \to true\mathsf{UM}_{(0,1),\#b=0\%2}(a \vee b))$ when asserted at a point i checks the existence of a point $j > i$ such that a or $b \in \sigma_j$, $\tau_j - \tau_i \in (0,1)$, $a \in \sigma_k$ for all $i < k < j$, and the number of points between i,j where b is true is even.

The subclass of RegMTL using only the URReg modality is denoted RegMTL[URReg]. Likewise, the subclass of $\mathsf{MTL}_{\mathsf{mod}}$ with only UM is denoted $\mathsf{MTL}_{\mathsf{mod}}[\mathsf{UM}]$, while $\mathsf{MTL}_{\mathsf{mod}}[\mathsf{MC}]$ denotes the subclass using just MC.

2.2 Temporal Projections

In this section, we discuss the technique of temporal projections used to show the satisfiability of RegMTL. Let Σ, X be finite sets of propositions such that $\Sigma \cap X = \emptyset$.

(Σ,X)-*simple extensions* and *Simple Projections*: A (Σ,X)-simple extension is a timed word $\rho' = (\sigma',\tau')$ over $X \cup \Sigma$ such that at any point $i \in dom(\rho')$, $\sigma'_i \cap \Sigma \neq \emptyset$. For $\Sigma{=}\{a,b\}, X{=}\{c\}$, $(\{a\},0.2)(\{a,c\},0.3)(\{b,c\},1.1)$ is a (Σ,X)-simple extension while $(\{a\},0.2)(\{c\},0.3)(\{b\},1.1)$ is not. Given a (Σ,X)-simple extension ρ, the *simple projection* of ρ with respect to X, denoted $\rho\backslash X$ is the word obtained by deleting elements of X from each σ_i. For $\Sigma{=}\{a,b\}, X{=}\{c\}$ and $\rho = (\{a\},0.1)(\{b\},0.9)(\{a,c\},1.1)$, $\rho \setminus X = (\{a\},0.1)(\{b\},0.9)(\{a\},1.1)$.

(Σ,X)-*oversampled behaviours* and *Oversampled Projections*: A (Σ,X)-oversampled behaviour is a timed word $\rho' = (\sigma',\tau')$ over $X \cup \Sigma$, such that $\sigma'_1 \cap \Sigma \neq \emptyset$ and $\sigma'_{|dom(\rho')|} \cap \Sigma \neq \emptyset$. Oversampled behaviours are more general than simple extensions since they allow occurrences of new points in between the first and the last position. These new points are called *oversampled points*. All other points are called *action points*. For $\Sigma = \{a,b\}, X = \{c\}$, $(\{a\},0.2)(\{c\},0.3)(\{b\},0.7)(\{a\},1.1)$ is a (Σ,X)-oversampled behaviour, while $(\{a\},0.2)(\{c\},0.3)(\{c\},1.1)$ is not. Given a (Σ,X)-oversampled behaviour $\rho' = (\sigma',\tau')$, the oversampled projection of ρ' with respect to Σ, denoted $\rho' \downarrow X$ is defined as the timed word obtained by removing the oversampled points, and then erasing the symbols of X from the action points. $\rho{=}\rho' \downarrow X$ is a timed word over Σ.

A *temporal projection* is either a simple projection or an oversampled projection. We now define *equisatisfiability modulo temporal projections*. Given MTL formulae ψ and ϕ, we

say that ϕ is equisatisfiable to ψ *modulo temporal projections* iff there exist disjoint sets X, Σ such that (1) ϕ is over Σ, and ψ over $\Sigma \cup X$, (2) For any timed word ρ over Σ such that $\rho \models \phi$, there exists a timed word ρ' such that $\rho' \models \psi$, and ρ is a temporal projection of ρ' with respect to X, (3) For any behaviour ρ' over $\Sigma \cup X$, if $\rho' \models \psi$ then the temporal projection ρ of ρ' with respect to X is well defined and $\rho \models \phi$.
If the temporal projection used above is a simple projection, we call it *equisatisfiability modulo simple projections* and denote it by $\phi = \exists X.\psi$. If the projection in the above definition is an oversampled projection, then it is called *equisatisfiability modulo oversampled projections* and is denoted $\phi \equiv \exists \downarrow X.\psi$. Equisatisfiability modulo simple projections are studied extensively [5, 12, 14]. It can be seen that if $\varphi_1 = \exists X_1.\psi_1$ and $\varphi_2 = \exists X_2.\psi_2$, with X_1, X_2 disjoint, then $\varphi_1 \wedge \varphi_2 = \exists (X_1 \cup X_2).(\psi_1 \wedge \psi_2)$ [8].

Unlike simple projections, when one considers oversampled projections, there is a need to *relativize* the formula with respect to the original alphabet Σ to preserve satisfiability. As an example, let $\phi = \Box_{(0,1)} a$ be a formula over $\Sigma = \{a\}$, and let $\psi_1 = \Box(b \leftrightarrow \neg a) \wedge (\neg b \, \mathsf{U}_{(0,1)} b)$, $\psi_2 = \Box(c \leftrightarrow \Box_{[0,1)} a) \wedge c$ be two formulae over $\Sigma_1 = \Sigma \cup \{b\}$ and $\Sigma_2 = \Sigma \cup \{c\}$ respectively. Clearly, $\phi = \exists \downarrow \{b\}\psi_1$ and $\phi = \exists \downarrow \{c\}\psi_2$. However, $\phi \neq \exists \downarrow \{b, c\}(\psi_1 \wedge \psi_2)$, since the non-action point b contradicts the condition $\Box_{[0,1)} a$ corresponding to c. However, if ψ_1, ψ_2 are relativized with respect to Σ_1, Σ_2 respectively, then we will not have this problem. Relativizing ψ_1, ψ_2 with respect to Σ_1, Σ_2 gives $Rel(\psi_1, \Sigma_1), Rel(\psi_1, \Sigma_2)$ as

- $\Box(act_1 \rightarrow (b \leftrightarrow \neg a)) \wedge [(act_1 \rightarrow \neg b) \, \mathsf{U}_{(0,1)} (b \wedge act_1)]$, and
- $\Box(act_2 \rightarrow (c \leftrightarrow \Box_{[0,1)}(act_2 \rightarrow a))) \wedge (act_2 \wedge c)$.

This resolves the problem and indeed $\phi = \exists \downarrow \{b, c\}(Rel(\psi_1, \Sigma_1) \wedge Rel(\psi_2, \Sigma_2))$.

3 Satisfiability, Complexity, Expressiveness

The main results of this section are as follows.

▶ **Theorem 2.** ***1.*** *Satisfiability of* RegMTL *is decidable.*
2. *Satisfiability of* $\mathsf{MITL}_{\mathsf{mod}}[\mathsf{UM}]$ *is* EXPSPACE*-complete.*
3. *Satisfiability of* MITL[UReg] *is in* 2EXPSPACE.
4. *Satisfiability of* $\mathsf{MITL}_{\mathsf{mod}}[\mathsf{MC}]$ *is* $\mathbf{F}_{\omega^\omega}$*-hard.*

We will use equisatisfiability modulo oversampled projections in the proof of Theorem 2. This technique is used to show the decidability of RegMTL, 2EXPSPACE-hardness of RegMITL[UReg], and Ackermannian-hardness of $\mathsf{MITL}_{\mathsf{mod}}[\mathsf{MC}]$. The proof of Theorem 2.1 follows from Lemmas 4 and 5, and from Theorem 1. Details of Theorems 2.2, 2.3, 2.4 can be found in Appendices B.2, B.3 and 3.3.

▶ **Theorem 3.** $\mathsf{RegMTL}[\mathsf{UReg}] \subseteq \mathsf{RegMTL}[\mathsf{Reg}]$, $\mathsf{MTL}_{\mathsf{mod}}[\mathsf{UM}] \subseteq \mathsf{MTL}_{\mathsf{mod}}[\mathsf{MC}]$.

Theorem 3 shows that the Reg modality can capture UReg (and likewise, MC captures UM). Thus, RegMTL $\equiv$ RegMTL[Reg]. The proofs can be seen in Appendix C.

3.1 Equisatisfiable Reduction

In this section, we describe the steps to obtain an equisatisfiable reduction from RegMTL to MTL which shows that satisfiability checking of RegMTL is decidable. Starting from a RegMTL formula φ, the following steps are taken.

1. Flattening. Each of the modalities $\mathsf{Reg}_I, \mathsf{UReg}$ that appear in the formula φ are replaced with fresh witness propositions to obtain a flattened formula. For example, if

$\varphi = \mathsf{Reg}_{(0,1)}[a\mathsf{UReg}_{(1,2),\mathsf{Reg}_{(0,1)}(a+b)^*}b]$, then flattening yields $\Box^{\mathsf{ns}}[w_1 \leftrightarrow \mathsf{Reg}_{(0,1)}w_2] \wedge \Box^{\mathsf{ns}}[w_2 \leftrightarrow a\mathsf{UReg}_{(1,2),w_3}b] \wedge \Box^{\mathsf{ns}}[w_3 \leftrightarrow \mathsf{Reg}_{(0,1)}(a+b)^*]$, where w_1, w_2, w_3 are fresh witness propositions. Let W be the set of fresh witness propositions such that $\Sigma \cap W = \emptyset$. After flattening, the modalities $\mathsf{Reg}_I, \mathsf{UReg}$ appear only within *temporal definitions*. Temporal definitions are of the form $\Box^{\mathsf{ns}}[a \leftrightarrow \mathsf{Reg}_I\mathsf{atom}]$ or $\Box^{\mathsf{ns}}[a \leftrightarrow x\mathsf{UReg}_{I',\mathsf{atom}}y]$, where atom is a regular expression over $\Sigma \cup W$, W being the set of fresh witness propositions used in the flattening, and I' is either a unit length interval or an unbounded interval.

2. Consider any temporal definition T and a timed word ρ over $\Sigma \cup W$. Each of the regular expression atom has a corresponding minimal DFA recognizing it. We first construct a simple extension ρ' which marks each position of ρ using the run information from the minimal DFA that accepts the regular expression atom. However, to check that the regular expression atom holds good in a particular time interval from a point in the timed word, we need to oversample ρ' by introducing some extra points. Based on this oversampling, each point of ρ' can be marked a as a witness of $\mathsf{Reg}_I\mathsf{atom}$ (or $x\mathsf{UReg}_{I',\mathsf{atom}}y$). The construction of the simple extension ρ' is in section 3.2, while details of the elimination of $\mathsf{Reg}_I\mathsf{atom}$, $x\mathsf{UReg}_{I',\mathsf{atom}}y$ using oversampling are in the lemmas 4 and 5.

3.2 Construction of Simple Extension ρ'

For any given ρ over $\Sigma \cup W$, where W is the set of witness propositions used in the temporal definitions T of the forms $\Box^{\mathsf{ns}}[a \leftrightarrow \mathsf{Reg}_I\mathsf{atom}]$ or $\Box^{\mathsf{ns}}[a \leftrightarrow x\mathsf{UReg}_{I',\mathsf{atom}}y]$, we construct a simple extension ρ' that marks points of ρ with the run information of the minimal DFA accepting atom. This results in the extended alphabet $\Sigma \cup W \cup \mathsf{Threads} \cup \mathsf{Merge}$ for ρ'. The behaviour of $\mathsf{Threads}$ and Merge are explained below.

Let AP denote the (sub)set of propositions over which atom is defined. Let $\mathcal{A}_{\mathsf{atom}} = (Q, 2^{AP}, \delta, q_1, Q_F)$ be the minimal DFA that accepts atom and let $Q = \{q_1, q_2, \ldots, q_m\}$. Let $\mathsf{In} = \{1, 2, \ldots, m\}$ be the indices of the states. We have to mark every point i in $dom(\rho')$ with a or $\neg a$ depending on the truth of $\mathsf{Reg}_I\mathsf{atom}$ or $x\mathsf{UReg}_{I,\mathsf{atom}}y$ at i. To do this, we "run" $\mathcal{A}_{\mathsf{atom}}$ starting from each point i in $dom(\rho')$. At any point i of $dom(\rho')$, we thus have the states reached in $\mathcal{A}_{\mathsf{atom}}$ after processing the $i-1$ prefixes of ρ', and we also start a new thread at position i. This way of book-keeping will lead to maintaining unbounded number of threads of the run of $\mathcal{A}_{\mathsf{atom}}$. To avoid this, we "merge" threads i, j if the states reached at points i, j are the same, and maintain the information of the merge. It can be seen then that we need to maintain at most m distinct threads at each point, given m is the number of states of $\mathcal{A}_{\mathsf{atom}}$. We mark the points in ρ' with the state information on each thread and the information about which threads are being merged (if any), with the following set of propositions :

1. Let $\mathsf{Th}_i(q_x)$ be a proposition that denotes that the ith thread is active and is in state q_x, while $\mathsf{Th}_i(\perp)$ be a proposition that denotes that the ith thread is not active. The set $\mathsf{Threads}$ consists of propositions $\mathsf{Th}_i(q_x), \mathsf{Th}_i(\perp)$ for $1 \leq i, x \leq m$.
2. If at a position e, we have $\mathsf{Th}_i(q_x)$ and $\mathsf{Th}_j(q_y)$ for $i < j$, and if $\delta(q_x, \sigma_e) = \delta(q_y, \sigma_e)$, then we can merge the threads i, j at position $e+1$. Let $\mathbb{M}(i, j)$ be a proposition that signifies that threads i, j have been merged. In this case, $\mathbb{M}(i, j)$ is true at position $e+1$. Let Merge be the set of all propositions $\mathbb{M}(i, j)$ for $1 \leq i < j \leq m$. At most m threads can be running at any point e of the word.

We now describe the conditions to be checked in ρ'.

- **Initial condition**- At the first point of the word, we start the first thread and initialize

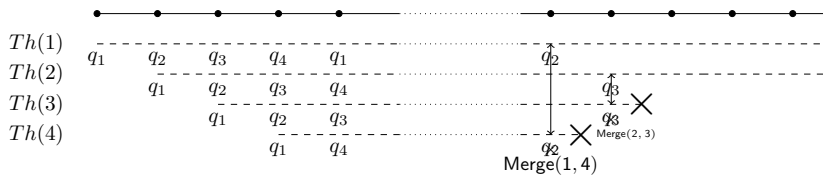

Figure 2 Encoding runs and merging of threads.

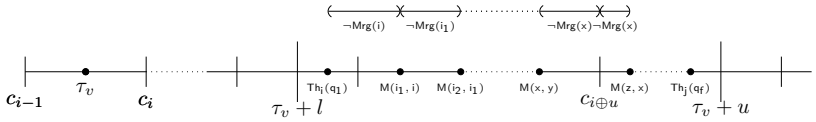

Figure 3 Linking of R_{pref} and R_{suf}.

all other threads as $\perp$. This could be specified as $\varphi_{init} = ((\mathsf{Th}_1(q_1)) \wedge \bigwedge_{i>1} \mathsf{Th}_i(\perp))$.

- **Initiating runs at all points**- To check the regular expression within an arbitrary interval, we need to start a new run from every point. $\varphi_{start} = \Box^{\mathsf{ns}}(\bigvee_{i \leq m} \mathsf{Th}_i(q_1))$
- **Disallowing Redundancy**- At any point of the word, if $i \neq j$ and $\mathsf{Th}_i(q_x)$ and $\mathsf{Th}_j(q_y)$ are both true, then $q_x \neq q_y$.
  $\varphi_{no-red} = \bigwedge_{x \in \mathsf{In}} \Box^{\mathsf{ns}}[\neg \bigvee_{1 \leq i < j \leq m} (\mathsf{Th}_i(q_x) \wedge \mathsf{Th}_j(q_x))]$
- **Merging Runs**- If two different threads $\mathsf{Th}_i, \mathsf{Th}_j (i < j)$ reach the same state q_x on reading the input at the present point, then we just keep one copy and merge thread Th_j with Th_i. We remember the merge with the proposition $\mathbb{M}(i,j)$. We define a macro $\mathsf{Nxt}(\mathsf{Th}_i(q_x))$ which is true at a point e if and only if $\mathsf{Th}_i(q_y)$ is true at e and $\delta(q_y, \sigma_e) = q_x$, where $\sigma_e \subseteq AP$ is the maximal set of propositions true at e. $\mathsf{Nxt}(\mathsf{Th}_i(q_x))$ is true at e iff thread Th_i reaches q_x after reading the input at e.
  $\mathsf{Nxt}(\mathsf{Th}_i(q_x)) = \bigvee_{(q_y, prop) \in \{(q,p) | \delta(q,p) = q_x\}} [prop \wedge \mathsf{Th}_i(q_y)]$.
  Let $\psi(i,j,k,q_x)$ be a formula that says that at the next position, $\mathsf{Th}_i(q_x)$ and $\mathsf{Th}_k(q_x)$ are true for $k > i$, but for all $j < i$, $\mathsf{Th}_j(q_x)$ is not. $\psi(i,j,k,q_x)$ is given by $\mathsf{Nxt}(\mathsf{Th}_i(q_x)) \wedge \bigwedge_{j<i} \neg\mathsf{Nxt}(\mathsf{Th}_j(q_x)) \wedge \mathsf{Nxt}(\mathsf{Th}_k(q_x))$. In this case, we merge threads $\mathsf{Th}_i, \mathsf{Th}_k$, and either restart Th_k in the initial state, or deactivate the kth thread at the next position. This is given by the formula $\mathsf{NextMerge(i,k)}$
  $\mathsf{O}[\mathbb{M}(i,k) \wedge (\mathsf{Th}_k(\perp) \vee \mathsf{Th}_k(q_1)) \wedge \mathsf{Th}_i(q_x)]$. $\varphi_{\mathbb{M}}$ is defined as

$$\bigwedge_{x,i,k \in \mathsf{In} \wedge k > i} \Box^{\mathsf{ns}}[\psi(i,j,k,q_x) \rightarrow \mathsf{NextMerge(i,k)}]$$

- **Propagating runs**- If $\mathsf{Nxt}(\mathsf{Th}_i(q_x))$ is true at a point, and if for all $j < i$, $\neg\mathsf{Nxt}(\mathsf{Th}_j(q_x))$ is true, then at the next point, we have $\mathsf{Th}_i(q_x)$. Let $\mathsf{NextTh}(i,j,q_x)$ denote the formula $\mathsf{Nxt}(\mathsf{Th}_i(q_x)) \wedge \neg\mathsf{Nxt}(\mathsf{Th}_j(q_x))$. The formula φ_{pro} is given by $\bigwedge_{i,j \in \mathsf{In} \wedge i < j} \Box^{\mathsf{ns}}[\mathsf{NextTh}(i,j,q_x) \rightarrow \mathsf{O}[\mathsf{Th}_i(q_x)) \wedge \neg\, \mathbb{M}(i,j)]]$. If $\mathsf{Th}_i(\perp)$ is true at the current point, then at the next point, either $\mathsf{Th}_i(\perp)$ or $\mathsf{Th}_i(q_1)$. The latter condition corresponds to starting a new run on thread Th_i.

$$\varphi_{NO-pro} = \bigwedge_{i \in \mathsf{In}} \Box^{\mathsf{ns}}\{\mathsf{Th}_i(\perp) \rightarrow \mathsf{O}(\mathsf{Th}_i(\perp) \vee \mathsf{Th}_i(q_1))\}$$

Once we construct the extension ρ', checking whether the regular expression atom holds in some interval I in the timed word ρ, is equivalent to checking that if a thread Th_i is at q_1 at the first action point in I, then the corresponding thread is at q_f at the last point in I. But the main challenge is that the indices of a particular thread might change because of merges. Thus the above condition reduces to checking that at the first action point u within I, if $\mathsf{Th}_i(q_1)$ holds, then after a series of merges of the form $\mathbb{M}(i_1, i)$,$\mathbb{M}(i_2, i_1)$, $\dots$ $\mathbb{M}(j, i_n)$, at the last point v in the interval I, $\mathsf{Th}_j(q_f)$ is true, for some final state q_f. Note that the number of possible sequences are bounded(and a function of size of the DFA). Figure 2 illustrates the threads and merging. Let Run be the formula obtained by conjuncting all formulae explained above. This captures the run information of $\mathcal{A}_{\mathsf{atom}}$. The formula Run then correctly captures the run information on ρ.

We can easily write a 1- TPTL formula that will check the truth of $\mathsf{Reg}_{[l,u)}\mathsf{atom}$ at a point v on the simple extension ρ' (see Appendix A). However, to write an MTL formula that checks the truth of $\mathsf{Reg}_{[l,u)}\mathsf{atom}$ at a point v, we need to oversample ρ' as shown below.

▶ **Lemma 4.** *Let $T = \Box^{\mathsf{ns}}[a \leftrightarrow \mathsf{Reg}_I\mathsf{atom}]$ be a temporal definition built from $\Sigma \cup W$. Then we synthesize a formula $\psi \in \mathsf{MTL}$ over $\Sigma \cup W \cup X$ such that $T \equiv \exists \downarrow X.\psi$.*

Proof. Lets first consider the case when the interval I is bounded of the form $[l, u)$. Starting with the simple extension ρ' having the information about the runs of $\mathcal{A}_{\mathsf{atom}}$, we explain the construction of the oversampled extension ρ'' as follows:

- We first oversample ρ' at all the integer timestamps and mark them with propositions in $C = \{c_0, \dots, c_{max-1}\}$ where max is the maximum constant used in timing constraints of the input formulae. An integer timestamp k is marked c_i if and only if $k = M(max) + i$ where $M(max)$ denotes a non-negative integral multiple of max and $0 \leq i \leq max - 1$. This can be done easily by the formula $c_0 \wedge \bigwedge_{i \in \{0,\dots max-1\}} \Box^{\mathsf{ns}}(c_i \rightarrow \neg\Diamond_{(0,1)}(\bigvee C) \wedge \Diamond_{(0,1]} c_{i\oplus 1})$ where $x \oplus y$ is addition of x, y modulo max.
- Next, a new point marked ovs is introduced at all time points τ whenever $\tau - l$ or $\tau - u$ is marked with $\bigvee \Sigma$. This ensures that for any time point t in ρ'', the points $t + l, t + u$ are also available in ρ''.

After the addition of integer time points, and points marked ovs, we obtain the oversampled extension $(\Sigma \cup W \cup \mathsf{Threads} \cup \mathsf{Merge}, C \cup \{\mathsf{ovs}\})$ ρ'' of ρ'.

To check the truth of $\mathsf{Reg}_{[l,u)}\mathsf{atom}$ at a point v, we need to assert the following: starting from the time point $\tau_v + l$, we have to check the existence of an accepting run R in $\mathcal{A}_{\mathsf{atom}}$ such that the run starts from the first action point in the interval $[\tau_v + l, \tau_v + u)$, is a valid run which goes through some possible sequence of merging of threads, and witnesses a final state at the last action point in $[\tau_v + l, \tau_v + u)$. To capture this, we start at the first action point in $[\tau_v + l, \tau_v + u)$ with initial state q_1 in some thread Th_i, and proceed for some time with Th_i active, until we reach a point where Th_i is merged with some Th_{i_1}. This is followed by Th_{i_1} remaining active until we reach a point where Th_{i_1} is merged with some other thread Th_{i_2} and so on, until we reach the last such merge where some thread say Th_n witnesses a final state at the last action point in $[\tau_v + l, \tau_v + u)$. A nesting of until formulae captures this sequence of merges of the threads, starting with Th_i in the initial state q_1. Starting at v, we have the point marked ovs at $\tau_v + l$, which helps us to anchor there and start asserting the existence of the run.

The issue is that the nested until can not keep track of the time elapse since $\tau_v + l$. However, note that the greatest integer point in $[\tau_v + l, \tau_v + u)$ is uniquely marked with $c_{i\oplus u}$ whenever $c_i \leq \tau_v \leq c_{i\oplus 1}$ are the closest integer points to τ_v. We make use of this by (i)

asserting the run of $\mathcal{A}_{\mathsf{atom}}$ until we reach $c_{i\oplus u}$ from $\tau_v + l$. Let the part of the run R that has been witnessed until $c_{i\oplus u}$ be R_{pref}. Let $R = R_{pref}.R_{suf}$ be the accepting run. (ii) From $\tau_v + l$, we jump to $\tau_v + u$, and assert the reverse of R_{suf} till we reach $c_{i\oplus u}$. This ensures that $R = R_{pref}.R_{suf}$ is a valid run in the interval $[\tau_v + l, \tau_v + u)$. Let $\mathsf{Mrg}(i) = [\bigvee_{j<i} \nwarrow\!\!\!\!\nwarrow(j,i) \vee c_{i\oplus u}]$.

We first write a formula that captures R_{pref}. Given a point v, the formula captures a sequence of merges through threads $i > i_1 > \cdots > i_{k_1}$, and m is the number of states of $\mathcal{A}_{\mathsf{atom}}$.

Let $\varphi_{Pref,k_1} = \bigvee_{m\geq i>i_1>\cdots>i_{k_1}} \mathsf{MergeseqPref}(k_1)$ where $\mathsf{MergeseqPref}(k_1)$ is the formula

$$\Diamond_{[l,l]}\{\neg(\bigvee \Sigma \vee c_{i\oplus u})\ \mathsf{U}[\mathsf{Th}_i(q_1) \wedge (\neg\mathsf{Mrg}(i)\ \mathsf{U}[\nwarrow\!\!\!\!\nwarrow(i_1,i)\wedge$$

$$(\neg\mathsf{Mrg}(i_1)\ \mathsf{U}[\nwarrow\!\!\!\!\nwarrow(i_2,i_1) \wedge \ldots (\neg\mathsf{Mrg}(i_{k_1})\ \mathsf{U}c_{i\oplus u})])])]\}$$

Note that this asserts the existence of a run till $c_{i\oplus u}$ going through a sequence of merges starting at $\tau_v + l$. Also, $\mathsf{Th}_{i_{k_1}}$ is the guessed last active thread till we reach $c_{i\oplus u}$ which will be merged in the continuation of the run from $c_{i\oplus u}$.

Now we start at $\tau_v + u$ and assert that we witness a final state sometime as part of some thread Th_{i_k}, and walk backwards such that some thread i_t got merged to i_k, and so on, we reach a thread Th_{i_c} to which thread $\mathsf{Th}_{i_{k_1}}$ merges with. Note that $\mathsf{Th}_{i_{k_1}}$ was active when we reached $c_{i\oplus u}$. This thread $\mathsf{Th}_{i_{k_1}}$ is thus the "linking point" of the forward and reverse runs. See Figure 3.

Let $\varphi_{Suf,k,k_1} = \bigvee_{1\leq i_k<\cdots<i_{k_1}\leq m} \mathsf{MergeseqSuf}(k,k_1)$ where $\mathsf{MergeseqSuf}(k,k_1)$ is the formula $\Diamond_{[u,u]}\{\neg(\bigvee \Sigma \vee c_{i\oplus u})\mathsf{S}[(\mathsf{Th}_{i_k}(q_f)) \wedge (\neg\mathsf{Mrg}(i_k)\mathsf{S}\ [\nwarrow\!\!\!\!\nwarrow(i_k,i_{k-1}) \wedge (\neg\mathsf{Mrg}(i_{k-1})\mathsf{S}[\nwarrow\!\!\!\!\nwarrow(i_{k-1},i_{k-2}) \wedge \cdots \nwarrow\!\!\!\!\nwarrow (i_c,i_{k_1}) \wedge (\neg\mathsf{Mrg}(i_{k_1})\ \mathsf{S}c_{i\oplus u})])])]\}$. For a fixed sequence of merges, the formula $\varphi_{k,k_1} = \bigvee_{k\geq k_1\geq 1}[\mathsf{MergeseqPref}(k_1) \wedge \mathsf{MergeseqSuf}(k,k_1)]$ captures an accepting run using the merge sequence. Disjuncting over all possible sequences for a starting thread Th_i, and disjuncting over all possible starting threads gives the required formula capturing an accepting run. Note that this resultant formulae is also relativized with respect to Σ and also conjuncted with $Rel(\Sigma, \mathsf{Run})$ (where Run is the formula capturing the run information in ρ' as seen in section 3.2) to obtain the equisatisfiable MTL formula. Note that S can be eliminated obtaining an equisatisfiable $\mathsf{MTL}[\,\mathsf{U}_I]$ formula modulo simple projections [12].

If I was an unbounded interval of the form $[l,\infty)$, then in formula φ_{k,k_1}, we do not require $\mathsf{MergeseqSuf}(k,k_1)$; instead, we will go all the way till the end of the word, and assert $\mathsf{Th}_{i_k}(q_f)$ at the last action point of the word. Thus, for unbounded intervals, we do not need any oversampling at integer points. ◀

▶ **Lemma 5.** *Let $T = \Box^{\mathsf{ns}}[a \leftrightarrow x\mathsf{UReg}_{I,\mathsf{re}}y]$ be a built from $\Sigma \cup W$. Then we synthesize a formula $\psi \in \mathsf{MTL}$ over $\Sigma \cup W \cup X$ such that $T \equiv \exists \downarrow X.\psi$.*

Proof. We discuss the case of bounded intervals here; the unbounded interval case can be seen in Appendix B. The proof technique is very similar to Lemma 4. The differences that arise are as below.

1. Checking re in $\mathsf{Reg}_I\mathsf{re}$ at point v is done at all points j such that $\tau_j - \tau_v \in I$. To ensure this, we needed the punctual modalities $\Diamond_{[u,u]}, \Diamond_{[l,l]}$. On the other hand, to check $\mathsf{UReg}_{I,\mathsf{re}}$ from a point v, the check on re is done from the first point after τ_v, and ends at some point within $[\tau_v + l, \tau_v + u)$. Assuming τ_v lies between integer points $c_i, c_{i\oplus 1}$, we can witness the forward run in $\mathsf{MergeseqPref}$ from the next point after τ_v till $c_{i\oplus 1}$, and for the reverse run, go to some point in $\tau_v + I$ where the final state is witnessed, and walk back till $c_{i\oplus 1}$. The punctual modalities are hence not required and we do not need points marked ovs.

2. The formulae $\mathsf{MergeseqPref}(k_1)$, $\mathsf{MergeseqSuf}(k, k_1)$ of the lemma 4 are replaced as follows:
   - $\mathsf{MergeseqPref}(k_1) : \{\neg(\bigvee \Sigma \vee c_{i\oplus 1})\ \mathsf{U}[\mathsf{Th}_i(q_1) \wedge (\neg\mathsf{Mrg}(i)\ \mathsf{U}[\mathbb{M}(i_1, i) \wedge (\neg\mathsf{Mrg}(i_1)\ \mathsf{U}\ [\mathbb{M}(i_2, i_1) \wedge \ldots (\neg\mathsf{Mrg}(i_{k_1})\ \mathsf{U}c_{i\oplus 1})])])]\}$.
   - $\mathsf{MergeseqSuf}(k, k_1) : \Diamond_I\{[(\mathsf{Th}_{i_k}(q_f)) \wedge (\neg\mathsf{Mrg}(i_k)\mathsf{S}\ [\mathbb{M}(i_k, i_{k-1}) \wedge (\neg\mathsf{Mrg}(i_{k-1})\mathsf{S}[\mathbb{M}(i_{k-1}, i_{k-2}) \wedge \cdots \mathbb{M}\ (i_c, i_{k_1}) \wedge (\neg\mathsf{Mrg}(i_{k_1})\ \mathsf{S}c_{i\oplus 1})])])]\}$

The above takes care of re in $x\mathsf{UReg}_{I,\mathsf{re}}y$: we also need to say that x holds continously from the current point to some point in I. This is done by pushing x into re (see the translation of $\varphi_1\mathsf{UReg}_{I,\mathsf{re}}\varphi_2$ to $\mathsf{Reg}_I\mathsf{re}'$ in Appendix C). The resultant formulae is relativized with respect to Σ and also conjuncted with $Rel(\Sigma, \mathsf{Run})$ to obtain the equisatisfiable MTL formula. ◀

The equisatisfiable reduction in Lemma 5 above hence gives an elementary upper bound for satisfiability checking when we work on MITL with UReg, since after elimination of UReg, we obtain an equisatisfiable MITL formula. This is very interesting since it shows an application of the oversampling technique : without oversampling, we can eliminate UReg using 1-TPTL as shown in Appendix A. However, 1-TPTL does not enjoy the benefits of non-punctuality. In particular, Appendix F.2 shows that non punctual 1-TPTL is already non-primitive recursive.

3.3 Complexity

In this section, we discuss the complexity of $\mathsf{MITL}_{\mathsf{mod}}[\mathsf{MC}]$, proving Theorem 2.4. To prove this, we obtain a reduction from the reachability problem of Insertion Channel Machines with Emptiness Testing (ICMET). We now show how to encode the reachability problem of ICMET in $\mathsf{MITL}_{\mathsf{mod}}[\mathsf{MC}]$.

ICMET

A channel machine is a tuple $\mathcal{A} = (S, M, \Delta, C)$ where S is a finite set of states, M is a finite channel alphabet, C is a finite set of channel names, and $\Delta \subseteq S \times Op \times S$ is the transition relation, where $Op = \{c!a, c?a, c = \epsilon \mid c \in C, a \in M\}$ is the set of transition operations. $c!a$ corresponds to writing message a to the tail of channel c, $c?a$ denotes reading the message a from the head of channel c, and $c = \epsilon$ tests channel c for emptiness.

We first define error-free channel machines. Given $\mathcal{A}$ as above, a configuration of $\mathcal{A}$ is a pair (q, U) where $q \in S$ and $U \in (M^*)^C$ gives the contents of each channel. Let Conf denote the configurations of $\mathcal{A}$. The rules in Δ induce an Op-labelled transition relation on Conf, as follows.

(a) $(q, c!a, q') \in \Delta$ yields a transition $(q, U) \xrightarrow{c!a} (q', U')$ where $U'(c) = U(c).a$, and $U'(d) = U(d)$ for $d \neq c$.

(b) $(q, c?a, q') \in \Delta$ yields a transition $(q, U) \xrightarrow{c?a} (q', U')$ where $U(c) = a.U'(c)$, and $U'(d) = U(d)$ for $d \neq c$.

(c) $(q, c = \epsilon, q') \in \Delta$ yields a transition $(q, U) \xrightarrow{c=\epsilon} (q', U')$ provided $U(c)$ is the empty word. All other channel contents remain the same.

If the only transitions allowed are as above, then we call $\mathcal{A}$ an error-free channel-machine. Error-free channel machines are Turing-powerful. We now look at channel machines with insertion errors. These augment the transition relation on Conf with the following rule:

(d) Insertion errors are then introduced by extending the transition relation on global states with the following clause: if $(q, U) \xrightarrow{\alpha} (q', V)$, and if $U' \sqsubseteq U$ and $V \sqsubseteq V'$, then $(q, U') \xrightarrow{\alpha} (q', V')$. $U' \sqsubseteq U$ if U' can be obtained from U by deleting any number of letters.

The channel machines as above are called ICMET. A run of an ICMET is a sequence of transitions $\gamma_0 \overset{op_0}{\rightarrow} \gamma_1 \cdots \overset{op_{n-1}}{\rightarrow} \gamma_n \ldots$ that is consistent with the above operational semantics.

Consider any ICMET $\mathcal{C} = (S, M, \Delta, C)$, with set of states $S = \{s_0, \ldots, s_n\}$ and channels $C = \{c_1, \ldots, c_k\}$. Let M be a finite set of messages used for communication in the channels.

We encode the set of all possible configurations of $\mathcal{C}$, with a timed language over the alphabet $\Sigma = M_a \cup M_b \cup \Delta \cup S \cup \{H\}$, where $M_a = \{m_a | m \in M\}$ $M_b = \{m_b | m \in M\}$, and H is a new symbol.

1. The jth configuration for $j \geq 0$ is encoded in the interval $[(2k+2)j, (2k+2)(j+1)-1)$ where k refers to number of channels.
2. At time $(2k+2)j + (2k-1)$, the current state s_w of the ICMET at configuration j is encoded by the truth of the proposition s_w.
3. The jth configuration begins at the time point $(2k+2)j$. At a distance $[2i-1, 2i]$ from this point, $1 \leq i \leq k$, the contents of the i^{th} channel are encoded as shown in the point 7. The intervals of the form $(2i, 2i+1)$, $0 \leq i \leq k+1$ from the start of any configuration are time intervals within which no action takes place.
4. Lets look at the encoding of the contents of channel i in the jth configuration. Let m_{h_i} be the message at the head of the channel i. Each message m_i is encoded using consecutive occurrences of symbols $m_{i,a}$ and $m_{i,b}$. In our encoding of channel i, the first point marked $m_{h_i,a}$ in the interval $(2k+2)j + [2i-1, 2i]$ is the head of the channel i and denotes that m_{h_i} is the message at the head of the channel. The last point marked $m_{t_i,b}$ in the interval is the tail of the channel, and denotes that message m_{t_i} is the message stored at the tail of the channel.
5. Exactly at $2k+1$ time units after the start of the j^{th} configuration, we encode the transition from the state at the j^{th} configuration to the $(j+1)^{st}$ configuration (starts at $(2k+2)(j+1)$). Note that the transition has the form $(s, c!m, s')$ or $(s, c?m, s')$ or $(s, c = \epsilon, s')$.
6. We introduce a special symbol H, which acts as separator between the head of the message and the remaining contents, for each channel.
7. A sequence of messages $w_1 w_2 w_3 \ldots w_z$ in any channel is encoded as a sequence $w_{1,a} w_{1,b} H w_{2,a} w_{2,b} w_{3,a} w_{3,b} \ldots w_{z,a} w_{z,b}$.

Let $S = \bigvee_{i=0}^{n} s_i$ denote the states of the ICMET, $\alpha = \bigvee_{i=0}^{m} \alpha_i$, denote the transitions α_i of the form $(s, c!m, s')$ or $(s, c?m, s')$ or $(s, c = \epsilon, s')$. Let $action = true$ and let $M_a = \bigvee_{m_x \in M}(m_{x,a})$, $M_b = \bigvee_{m_x \in M}(m_{x,b})$, with $M = M_a \vee M_b$.

1. All the states must be at distance $2k+2$ from the previous state (first one being at 0) and all the propositions encoding transitions must be at the distance $2k+1$ from the start of the configuration.

   $$\varphi_S = s_0 \wedge \Box[S \Rightarrow \{\Diamond_{(0,2k+2]}(S) \wedge \Box_{(0,2k+2)}(\neg S) \wedge \Diamond_{(0,2k+1]}\alpha \wedge \Box_{[0,2k+1)}(\neg\alpha) \wedge \Diamond_{(2k+1,2k+2)}(\neg\alpha)\}]$$

2. All the messages are in the interval $[2i-1, 2i]$ from the start of configuration. No action takes place at $(2i-2, 2i-1)$ from the start of any configuration.

   $$\varphi_m = \Box\{S \Rightarrow \bigwedge_{i=1}^{k} \Box_{[2i-1,2i]}(M \vee H) \wedge \Box_{(2i-2,2i-1)}(\neg action)\}$$

3. Consecutive source and target states must be in accordance with a transition α. For example, s_j appears consecutively after s_i reading α_i iff α_i is of the form $(s_i, y, s_j) \in \Delta$, with $y \in \{c_i!m, c_i?m, c_i = \emptyset\}$.

   $\varphi_\Delta = \bigwedge_{s,s' \in S} \Box\{(s \wedge \Diamond_{(0,2k+2]} s') \Rightarrow (\Diamond_{(0,2k+1]} \bigvee \Delta_{s,s'})\}$ where $\Delta_{s,s'}$ are possible α_i between s, s'.

4. We introduce a special symbol H along with other channel contents which acts as a separator between the head of the channel and rest of the contents. Thus H has following properties
   - There is one and only one time-stamp in the interval $(2i-1, 2i)$ from the start of the configuration where H is true. The following formula says that there is an occurrence of a H:

     $\varphi_{H_1}{=}\Box[(S{\wedge}\Diamond_{(2i-1,2i)}M){\Rightarrow}(\bigwedge_{i=1}^{k}\Diamond_{(2i-1,2i)}(H))]$

     The following formula says that there can be only one H: $\varphi_{H_2}{=}\Box(H{\Rightarrow}\neg\Diamond_{(0,1)}H)$
   - Every message m_x is encoded by truth of proposition $m_{x,a}$ immediately followed by $m_{x,b}$. Thus for any message m_x, the configuration encoding the channel contents has a sub-string of the form $(m_{x,a}m_{x,b})^*$ where m_x is some message in M.

     $\varphi_m{=}\Box[m_{x,a}{\Rightarrow}\mathsf{O}_{(0,1]}m_{x,b}]{\wedge}\Box[m_{x,b}{\Rightarrow}\mathsf{O}_{(0,1)}M_a\ {\vee}\mathsf{O}(\bigvee\Delta\vee H)]{\wedge}(\neg M_b\ \mathsf{U}M_a)$
   - If the channel is not empty (there is at least one message m_am_b in the interval $(2i{-}1, 2i)$ corresponding to channel i contents) then there is one and only one m_b before H. The following formula says that there can be at most one m_b before H.

     $\varphi_{H_3}{=}\Box[\neg\{M_b\wedge\Diamond_{(0,1)}(M_a\wedge\Diamond_{(0,1)}H)\}]$

     The following formula says that there is one M_b before H in the channel, if the channel is non-empty.

     $\varphi_{H_4}{=}\Box[S{\Rightarrow}\{\bigwedge_{j=1}^{k}(\Diamond_{[2j-1,2j]}(M_b){\Rightarrow}\ \Diamond_{[2j-1,2j]}(M_b\wedge\Diamond_{(0,1)}H))\}]$

     Let $\varphi_H{=}\varphi_{H_1}\wedge\varphi_{H_2}\wedge\varphi_{H_3}\wedge\varphi_{H_4}$.
5. Encoding transitions:
   - We first define a macro for copying the contents of the i^{th} channel to the next configuration with insertion errors. If there were some $m_{x,a}, m_{x,b}$ at times t, t', $m_{x,b}$ is copied to $t''+2k+2$ (where $t''\in[t,t')$), representing the channel contents in the next configuration. This is specified by means of an even count check.
     - From any 3 consecutive points u, v, w such that $m_{x,a}$ and $m_{x,b}$ are true at v and w, respectively, if there are even (or odd) number of $m_{x,b}$ within $(0, 2k+2)$ from both v and w , then there must be odd number of $m_{x,b}$'s within time interval $[\tau_v+2k+2, \tau_w+2k+2]$. Thus there must be at least one $m_{x,b}$ copied from the point w to some point in the interval $[\tau_v+2k+2, \tau_w+2k+2]$. The rest of the even number of erroneous $m_{x,b}$ in $[\tau_v+2k+2, \tau_w+2k+2]$, along with the arbitrary insertion errors within $[\tau_u+2k+2, \tau_v+2k+2]$ models the insertion error of the ICMET. The formula copy_g is as follows.

     $\Box_{[2g-1,2g]}[\bigwedge_{m_x\in M}(m_{x,a}{\wedge}\mathsf{iseven}_{(0,2k+2)}(m_{x,b})){\Rightarrow}\mathsf{O}(\mathsf{iseven}_{(0,2k+2)}(m_{x,b}))]$
     $\wedge\Box_{[2i-1,2i]}[\bigwedge_{m_x\in M}(m_{x,a}{\wedge}\neg\mathsf{iseven}_{(0,2k+2)}(m_{x,b})){\Rightarrow}\mathsf{O}(\neg\mathsf{iseven}_{(0,2k+2)}(m_{x,b}))]$
   - If the transition is of the form $c_i=\epsilon$. The following formulae checks that there are no events in the interval $(2i-1, 2i)$ corresponding to channel i, while all the other channel contents are copied.

     $$\varphi_{c_i=\epsilon}{=}S\wedge\Box_{(2i-1,2i)}(\neg action){\wedge}\bigwedge_{g=1}^{k}\mathsf{copy}_g$$
   - If the transition is of the form $c_i!m_x$ where $m\in M$. An extra message is appended to the tail of channel i, and all the m_am_b's are copied to the next configuration. $M_b\wedge\Box_{(0,1)}(\neg M))$ denotes the last time point of channel i; if this occurs at time t, we know that this is copied at a timestamp strictly less than $2k+2+t$.Thus we assert that truth of $\Diamond_{(2k+2,2k+3)}m_{x,b}$ at t.

     $$\varphi_{c_i!m}{=}S{\wedge}\bigwedge_{g=1}^{k}\mathsf{copy}_g{\wedge}\Diamond_{[2i-1,2i)}\{(M{\wedge}\Box_{(0,1)}(\neg M)){\Rightarrow}(\Diamond_{(2k+2,2k+3)}(m_{x,b}))\}$$

- If the transition is of the form $c_i?m$ where $m \in M$. The contents of all channels other than i are copied to the intervals encoding corresponding channel contents in the next configuration. We also check the existence of a first message in channel i; such a message has a H at distance $(0,1)$ from it.

$$\varphi_{c_i?m_x} = S \wedge \bigwedge_{j\neq i, g=1}^{k} \mathsf{copy}_g \wedge \Diamond_{(2i-1,2i)}\{m_{x,b} \wedge \Diamond_{(0,1)}(H)\} \wedge \Box_{[2i-1,2i]}[\bigwedge_{m_x \in M}(m_{x,a} \wedge \mathsf{iseven}_{(0,2k+2)}(m_{x,b}) \wedge \neg\Diamond_{(0,1)}H) \Rightarrow \mathsf{O}(\mathsf{iseven}_{(0,2k+2)}(m_{x,b}))] \wedge \Box_{[2i-1,2i]}[\bigwedge_{m_x \in M}(m_{x,a} \wedge \neg\mathsf{iseven}_{(0,2k+2)}(m_{x,b}) \wedge \neg\Diamond_{(0,1)}H) \Rightarrow \mathsf{O}(\neg\mathsf{iseven}_{(0,2k+2)}(m_{x,b}))]$$

6. Channel contents must change in accordance to the relevant transition. Let L be a set of labels (names) for the transitions. Let $l \in L$ and α_l be a transition labeled l.

   $\varphi_C = \Box[S \Rightarrow \bigwedge_{l\in L}(\Diamond_{(0,2k+1]}(\bigvee \alpha_l \Rightarrow \phi_l))]$

   where ϕ_l are the formulae as seen in 5.
7. Let t be a state of the ICMET whose reachability we are interested in. Check s_t is reachable from s_0.
   $\phi_{reach} = \Diamond(s_t)$
   Thus the formula encoding ICMET is:

   $\varphi^3 = \varphi_S \wedge \varphi_\Delta \wedge \varphi_m \wedge \varphi_H \wedge \varphi_C \wedge \varphi_{reach}$

4 Main Equivalences

In this section, we discuss the two equivalences : the equivalence between po-1-clock ATA and 1−TPTL, and that between po-1-clock ATA and SfrMTL. SfrMTL is the fragment of RegMTL where the regular expressions are all star-free. This gives the equivalence between 1−TPTL and SfrMTL.

4.1 Automaton-Logic Characterization

In this section, we show that partially ordered 1-clock alternating timed automata (po-1-clock ATA) capture exactly the same class of languages as 1−TPTL. We also show that 1−TPTL is equivalent to the class RegMTL where the regular expressions re involved in the formulae are star-free. We denote by SfrMTL this subclass RegMTL. This also shows for the first time in pointwise timed logics, an equivalence between freeze point logics and logics with interval constraints.

A 1-clock ATA [10] is a tuple $\mathcal{A} = (\Sigma, S, s_0, F, \delta)$, where Σ is a finite alphabet, S is a finite set of locations, $s_0 \in S$ is the initial location and $F \subseteq S$ is the set of final locations. Let x denote the clock variable in the 1-clock ATA, and $x \bowtie c$ denote a clock constraint where $c \in \mathbb{N}$ and $\bowtie \in \{<, \leq, >, \geq\}$. Let X denote a finite set of clock constraints of the form $x \bowtie c$. The transition function is defined as $\delta : S \times \Sigma \to \Phi(S \cup \Sigma \cup X)$ where $\Phi(S \cup \Sigma \cup X)$ is a set of formulae defined by the grammar below. Let $s \in S$. The grammar is defined as

$$\varphi ::= \top | \bot | \varphi_1 \wedge \varphi_2 | \varphi_1 \vee \varphi_2 | s | x \bowtie c | x.\varphi$$

$x.\varphi$ is a binding construct correspinding to resetting the clock x to 0.

The notation $\Phi(S \cup \Sigma \cup X)$ thus allows boolean combinations as defined above of locations, symbols of Σ, clock constraints and $\top, \bot$, with or without the binding construct $(x.)$. A configuration of a 1-clock ATA is a set consisting of locations along with their clock valuation. Given a configuration C, we denote by $\delta(C, a)$ the configuration D obtained by applying $\delta(s, a)$ to each location s such that $(s, \nu) \in C$. A run of the 1-clock ATA starts from the

initial configuration $\{(s_0, 0)\}$, and proceeds with alternating time elapse transitions and discrete transitions obtained on reading a symbol from Σ. A configuration is accepting iff it is either empty, or is of the form $\{(s, \nu) \mid s \in F\}$. The language accepted by a 1-clock ATA $\mathcal{A}$, denoted $L(\mathcal{A})$ is the set of all timed words ρ such that starting from $\{(s_0, 0)\}$, reading ρ leads to an accepting configuration. A po-1-clock ATA is one in which

- there is a partial order denoted $\prec$ on the locations, such that whenever s_j appears in $\Phi(s_i)$, $s_j \prec s_i$, or $s_j = s_i$. Let $\downarrow s_i = \{s_j \mid s_j \prec s_i\}$.
- $x.s$ does not appear in $\delta(s, a)$ for all $s \in S, a \in \Sigma$.

Example. Consider $\mathcal{A} = (\{a, b\}, \{s_0, s_a, s_\ell\}, s_0, \{s_0, s_\ell\}, \delta)$ with transitions $\delta(s_0, b) = s_0, \delta(s_0, a) = (s_0 \wedge x.s_a) \vee s_\ell$, $\delta(s_a, a) = (s_a \wedge x < 1) \vee (x > 1) = \delta(s_a, b)$, and $\delta(s_\ell, b) = s_\ell, \delta(s_\ell, a) = \bot$. The automaton accepts all strings where every non-last a has no symbols at distance 1 from it. Note that this is a po-1-clock ATA.

▶ **Lemma 6.** po-*1-clock ATA and* 1−TPTL *are equivalent in expressive power.*

4.1.1 po-1-clock ATA to 1−TPTL

In this section, we explain the algorithm which converts a po-1-clock ATA $\mathcal{A}$ into a 1−TPTL formula φ such that $L(\mathcal{A}) = L(\varphi)$. The translation from 1−TPTL to po-1-clock ATA is easy, as in the translation from MTL to po-1-clock ATA. We illustrate the key steps of the reverse direction, and apply it on the example above, while the step by step details can be seen in Appendix D. There are 4 main steps.

1. In step 1, we write each transition $\delta(s, a)$ into a disjunction $C_1 \vee C_2$ or C_1 or C_2, where $C_1 = s \wedge \varphi_1$, with $\varphi_1 \in \Phi(\downarrow s \cup \{a\} \cup X)$, and $C_2 = \varphi_2$, where $\varphi_2 \in \Phi(\downarrow s \cup \{a\} \cup X)$.
2. In step 2, we combine all transitions possible from a location s by disjuncting them, and denote the resultant as $\Delta(s)$. In the example above, we obtain $\Delta(s_0) = s_0 \wedge [(a \wedge x.s_a)) \vee b] \vee (a \wedge s_\ell)$.
3. In step 3, we take the first step towards obtaining a 1−TPTL formula corresponding to each location, by replacing all locations s' appearing in $\Delta(s)$ with $\mathsf{O}s'$. This is denoted $\mathcal{N}(s)$. Continuing with the example, we obtain $\mathcal{N}(s_0) = \mathsf{O}s_0 \wedge [(a \wedge x.\mathsf{O}s_a)) \vee b] \vee (a \wedge \mathsf{O}s_\ell)$, $\mathcal{N}(s_a) = (\mathsf{O}s_a \wedge x < 1) \vee (x > 1)$, $\mathcal{N}(s_\ell) = \mathsf{O}s_\ell \wedge b$.
4. In the last step, we solve each $\mathcal{N}(s)$ starting with the lowest location in the partial order. We make use of the fact that for the lowest locations s_n in the partial order, we have $\mathcal{N}(s_n) = (\mathsf{O}s_n \wedge \varphi_1) \vee \varphi_2$, where $\varphi_1, \varphi_2 \in \Phi(\Sigma, X)$. Hence, a solution to this, denoted $F(s_n)$ is $\varphi_1 \,\mathsf{W}\, \varphi_2$ if s_n is an accepting location, and as $\varphi_1 \,\mathsf{U}^{\mathsf{ns}}\, \varphi_2$ if s_n is non-accepting. This is recursively continued as we go up the partial order, where each $\mathcal{N}(s_i)$ has the form $(\mathsf{O}s_i \wedge \varphi_1) \vee \varphi_2$ such that $F(s')$ is computed for all locations s' appearing in φ_1, φ_2. Solving for s_i is then similar to that of s_n. $F(s_0)$ then gives the TPTL formula that we are looking for.
   In our example, $F(s_\ell) = \Box^{\mathsf{ns}} b$, $F(s_a) = x < 1 \,\mathsf{U}^{\mathsf{ns}}\, x > 1$. Finally, $F(s_0) = [(a \wedge x.\mathsf{O}F(s_a)) \vee b] \,\mathsf{W}\, (a \wedge \mathsf{O}F(s_\ell))$ as $((a \wedge (x.\mathsf{O}[(x < 1) \,\mathsf{U}^{\mathsf{ns}}\, x > 1])) \vee b) \,\mathsf{W}\, (a \wedge \mathsf{O}\Box^{\mathsf{ns}} b)$.

4.2 1−TPTL and SfrMTL

In this section, we prove the following result.

▶ **Theorem 7.** *1*−TPTL *and* SfrMTL *are equivalent.*

The proof uses Lemmas 8 and 9. We first show that starting from a SfrMTL formula φ, we can construct an equivalent 1−TPTL formula ψ.

▶ **Lemma 8.** SfrMTL $\subseteq 1-$ TPTL

The proof can be found in Appendix E. We illustrate the technique on an example here. *Example.* Consider $\varphi = \mathsf{Reg}_{(0,1)}[\mathsf{Reg}_{(1,2)}(a+b)^*]$. We first obtain $\mathsf{Reg}_{(0,1)}(w_{(0,1)} \wedge [\mathsf{Reg}_{(1,2)}(a+b)^*])$, followed by $\mathsf{Reg}_{(0,1)}(w_{(0,1)} \wedge w)$ where w is a witness for $\mathsf{Reg}_{(1,2)}(a+b)^*$. This is then rewritten as $\mathsf{Reg}((w_{(0,1)} \wedge w).(\neg w_{(0,1)})^*)$, and subsequently as $\mathsf{Reg}((x \in (0,1) \wedge w) \wedge \Box[\neg(x \in (0,1)])$. This is equivalent to $x.[x \in (0,1) \wedge w \wedge \Box[\neg(x \in (0,1)]]$. Now we replace w, and do one more application of this technique to obtain the formula $x.[x \in (0,1) \wedge [x.(\psi \wedge x \in (1,2) \wedge \Box[\neg(x \in (1,2))])] \wedge \Box[\neg(x \in (0,1))]]$, where ψ is the LTL formula equivalent to $(a+b)^*$.

4.3 po-1-clock ATA to SfrMTL

▶ **Lemma 9.** *Given a* po*-1-clock ATA* $\mathcal{A}$*, we can construct a* SfrMTL *formula* φ *such that* $L(\mathcal{A}) = L(\varphi)$.

Let $\mathcal{A}$ be a po-1-clock ATA with locations $S = \{s_0, s_1, \ldots, s_n\}$. Let K be the maximal constant used in the guards $x \sim c$ occurring in the transitions. The idea of the proof is to partition the behaviour of each location s_i across the regions $R_0{=}[0,0], R_1{=}(0,1), \ldots, R_{2K}{=}[K,K], R_K^+{=}(K,\infty)$ with respect to the last reset of the clock. Let $\mathcal{R}$ denote the set of regions. Let $R_h < R_g$ denote that the region R_h comes before R_g.

The behaviour in each region is captured as an LTL formula that is invariant in each region. From this, we obtain an SfrMTL formula that represents the behaviour starting from each region while at location s. The fact that the behaviours are captured by LTL formulae asserts the star-freeness of the regular expressions in the constructed RegMTL formulae. In the following, we describe this construction step by step. Let a behaviour distribution (BD) be described as a sequence of length $2K+1$ of the form $[\varphi_0, \varphi_1, \ldots, \varphi_{2K}]$ where each φ_i is a LTL formula (which does not evaluate to false) that is asserted in region R_i. For any location s in $\mathcal{A}$, and a region R we define a function that associates a set of possible behaviours. As seen in section 4.1.1, assume that we have computed $F(s)$ for all locations s. Let $F(S) = \{F(s) \mid s \in S\}$. Let $\mathcal{B}(F(S))$ represent the boolean closure of $F(S)$ (we require only conjunctions and disjunctions of elements from $F(S)$). We define $\mathsf{Beh} : \mathcal{B}(F(S)) \times \mathcal{R} \to 2^{\mathsf{BD}}$. Intuitively, $\mathsf{Beh}(F(s), R_i)$ provides all the possible behaviours in all the regions of $\mathcal{R}$, while asserting $F(s)$ at any point in R_i. Thus, $\mathsf{Beh}(F(s), R_i) = \{[\varphi_{g,0}, \ldots, \varphi_{g,2K}] \mid 1 \leq g \leq \alpha\}$, where α is a number that depends on the number of locations and the transitions of $\mathcal{A}$ and the maximal constant K. Now we describe the construction of $\mathsf{Beh}(F(s), R_i)$. If s is the lowest in the partial order, then $F(s)$ has the form $\varphi_1 \,\mathsf{W} \varphi_2$ or $\varphi_1 \,\mathsf{U}^{\mathsf{ns}} \varphi_2$, where φ_1, φ_2 are both disjunctions of conjunctions over $\Phi(\Sigma, X)$. Each conjunct has the form $\psi \wedge x \in I$ where $\psi \in \Phi(\Sigma)$ and $I \in \mathcal{R}$.

- Let s be a lowest location in the partial order. $F(s)$ then has the form

  $$\varphi{=}[(P_1 \wedge C_1) \vee (P_2 \wedge C_2) \ldots (P_n \wedge C_n)](\,\mathsf{U}|\,\mathsf{W})[(Q_1 \wedge E_1) \vee (Q_2 \wedge E_2) \ldots (Q_m \wedge E_m)]$$

  where P_i and Q_j are propositional formulae in $\Phi(\Sigma)$ and C_i and E_j are clock constraints. Without loss of generality, we assume that clock constraints are of the form $x \in R_y$, where $R_y \in \mathcal{R}$, and that no two C_i and no two E_j are the same. We now construct $\mathsf{Beh}(F(s), R)$ for $F(s)$.

  1. $\mathsf{Beh}(F(s), R_i) = \emptyset$ if and only if there are no constraints $x \in R_i$ in $F(s)$. This is because $F(s)$ does not allow any behaviour within R_i and $\mathsf{Beh}(F(s), R_i)$ asserts the behaviour when the clock valuation lies in R_i.
     Consider an $E_j = x \in R_y$ with $R_y \geq R_i$. For each such E_j, a behaviour BD is added to the set $\mathsf{Beh}(F(s), R_i)$ as follows.

- Assume that there exist some k such that $E_j = C_k$. In this case, the LTL formulae that is satisfied in region R_y is $P_k(\,\mathsf{U}|\,\mathsf{W})Q_j$. Thus the y^{th} element of the sequence is $P_k(\,\mathsf{U}|\,\mathsf{W})Q_j$.
- Assume that there is no C_k such that $E_j = C_k$. Then the LTL formula that is satisfied in R_y is Q_j. Thus the y^{th} element of the sequence is Q_j. For every sequence that has R_y as one of the above, we have:
  * The assertion in all regions $< R_i$ is $\top$ as there is no restriction on the region before the present point, since we only consider future temporal modalities. Similarly the formulae in regions $R_z > R_y$ are also set to $\top$ as there are no restrictions on the behaviour once we come out of the state s.
  * For all $C_g = x \in R_w$, where $R_y > R_w > R_i$, the region R_w will satisfy $\Box^{\mathsf{ns}} P_g \vee \Box^{\mathsf{ns}} \bot$. Thus the assertion in R_g in every sequence is $\Box^{\mathsf{ns}} P_g$ or $\Box^{\mathsf{ns}} \bot$, depending on whether or not we have points lying in R_g. Recall that $\Box^{\mathsf{ns}} \bot$ is the LTL formula whose only model is the empty word ϵ. If for some C_g such that $E_j \neq C_g$ and $C_g = x \in R_i$, then in region R_i, we assert $\Box^{\mathsf{ns}} P_g$. Thus the i^{th} entry is $\Box^{\mathsf{ns}} P_g$.
  * Note that all the remaining regions (if any), are between i and y. There is no behaviour allowed at this point. At these points $\Box^{\mathsf{ns}} \bot$ is true as only the empty string is accepted.

- **Boolean combinations of** Beh- Given two locations s_1, s_2, with $F(s_1) = \varphi_1$ and $F(s_2) = \varphi_2$, we construct $\mathsf{Beh}(F(s_1), R)$ and $\mathsf{Beh}(F(s_2), R)$ as shown above for all $R \in \mathcal{R}$. Given these Beh's we now define boolean operations $\wedge$ and $\vee$ on these sets, such that $\mathsf{Beh}(\varphi_1, R) \wedge \mathsf{Beh}(\varphi_2, R) = \mathsf{Beh}(\varphi_1 \wedge \varphi_2, R)$.
  1. For every $R_i \in \mathcal{R}$, we first take the cross product $\mathsf{Beh}(\varphi_1, R_i) \times \mathsf{Beh}(\varphi_2, R_i)$, obtaining a set consisting of ordered pairs of BDs. All the possible behaviours of $\varphi_1 \wedge \varphi_2$ starting in region R_i is equivalent to the conjunction of all possible behaviours of φ_1 conjuncted with all the possible behaviours of φ_2.
  2. For every pair $(\mathsf{BD}_1, \mathsf{BD}_2) \in \mathsf{Beh}(\varphi_1, R_i) \times \mathsf{Beh}(\varphi_2, R_i)$, construct a behaviour $\mathsf{BD}' \in \mathsf{Beh}(\varphi_1 \wedge \varphi_2, R_i)$ such that the i^{th} entry of BD' is equal to the conjunction of the i^{th} entry of BD_1 with that of BD_2. This will ensure that we take all the possible behaviours of $F(s_1)$ at region R_i and conjunct it with all the possible behaviours of $F(s_2)$ in the same region. In a similar way we can also compute the $\mathsf{Beh}(\varphi_1 \vee \varphi_2, R)$.
- **Elimination of nested** Beh: Given any $F(s)$ of the form

$$\varphi = [(P_1 \wedge C_1) \vee (P_2 \wedge C_2) \ldots (P_n \wedge C_n)](\,\mathsf{U}|\,\mathsf{W})[(Q_1 \wedge E_1) \vee (Q_2 \wedge E_2) \ldots (Q_m \wedge E_m)]$$

with $P_i, Q_j \in \Phi(\Sigma \cup \mathsf{O}S)$, and C_i, E_j being clock constraints of the form $x \in R$. Assume that we have calculated $\mathsf{Beh}(F(s_i), R)$ for all $s_i \in{\downarrow} s$. We construct $\mathsf{Beh}(F(s), R)$ as shown above. After the construction, there might be some propositions of the form $\mathsf{O}(s_j)$ as a conjunct in some of the BD's in $\mathsf{Beh}(F(s), R)$. This occurrence of s_j is eliminated by stitching $\mathsf{Beh}(F(s_j))$ with BD as follows:
  1. Given a sequence $\mathsf{BD} = [X_0, \ldots, X_{g-1}, Q_j \wedge \mathsf{O}(T_j), X_{g+1}, \ldots, X_{2K}]$, we show how to eliminate T_j in the g^{th} entry.
  2. There are $2K - g + 1$ possibilities, depending on which region $\geq g$ the next point lies with respect to $Q_j \wedge \mathsf{O}(T_j)$.
  3. Suppose the next point can be taken in R_g itself. This means that from the next point, all the possible behaviours described by $\mathsf{Beh}(F(T_j), R_g)$ would apply along with the behaviour in this sequence BD. Thus, we first take a cross product $\mathsf{BD} \times \mathsf{Beh}(F(T_j), R_g)$ which will give us pairs of sequences of the form $[X_0, \ldots, X_{g-1}, Q_j \wedge \mathsf{O}(T_j), X_{g+1}, \ldots, X_{2K}], [Y_0, \ldots, Y_{2K}]$. We define a binary operation combine which combines two sequences. Let $[X'_0, \ldots, X'_{2K}]$ denote the combined sequence. To combine

the behaviours from the point where T_j was called, we substitute T_j with the LTL formula asserted at region R_g in $\mathsf{Beh}(F(T_j), R_g)$. This is done by replacing T_j with Y_g. For all $w < g$, $X'_w = X_w$. For all $w > g$, $X'_w = X_w \wedge Y_w$. Let the set of BDs obtained thus be called Seq_g.

4. Now consider the case when the next point is taken a region $> R_g$. In this case, we consider all the possible regions from R_{g+1} onwards. For every $b \in \{g+1, \ldots, 2K\}$ we do the following operation: we first take the cross product of $[X_0, \ldots, X_{g-1}, Q_j \wedge \mathsf{O}(T_j), X_{g+1}, \ldots, X_{2K}]$ and $\mathsf{Beh}(F(T_j), R_b)$. Consider any pair of sequences $[X_0, \ldots, X_{g-1}, Q_j \wedge \mathsf{O}(T_j), X_{g+1}, \ldots, X_{2K}], [Y_0, \ldots, Y_{2K}]$. We define an operation $\mathsf{stitch}(b)$ on this pair which gives us a sequence $[X'_0, \ldots, X'_{2K}]$. For all $w < g$, $X'_w = X_w$. For $w = g$, $X'_w = Q_j$. For all $b > w > g$, $X'_w = X_w \wedge \Box^{\mathsf{ns}} \bot$. This implies the next point from where the assertion $Q_j \wedge \mathsf{O}(T_j)$ was made is in a region $\geq R_b$. For all $w \geq b$, $X'_w = X_w \wedge Y_w$. This combines the assertions of both the behaviours from the next point onwards. Let the set of BDs we get in this case be $\mathsf{Seq}_{\geq g}$.
5. The final operation is to substitute $[X_0, \ldots, X_{g-1}, Q_j \wedge \mathsf{O}(T_j), X_{g+1}, \ldots, X_{2K}]$ with BDs from any of $\mathsf{Seq}_g, \mathsf{Seq}_{g+1}, \ldots, \mathsf{Seq}_{2K}$.
6. Note that a similar technique will work while eliminating U_j from $\mathsf{BD} = [X_0, \ldots, X_{g-1}, \Box^{\mathsf{ns}} P_j \wedge \mathsf{O}(U_j), X_{g+1}, \ldots, X_{2K}]$.

- Given $\mathsf{BD}=[X_0, \ldots, X_{g-1}, P_i \wedge \mathsf{O}(U_i)\ \mathsf{U}^{\mathsf{ns}} Q_j \wedge \mathsf{O}(T_j), X_{g+1}, \ldots, X_{2K}]$, we need to eliminate both the U_i and T_j. The formulae says either $Q_j \wedge \mathsf{O}(T_j)$ is true at the present point or, $P_i \wedge \mathsf{O}(U_i)$ true until some point in the future within the region R_g, when $Q_j \wedge \mathsf{O}(T_j)$ becomes true. Thus, we can substitute BD with two sequences $\mathsf{BD}_1=[X_0, \ldots, X_{g-1}, Q_j \wedge \mathsf{O}(T_j), X_{g+1}, \ldots, X_{2K}]$, and $\mathsf{BD}_2=[X_0, \ldots, X_{g-1},\ P_i \wedge \mathsf{O}(U_i)\ \mathsf{U} Q_j \wedge \mathsf{O}(T_j), X_{g+1}, \ldots, X_{2K}]$. We can eliminate T_j from BD_1 as shown before. Consider BD_2 which guarantees that the next point from which the assertion $P_i \wedge \mathsf{O}(U_i)\ \mathsf{U} Q_j \wedge \mathsf{O}(T_j)$ is made is within R_g, and that U_i is called for the last time within R_g. Such a BD_2 has to be combined with $\mathsf{Beh}(F(U_i), R_g)$. T_j can be called from any point either within region R_g or succeeding regions.
- Consider the case where T_j is called from within the region R_g. First let us take a cross product of BD with $\mathsf{Beh}(F(U_i), R_g) \times \mathsf{Beh}(F(T_j), R_g)$. This gives a triplet of sequences of the form $[X_0, \ldots, X_{g-1}, Q_j \wedge \mathsf{O}(T_j), X_{g+1}, \ldots, X_{2K}], [Y_{U,0}, \ldots, Y_{U,2K}], [Y_{T,0}, \ldots, Y_{T,2K}]$. We now show to combine the behaviours and get a sequence $[X'_0, \ldots, X'_{2K}]$.
  - For every $w < g$, $X'_w = X_w$. For $w = g$, X'_g is obtaining by replacing U_i with $Y_{U,g}$ and T_j with $Y_{T,g}$ in the gth entry of BD_2. For all $w > g$, $X'_w = X_w \wedge Y_{U,g} \wedge Y_{T,g}$. Let the set of these BDs be denoted Seq_g.
- Now consider the case where T_j was called from any region $R_b > R_g$. Take a cross product of BD with $\mathsf{Beh}(F(U_i), R_g) \times \mathsf{Beh}(F(T_j), R_b)$. This gives us triplets of the form $[X_0, \ldots, X_{g-1}, Q_j \wedge \mathsf{O}(T_j), X_{g+1}, \ldots, X_{2K}]$, $[Y_{U,0}, \ldots, Y_{U,2K}], [Y_{T,0}, \ldots, Y_{T,2K}]$. The one difference in combining this triplet as compared to the last one is that we have to assert that from the last point in R_g, the next point only occurs in the region R_b. Thus all the regions between R_g and R_b should be conjuncted with $\Box^{\mathsf{ns}} \bot$. We get a sequence $[X'_0, \ldots, X'_{2K}]$ after combining, such that
  - For all $w < g$, $X'_w = X_w$.
  - For $w = g$, $X'_g = (P_i \wedge \mathsf{O} Y_{U,g})\ \mathsf{U} Q_j$.
  - For $b > w > g$, $X'_w = X_w \wedge Y_{U,w} \wedge \Box^{\mathsf{ns}} \bot$.
  - For $w \geq b$, $X'_w = X_w \wedge Y_{U,g} \wedge Y_{T,g}$.
- In case of formulae of the form $[P_i \wedge \mathsf{O}(U_i)] \mathsf{W} [Q_j \wedge \mathsf{O}(T_j)]$ in R_g, we convert it into $(\alpha_1 \mathsf{U} \alpha_2) \vee \Box^{\mathsf{ns}} \alpha_1$ where $\alpha_1 = (P_i \wedge \mathsf{O}(U_i))$ and $\alpha_2 = (Q_j \wedge \mathsf{O}(T_j))$. Then BD is $[X_0, \ldots, X_{g-1}, [\alpha_1\ \mathsf{W} \alpha_2], X_{g+1}, \ldots, X_{2K}]$ and can be replaced by 2 BDs

- BD_1=$[X_0, \ldots, X_{g-1}, \alpha_1 \, \mathsf{U}^{\mathsf{ns}} \alpha_2, X_{g+1}, \ldots, X_{2K}]$
- BD_2=$[X_0, \ldots, X_{g-1}, \Box^{\mathsf{ns}}(\alpha_1), X_{g+1}, \ldots, X_{2K}]$.

For BD_1 and BD_2, we apply the operations defined previously.

- Finally, we show that given a Beh for $F(s)$, how to construct an SfrMTL formula, $\mathsf{Expr}(s)$, equivalent to $x.\mathsf{O}(s)$. That is, $\rho, i \models \mathsf{Expr}(s)$ if and only if $\rho, i, \nu \models x.\mathsf{O}(F(s))$, for any ν. We give a constructive proof as follows:
  - Assume $\rho, i, \nu \models x.\mathsf{O}(F(s))$. Note that according to the syntax of TPTL, every constraint $x \in I$ checks the time elapse between the last point where x was frozen. Thus satisfaction of formulae of the form $x.\phi$ at a point is independent of the clock valuation. $\rho, i, \nu \models x.\mathsf{O}(F(s))$ iff $\rho, i, \nu[x \leftarrow \tau_i] \models \mathsf{O}F(s)$. We have precomputed $\mathsf{Beh}(F(s), R)$ for all regions R. Thus, $\rho, i, \nu \models x.\mathsf{O}(F(s))$ iff for all $w \in 0, \ldots, 2K$, $\rho, i+1, \tau_i \models (x \in R_w)$. This implies that there exists $\mathsf{BD} \in \mathsf{Beh}(F(s), R_w)$ such that for all $j \in \{0, \ldots, 2K\}$, the jth entry $\mathsf{BD}[j]$ of BD is the LTL formulae satisfied within region R_j. Note that, $\rho, i+1, \tau_i \models (x \in R_w)$ is true, iff, $\rho, i \models \bigwedge_{g \in \{1,\ldots,w-1\}} [\mathsf{Reg}_{R_g} \emptyset] \wedge \mathsf{Reg}_{R_w} \Sigma^+$. This is true iff $\rho, i \models \bigvee_{\mathsf{BD} \in \mathsf{Beh}(F(s), R_w)} \bigwedge_{j \in \{1,\ldots,2K\}} \mathsf{Reg}_{R_j}(\mathsf{re}(\mathsf{BD}[j]))$, where $\mathsf{re}(\mathsf{BD}[j])$ is a regular expression equivalent to $\mathsf{BD}[j]$. As $\mathsf{BD}[j]$ is an LTL formula, the resultant expression will definitely have an equivalent star-free expression. Thus, $\rho, i, \nu \models x.\mathsf{O}(F(s))$, iff, $\rho, i \models (\psi_1 \rightarrow \psi_2)$ where
    - $\psi_1 = \bigwedge_{w \in \{0,\ldots,2K\} \setminus E} \bigwedge_{g \in \{1,\ldots,w-1\}} \mathsf{Reg}_{R_g} \emptyset \wedge \mathsf{Reg}_{R_w} \Sigma^+$ and
    - $\psi_2 = \bigvee_{\mathsf{BD} \in \mathsf{Beh}(F(s), R_w) j \in \{1,\ldots,2K\}} \bigwedge \mathsf{Reg}_{R_j}(\mathsf{re}(\mathsf{BD}[j]))\}]$.

  where E is the set of regions such that, for all $e \in E$, $\mathsf{Beh}(F(s), R_e)$ is an empty set. The SfrMTL formula $\mathsf{Expr}(s_0)$ is such that $\rho, 1 \models F(s_0)$ iff $\rho, 0 \models \mathsf{Expr}(s_0)$.

4.4 Discussion

1. **Generalization of other Extensions**: In this paper, we study extensions of MTL with ability to specify periodic properties by adding constructs which can specify regular expressions over subformulae within a time interval. This construct also generalizes most of the extensions studied in the literature (for example, Pnueli modalities, threshold counting, modulo counting and so on) still retaining decidability. To the best of our knowledge this is the most expressive decidable extension of MTL in the literature in point-wise semantics.
2. **Automaton Logic Connection**: We give an interval logic characterization for po-1-clock-ATA. The only other such equivalences we know of are [11] is between the logic MITL with only unary future and past modalities, restricted to unbounded intervals and partially ordered 2-way deterministic timed automata. Unlike interval logics, automata and logics with freeze quantifiers do not enjoy the perks of non punctuality see Appendix F.2.
3. **Interval Constraint vs. Freeze point quantification**: This was always an interesting question in the literature. Ours is the first such equivalence in point wise semantics. In continuous semantics, these logics are equivalent if we extend it with counting modality [4].
4. **Exploiting Non-punctuality**: We also give two natural non-punctual fragments $\mathsf{RegMITL}[\mathsf{UReg}]$ and $\mathsf{MITL}_{\mathsf{mod}}[\mathsf{UM}]$ of our logic having elementary complexity for satisfiability over both finite and infinite words proving the benefits of characterization using interval logics. We claim that these logics are the **most expressive** logics in pointwise

semantics which have **elementary** satisfiability checking for both finite and infinite timed words.

Finally, we show that if we allow mod counting within the next unit interval, we fail to achieve benefits of relaxing punctuality.

References

1. R. Alur, T. Feder, and T. Henzinger. The benefits of relaxing punctuality. *J.ACM*, 43(1):116–146, 1996.
2. Augustin Baziramwabo, Pierre McKenzie, and Denis Thérien. Modular temporal logic. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*, pages 344–351, 1999.
3. Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3), 2009.
4. P. Hunter. When is metric temporal logic expressively complete? In *CSL*, pages 380–394, 2013.
5. D. Kini, S. N. Krishna, and P. K. Pandya. On construction of safety signal automata for $\mathsf{MITL}[\mathcal{U}, \mathcal{S}]$ using temporal projections. In *FORMATS*, pages 225–239, 2011.
6. F. Laroussinie, A. Meyer, and E. Petonnet. Counting ltl. In *TIME*, pages 51–58, 2010.
7. K. Lodaya and A. V. Sreejith. Ltl can be more succinct. In *ATVA*, pages 245–258, 2010.
8. K. Madnani, S. N. Krishna, and P. K. Pandya. Partially punctual metric temporal logic is decidable. In *http://arxiv.org/abs/1404.6965*, 2014.
9. M. Minsky. *Finite and Infinite Machines*. Prentice-Hall, 1967.
10. J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *LICS*, pages 188–197, 2005.
11. Paritosh K. Pandya and Simoni S. Shah. The unary fragments of metric interval temporal logic: Bounded versus lower bound constraints. In *Automated Technology for Verification and Analysis - 10th International Symposium, ATVA 2012, Thiruvananthapuram, India, October 3-6, 2012. Proceedings*, pages 77–91, 2012.
12. Pavithra Prabhakar and Deepak D'Souza. On the expressiveness of MTL with past operators. In *FORMATS*, pages 322–336, 2006.
13. A. Rabinovich. Complexity of metric temporal logic with counting and pnueli modalities. In *FORMATS*, pages 93–108, 2008.
14. Jean Francois Raskin. *Logics, Automata and Classical Theories for Deciding Real Time*. PhD thesis, Universite de Namur, 1999.

Appendix

A 1-TPTL for $Reg_{[l,u)}$atom

To encode an accepting run going through a sequence of merges capturing $Reg_{[l,u)}$atom at a point e, we assert $\varphi_{chk1} \vee \varphi_{chk2}$ at e, assuming $l \neq 0$. If $l = 0$, we assert φ_{chk3}. Recall that m is the number of states in the minimal DFA accepting atom.

- Let $\mathsf{cond1} = 0 \leq n < m$, and
- Let $\mathsf{cond2} = 1 \leq i_1 < i_2 < \ldots < i_n < i \leq m$.
- $\varphi_{chk1} = \bigvee_{\mathsf{cond1}} \bigvee_{\mathsf{cond2}} x.\Diamond(x < l \wedge \mathsf{O}[(x \geq l) \wedge \mathsf{GoodRun}])$
- $\varphi_{chk2} = \bigvee_{\mathsf{cond1}} \bigvee_{\mathsf{cond2}} x.(\mathsf{O}[(x \geq l) \wedge \mathsf{GoodRun}])$
- $\varphi_{chk3} = \bigvee_{\mathsf{cond1}} \bigvee_{\mathsf{cond2}} x.\mathsf{GoodRun}$

where GoodRun is the formula which describes the run starting in q_1, going through a sequence of merges, and witnesses q_f at a point when $x \in [l, u)$, and is the maximal point in $[l, u)$. GoodRun is given by
$\mathsf{Th}_i(q_1) \wedge [\{\neg \mathsf{Mrg}(i)\}\ \mathsf{U}[\mathbb{M}(i_n, i) \wedge$
$\{\neg \mathsf{Mrg}(i_n)\}\ \mathsf{U}[\mathbb{M}(i_{n-1}, i_n) \ldots \{\neg \mathsf{Mrg}(i_2)\}\ \mathsf{U}[\mathbb{M}(i_1, i_2) \wedge \bigvee_{q \in Q_F} \mathsf{Nxt}(\mathsf{Th}_{i_1}(q)) \wedge x \in [l, u) \wedge \mathsf{O}(x > u)] \ldots]]]]$
where $\mathsf{Mrg}(i) = \bigvee_{j<i} \mathbb{M}(j, i)$.

The idea is to freeze the clock at the current point e, and start checking a good run from the first point in the interval $[l, u)$. φ_{chk1} is the case when the first point in $[l, u)$ is not the next point from the current point e, while φ_{chk2} handles the case when the next point is in $[l, u)$. In both cases, $l > 0$. Let Th_i be the thread having the initial state q_1 in the start of the interval I. Let i_1 be the index of the thread to which Th_i eventually merged (at the last point in the interval $[l, u)$ from e). The next expected state of thread Th_{i_1} is one of the final states if and only if the sub-string within the interval $[l, u)$ from the point e satisfies the regular expression atom. Note that when the frozen clock is $\geq l$, we start the run with $\mathsf{Th}_i(q_1)$, go through the merges, and check that $x \in I$ when we encounter a thread $\mathsf{Th}_{i_1}(q)$, with q being a final state. To ensure that we have covered checking all points in $\tau_e + I$, we ensure that at the next point after $\mathsf{Th}_{i_1}(q)$, $x > u$. The decidability of $1-$TPTL gives the decidability of RegMTL.

B Proof of Lemma 5 : Unbounded Intervals

The major challenge for the unbounded case is that the point from where we start asserting $\mathsf{Th}_i(q_f)$ (call this point w) and the point from where we start the counting, (this point is v) may be arbitrarily far. This may result in more than one point marked $c_{i\oplus 1}$. In the bounded interval case, the unique point marked $c_{i\oplus 1}$ was used as the "linking point" to stitch the sequences of the run after v till $c_{i\oplus 1}$, and from some point in $\tau_v + I$ witnessing a final state back to $c_{i\oplus 1}$. The possible non-uniqueness of $c_{i\oplus 1}$ thus poses a problem in reusing what we did in the bounded interval case. Thus we consider two cases:
Case 1: In this case, we assume that our point w lies within $[\tau_v + l, \lceil \tau_v + l \rceil)$. Note that $\lceil \tau_v + l \rceil$ is the nearest point from v marked with $c_{i\oplus l\oplus 1}$. This can be checked by asserting $\neg c_{i\oplus l\oplus 1}$ all the way till $c_{i\oplus 1}$ while walking backward from w, where $\mathsf{Th}_{i_k}(q_f)$ is witnessed.

The formula $\mathsf{MergeseqPref}(k_1)$ does not change. $\mathsf{MergeseqSuf}(k, k_1)$ is as follows:

$$\Diamond_{[l,l+1)}\{[(\mathsf{Th}_{i_k}(q_f)) \wedge (\neg\mathsf{Mrg}'(i_k)\ \mathsf{S}[\not\prec\!\!\prec(i_k, i_{k-1}) \wedge (\neg\mathsf{Mrg}'(i_{k-1})$$
$$\mathsf{S}[\not\prec\!\!\prec(i_{k-1}, i_{k-2}) \wedge \cdots \not\prec\!\!\prec(i_c, i_{k_1}) \wedge (\neg\mathsf{Mrg}'(i_{k_1})\ \mathsf{S}c_{i\oplus 1})]])]]\}$$

where $\mathsf{Mrg}'(i) = [\bigvee_{j<i} \not\prec\!\!\prec(j, i) \vee c_{i\oplus l\oplus 1}]$

Case 2: In this case, we assume the complement. That is the point w occurs after $\lceil \tau_v + l \rceil$. In this case, we assert the prefix till $c_{i\oplus l\oplus 1}$ and then continue asserting the suffix from this point in the forward fashion unlike other cases. The changed MergeseqPref and MergeseqSuf are as follows:

- $\mathsf{MergeseqPref}(k_1)$:

$$\{\neg(\bigvee \Sigma \vee c_{i\oplus l\oplus 1})\ \mathsf{U}[\mathsf{Th}_i(q_1) \wedge (\neg\mathsf{Mrg}(i)\ \mathsf{U}[\not\prec\!\!\prec(i_1, i) \wedge$$
$$(\neg\mathsf{Mrg}(i_1)\ \mathsf{U}[\not\prec\!\!\prec(i_2, i_1) \wedge \ldots (\neg\mathsf{Mrg}(i_{k_1})\ \mathsf{U}c_{i\oplus l\oplus 1})]])]]\}$$

- $\mathsf{MergeseqSuf}(k, k_1)$:

$$\Diamond_{[l+1,l+2)}\{[c_{i\oplus l\oplus 1} \wedge (\neg\mathsf{Mrg}(i_{k_1})\ \mathsf{U}[\not\prec\!\!\prec(i_c, i_{k_1}) \wedge (\neg\mathsf{Mrg}(i_c)$$
$$\mathsf{U}[\not\prec\!\!\prec(i_c, i_{k_1}) \wedge \cdots \not\prec\!\!\prec(i_{k-1}, i_{k-2}) \wedge (\neg\mathsf{Mrg}(i_{k-1})\ \mathsf{U}$$
$$(\mathsf{Th}_{i_k}(q_f))])])]\}$$ where $\mathsf{Mrg}(i) = [\bigvee_{j<i} \not\prec\!\!\prec(j, i)]$

B.1 Complexity of RegMTL Fragments

To prove the complexity results we need the following lemma.

▶ **Lemma 10.** *Given any* MITL *formulae φ with $\mathcal{O}(2^n)$ modalities and maximum constant used in timing intervals K, the satisfiability checking for φ is* EXPSPACE *in n, K.*

Proof. Given any MITL formula with $\mathsf{expn} = O(2^n)$ number of modalities, we give a satisfiability preserving reduction from φ to $\psi \in \mathsf{MITL}[\mathsf{U}_{0,\infty}, \mathsf{S}]$ as follows:

(a) Break each U_I formulae where I is a bounded interval, into disjunctions of U_{I_i} modality, where each I_i is a unit length interval and union of all I_i is equal to I. That is, $\phi_1\ \mathsf{U}_{\langle l,u\rangle}\phi_2 \equiv \phi_1\ \mathsf{U}_{\langle l,l+1)}\phi_2 \vee \phi_1\ \mathsf{U}_{[l+1,l+2)}\phi_2 \ldots \vee \phi_1\ \mathsf{U}_{[u-1,u\rangle}\phi_2$. This at most increases the number of modalities from expn to $\mathsf{expn} \times K$.

(b) Next, we flatten all the modalities containing bounded intervals. This results in replacing subformulae of the form $\phi_1\ \mathsf{U}_{[l,l+1)}\phi_2$ with new witness variables. This results in the conjunction of temporal definitions of the form $\Box^{\mathsf{ns}}[a \leftrightarrow \phi_1\ \mathsf{U}_{[l,l+1)}\phi_2]$ to the formula. This will result in linear blow up in number of temporal modalities $(2 \times \mathsf{expn} \times K)$.

(c) Now consider any temporal definition $\Box^{\mathsf{ns}}[a \leftrightarrow \phi_1\ \mathsf{U}_{[l,l+1)}\phi_2]$.

We show a reduction to an equisatisfiable MITL formula containing only intervals of the form $\langle 0, u\rangle$ or $\langle l, \infty)$.

- First we oversample the words at integer points $C = \{c_0, c_1, c_2, \ldots, c_{K-1}\}$. An integer timestamp k is marked c_i if and only if $k = M(K)+i$, where $M(K)$ denotes a non-negative integer multiple of K, and $0 \le i \le K-1$. This can be done easily by the formula

$$c_0 \wedge \bigwedge_{i\in\{0,\ldots K-1\}} \Box^{\mathsf{ns}}(c_i \rightarrow \neg\Diamond_{(0,1)}(\bigvee C) \wedge \Diamond_{(0,1]}c_{i\oplus 1})$$

where $x \oplus y$ is $(x+y)\%K$ (recall that $(x+y)\%K = M(K) + (x+y), 0 \le x+y \le K-1$).

- Consider any point i within a unit integer interval marked c_{i-1}, c_i. Then $\phi_1 \mathsf{U}_{[l,l+1)}\phi_2$ is true at that point i if and only if, ϕ_1 is true on all the action points till a point j in the future, such that
  - either j occurs within $[l, \infty)$ from i and there is no $c_{i\oplus l}$ between i and j ($\tau_j \in [\tau_i + l, \lceil \tau + l \rceil]$)
    $\phi_{C1,i} = (\phi_1 \wedge \neg c_{i\oplus l}) \mathsf{U}_{[l,\infty)}\phi_2$
  - or, j occurs within $[0, l+1)$ from i, and j is within a unit interval marked $c_{i\oplus l}$ and $c_{i\oplus l\oplus 1}$ ($\tau_j \in [\lceil \tau + l \rceil, \tau_i + l + 1)$).
    $\phi_{C2,i} = \phi_1 \mathsf{U}_{[0,l+1)}(\phi_2 \wedge (\neg(\bigvee C)) \mathsf{S} c_{i\oplus l})$

  The temporal definition $\Box^{\mathsf{ns}}[a \leftrightarrow \phi_1 \mathsf{U}_{[l,l+1)}\phi_2]$ is then captured by
  $\bigvee_{i=1}^{K-1} \Box^{\mathsf{ns}}[\{a \wedge (\neg(\bigvee C) \mathsf{U} c_i)\} \leftrightarrow \phi_{C1,i} \vee \phi_{C2,i}]$

To eliminate each bounded interval modality as seen (a)-(c) above, we need $\mathcal{O}(K)$ modalities. Thus the total number of modalities is $\mathcal{O}(2^n) \times \mathcal{O}(K) \times \mathcal{O}(K)$ and the total number of propositions $2^{\Sigma} \cup \{c_0, \ldots, c_{K-1}\}$. Assuming binary encoding for K, we get a $\mathsf{MITL}[\mathsf{U}_{0,\infty}, \mathsf{S}]$ formulae of exponential size. As the satisfiability checking for $\mathsf{MITL}[\mathsf{U}_{0,\infty}, \mathsf{S}]$ is in PSPACE [1], we get EXPSPACE upper bound. EXPSPACE hardness of MITL can be found in [1].

◀

B.2 Proof of Theorem 2.2

Starting from an $\mathsf{MITL}_{\mathsf{mod}}[\mathsf{UM}]$ formula, we first show how to obtain an equisatisfiable MITL formula modulo simple projections.

Elimination of UM

In this section, we show how to eliminate UM from $\mathsf{MTL}_{\mathsf{mod}}[\mathsf{UM}]$ over strictly monotonic timed words. This can be extended to weakly monotonic timed words. Given any $\mathsf{MTL}_{\mathsf{mod}}[\mathsf{UM}]$ formula φ over Σ, we first "flatten" the UM modalities of φ and obtain a flattened formula. *Example.* The formula $\varphi = [a \mathsf{U}(e \wedge (f \mathsf{U}_{(2,3),\#b=2\%5}y))]$ can be flattened by replacing the UM with a fresh witness proposition w to obtain
$\varphi_{flat} = [a \mathsf{U}(e \wedge w)] \wedge \Box^{\mathsf{ns}}\{w \leftrightarrow (f \mathsf{U}_{(2,3),\#b=2\%5}y)\}$.

Starting from $\chi \in \mathsf{MTL}_{\mathsf{mod}}[\mathsf{UM}]$, in the following, we now show how to obtain equisatisfiable MTL formulae corresponding to each temporal projection containing a UM modality.

1. *Flattening* : Flatten χ obtaining χ_{flat} over $\Sigma \cup W$, where W is the set of witness propositions used, $\Sigma \cap W = \emptyset$.
2. *Eliminate Counting* : Consider, one by one, each temporal definition T_i of χ_{flat}. Let $\Sigma_i = \Sigma \cup W \cup X_i$, where X_i is a set of fresh propositions, $X_i \cap X_j = \emptyset$ for $i \neq j$.
   - For each temporal projection T_i containing a UM modality of the form $x \mathsf{U}_{I,\#b=k\%_n y}$, Lemma 11 gives $\zeta_i \in \mathsf{MTL}$ over Σ_i such that $T_i \equiv \exists X_i.\zeta_i$.
3. *Putting it all together* : The formula $\zeta = \bigwedge_{i=1}^{k} \zeta_i \in \mathsf{MTL}$ is such that $\bigwedge_{i=1}^{k} T_i \equiv \exists X. \bigwedge_{i=1}^{k} \zeta_i$ where $X = \bigcup_{i=1}^{k} X_i$.

For elimination of UM, marking witnesses correctly is ensured using an extra set of symbols $B = \{b_0, ..., b_n\}$ which act as counters incremented in a circular fashion. Each time a witness of the formula which is being counted is encountered, the counter increments, else it remains same. The evaluation of the mod counting formulae can be reduced to checking the difference

between indices between the first and the last symbol in the time region where the counting constraint is checked.

B.2.1 Construction of Simple Extension

Consider a temporal definition $T = \Box^{\mathsf{ns}}[a \leftrightarrow x\mathsf{UM}_{I,\#b=k\%n}y]$, built from $\Sigma \cup W$. Let $\oplus$ denote addition modulo $n+1$.

1. *Construction of a $(\Sigma \cup W, B)$- simple extension.* We introduce a fresh set of propositions $B = \{b_0, b_1, \ldots, b_{n-1}\}$ and construct a family of simple extensions $\rho' = (\sigma', \tau)$ from $\rho = (\sigma, \tau)$ as follows:
   - $C1$: $\sigma'_1 = \sigma_1 \cup \{b_0\}$. If $b_k \in \sigma'_i$ and if $b \in \sigma_{i+1}$, $\sigma'_{i+1} = \sigma_{i+1} \cup \{b_{k\oplus 1}\}$.
   - $C2$: If $b_k \in \sigma'_i$ and $b \notin \sigma_{i+1}$, then $\sigma'_{i+1} = \sigma_{i+1} \cup \{b_k\}$.
   - $C3$: σ'_i has exactly one symbol from B for all $1 \leq i \leq |dom(\rho)|$.
2. *Formula specifying the above behaviour.* The variables in B help in counting the number of b's in ρ. $C1, C2$ and $C3$ are written in MTL as follows:
   - $\delta_1 = \bigwedge_{k=0}^{n} \Box^{\mathsf{ns}}[(\mathsf{O}b \wedge b_k) \rightarrow \mathsf{O}b_{k\oplus 1}]$ and
   - $\delta_2 = \bigwedge_{k=0}^{n} \Box^{\mathsf{ns}}[(\mathsf{O}\neg b \wedge b_k) \rightarrow \mathsf{O}b_k]$
   - $\delta_3 = \bigwedge_{k=0}^{n} \Box^{\mathsf{ns}}[b_k \rightarrow \bigwedge_{j \neq k} \neg b_j]$

▶ **Lemma 11.** *Consider a temporal definition $T = \Box^{\mathsf{ns}}[a \leftrightarrow x\,\mathsf{U}_{I,\#b=k\%n}y]$, built from $\Sigma \cup W$. Then we synthesize a formula $\psi \in$ MTL over $\Sigma \cup W \cup X$ such that $T \equiv \exists X.\psi$.*

Proof. 1. Construct a simple projection ρ' as shown in B.2.1.

2. Now checking whether at point i in ρ, $x\,\mathsf{U}_{I,\#b=k\%n}y$ is true, is equivalent to checking that at point i in ρ' there exist a point j in the future where y is true and for all the points between j and i, x is true and the difference between the index values of the symbols from B at i and j is $k\%n$. $\phi_{mark,a} = \Box^{\mathsf{ns}} \bigwedge_{i \in \{1,\ldots n-1\}} (a \wedge b_i \leftrightarrow [x\,\mathsf{U}_I(y \wedge b_j)])$ where $j = k + i\%n$.
3. The formula $\delta_1 \wedge \delta_2 \wedge \delta_3 \wedge \phi_{mark,a}$ is equivalent to T modulo simple projections.

◀

▶ **Lemma 12.** *Satisfiability of* $\mathsf{MITL}_{\mathsf{mod}}[\mathsf{UM}]$ *is* EXPSPACE*-complete.*

Proof. Assume that we have a $\mathsf{MITL}_{\mathsf{mod}}[\mathsf{UM}]$ formula ϕ with m UM modalities, alphabet Σ (number of propositions used is 2^{Σ}) and let K be the maximal constant appearing in the intervals of ϕ. Let $k_1\%n_1, \ldots, k_m\%n_m$ be the modulo counting entities in these UM formulae. Let n_{max} be the maximum of $n_1, \ldots, n_m$. Going by the construction above, we obtain m temporal definitions $T_1, \ldots, T_m$. To eliminate each T_i, we introduce n_{max} formulae of the form $\phi_{mark,a}$, evaluated on timed words over $2^{\Sigma} \cup B_1 \cup \cdots \cup B_m$. This is enforced by $\delta_1, \delta_2, \delta_3$. The number of propositions in the obtained MITL formula is hence $|2^{\Sigma}|.|B_1 + B_2 + \cdots + B_m|$, while the number of boolean connectives and temporal operators is $\mathcal{O}(m.n_{max})$, while the maximum constant appearing in the intervals is K. Thus we have an exponential size MITL formulae with max constant as K. Hence, by lemma 10, satisfiability checking for $\mathsf{MITL}_{\mathsf{mod}}[\mathsf{UM}]$ is EXPSPACE is complete. ◀

B.3 Proof of Theorem 2.3

Proof. In Lemma 5, temporal definitions of the form $\Box^{\mathsf{ns}}[a \leftrightarrow x\mathsf{UReg}_{I,\mathsf{re}}y]$ were eliminated to obtain equisatisfiable MTL formulae. The proof of Lemma 5 produces an equisatisfiable MITL formula if the input formulae does not contain punctual intervals. The alphabet contains sets Merge, Threads both of which have size m^2 where m is the size of the DFA corresponding to the regular expression re in the temporal definition $\Box^{\mathsf{ns}}[a \leftrightarrow x\mathsf{UReg}_{I,\mathsf{re}}y]$. m is exponential in the size of re. When it comes to the propositions marking the simple extension, at each point, we maintain $\leq m$ threads $\mathsf{Th}_1, \ldots, \mathsf{Th}_m$. The states on each thread are different, and thus, the propositions used in marking each position belongs to the set of sequences $\{Th_1(s_1)Th_2(s_2)\ldots Th_m(s_m) \mid s_i \neq s_j\}$. The number of propositions in the MITL formula is hence $\leq 2^{\Sigma} \times 2^{m^m}$. By construction, the size of the reduced MITL formulae will be of the order of $\mathcal{O}(2^{m.2^m})$. Note that the number of modalities (and nesting) in $\mathsf{MergeseqPref}(k_1)$ and $\mathsf{MergeseqSuf}(k, k_1)$ is at most m. The possible number of sequences over which φ_{Pref,k_1} and φ_{Suf,k,k_1} disjunct is $\mathcal{O}(m!)$. There are at most m^2 such formulae, giving a blow up of $\mathcal{O}(m!) \times \mathcal{O}(Poly(m))$. Thus the blow up is exponential $\mathcal{O}(2^m.m)$ with respect to m and thus doubly exponential with respect to the size of regular expression.

Applying the same reduction as shown in proof of Lemma 10 for the reduced MITL formulae will result in an equisatisfiable $\mathsf{MITL}[\,\mathsf{U}_{0\infty}, \mathsf{S}]$ of the size of $\mathcal{O}(2^{n.2^n})$. Using the PSPACE complexity of $\mathsf{MITL}[\,\mathsf{U}_{0\infty}, \mathsf{S}]$, we obtain a 2EXPSPACE upper bound for MITL[UReg]. Arriving at a tighter complexity for this class is an interesting problem and is open.

◀

C Details on Expressiveness

▶ **Theorem 13.** *1.* $\mathsf{RegMTL}[\mathsf{UReg}] \subseteq \mathsf{RegMTL}[\mathsf{Reg}]$
2. $\mathsf{MTL}_{\mathsf{mod}}[\mathsf{UM}] \subseteq \mathsf{MTL}_{\mathsf{mod}}[\mathsf{MC}]$

Proof. 1. We first prove $\mathsf{RegMTL}[\mathsf{UReg}] \subseteq \mathsf{RegMTL}[\mathsf{Reg}]$.

Note that $\phi_1\mathsf{UReg}_{I,\mathsf{re}}\phi_2 \equiv true\mathsf{UReg}_{I,\mathsf{re}'}\phi_2$, where re' is a regular expression obtained by conjuncting ϕ_1 to all formulae ψ occurring in the top level subformulae of re. For example, if we had $a\mathsf{UReg}_{(0,1),(\mathsf{Reg}_{(1,2)}[\mathsf{Reg}_{(2,3)}(b+c)^*])}d$, then we obtain $true\mathsf{UReg}_{(0,1),(a\wedge\mathsf{Reg}_{(1,2)}[\mathsf{Reg}_{(2,3)}(b+c)^*])}d$. When evaluated at a point i, the conjunction ensures that ϕ_1 holds good at all the points between i and j, where $\tau_j - \tau_i \in I$. To reduce $true\mathsf{UReg}_{I,\mathsf{re}'}\phi_2$ to a Reg_I formula, we need the following lemma.

▶ **Lemma 14.** *Given any regular expression R, there exist finitely many regular expressions $R_1^1, R_2^1, \ldots, R_1^n, R_2^n$ such that $R = \bigcup_{i=1}^n R_1^i.R_2^i$. That is, for any string $\sigma \in R$ and for any decomposition of σ as $\sigma_1.\sigma_2$, there exists some $i \leq n$ such that $\sigma_1 \in R_1^i$ and $\sigma_2 \in R_2^i$.*

Proof. Let $\mathcal{A}$ be the minimal DFA for R. Let the number of states in $\mathcal{A}$ be n. The set of strings that leads to some state q_i from the initial state q_1 is definable by a regular expression R_1^i. Likewise, the set of strings that lead from q_i to some final state of $\mathcal{A}$ is also definable by some regular expression R_2^i. Given that there are n states in the DFA $\mathcal{A}$, we have $L(\mathcal{A}) = \bigcup_{i=1}^n R_1^i.R_2^i$. Consider any string $\sigma \in L(\mathcal{A})$, and any arbitrary decomposition of σ as $\sigma_1.\sigma_2$. If we run the word σ_1 over $\mathcal{A}$, we might reach at some state q_i. Thus $\sigma_1 \in L(R_1^i)$. If we read σ_2 from q_i, it should lead us to one of the final states (by assumption that $\sigma \in R$) . Thus $\sigma_2 \in L(R_2^i)$. ◀

Lets consider $true\mathsf{UReg}_{I,\mathsf{re}'}\phi_2$ when $I = [l, l+1)$. Let Γ be the set of subformulae and their negations occurring in re'. When evaluating $true\mathsf{UReg}_{[l,l+1),\mathsf{re}'}\phi_2$ at a point i, we know that ϕ_2 holds good at some point j such that $\tau_j - \tau_i \in [l, l+1)$, and that $\mathsf{Seg}(\mathsf{re}', i, j) \in L(\mathsf{re}')$. We know that by the above lemma, any word $\sigma \in L(\mathsf{re}')$,for any decomposition $\sigma = \sigma_1.\sigma_2$, there exist an $i \in \{1, 2, \ldots, n\}$ such that $\sigma_1 \in L(R_1^i)$ and $\sigma_2 \in L(R_2^i)$. Thus we decompose at j' with every possible $R_1^k.R_2^k$ pair such that

- $\tau_{j'} \in \tau_i + [l, l+1)$, $\mathsf{TSeg}(\Gamma, (0, l), i) \in L(R_1^k)$,
- $\mathsf{TSeg}(\Gamma, [l, l+1), i) \in L(R_2^k).S.\Sigma^*$, where $\phi_2 \in S$, $S \in \mathsf{Cl}(\Gamma)$.

Note that ϕ_2 holds good at the point j such that $\tau_j \in [\tau_i + l, \tau_i + l + 1)$, and in $[l, \tau_j)$, the expression R_2^k evaluates to true. We simply assert Σ^* on the remaining part $(\tau_j, l+1)$ of the interval. Thus $true\mathsf{UReg}_{[l,l+1),\mathsf{re}'}\phi_2 \equiv \bigvee_{i\in\{1,2\ldots,n\}} \mathsf{Reg}_{(0,l)}R_1^i \wedge \mathsf{Reg}_{[l,l+1)}R_2^i.\phi_2.\Sigma^*$.

2. We first show that the UM modality can be captured by MC. Consider any formula $\phi_1\mathsf{UM}_{I,\#\phi_3=k\%n}\phi_2$. At any point i this formulae is true if and only if there exists a point j in future such that $\tau_j - \tau_i \in I$ and the number of points between i and j where ϕ_3 is true is $k\%n$, and ϕ_1 is true at all points between i and j. To count between i and j, we can first count the behaviour ϕ_3 from i to the last point of the word, followed by the counting from j to the last point of the word. Then we check that the difference between these counts to be $k\%n$.
   Let $cnt_\phi(x, \phi_3) = \{\phi \wedge \mathsf{MC}^{x\%n}_{(0,\infty)}(\phi_3)\}$. Using this macro, $\phi_1\mathsf{UM}_{I,\#\phi_3=k\%n}\phi_2$ is equivalent to $\bigvee_{k_1=0}^{n-1}[\psi_1 \vee \psi_2]$ where
   - ψ_1=$\{cnt_{true}(k_1, \phi_3) \wedge (\phi_1\ \mathsf{U}_I cnt_{\phi_2\wedge\neg\phi_3}(k_2, \phi_3))\}$,
   - ψ_2=$\{cnt_{true}(k_1, \phi_3) \wedge (\phi_1\ \mathsf{U}_I cnt_{\phi_2\wedge\phi_3}(k_2-1, \phi_3))\}$,
   - k_1−k_2=k

   The only difference between ψ_1, ψ_2 is that in one, ϕ_3 holds at position j, while in the other, it does not. The $k_2 - 1$ is to avoid the double counting in the case ϕ_3 holds at j.

◀

D po-1-clock ATA to 1−TPTL

In this section, we explain the algorithm which converts a po-1-clock ATA $\mathcal{A}$ into a 1−TPTL formula φ such that $L(\mathcal{A}) = L(\varphi)$.

1. **Step 1**. Rewrite the transitions of the automaton. Each $\delta(s, a)$ can be written in an equivalent form $C_1 \vee C_2$ or C_1 or C_2 where
   - C_1 has the form $s \wedge \varphi_1$, where $\varphi_1 \in \Phi(\downarrow s \cup \{a\} \cup X)$,
   - C_2 has the form φ_2, where $\varphi_2 \in \Phi(\downarrow s \cup \{a\} \cup X)$

   In particular, if s is the lowest location in the partial order, then $\varphi_1, \varphi_2 \in \Phi(\{a\} \cup X)$. Denote this equivalent form by $\delta'(s, a)$.
   For the example above, we obtain $\delta'(s_0, a) = (s_0 \wedge (a \wedge x.s_a)) \vee (a \wedge s_\ell), \delta'(s_0, b) = s_0 \wedge b$, $\delta'(s_a, a) = (s_a \wedge x < 1) \vee (x > 1)$ $\delta'(s_\ell) = (s_\ell \wedge b)$
2. **Step 2**. For each location s, construct $\Delta(s)$ which combines $\delta'(s, a)$ for all $a \in \Sigma$, by disjuncting them first, and again putting them in the form in step 1. Thus, we obtain $\Delta(s) = D_1 \vee D_2$ or D_1 or D_2 where D_1, D_2 have the forms $s \wedge \varphi_1$ and φ_2 respectively, where $\varphi_1, \varphi_2 \in \Phi(\downarrow s \cup \Sigma \cup X)$.
   For the example above, we obtain $\Delta(s_0) = s_0 \wedge [(a \wedge x.s_a)) \vee b] \vee (a \wedge s_\ell)$ $\Delta(s_a) = (s_a \wedge x < 1) \vee (x > 1)$ $\Delta(s_\ell) = s_\ell \wedge b$.
3. **Step 3**. We now convert each $\Delta(s)$ into a normal form $\mathcal{N}(s)$. $\mathcal{N}(s)$ is obtained from $\Delta(s)$ as follows.

- If s occurs in $\Delta(s)$, replace it with $\mathsf{O}s$.
- Replace each s' occurring in each $\Phi_i(\downarrow s)$ with $\mathsf{O}s'$.

Let $\mathcal{N}(s) = \mathcal{N}_1 \vee \mathcal{N}_2$, where $\mathcal{N}_1, \mathcal{N}_2$ are normal forms. Intuitively, the states appearing on the right side of each transition are those which are taken up in the next step. The normal form explicitely does this, and takes us a step closer to $1-$ – TPTL.

Continuing with the example, we obtain $\mathcal{N}(s_0) = \mathsf{O}s_0 \wedge [(a \wedge x.\mathsf{O}s_a)) \vee b] \vee (a \wedge \mathsf{O}s_\ell)$ $\mathcal{N}(s_a) = (\mathsf{O}s_a \wedge x < 1) \vee (x > 1)$ $\mathcal{N}(s_\ell) = \mathsf{O}s_\ell \wedge b$.

4. **Step 4**.
   - Start with the state s_n which is the lowest in the partial order. Let $\mathcal{N}(s_n) = (\mathsf{O}s_n \wedge \varphi_1) \vee \varphi_2$, where $\varphi_1, \varphi_2 \in \Phi(\Sigma, X)$.
     Solving $\mathcal{N}(s_n)$, one obtains the solution $F(s_n)$ as $\varphi_1 \,\mathsf{W} \varphi_2$ if s_n is an accepting location, and as $\varphi_1 \,\mathsf{U}^{\mathsf{ns}} \varphi_2$ if s_n is non-accepting.
     In the running example, we obtain $F(s_\ell) = b \,\mathsf{W} \bot = \Box^{\mathsf{ns}} b$ $F(s_a) = (x < 1) \,\mathsf{U}^{\mathsf{ns}} x > 1$
   - Consider now some $\mathcal{N}(s_i) = (\mathsf{O}s_i \wedge \varphi_1) \vee \varphi_2$. First replace each s' in φ_i with $F(s')$, and call the resultant expression as $F(\varphi_i)$, and $F(s_i)$ is then obtained as $F(\varphi_1) \,\mathsf{W} F(\varphi_2)$ if s_i is an accepting location, and as $F(\varphi_1) \,\mathsf{U}^{\mathsf{ns}} F(\varphi_2)$ if s_i is non-accepting.
     Substituting $F(s_a)$ and $F(s_\ell)$, we obtain $\mathcal{N}(s_0) = \mathsf{O}s_0 \wedge [(a \wedge x.\mathsf{O}F(s_a)) \vee b] \vee (a \wedge \mathsf{O}F(s_\ell))$, and hence $F(s_0) = [(a \wedge x.\mathsf{O}F(s_a)) \vee b] \mathsf{W} (a \wedge \mathsf{O}F(s_\ell))$ which is $((a \wedge (x.\mathsf{O}[(x < 1) \,\mathsf{U}^{\mathsf{ns}} x > 1])) \vee b) \,\mathsf{W} (a \wedge \mathsf{O}\Box^{\mathsf{ns}} b)$
   - The $1-$TPTL formula equivalent to $L(\mathcal{A})$ is then given by $F(s_0)$.

D.1 Correctness of Construction

The above algorithm is correct; that is, the $1-$TPTL formula $F(s_0)$ indeed captures the language accepted by the po-1-clock ATA.

For the proof of correctness, we define a 1-clock ATA with a TPTL look ahead. That is, $\delta : S \times \Sigma \to \Phi(S \cup X \cup \chi(\Sigma \cup \{x\}))$, where $\chi(\Sigma \cup \{x\})$ is a TPTL formula over alphabet Σ and clock variable x. We allow open TPTL formulae for look ahead; that is, one which is not of the form $x.\varphi$. All the freeze quantifications $x.$ lie within φ. The extension now allows to take a transition $(s, \nu) \to [\kappa \wedge \psi(x)]$, where $\psi(x)$ is a TPTL formula, if and only if the suffix of the input word with value of x being ν satisfies $\psi(x)$. We induct on the level of the partial order on the states.

Base Case: Let the level of the partial order be zero. Consider 1-clock ATA having only one location s_0. Let the transition function be $\delta(s_0, a) = \mathcal{B}_a(\psi_a(x), X, s_0)$ for every $a \in \Sigma$. By our construction, we reduce s_0 into $\Delta(s_0) = \bigvee\limits_{a \in \Sigma} [\mathcal{B}_a(\psi_a(x), X, \mathsf{O}(s_0))]$. Let $\Delta(s_0) = \bigvee(P_i \wedge \psi_i(x) \wedge X_i \wedge \mathsf{O}s_0) \vee \bigvee(Q_j \wedge \psi_j(x) \wedge X_j)$. $\delta(s_0, a) = s_0 \wedge X_1 \wedge \psi_1(x)$ specifies that the clock constraints X_1 are satisfied and the suffix satisfies the formulae $\psi_1(x)$ on reading an a. Thus for this $\delta(s_0, a)$, we have $\mathsf{O}s_0 \wedge X_1 \wedge \psi_1(x) \wedge a$ as a corresponding disjunct in Δ which specifies the same constraints on the word. Thus the solution to the above will be satisfied at a point with some $x = \nu$ if and only if there is an accepting run from s_0 to the final configuration with $x = \nu$.

If the s_0 is a final location, the solution to this is, $\varphi = \bigvee(P_i \wedge \psi_i(x) \wedge X_i \wedge \mathsf{O}s_0) \,\mathsf{W} \bigvee(Q_j \wedge \psi_j(x) \wedge X_j)$. If it is non-final, then it would be U instead of W. Note that this implies that whenever s_0 is invoked with value of x being ν, the above formulae would be true with $x = \nu$ thus getting an equivalent $1 -$ TPTL formulae.

Assume that for automata with $n-1$ levels in the partial order, we can construct $1-$TPTL formulae equivalent to the input automaton as per the construction. Consider any input automaton with n levels. Consider all the locations at the lowest level (that is, the location

can call only itself), $s_0, \ldots, s_k$. Apply the same construction. As explained above, the constructed formulae while eliminating a location will be true at a point if and only if there is an accepting run starting from the corresponding location with the same clock value. Let the formulae obtained for any s_i be φ_i.

The occurrence of an s_i in any $\Delta(s_{i<n})$ can be substituted with φ_i as a look ahead. This gives us an $n-1$ level 1-clock ATA with TPTL look ahead. By by induction, we obtain that every 1-clock po-ATA can be reduced to $1-$ TPTL formulae.

E Proof of Lemma 8

Proof. Let ρ be a timed word such that $\rho, i \models \mathsf{Reg}_I \mathsf{re}$. Note that re could be a compound regular expression containing formulae of the form $\mathsf{Reg}_{I'}\mathsf{re}'$. As a first step, we introduce an atomic proposition w_I which evaluates to true at all points j in ρ such that $\tau_j - \tau_i \in I$. Then it is easy to see that $\rho, i \models \mathsf{Reg}_I \mathsf{re}$ iff $\rho, i \models \mathsf{Reg}_I(\mathsf{re} \wedge w_I)$, since Reg_I covers exactly all points which are within the interval I from i. As the next step, we replace re, with an atomic proposition w obtaining the formula $\mathsf{Reg}_I[w \wedge w_I]$. Assume that I is a bounded interval. $\mathsf{Reg}_I[w \wedge w_I]$ is equivalent to $\mathsf{Reg}[(w \wedge w_I).(\neg w_I)^*]$, since $\mathsf{Reg}[]$ covers the entire suffix of ρ starting at point i. Now, replace w_I with the clock constraint $x \in I$, and rewrite the formula as $x.[(w \wedge (x \in I)).\Box(\neg(x \in I))]$, which is in $1-$ TPTL. Note that this step also preserves equivalence of teh formulae. Replacing w with re now eliminates one level of the Reg operator in the above formula. Doing the same technique as above to re which has the form $\mathsf{Reg}_{I'}(\mathsf{re}')$, will eliminate one more level of Reg and so on. Continuing this process will result in a $1-$TPTL formula which has k freeze quantifications iff the starting SfrMTL formula had k nestings of the Reg modality.

In case I is an unbounded interval, then we need not concatenate $(\neg w_I)^*$ at the end of $(w \wedge w_I)$. The rest of the proof is the same. ◀ ◀

F Example

Example. Consider the example of the po-1-clock ATA in the main paper. We now work out a few steps to illustrate the construction. The lowest locations are s_ℓ, s_a and we know $F(s_\ell) = \Box^{\mathsf{ns}} b$, $F(s_a) = (x < 1)\, \mathsf{U}^{\mathsf{ns}} (x > 1)$ and $F(s_0) = [(a \wedge x.\mathsf{O}F(s_a)) \vee b]\, \mathsf{W} (a \wedge \mathsf{O}F(s_\ell))$. The regions are R_0, R_1, R_2, R_3.

From this, we obtain $\mathsf{Beh}(s_a, R) = \{[\top, \top, \Box^{\mathsf{ns}} \bot, \top]\}$ for all $R \in \mathcal{R} \setminus \{R_2\}$. Note that as there is no constraint of the form $x = 1$ in the formulae, $\mathsf{Beh}(s_a, R_2) = \emptyset$. As there are no clock constraints in s_ℓ, there is no need to compute the Beh for it. We do it here just for the purpose of illustration. $\mathsf{Beh}(s_\ell, R_0)$ consists of BD's

- $\mathsf{BD}_1 = [\Box^{\mathsf{ns}} b, \Box^{\mathsf{ns}} b, \Box^{\mathsf{ns}} b, \Box^{\mathsf{ns}} b]$
- $\mathsf{BD}_2 = [\top, \Box^{\mathsf{ns}} b, \Box^{\mathsf{ns}} b, \Box^{\mathsf{ns}} b]$
- $\mathsf{BD}_3 = [\top, \top, \Box^{\mathsf{ns}} b, \Box^{\mathsf{ns}} b]$ and $\mathsf{BD}_4 = [\top, \top, \top, \Box^{\mathsf{ns}} b]$.

$\mathsf{Beh}(s_\ell, R_1)$ consists of $\mathsf{BD}_1, \mathsf{BD}_2, \mathsf{BD}_3$, $\mathsf{Beh}(s_\ell, R_2)$ consists of $\mathsf{BD}_2, \mathsf{BD}_3$, while $\mathsf{Beh}(s_\ell, R_3)$ consists of BD_3.

It can be seen that $\mathsf{Expr}(s_a)$ is given by the disjunction of

- $[\mathsf{Reg}_{R_0}(\top) \wedge \mathsf{Reg}_{R_1}(\top) \wedge \mathsf{Reg}_{R_2}(\emptyset) \wedge \mathsf{Reg}_{R_3}(\top)]$
- $[\mathsf{Reg}_{R_0}(\emptyset) \wedge \mathsf{Reg}_{R_1}(\Sigma^+)] \rightarrow [\mathsf{Reg}_{R_1}(\top) \wedge \mathsf{Reg}_{R_2}(\emptyset) \wedge \mathsf{Reg}_{R_3}(\top)]$
- $[\mathsf{Reg}_{R_0}(\emptyset) \wedge \mathsf{Reg}_{R_1}(\emptyset) \wedge \mathsf{Reg}_{R_3}(\Sigma^+)] \rightarrow \mathsf{Reg}_{R_3}(\top)$

It can be seen that $\mathsf{Expr}(s_\ell)$ will be equivalent to $\mathsf{O}\Box^{\mathsf{ns}} b$. Let $F'(s_0)$ be the formulae we get by substituting s_a with $\mathsf{Expr}(s_a)$ and s_ℓ by $\Box^{\mathsf{ns}} b$. Again note that there are no clock constraints

in $F(s_0)$. thus we do not need to make Beh for it. The final formulae which is supposed to be asserted at 0 is $x.\mathsf{O}(F'(s_0)$ which is equivalent to $\mathsf{O}(F'(s_0)$(as there are no timing constraints in $F(s_0)$). Observe what is the behaviour of $\mathsf{O}(F'(s_0) = \mathsf{O}[(a \wedge \mathsf{Expr}(s_a)) \vee b]\, \mathsf{W}(a \wedge \mathsf{O}(\Box^{\mathsf{ns}} b))$. Note that if any point satisfies $\mathsf{Expr}(s_a)$ if and only if there is no action point exactly after 1 unit time. Thus, $\mathsf{O}(F'(s_0)) = \mathsf{O}[[(a \wedge \mathsf{Expr}(s_a)) \vee b]\, \mathsf{W}(a \wedge \mathsf{O}(\Box^{\mathsf{ns}} b))]]$ implies that from the beginning of the timed word till the last occurrence of a, there is no a which has an action point exactly after unit time from it. This is what exactly the input ATA was specifying.

F.1 Two Counter Machines

A deterministic k-counter machine is a $k+1$ tuple $\mathcal{M} = (P, C_1, \ldots, C_k)$, where

1. $C_1, \ldots, C_k$ are counters taking values in $\mathcal{N} \cup \{0\}$ (their initial values are set to zero);
2. P is a finite set of instructions with labels $p_1, \ldots, p_{n-1}, p_n$. There is a unique instruction labelled HALT. For $E \in \{C_1, \ldots, C_k\}$, the instructions P are of the following forms:
   a. p_g: $Inc(E)$, goto p_h,
   b. p_g: If $E = 0$, goto p_h, else go to p_d,
   c. p_g: $Dec(E)$, goto p_h,
   d. p_n: HALT.

A configuration $W = (i, c_1, \ldots, c_k)$ of $\mathcal{M}$ is given by the value of the current program counter i and values $c_1, c_2, \ldots, c_k$ of the counters $C_1, C_2, \ldots, C_k$. A move of the counter machine $(l, c_1, c_2, \ldots, c_k) \to (l', c_1', c_2', \ldots, c_k')$ denotes that configuration $(l', c_1', c_2', \ldots, c_k')$ is obtained from $(l, c_1, c_2, \ldots, c_k)$ by executing the l^{th} instruction p_l. If p_l is an increment or decrement instruction, $c_l' = c_l + 1$ or $c_l - 1$, while $c_i' = c_i$ for $i \neq l$ and p_l' is the respective next instruction, while if p_l is a zero check instruction, then $c_i' = c_i$ for all i, and $p_l' = p_j$ if $c_l = 0$ and p_k otherwise.

Incremental Error Counter Machine

An incremental error counter machine is a counter machine where a particular configuration can have counter values with arbitrary positive error. Formally, an incremental error k-counter machine is a $k+1$ tuple $\mathcal{M} = (P, C_1, \ldots, C_k)$ where P is a set of instructions like above and C_1 to C_k are the counters. The difference between a counter machine with and without incremental counter error is as follows:

1. Let $(l, c_1, c_2 \ldots, c_k) \to (l', c_1', c_2' \ldots, c_k')$ be a move of a counter machine without error when executing l^{th} instruction.
2. The corresponding move in the increment error counter machine is

$$(l, c_1, c_2 \ldots, c_k) \to \{(l', c_1'', c_2'' \ldots, c_k'') | c_i'' \geq c_i', 1 \leq i \leq k\}$$

   Thus the value of the counters are non deterministic.

▶ **Theorem 15.** *[9] The halting problem for deterministic k counter machines is undecidable for $k \geq 2$.*

▶ **Theorem 16.** *[3] The halting problem for incremental error k-counter machines is non primitive recursive.*

F.2 Non-punctual 1-TPTL is NPR

In this section, we show that non-punctuality does not provide any benefits in terms of complexity of satisfiability for TPTL as in the case of MITL. We show that satisfiability checking of non-punctual TPTL is itself non-primitive recursive. This highlights the importance of our oversampling reductions from RegMTL and RegMITL to MTL and MITL respectively, giving RegMITL an elementary complexity. It is easier to reduce RegMITL to 1-variable, non-punctual, TPTL without using oversampling, but this gives a non-primitive recursive bound on complexity.

Non-punctual TPTL with 1 Variable ($1-$OpTPTL)

We study a subclass of $1-$TPTL called open $1-$TPTL and denoted as $1-$OpTPTL. The restrictions are mainly on the form of the intervals used in comparing the clock x as follows:

- Whenever the single clock x lies in the scope of even number of negations, x is compared only with open intervals, and
- Whenever the single clock x lies in the scope of an odd number of negations, x is compared to a closed interval.

Note that this is a stricter restriction than non-punctuality as it can assert a property only within an open timed regions.

F.2.1 Satisfiability Checking for $1-$OpTPTL

In this section we will investigate the benefits of relaxing punctuality in TPTL by exploring the hardness of satisfiability checking for $1-$OpTPTL over timed words.

▶ **Theorem 17.** *Satisfiability Checking of* $1-\mathsf{OpTPTL}[\Diamond,\mathsf{O}]$ *is decidable with non primitive recursive lower bound over finite timed words and it is undecidable over infinite timed words.*

Proof. We encode the runs of k counter incremental error channel machine using $1-$OpTPTL formulae with $\Diamond,\mathsf{O}$ modalities. We will encode a particular computation of any CM using timed words. The main idea is to construct an $1-\mathsf{OpTPTL}[\Diamond,\mathsf{O}]$ formula φ_{ICM} for any given k-incremental counter machine ICM such that it is satisfied by only those timed words that encode the halting computation of ICM. Moreover, for every halting computation $\mathcal{C}$ of ICM at least one timed word ρ_C satisfies φ_{ICM} such that ρ_C encodes $\mathcal{C}$.

We encode each computation of some k-incremental counter machine $ICM = (P, C)$ where $P = \{p_1, \ldots, p_n\}$ and $C = \{c_1, \ldots, c_k\}$ using timed words over the alphabet $\Sigma_{ICM} = \bigcup_{i\in\{1,\ldots,k\}}(S \cup F \cup \{a_j, b_j\})$ where $S = \{s^p | p \in 1, \ldots, n\}$ and $F = \{f^p | p \in 1, \ldots, n\}$ as follows:
A i^{th} configuration, $(p, c_1, \ldots, c_k)$ is encoded in the time region $[i, i+1)$ with sequence :

$$s^p((a_1b_1)^{c_1}(a_2b_2)^{c_2}\ldots(a_kb_k)^{c_k}f^p.$$

The concatenation of these time segments of a timed word encodes the whole computation. Thus the untimed projection of our language will be of the form:

$$\mathcal{S}(a_1b_1)^*(a_2b_2)^*\ldots(a_kb_k)^*\mathcal{F})^*$$

where $\mathcal{S} = \bigvee_{p\in\{1,2,\ldots,n\}} s^p$ and $\mathcal{F} = \bigvee_{p\in\{1,2,\ldots,n\}} f^p$.

To construct a formula φ_{ICM}, the main challenge is to write down some finite specifications which propagate the behaviour from the time segment $[i, i+1)$ to the time segment $[i+1, i+2)$

such that the later encodes the $i+1^{th}$ configuration of ICM (in accordance with the program counter value at i^{th} configuration). The usual idea is to copy all the a's from one configuration to another using punctuality. This is not possible in a non-punctual logic. Thus we try to preserve the number (or copy a time point) using following idea:

- Given any non last $(a_j,t)(b_j,t')$ before $\mathcal{F}$(for some counter c_j) , of a timed word encoding a computation. We assert that the last symbol in $(t,t+1)$ is a_j and the symbol in $(t',t'+1)$ is b_j.
- We can easily assert that the untimed sequence of the timed word is of the form

$$\mathcal{S}(a_1b_1)^*(a_2b_2)^*\dots(a_kb_k)^*\mathcal{F})^*$$

- The above two conditions imply that there is at least one a_j within time(t_1+1,t_2+1). Thus all the non last a_jb_j is copied to the segment encoding next configuration. Now appending one a_jb_j,two a_jb_j's or no a_jb_j's depends on whether the instruction was copy, increment or decrement operation.

φ_{ICM} is obtained as a conjunction of several formulae. Let $\mathcal{S},\mathcal{F}$ be a shorthand for $\bigwedge_{p\in\{1,\dots,n\}} s^p$ and $\bigwedge_{p\in\{1,\dots,n\}} f^p$, respectively. We also define macros $A_j=\bigvee_{w\geq j} a_w$ and $A_{k+1}=\bot$ We now give formula for encoding the machine. Let $C=\{1,\dots,k\}$ and $P=\{1,\dots,n\}$.

- **Expressing untimed sequence**: The words should be of the form

$$(\mathcal{S}(a_1b_1)^*(a_2b_2)^*\dots(a_kb_k)^*\mathcal{F})^*$$

. This could be expressed in the formula below

$\varphi_1=\bigwedge_{j\in C,p\in P}\Box^{\mathsf{ns}}[s^p\to \mathsf{O}(A_1\vee f^p)]\wedge\Box^{\mathsf{ns}}[a_j\to\mathsf{O}(b_j)]\wedge$
$\Box^{\mathsf{ns}}[b_j\to\mathsf{O}(A_{j+1}\vee f^p)]\wedge\Box^{\mathsf{ns}}[f^p\to\mathsf{O}(\mathcal{S}\vee\Box^{\mathsf{ns}}(false))]$

- **Initial Configuration**: There is no occurrence of a_jb_j within $[0,1]$. The program counter value is 1.

$\varphi_2=x.\{s_1^1\wedge\mathsf{O}(f_j^1\wedge T-x\in(0,1))$

- **Copying** $\mathcal{S},\mathcal{F}$: Every $(\mathcal{S},u)$, $(\mathcal{F},v)$ has a next occurrence $(\mathcal{S},u')$, $(\mathcal{F},v')$ in future such that $u'-u\in(k,k+1)$ and $v'-v\in(k-1,k)$. Note that this condition along with φ_1 and φ_2 makes sure that $\mathcal{S}$ and $\mathcal{F}$ occur only within the intervals of the form $[i,i+1)$ where i is the configuration number.

$\varphi_3=[\Box^{\mathsf{ns}}x.\{(\mathcal{S}\wedge\neg s^n)\to\neg\Diamond(T-x\in[0,1]\wedge\mathcal{S})\wedge\Diamond(\mathcal{S}\wedge T-x\in(1,2))\}\wedge\Box^{\mathsf{ns}}x.\{(\mathcal{F}\wedge\neg f^n)\to\Diamond(\mathcal{F}\wedge T-x\in(0,1))\}]$

- Beyond p_n=HALT, there are no instructions

$\varphi_4=\Box^{\mathsf{ns}}[f^n\to\Box(false)]$

- At any point of time, exactly one event takes place. Events have distinct time stamps.

$\varphi_6=[\bigwedge_{y\in\Sigma_{ICM}}\Box^{\mathsf{ns}}[y\to\neg(\bigwedge_{\Sigma_{ICM}\setminus\{y\}}(x))]\wedge\Box^{\mathsf{ns}}[\Box(false)\vee\mathsf{O}(T-x\in(0,\infty))]$

- Eventually we reach the halting configuration $\langle p_n,c_1,\dots,c_k\rangle$: $\varphi_6=\widetilde{\Diamond}s^n$

- Every non last $(a_j,t)(b_j,t')$ occurring in the interval $(i,i+1)$ should be copied in the interval $(i+1,i+2)$. We specify this condition by stating that from every non last a_j (before A_{j+1} or f^p) the last symbol within $(0,1)$ is a_j. Similarly from every non last b_j(before A_{j+1} or f^p) the last symbol within $(k-1,k)$ is b_j. Thus $(a_j,t)(b_j,t')$ will have

a $(b_j, t'+1-\epsilon)$ where $\epsilon \in (0, t'-t)$. Thus all the non last a_jb_j will incur a b_j in the next configuration . φ_2 makes sure that there is an a_j between two b_j's. Thus this condition along with φ_1 makes sure that the non last a_jb_j sequence is conserved. Note that there can be some a_jb_j which are arbitrarily inserted. These insertions errors model the incremental error of the machine. Thus if we consider a mapping where $(a_j, t_{ins})(b_j, t'_{ins})$ is mapped to $(a_j, t)(b_j, t')$ such that $t'_{ins} \in \{t+1, t'+1\}$, this is an injective function. Just for the sake of simplicity we assume that $a_{k+1} = false$.

$$\varphi_7 = \bigwedge_{j \in C} \Box^{\mathsf{ns}} x.[(nl(a_j) \wedge \psi_{nh}) \rightarrow \Diamond(a_j \wedge T - x \in (0,1) \wedge \mathsf{O}(T-x \in (1,2)))] \wedge$$
$$\Box^{\mathsf{ns}} x.[(nl(b_j) \wedge \psi_{nh}) \rightarrow \Diamond(b_j \wedge T - x \in (0,1) \wedge \mathsf{O}(T-x \in (1,2)))]$$

Let $nl(a_j) = a_j \wedge last(a_j)$, $nl(b_j) = b_j \wedge last(b_j)$, $\psi_{nh} = \neg\Diamond(f^n_k \wedge T - x \in [0,1])$, $last(a_j) = a_j \wedge \mathsf{O}(\mathsf{O}(\mathcal{F} \vee A_{j+1})))$ and $last(b_j) = b_j \wedge \mathsf{O}(\mathcal{F} \vee A_{j+1})$.

We define a short macro $Copy_{C \setminus W}$: Copies the content of all the intervals encoding counter values except counters in W. Just for the sake of simplicity we denote

$$Copy_{C\setminus W} = \bigwedge_{j \in C \setminus W} \Box^{\mathsf{ns}} x.\{last(a_j) \rightarrow (a_j \wedge T - x \in (0,1) \wedge \mathsf{O}(b_j \wedge T - x \in (1,2) \wedge \mathsf{O}(\mathcal{F})))\}$$

Using this macro we define the increment,decrement and jump operation.

1. p_g: If $C_j = 0$ goto p_h, else goto p_d. δ_1 specifies the next configuration when the check for zero succeeds. δ_2 specifies the else condition.

   $$\varphi_8^{g,j=0} = Copy_{C\setminus\{\emptyset\}} \wedge \delta_1 \wedge \delta_2$$

   $$\delta_1 = \Box^{\mathsf{ns}}[\{s^x_1 \wedge ((\neg a_j)\, \mathsf{U} \mathcal{F})\} \rightarrow (\neg \mathcal{S})\, \mathsf{U} s^y)]$$
   $$\delta_2 = \Box^{\mathsf{ns}}[\{s^x_1 \wedge ((\neg a_j)\, \mathsf{U} a_j)\} \rightarrow (\neg \mathcal{S})\, \mathsf{U} s^d)].$$

2. p_g: $Inc(C_j)$ goto p_h. The increment is modelled by appending exactly one a_jb_j in the next interval just after the last copied a_jb_j

   $$\varphi_8^{g,inc_j} = Copy_{C\setminus\emptyset} \wedge \Box^{\mathsf{ns}}(s^g \rightarrow (\neg\mathcal{S})\, \mathsf{U} s^h) \wedge \psi_0^{inc} \wedge \psi_1^{inc}$$

   The formula $\psi_0^{inc} = \Box^{\mathsf{ns}}[(s^g \wedge (\neg a_j\, \mathsf{U} f^g)) \rightarrow (\neg\mathcal{S}\, \mathsf{U} x.(s^h \wedge \Diamond(T - x \in (0,1) \wedge a_j))]$ specifies the increment of the counter j when the value of j is zero. The formula $\psi_1^{inc} = \Box^{\mathsf{ns}}[\{s^g \wedge ((\neg\mathcal{F})\mathsf{U}(a_j))\} \rightarrow (\neg\mathcal{F})\mathsf{U}x.\{last(a_j) \wedge \Diamond(T - x \in (0,1) \wedge (a_j \wedge \mathsf{OO}(last(a_j) \wedge T - x \in (1,2))))\}]$ specifies the increment of counter j when j value is non zero by appending exactly one pair of a_jb_j after the last copied a_jb_j in the next interval.
3. p_g: $Dec(C_j)$ goto p_h. Let $second - last(a_j) = a_j \wedge \mathsf{O}(\mathsf{O}(last(a_j)))$. Decrement is modelled by avoiding copy of last a_jb_j in the next interval.

   $$\varphi_8^{g,dec_j} = Copy_{C\setminus j} \wedge \Box^{\mathsf{ns}}(s^g \rightarrow (\neg\mathcal{S})\, \mathsf{U} s^h) \wedge \psi_0^{dec} \wedge \psi_1^{dec}$$

   The formula $\psi_0^{dec} = \Box^{\mathsf{ns}}[\{s^g \wedge (\neg a_j)\, \mathsf{U} f^g)\} \rightarrow \{(\neg\mathcal{S})\, \mathsf{U}\{s^h \wedge ((\neg a_j)\, \mathsf{U}(\mathcal{F})\}]$ specifies that the counter remains unchanged if decrement is applied to the j when it is zero. The formula $\psi_1^{dec} = \Box^{\mathsf{ns}}[\{s^g \wedge ((\neg\mathcal{F})\, \mathsf{U}(a_j))\} \rightarrow (\neg\mathcal{F})\, \mathsf{U}x.\{second - last(a_j) \wedge \Diamond(T - x \in (0,1) \wedge (a_j \wedge \mathsf{OO}(A_{j+1} \wedge T - x \in (1,2))))\}]$ decrements the counter j, if the present value of j is non zero. It does that by disallowing copy of last a_jb_j of the present interval to the next.

The formula $\varphi_{ICM} = \bigwedge_{i\in\{1,\ldots,7\}} \varphi_i \wedge \bigwedge_{p\in P} \varphi_8^p$. 2) To prove the undecidability we encode the k counter machine without error. Let the formula be φ_{CM}. The encoding is same as above. The only difference is while copying the non-last a in the $\varphi_{\mathcal{M}}$ we allowed insertion errors i.e. there were arbitrarily extra a and b allowed in between apart from the copied ones in the next configuration while copying the non-last a and b. To encode counter machine without error we need to take care of insertion errors. Rest of the formula are same. The following formula will avoid error and copy all the non-last a and b without any extra a and b inserted in between.

$$\varphi_9 = \bigwedge_{j\in C} \Box^{\mathsf{ns}} x.[(a_j \wedge \neg last(a_j)) \rightarrow \Diamond(x - T \in (1,2) \wedge \mathsf{O}(a_j \wedge x - T \in (0,1)))] \wedge$$
$$\Box^{\mathsf{ns}} x.[(b_k \wedge \neg last(b_j)) \rightarrow \Diamond(x - T \in (1,2) \wedge \mathsf{O}(b_j \wedge x - T \in (0,1)))]$$

Now, $\varphi_{CM} = \varphi_{ICM} \wedge \varphi_9$

F.2.1.1 Correctness Argument

- Note that increment errors occurred only while copying the non last ab sequence in (1). The similar argument for mapping a_j with a unique a_j in the next configuration can be applied in past and thus using φ_9 mapping we can say that non last a_j, b_j in the previous configuration can be mapped to a copied a_j, b_j in the next configuration with an injective mapping. This gives as an existence of bijection between the set of non-last a_k, b_k in the previous configuration and the set of copied a_k, b_k by φ_7. Thus "there are no insertion errors" is specified with φ_9.

◀