Making Metric Temporal Logic Rational*

Shankara Narayanan Krishna[1], Khushraj Madnani[1], and Paritosh K. Pandya[3]

1 **IIT Bombay, Mumbai, India**
krishnas@cse.iitb.ac.in
2 **IIT Bombay, Mumbai, India**
khushraj@cse.iitb.ac.in
3 **Tata Institute of Fundamental Research, Mumbai, India**
pandya@tifr.res.in

Abstract

We study an extension of MTL in pointwise time with regular expression guarded modality $\mathsf{Rat}_I(\mathsf{re})$ where re is a rational expression over subformulae. We study the decidability and expressiveness of this extension (MTL+ URat+Rat), called RatMTL, as well as its fragment SfrMTL where only star-free rational expressions are allowed. Using the technique of temporal projections, we show that RatMTL has decidable satisfiability by giving an equisatisfiable reduction to MTL. We also identify a subclass MITL + URat of RatMTL for which our equisatisfiable reduction gives rise to formulae of MITL, yielding elementary decidability. As our second main result, we show a tight automaton-logic connection between SfrMTL and partially ordered (or very weak) 1-clock alternating timed automata.

1998 ACM Subject Classification F.4.1. Mathematical Logic

Keywords and phrases Metric Temporal Logic, Timed Automata, Regular Expression, Equisatisfiability, Expressiveness

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.77

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. Logics MTL and TPTL are amongst the prominent real time logics [2]. Two notions of MTL semantics have been studied in the literature : continuous and pointwise [5]. The expressiveness and decidability results vary considerably with the semantics used : while the satisfiability checking of MTL is undecidable in the continuous semantics even for finite timed words [1], it is decidable in pointwise semantics with non-primitive recursive complexity over finite timed words [15]. The satisfiability checking over infinite timed words is undecidable for both the semantics. Due to the hardness of analysis, quest for a decidable subclass and extension was started.

Related Work. Due to limited expressive power of MTL, several additional modalities have been proposed : the threshold counting modality [16] $\mathsf{C}_I^{\geq n}\phi$ states that in time interval I relative to current point, ϕ occurs at least n times. Note that we represent the set of modalities C_I is represented by C. The Pnueli modality [16] $\mathsf{Pn}_I(\phi_1, \ldots, \phi_n)$ states that there

[*] Please refer url <http://arxiv.org/abs/1705.01501> for full version

is a subsequence of n time points inside interval I where at i^{th} point the formula ϕ_i holds. In a recent result, Hunter [10] showed that, in continuous time semantics, MTL enriched with C modality (denoted $\mathsf{MTL} + \mathsf{C}$) is as expressive as $\mathsf{FO}[<, +1]$, which is as expressive as TPTL. Unfortunately, satisfiability and model checking of all these logics are undecidable. This has led us to focus on the pointwise case with only the until modality, i.e. logic $\mathsf{MTL}[\,\mathsf{U}_I]$, which we abbreviate as MTL in rest of the paper.Also, $\mathsf{MTL} + op$ means MTL with modalities $\,\mathsf{U}_I$ as well as op.

In pointwise semantics, it can be shown that $\mathsf{MTL}{+}\mathsf{C}$ is strictly more expressive than MTL and remains decidable for finite words (see [12]). In this paper, we propose a generalization of threshold counting and Pnueli modalities by a $\mathsf{rational\ expression}$ modality $\mathsf{Rat}_I\mathsf{re}(\phi_1, \ldots, \phi_k)$, which specifies that the truth of the subformulae, $\phi_1, \ldots, \phi_k$, at the set of points within interval I is in accordance with the regular expression $\mathsf{re}(\phi_1, \ldots, \phi_k)$. The resulting logic is called RatMTL and is the subject of this paper. The inability to specify rational expression constraints has been an important lacuna of LTL and its practically useful extensions such as PSL sugar [7], [6] (based on Dynamic Logic [8]) which extend LTL with both counting and rational expressions were studied. This indicates that our logic RatMTL is a natural and useful logic for specifying properties. Adding timing constraints to regular expressions was first given by Asarin, Caspi and Maler in [3] and was called as Timed Regular Expressions. They also show that these expressions exactly characterize the expressive power of Timed Automata. But this equivalence relies indispensably on the addition of renaming operation within there syntax [9] and are not closed under negations. In fact the validity checking for this extension was undecidable. Thus we propose a boolean closed decidable logic which can express regular expressions along with timing constraints. To our knowledge, impact of rational expression constraints on metric temporal modalities have not been studied before. The expressive power of logic RatMTL raises several points of interest.

As our first main result, we show that satisfiability of RatMTL is decidable by giving an equisatisfiable reduction to MTL. The reduction makes use of the technique of *oversampled temporal projections* which was previously proposed [11], [12] and used for proving the decidability of $\mathsf{MTL} + \mathsf{C}$. The reduction given here has several novel features such as an MTL encoding of the run tree of an alternating automaton which restarts the DFA of a given rational expression at each time point (section 3.1). We identify two syntactic subsets of RatMTL, the first denoted as $\mathsf{MITL} + \mathsf{URat}$ with $\mathsf{2EXPSPACE}$ easy satisfiability, and its further subset $\mathsf{MITL}{+}\mathsf{UM}$ with $\mathsf{EXPSPACE}$-complete satisfiability. As our second main result, we show that the star-free fragment SfrMTL of RatMTL characterizes exactly the class of partially ordered 1-clock alternating timed automata, thereby giving a tight logic automaton connection. The most non-trivial part of this proof is the construction of SfrMTL formula equivalent to a given partially ordered 1-clock alternating timed automaton $\mathcal{A}$ (Lemma 11).

2 Timed Temporal Logics

This section describes 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 $\tau_1\tau_2 \ldots \tau_n$ respectively, with $\sigma_i \in \mathcal{P}(\Sigma) - \emptyset$, and $\tau_i \in \mathbb{R}_{\geq 0}$ for $1 \leq i \leq n$ and $\forall i \in dom(\rho)$, $\tau_i \leq \tau_{i+1}$, where $dom(\rho)$ is the set of positions $\{1, 2, \ldots, n\}$ in the timed word. For convenience, we assume $\tau_1 = 0$. The σ_i's can be thought of as labelling positions i in $dom(\rho)$. For example, given $\Sigma = \{a, b, c\}$, $\rho = (\{a, c\}, 0)(\{a\}, 0.7)(\{b\}, 1.1)$ is a timed word. ρ is strictly monotonic iff $\tau_i < \tau_{i+1}$ for all $i, i + 1 \in dom(\rho)$. Otherwise, it is weakly monotonic. The set of finite timed words over Σ is denoted $T\Sigma^*$. Given $\rho=(\sigma,\tau)$ with $\sigma=\sigma_1\dots\sigma_n$, σ^{single} denotes the set of words $\{w_1w_2\dots w_n \mid w_i\in\sigma_i\}$. For ρ as above, σ^{single} consists of $(\{a\},0)(\{a\},0.7)(\{b\},1.1)$ and $(\{c\},0)(\{a\},0.7)(\{b\},1.1)$. Let $I\nu$ be a set of open, half-open or closed time intervals. The end points of these intervals are in $\mathbb{N}\cup\{0,\infty\}$. For example, $[1,3),[2,\infty)$. For $\tau\in\mathbb{R}_{\geq 0}$ and interval $\langle a,b\rangle$, with $<\in\{(,[\}$ and $>\in\{],)\}$, $\tau+\langle a,b\rangle$ stands for the interval $\langle\tau+a,\tau+b\rangle$.

Metric Temporal Logic (MTL). 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: $\varphi ::= a(\in\Sigma)\ |true\ |\varphi\wedge\varphi\mid\neg\varphi\mid\varphi\,\mathsf{U}_I\varphi$, where $I\in I\nu$. For a timed word $\rho=(\sigma,\tau)\in T\Sigma^*$, a position $i\in dom(\rho)$, 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: (i) $\rho,i\models a\leftrightarrow a\in\sigma_i$, (ii) $\rho,i\models\neg\varphi\leftrightarrow\rho,i\nvDash\varphi$, (iii) $\rho,i\models\varphi_1\wedge\varphi_2\leftrightarrow\rho,i\models\varphi_1$ and $\rho,i\ \models\ \varphi_2$, (iv) $\rho,i\ \models\ \varphi_1\,\mathsf{U}_I\varphi_2\leftrightarrow\exists j>i$, $\rho,j\ \models\ \varphi_2,\tau_j-\tau_i\in I$, and $\rho,k\ \models\ \varphi_1\ \forall\ i<k<j$.

The language of a MTL formula φ is $L(\varphi)=\{\rho\mid\rho,1\models\varphi\}$. 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 $\mathsf{O}_I\phi\equiv\bot\,\mathsf{U}_I\phi$. Non-strict versions of operators are defined as $\Diamond^{\mathsf{ns}}_I a=a\vee\Diamond_I a,\Box^{\mathsf{ns}}_I a\equiv a\wedge\Box_I a$, $a\,\mathsf{U}^{\mathsf{ns}}_I b\equiv b\vee[a\wedge(a\,\mathsf{U}_I b)]$ 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.

Timed Propositional Temporal Logic (TPTL). TPTL is a prominent real time extension of LTL, where timing constraints are specified with the help of freeze clocks. The set of TPTL formulas are defined inductively as $\varphi ::= a(\in\Sigma)\ |true\ |\varphi\wedge\varphi\mid\neg\varphi\mid\varphi\,\mathsf{U}\varphi\mid y.\varphi\mid y\in I$. $\mathcal{C}$ is a set of clock variables progressing at the same rate, $y\in\mathcal{C}$, and I is an interval as above. For a timed word $\rho=(\sigma_1,\tau_1)\dots(\sigma_n,\tau_n)$, 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 as follows: (1) $\rho,i,\nu\models a\leftrightarrow a\in\sigma_i$, (2) $\rho,i,\nu\models\neg\varphi\leftrightarrow\rho,i,\nu\nvDash\varphi$, (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) $\rho,i,\nu\models x.\varphi\leftrightarrow\rho,i,\nu[x\leftarrow\tau_i]\models\varphi$, (5) $\rho,i,\nu\models x\in I\leftrightarrow\tau_i-\nu(x)\in I$, (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.

▶ **Theorem 1** ([15]). MTL *satisfiability is decidable over finite timed words and is non-primitive recursive.*

MTL with Rational Expressions (RatMTL)

We propose an extension of MTL with rational expressions, that forms the core of the paper. These modalities can assert the truth of a rational expression (over subformulae) within a particular time interval with respect to the present point. For example, $\mathsf{Rat}_{(0,1)}(\varphi_1.\varphi_2)^+$ when evaluated at a point i, asserts the existence of $2k$ points $\tau_i<\tau_{i+1}<\tau_{i+2}<\dots<\tau_{i+2k}<\tau_i{+}1$, $k>0$, such that φ_1 evaluates to true at τ_{i+2j+1}, and φ_2 evaluates to true at τ_{i+2j+2}, for all $0\leq j<k$.

RatMTL Syntax Formulae of RatMTL are built from Σ (atomic propositions) as follows:
$\varphi ::= a(\in\Sigma)\ |true\ |\varphi\wedge\varphi\mid\neg\varphi\mid\mathsf{Rat}_I\mathsf{re}(\mathsf{S})\mid\varphi\mathsf{URat}_{I,\mathsf{re}(\mathsf{S})}\varphi$, where $I\in I\nu$ and S is a finite set of formulae of interest generated by this grammar, and $\mathsf{re}(\mathsf{S})$ is defined as a

rational expression over S. $\mathsf{re(S)} ::= \varphi(\in \mathsf{S}) \mid \mathsf{re(S).re(S)} \mid \mathsf{re(S)} + \mathsf{re(S)} \mid [\mathsf{re(S)}]^*$. Thus, RatMTL is $\mathsf{MTL} + \mathsf{URat} + \mathsf{Rat}$. An *atomic* rational expression re is any well-formed formula $\varphi \in \mathsf{RatMTL}$.

RatMTL Semantics For a timed word $\rho = (\sigma, \tau) \in T\Sigma^*$, a position $i \in dom(\rho)$, and a RatMTL formula φ, a finite set S of formulae, we define the satisfaction of φ at a position i as follows. For positions $i < j \in dom(\rho)$, let $\mathsf{Seg}(\rho, \mathsf{S}, i, j)$ denote the untimed word over $\mathcal{P}(\mathsf{S})$ obtained by marking the positions $k \in \{i+1, \ldots, j-1\}$ of ρ with $\psi \in \mathsf{S}$ iff $\rho, k \models \psi$. For a position $i{\in}dom(\rho)$ and an interval I, let $\mathsf{TSeg}(\rho, S, I, i)$ denote the untimed word over $\mathcal{P}(\mathsf{S})$ obtained by marking all the positions k such that $\tau_k - \tau_i \in I$ of ρ with $\psi \in \mathsf{S}$ iff $\rho, k \models \psi$.

1. $\rho, i \models \varphi_1 \mathsf{URat}_{I,\mathsf{re(S)}} \varphi_2 \leftrightarrow \exists j{>}i,\ \rho, j {\models} \varphi_2, \tau_j - \tau_i {\in} I,\ \rho, k \models \varphi_1\ \forall i{<}k{<}j$ and, $[\mathsf{Seg}(\rho, \mathsf{S}, i, j)]^{\mathsf{single}} \cap L(\mathsf{re}(S)) \neq \emptyset$, where $L(\mathsf{re(S)})$ is the language of the rational expression re formed over the set S. The subclass of RatMTL using only the URat modality is denoted $\mathsf{RatMTL[URat]}$ or $\mathsf{MTL} + \mathsf{URat}$ and if only non-punctual intervals are used, then it is denoted $\mathsf{RatMITL[URat]}$ or $\mathsf{MITL} + \mathsf{URat}$.
2. $\rho, i \models \mathsf{Rat}_I\mathsf{re} \leftrightarrow [\mathsf{TSeg}(\rho, S, I, i)]^{\mathsf{single}} \cap L(\mathsf{re}(S)) \neq \emptyset$.

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

▶ **Example 2.** Consider the formula $\varphi = a\mathsf{URat}_{(0,1),ab^*}b$. Then $\mathsf{re}{=}ab^*$, and the subformulae of interest are a, b. For $\rho{=}(\{a\}, 0)(\{a, b\}, 0.3)(\{a, b\}, 0.99)$, $\rho, 1 \models \varphi$, since $a{\in}\sigma_2, b{\in}\sigma_3$, $\tau_3{-}\tau_1{\in}(0, 1)$ and $a \in [\mathsf{Seg}(\rho, \{a, b\}, 1, 3)]^{\mathsf{single}} \cap L(ab^*)$. On the other hand, for the word $\rho = (\{a\}, 0)(\{a\}, 0.3)(\{a\}, 0.5)(\{a\}, 0.9)(\{b\}, 0.99)$, we know that $\rho, 1 \nvDash \varphi$, since even though $b \in \sigma_5, a \in \sigma_i$ for $i < 5$, $[\mathsf{Seg}(\rho, \{a, b\}, 1, 5)]^{\mathsf{single}} = aaa$ and $aaa \notin L(ab^*)$.

▶ **Example 3.** Consider the formula $\varphi = \mathsf{Rat}_{(0,1)}[\mathsf{Rat}_{(0,1)}a]^*$.
For $\rho = (\{a, b\}, 0)(\{a, b\}, 0.7)(\{b\}, 0.98)(\{a, b\}, 1.4)$, we have $\rho, 1{\nvDash}\mathsf{Rat}_{(0,1)}[\mathsf{Rat}_{(0,1)}a]^*$, since point 2 is not marked $\mathsf{Rat}_{(0,1)}a$, even though point 3 is.

Generalizing Counting, Pnueli & Mod Counting Modalities. The following reductions show that RatMTL subsumes most of the extensions of MTL studied in the literature.

(1) Threshold Counting constraints [16], [13], [12] specify the number of times a property holds within some time region is at least (or at most) n. These can be expressed in RatMTL: (i) $\mathsf{C}_I^{\geq n}\varphi \equiv \mathsf{Rat}_I(\mathsf{re}_{th})$, (ii) $\phi_1\mathsf{UT}_{I,\varphi\geq n}\phi_2 \equiv \phi_1\mathsf{URat}_{I,\mathsf{re}_{th}}\phi_2$, where $\mathsf{re}_{th} = true^*\underbrace{\varphi.true^*.\ldots.\varphi.true^*}_{n \text{ times}}$.

(2) Pnueli Modalities[1] [16], which enhance the expressiveness of MITL in continuous semantics preserving the complexity, can be written in RatMTL: $\mathsf{Pn}_I(\phi_1, \phi_2, \ldots, \phi_k)$ can be written as $\mathsf{Rat}_I(true^*.\phi_1.true^*\phi_2.\ldots.true^*.\phi_k.true^*)$.

(3) Modulo Counting constraints [4], [14] specify the number of times a property holds modulo $n \in \mathbb{N}$, in some region. We extend these to the timed setting by proposing two modalities $\mathsf{MC}_I^{k\%n}$ and $\mathsf{UM}_{I,\varphi=k\%n}$. $\mathsf{MC}_I^{k\%n}\varphi$ checks if the number of times φ is true in interval I is $M(n) + k$, where $M(n)$ denotes a non-negative integer multiple of n, and $0 \leq k \leq n-1$, while $\varphi_1\mathsf{UM}_{I,\#\psi=k\%n}\varphi_2$ when asserted at a point i, checks

[1] The version of the modality only specified sequences for the next unit interval. We talk about a more general version of this operator which is appended by timing interval.

the existence of $j > i$ such that $\tau_j - \tau_i \in I$, φ_2 is true at j, φ_1 holds between i, j, and the number of times ψ is true between i, j is $M(n) + k$, $0 \leq k \leq n-1$. As an example, $\psi = \mathit{true}\mathsf{UM}_{(0,1),\#b=1\%2}(a \vee b)$, when asserted at a point i, checks the existence of a point $j > i$ such that a or $b \in \sigma_j$, $\tau_j - \tau_i \in (0,1)$, and the number of points between i, j where b is true is odd. Both these modalities can be rewritten equivalently in RatMTL as follows: $\mathsf{MC}_I^{k\%n}\varphi \equiv \mathsf{Rat}_I(\mathsf{re}_{mod})$ and $\phi_1\mathsf{UM}_{I,\varphi=k\%n}\phi_2 \equiv \phi_1\mathsf{URat}_{I,\mathsf{re}_{mod}}\phi_2$ where $\mathsf{re}_{mod} = ([\underbrace{(\neg\varphi)^*.\varphi.\ldots.(\neg\varphi)^*.\varphi}_{n \text{ times}}]^*.[\underbrace{(\neg\varphi)^*.\varphi.\ldots.(\neg\varphi)^*.\varphi}_{k \text{ times}}]$. The extension of MTL (MITL) with only UM is denoted $\mathsf{MTL}+\mathsf{UM}$ ($\mathsf{MITL}+\mathsf{UM}$) while $\mathsf{MTL}+\mathsf{MC}$ ($\mathsf{MITL}+\mathsf{MC}$) denotes the extension using MC.

3 Satisfiability of RatMTL and Complexity

The main results of this section are as follows.

▶ **Theorem 4.** *(1) Satisfiability of* RatMTL *is decidable over finite timed words. (2) Satisfiability of* $\mathsf{MITL}+\mathsf{UM}$ *is* $\mathsf{EXPSPACE}$*-complete. (3) Satisfiability of* $\mathsf{MITL}+\mathsf{URat}$ *is within* $\mathsf{2EXPSPACE}$*. (4) Satisfiability of* $\mathsf{MITL}+\mathsf{MC}$ *is* $\mathbf{F}_{\omega^\omega}$*-hard.*

Details of 4.2, 4.3, 4.4 are in appendices E.2,E.3 and E.4 of the full version, respectively.

▶ **Theorem 5.** $\mathsf{MTL}+\mathsf{URat} \subseteq \mathsf{MTL}+\mathsf{Rat}$, $\mathsf{MTL}+\mathsf{UM} \subseteq \mathsf{MTL}+\mathsf{MC}$.

Theorem 5 shows that the Rat modality can capture URat (and likewise, MC captures UM). Thus, $\mathsf{RatMTL} \equiv \mathsf{MTL}+\mathsf{Rat}$. Observe that any re can be decomposed into finitely many factors, i.e. $\mathsf{re} = \sum_{i=1}^{n} R_1^i.R_2^i$. Given $\mathit{true}\mathsf{URat}_{[l,u),\mathsf{re}}\phi_2$, we assert R_1^i within interval $(0, l]$ and R_2^i in the prefix of the latter part within $[l, u)$, followed by ϕ_2. $\mathit{true}\mathsf{URat}_{[l,u),\mathsf{re}}\phi_2 \equiv \bigvee_{i\in\{1,2\ldots,n\}} \mathsf{Rat}_{(0,l)}R_1^i \wedge \mathsf{Rat}_{[l,u)}R_2^i.\phi_2.\Sigma^*$. The proofs are in appendix G of the full version.

3.1 Proof of Theorem 4.1

Equisatisfiability. We will use the technique of equisatisfiability modulo oversampling [11] in the proof of Theorem 4. Using this technique, formulae φ in one logic (say RatMTL) can be transformed into formulae ψ over a simpler logic (say MTL) such that whenever $\rho \models \varphi$ for a timed word ρ over alphabet Σ, one can construct a timed word ρ' over an extended set of positions and an extended alphabet Σ' such that $\rho' \models \psi$ and vice-versa [11], [12]. In *oversampling*, (i) $dom(\rho')$ is extended by adding some extra positions between the first and last point of ρ, (ii) the labeling of a position $i \in dom(\rho)$ is over the extended alphabet $\Sigma' \supset \Sigma$ and can be a superset of the previous labeling over Σ, while the new positions are labeled using only the new symbols $\Sigma' - \Sigma$. We can recover ρ from ρ' by erasing the new points and the new symbols. A restricted use of oversampling, when one only extends the alphabet and not the set of positions of a timed word ρ is called *simple extension*. In this case, if ρ' is a simple extension of ρ, then $dom(\rho) = dom(\rho')$, and by erasing the new symbols from ρ', we obtain ρ. See Figure 1 for an illustration. The formula ψ over the larger alphabet $\Sigma' \supset \Sigma$ such that $\rho' \models \psi$ iff $\rho \models \varphi$ is said to be equisatisfiable modulo temporal projections to φ. In particular, ψ is equisatisfiable to φ modulo simple extensions or modulo oversampling, depending on how the word ρ' is constructed from the word ρ.

The oversampling technique is used in the proofs of parts 4.1, 4.3 and 4.4.

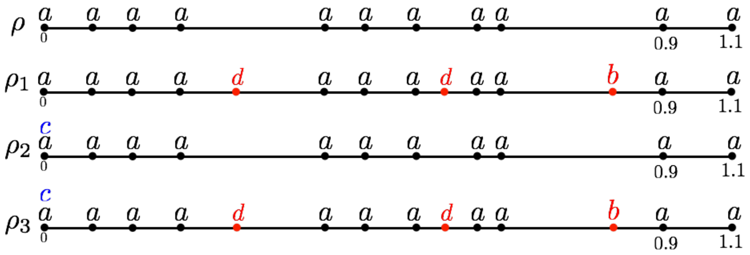

Figure 1 ρ is over $\Sigma = \{a\}$ and satisfies $\varphi = \Box_{(0,1)}a$. ρ_1 is an oversampling of ρ over an extended alphabet $\Sigma_1 = \Sigma \cup \{b, d\}$ and satisfies $\psi_1 = \Box(b \leftrightarrow \neg a) \wedge (\neg b \,\mathsf{U}_{(0,1)} b)$. The red points in ρ_1 are the oversampling points. ρ_2 is a simple extension of ρ over an extended alphabet $\Sigma_2 = \Sigma \cup \{c\}$ and satisfies $\psi_2 = \Box(c \leftrightarrow \Box_{(0,1)}a) \wedge c$. It can be seen that ψ_1 is equivalent to φ modulo oversampling, and ψ_2 is equivalent to φ modulo simple extensions using the (respectively oversampling, simple) extensions ρ_1, ρ_2 of ρ. However, ρ_3 above, obtained by merging ρ_1, ρ_2, eventhough an oversampling of ρ, is not a good model for the formula $\psi_1 \wedge \psi_2$ over $\Sigma_1 \cup \Sigma_2$. However, we can relativize ψ_1 and ψ_2 with respect to Σ as $\Box(act_1 \rightarrow (b \leftrightarrow \neg a)) \wedge [(act_1 \rightarrow \neg b) \,\mathsf{U}_{(0,1)} (b \wedge act_1)]$, and $\Box(act_2 \rightarrow (c \leftrightarrow \Box_{[0,1)}(act_2 \rightarrow a))) \wedge (act_2 \wedge c)$ where $act_1 = \bigvee \Sigma_1, act_2 = \bigvee \Sigma_2$. The relativized formula $\kappa = Rel(\psi_1, \Sigma) \wedge Rel(\psi_2, \Sigma)$ is then equisatisfiable to φ modulo oversampling, and ρ_3 is indeed an oversampling of ρ satisfying κ. This shows that while combining formulae ψ_1, ψ_2 which are equivalent to formulae φ_1, φ_2 modulo oversampling, we need to relativize ψ_1, ψ_2 to obtain a conjunction which will be equisatisfiable to $\varphi_1 \wedge \varphi_2$ modulo oversampling. See [11] for details.

Equisatisfiable Reduction : RatMTL to MTL

Let φ be a RatMTL formula. To obtain equisatisfiable MTL formula ψ, we do the following.

1. We "flatten" the rational(Rat & URat) modalities to simplify the formulae, eliminating nested rational modalities by allotting witness variable for each rational subformulae . Thus the resulting formulae will be of the form $\mathsf{prop} \wedge \Box^{\mathsf{ns}}[w_1 \leftrightarrow \mathsf{Rat}_I, \mathsf{URat}] \cdots \wedge \Box^{\mathsf{ns}}[w_k \leftrightarrow \mathsf{Rat}_I, \mathsf{URat}]$ where prop refers to some boolean formulae over atoms and $\mathsf{Rat}_I, \mathsf{URat}$ denotes formulae of the form $\mathsf{Rat}_I \mathsf{re{-}atom}$, $\mathsf{prop}\mathsf{URat}_{I,\mathsf{re-atom}}\mathsf{prop}$, respectively. Each conjunct of the form $\Box^{\mathsf{ns}}[w_1 \leftrightarrow \mathsf{Rat}_I, \mathsf{URat}]$ is called as *temporal definition.*
2. The elimination of rational modalities is achieved by obtaining equisatisfiable MTL formulae ψ_i over X_i, possibly a larger set of propositions than $\Sigma \cup W_i$ corresponding to each temporal definition T_i of φ_{flat}. Relativizing these MTL formulae and conjuncting them, we obtain an MTL formula $\bigwedge_i Rel(\psi_i, \Sigma)$ that is equisatisfiable to φ (see Figure 1 for relativization).

The above steps are routine [11], [12]. What remains is to handle the temporal definitions.

Embedding the Runs of the DFA

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 $\Box^{\mathsf{ns}}[w \leftrightarrow \mathsf{Rat}_I \mathsf{re{-}atom}]$ or $\Box^{\mathsf{ns}}[w \leftrightarrow x\mathsf{URat}_{I',\mathsf{re-atom}}y]$, the rational expression re−atom has a corresponding minimal DFA recognizing it. We define an LTL formula $\mathsf{GOODRUN}(\phi_e)$ which takes a formula ϕ_e as a parameter with the following behaviour. $\rho, i \models \mathsf{GOODRUN}(\phi_e)$ iff for all $k > i$, $(\rho, k \models \phi_e) \rightarrow (\rho[i, k] \in L(\mathsf{re{-}atom}))$. To achieve this, we use two new sets of symbols Threads and Merge for this information. This results in the extended alphabet $\Sigma \cup W \cup \mathsf{Threads} \cup \mathsf{Merge}$ for the simple extension ρ' of ρ. The behaviour of Threads and Merge are explained below.

Consider $\mathsf{re{-}atom} = \mathsf{re}(\mathsf{S})$. Let $\mathcal{A}_{\mathsf{re-atom}} = (Q, 2^{\mathsf{S}}, \delta, q_1, Q_F)$ be the minimal DFA for re−atom and let $Q = \{q_1, q_2, \ldots, q_m\}$. Let $\mathsf{In} = \{1, 2, \ldots, m\}$ be the indices of the states.

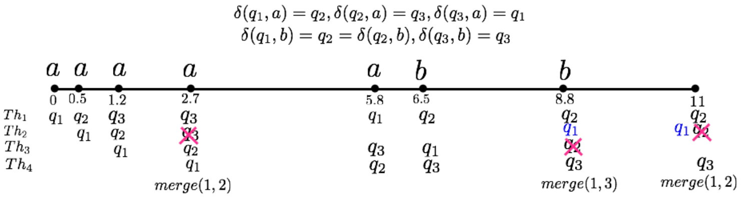
$\delta(q_1,a)=q_2,\delta(q_2,a)=q_3,\delta(q_3,a)=q_1$
$\delta(q_1,b)=q_2=\delta(q_2,b),\delta(q_3,b)=q_3$
a a a a a b b
0 0.5 1.2 2.7 5.8 6.5 8.8 11
Th_1
Th_2
Th_3
Th_4
q_1 q_2 q_3 q_3 q_1 q_2 q_2 q_2
q_1 q_2 $\cancel{q_3}$ q_1 q_1 $\cancel{q_2}$
q_1 q_2 q_3 q_1 $\cancel{q_2}$
q_1 q_2 q_3 q_3 q_3
$merge(1,2)$
$merge(1,3)$
$merge(1,2)$

Figure 2 Depiction of threads and merging. At time point 2.7, thread 2 is merged with 1, since they both had the same state information. This thread remains inactive till time point 8.8, where it becomes active, by starting a new run in state q_1. At time point 8.8, thread 3 merges with thread 1, while at time point 11, thread 2 merges with 1, but is reactivated in state q_1.

Conceptually, we consider multiple runs of $\mathcal{A}_{\mathsf{re-atom}}$ with a new run (new thread) started at each point in ρ. $\mathsf{Threads}$ records the state of each previously started run. At each step, each thread is updated from it previous value according to the transition function δ of $\mathcal{A}_{\mathsf{re-atom}}$ and also augmented with a new run in initial state. Potentially, the number of threads would grow unboundedly in size but notice that once two runs are the same state at position i they remain identical in future. Hence they can be merged into single thread (see Figure2). As a result, m threads suffice. We record whether threads are merged in the current state using variables Merge. An LTL formula records the evolution of $\mathsf{Threads}$ and Merge over any behaviour ρ. We can define formula $\mathsf{GOODRUN}(\phi_e)$ in LTL over $\mathsf{Threads}$ and Merge.

1. At each position, let $\mathsf{Th}_i(q_x)$ be a proposition that denotes that the ith thread is active and is in state q_x, while $\mathsf{Th}_i(\bot)$ be a proposition that denotes that the ith thread is not active. The set $\mathsf{Threads}$ consists of propositions $\mathsf{Th}_i(q_x), \mathsf{Th}_i(\bot)$ 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 $\mathsf{merge}(i,j)$ be a proposition that signifies that threads i, j have been merged. In this case, $\mathsf{merge}(i,j)$ is true at position $e+1$. Let Merge be the set of all propositions $\mathsf{merge}(i,j)$ for $1 \leq i < j \leq m$.

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

- **Initial condition**(φ_{init})- At the first point of the word, we start the first thread and initialize all other threads as $\bot$: $\varphi_{init} = ((\mathsf{Th}_1(q_1)) \wedge \bigwedge\limits_{1<i\leq m} \mathsf{Th}_i(\bot))$.
- **Initiating runs at all points**(φ_{start})- To check the rational expression within an arbitrary interval, we need to start a new run from every point. $\varphi_{start} = \Box^{\mathsf{ns}}(\bigvee\limits_{i\leq m} \mathsf{Th}_i(q_1))$
- **Disallowing Redundancy**(φ_{no-red})- At any point of the word, if $i < j$ and $\mathsf{Th}_i(q_x)$ and $\mathsf{Th}_j(q_x)$ are both true, $q_x \neq q_y$. $\varphi_{no-red} = \bigwedge\limits_{x \in \mathsf{In}} \Box^{\mathsf{ns}}[\neg \bigvee\limits_{1\leq i<j\leq m} (\mathsf{Th}_i(q_x) \wedge \mathsf{Th}_j(q_x))]$
- **Merging Runs**(φ_{merge})- 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 merge thread Th_j with Th_i. We remember the merge with the proposition $\mathsf{merge}(i,j)$. 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: $\bigvee\limits_{\{(q_y,prop)\in(Q,2^{AP})|\delta(q_y,prop)=q_x\}} [prop \wedge \mathsf{Th}_i(q_y)]$.
  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 $\mathsf{Nxt}(\mathsf{Th}_i(q_x)) \wedge \bigwedge\limits_{j<i} \neg \mathsf{Nxt}(\mathsf{Th}_j(q_x)) \wedge \mathsf{Nxt}(\mathsf{Th}_k(q_x))$. 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 $\mathsf{NextMerge(i,k)} = \mathsf{O}[\mathsf{merge}(i,k) \wedge (\mathsf{Th}_k(\bot) \vee \mathsf{Th}_k(q_1)) \wedge \mathsf{Th}_i(q_x)]$.
  $\varphi_{\mathsf{merge}} = \bigwedge\limits_{x,i,k \in \mathsf{In} \wedge k>i} \Box^{\mathsf{ns}}[\psi(i,j,k,q_x) \rightarrow \mathsf{NextMerge(i,k)}]$.

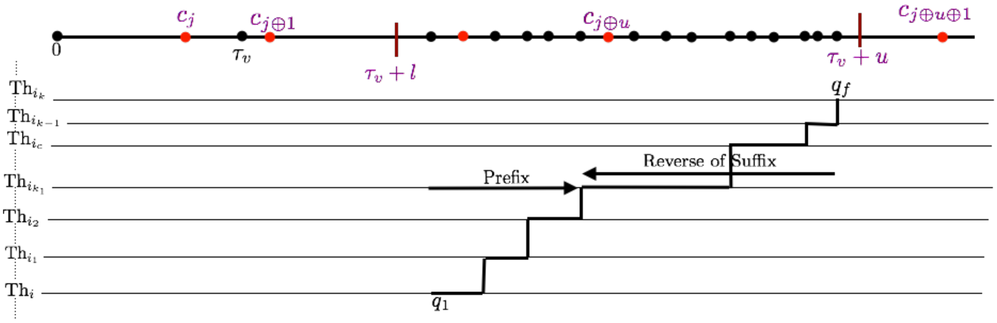

Figure 3 The linking thread at $c_{j\oplus u}$. The points in red are the oversampling integer points, and so are $\tau_v + l$ and $\tau_v + u$.

- **Propagating runs**($\varphi_{pro}, \varphi_{NO-pro}$)- 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 $\bigwedge\limits_{i,j\in\mathsf{In}\wedge i<j} \Box^{\mathsf{ns}}[\mathsf{NextTh}(i, j, q_x) \rightarrow \mathsf{O}[\mathsf{Th}_i(q_x) \wedge \neg\mathsf{merge}(i, j)]]$. If $\mathsf{Th}_i(\bot)$ is true at the current point, then at the next point, either $\mathsf{Th}_i(\bot)$ or $\mathsf{Th}_i(q_1)$. The latter condition corresponds to starting a new run on thread Th_i. $\varphi_{NO-pro}= \bigwedge\limits_{i\in\mathsf{In}} \Box^{\mathsf{ns}}\{\mathsf{Th}_i(\bot) \rightarrow \mathsf{O}(\mathsf{Th}_i(\bot) \vee \mathsf{Th}_i(q_1))\}$

Let Run be the formula obtained by conjuncting all formulae explained above. Once we construct the simple extension ρ', checking whether the rational expression $\mathsf{re{-}atom}$ holds in some interval I in the timed word ρ, is equivalent to checking that if u is the first action point within I, and if $\mathsf{Th}_i(q_1)$ holds at u, then after a series of merges of the form $\mathsf{merge}(i_1, i)$,$\mathsf{merge}(i_2, i_1)$, $\ldots \mathsf{merge}(j, i_n)$, at the last point v in the interval I, $\mathsf{Th}_j(q_f)$ is true, for some final state q_f. This is encoded as $\mathsf{GOODRUN}(q_f)$. It can be seen that the number of possible sequences of merges are bounded. Figure 2 illustrates the threads and merging. To write an MTL formula that checks the truth of $\mathsf{Rat}_{[l,u)}\mathsf{re{-}atom}$ at a point v, we need to oversample ρ' as shown below.

▶ **Lemma 6.** *Let $T = \Box^{\mathsf{ns}}[w \leftrightarrow \mathsf{Rat}_I\mathsf{re{-}atom}]$ 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 is equivalent to ψ modulo oversampling.*

Proof. Lets first consider the case when the interval I is bounded of the form $[l, u)$. Consider a point in ρ' with time stamp τ_v. To assert w at τ_v, we look at the first action point after time point $\tau_v + l$, and check that $\mathsf{GOODRUN}(last(q_f))$ holds, where $last(q_f)$ identifies the last action point just before $\tau_v + u$. The first difficulty is the possible absence of time points $\tau_v + l$ and $\tau_v + u$. To overcome this difficulty, we oversample ρ' by introducing points at times $t + l, t + u$, whenever t is a time point in ρ'. These new points are labelled with a new proposition ovs. Sadly, $last(q_f)$ cannot be written in MTL.

To address this, we introduce new time points at every integer point of ρ'. The starting point 0 is labelled c_0. Consecutive integer time points are marked $c_i, c_{i\oplus 1}$, where $\oplus$ is addition modulo the maximum constant used in the time interval in the RatMTL formula. This helps in measuring the time elapse since the first action point after $\tau_v + l$, till the last action point before $\tau_v + u$ as follows: if $\tau_v + l$ lies between points marked $c_j, c_{j\oplus 1}$, then the last integer point before $\tau_v + u$ is **uniquely** marked $c_{j\oplus u}$.

- Anchoring at τ_v, we assert the following at distance l: no action points are seen until the first action point where $\mathsf{Th}_i(q_1)$ is true for some thread Th_i. Consider the next point where $c_{j\oplus u}$ is seen. Let $\mathsf{Th}_{i_{k_1}}$ be the thread to which Th_i has merged at the last action point just before $c_{j\oplus u}$. Let us call $\mathsf{Th}_{i_{k_1}}$ the "last merged thread" before $c_{j\oplus u}$. The sequence of merges from Th_i till $\mathsf{Th}_{i_{k_1}}$ asserts a prefix of the run that we are looking for between $\tau_v + l$ and $\tau_v + u$. To complete the run we mention the sequence of merges from $\mathsf{Th}_{i_{k_1}}$ which culminates in some $\mathsf{Th}_{i_k}(q_f)$ at the last action point before $\tau_v + u$.

- Anchoring at τ_v, we assert the following at distance u: we see no action points since $\mathsf{Th}_{i_k}(q_f)$ at the action point before $\tau_v + u$ for some thread Th_{i_k}, and there is a path linking thread $\mathsf{Th}_{i_{k_1}}$ to Th_{i_k} since the point $c_{j\oplus u}$. We assert that the "last merged thread", $\mathsf{Th}_{i_{k_1}}$ is active at $c_{j\oplus u}$: this is the linking thread which is last merged into before $c_{j\oplus u}$, and which is the first thread which merges into another thread after $c_{j\oplus u}$.

These two formulae thus "stitch" the actual run observed between points $\tau_v + l$ and $\tau_v + u$. The formal technical details can be seen in Appendix D in the full version. If I was an unbounded interval of the form $[l, \infty)$, then 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. ◀

In a similar manner, we can eliminate the URat modality, the proof of which can be found in Appendix E in the full version. If we choose to work on logic $\mathsf{MITL} + \mathsf{URat}$, we obtain a 2EXPSPACE upper bound for satisfiability checking, since elimination of URat results in an equisatisfiable MITL formula. This is an interesting consequence of the oversampling technique; without oversampling, we can eliminate URat obtaining 1-TPTL (Appendix C, full version). However, 1-TPTL does not enjoy the benefits of non-punctuality, and is non-primitive recursive (Appendix F, full version).

4 Automaton-Metric Temporal Logic-Freeze Logic Equivalences

The focus of this section is to obtain equivalences between automata, temporal and freeze logics. First of all, we identify a fragment of RatMTL denoted SfrMTL, where the rational expressions in the formulae are all star-free. We then show the equivalence between po-1-clock ATA, $1-\mathsf{TPTL}$, and SfrMTL (po-1-clock ATA $\subseteq$ SfrMTL $\subseteq$ $1-\mathsf{TPTL}$ $\equiv$ po-1-clock ATA). The main result of this section gives a tight automaton-logic connection in Theorem 7, and is proved using Lemmas 9, 10 and 11.

▶ **Theorem 7.** *1−TPTL, SfrMTL and po-1-clock ATA are all equivalent.*

We first show that partially ordered 1-clock alternating timed automata (po-1-clock ATA) capture exactly the same class of languages as $1-\mathsf{TPTL}$. We also show that $1-\mathsf{TPTL}$ is equivalent to the subclass SfrMTL of RatMTL where the rational expressions re involved in the formulae are such that $L(\mathsf{re})$ is star-free.

A 1-clock ATA [15] 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 $\delta : S \times \Sigma \rightarrow \Phi(S \cup \Sigma \cup X)$ where $\Phi(S \cup \Sigma \cup X)$ is a set of formulae defined by the grammar $\varphi ::= \top | \bot | \varphi_1 \wedge \varphi_2 | \varphi_1 \vee \varphi_2 | s | x \bowtie c | x.\varphi$ where $s \in S$, and $x.\varphi$ is a binding construct corresponding 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 (i) 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 $\downarrow s_i = \{s_j \mid s_j \prec s_i\}$, (ii) $x.s$ does not appear in $\delta(s,a)$ for all $s \in S, a \in \Sigma$.

▶ **Example 8.** Consider the po-1-clock ATA $\mathcal{A} = (\{a,b\}, \{s_0, s_a, s_\ell\}, s_0, \{s_0, s_\ell\}, \delta)$ 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, and has some symbol at distance > 1 from it.

▶ **Lemma 9.** po-*1-clock ATA and* $1{-}\mathsf{TPTL}$ *are equivalent in expressive power.*

The translation from $1{-}\mathsf{TPTL}$ to po-1-clock ATA is easy, as in the translation from MTL to po-1-clock ATA. For the reverse direction, we start from the lowest location (say s) in the partial order, and replace the transitions of s by a 1-TPTL formula that models timed words which are accepted, when started in s. The accepting behaviours of each location s, denoted $\mathsf{Beh}(s)$ is computed bottom up. The 1-TPTL formula that we are looking for is $\mathsf{Beh}(s_0)$ where s_0 is the initial location. In example 8, $\mathsf{Beh}(s_\ell) = \Box^{\mathsf{ns}} b$, $\mathsf{Beh}(s_a)$=$(x < 1)\ \mathsf{U}^{\mathsf{ns}}(x > 1)$, $\mathsf{Beh}(s_0) = [(a \wedge x.\mathsf{O}\mathsf{Beh}(s_a)) \vee b]\ \mathsf{W}(a \wedge \mathsf{O}\mathsf{Beh}(s_\ell))$ =$((a \wedge (x.\mathsf{O}[(x < 1)\ \mathsf{U}^{\mathsf{ns}} x > 1])) \vee b)\ \mathsf{W}(a \wedge \mathsf{O}\Box^{\mathsf{ns}} b)$. Step by step details for Lemma 9 can be seen in Appendix H of the full version.

▶ **Lemma 10.** $\mathsf{SfrMTL} \subseteq 1 - \mathsf{TPTL}$.

The proof of Lemma 10 can be found in Appendix I of the full version. The intuition is to freeze a clock x at the current point, and write an LTL formula equivalent to the star-free expression over an interval I which can be constrained checking $x \in I$ in the LTL formula.

▶ **Lemma 11.** *(*po-*1-clock ATA to* SfrMTL*) Given a* po-*1-clock ATA* $\mathcal{A}$*, we can construct a* SfrMTL *formula* φ *such that* $L(\mathcal{A}) = L(\varphi)$*.*

Proof. (Sketch) We give a proof sketch here, a detailed proof can be found in Appendix J of the full version. 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. Let $R_{2i} = [i,i], R_{2i+1} = (i, i+1), 0 \leq i < K$ and $R_K^+ = (K, \infty)$ be the regions $\mathcal{R}$ of x. Let $R_h \prec R_k$ denote that region R_h precedes region R_k. For each location s,$\mathsf{Beh}(s)$ as computed in Lemma 9 is a 1-TPTL formula that gives the timed behaviour starting at s, using constraints $x \sim c$ since the point where x was frozen. In example 8, $\mathsf{Beh}(s_a)$=$(x < 1)\ \mathsf{U}^{\mathsf{ns}}(x > 1)$, allows symbols a, b as long as $x < 1$ keeping the control in s_a, has no behaviour at $x = 1$, and allows control to leave s_a when $x > 1$. For any s, we "distribute" $\mathsf{Beh}(s)$ across regions by untiming it. In example 8, $\mathsf{Beh}(s_a)$ is $\Box^{\mathsf{ns}}(a \vee b)$ for regions R_0, R_1, it is $\bot$ for R_2 and is $(a \vee b)$ for R_1^+. Given any $\mathsf{Beh}(s)$, and a pair of regions $R_j \preceq R_k$, such that s has a non-empty behaviour in region R_j, and control leaves s in R_k, the untimed behaviour of s between regions $R_j, \ldots, R_k$ is written as LTL formulae $\varphi_j, \ldots, \varphi_k$. This results in a "behaviour description" (or BD for short) denoted $\mathsf{BD}(s, R_j, R_k) = \{\mathsf{BD}_1, \mathsf{BD}_2, \ldots, \mathsf{BD}_w\}$[2] where each BD_i is a $2K + 1$

[2] Note that if s is one of the lowest locations in the partial order, this is a singleton set. We will denote the elements of $\mathsf{BD}(s, R_j, R_k)$ as $BD_{no.}$.

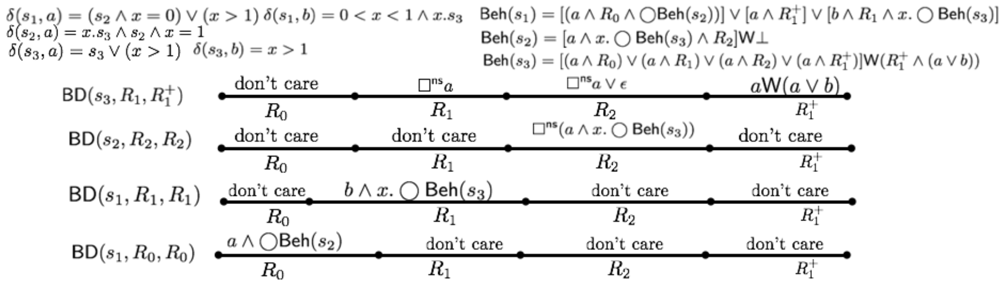
$\delta(s_1,a)=(s_2\wedge x=0)\vee(x>1)$ $\delta(s_1,b)=0<x<1\wedge x.s_3$
$\delta(s_2,a)=x.s_3\wedge s_2\wedge x=1$
$\delta(s_3,a)=s_3\vee(x>1)$ $\delta(s_3,b)=x>1$
$\mathsf{Beh}(s_1)=[(a\wedge R_0\wedge\bigcirc\mathsf{Beh}(s_2))]\vee[a\wedge R_1^+]\vee[b\wedge R_1\wedge x.\bigcirc\mathsf{Beh}(s_3)]$
$\mathsf{Beh}(s_2)=[a\wedge x.\bigcirc\mathsf{Beh}(s_3)\wedge R_2]\mathsf{W}\bot$
$\mathsf{Beh}(s_3)=[(a\wedge R_0)\vee(a\wedge R_1)\vee(a\wedge R_2)\vee(a\wedge R_1^+)]\mathsf{W}(R_1^+\wedge(a\vee b))$
$\mathsf{BD}(s_3,R_1,R_1^+)$
don't care
$\square^{\mathsf{ns}}a$
$\square^{\mathsf{ns}}a\vee\epsilon$
$a\mathsf{W}(a\vee b)$
R_0
R_1
R_2
R_1^+
$\mathsf{BD}(s_2,R_2,R_2)$
don't care
don't care
$\square^{\mathsf{ns}}(a\wedge x.\bigcirc\mathsf{Beh}(s_3))$
don't care
R_0
R_1
R_2
R_1^+
$\mathsf{BD}(s_1,R_1,R_1)$
don't care
$b\wedge x.\bigcirc\mathsf{Beh}(s_3)$
don't care
don't care
R_0
R_1
R_2
R_1^+
$\mathsf{BD}(s_1,R_0,R_0)$
$a\wedge\bigcirc\mathsf{Beh}(s_2)$
don't care
don't care
don't care
R_0
R_1
R_2
R_1^+

Figure 4 A po-1-clock ATA with initial location s_1 and s_2, s_3 are accepting.

tuples with $\mathsf{BD}_i[R_l]=\varphi_l$ for $j\le l\le k$, and $\mathsf{BD}[R]=\top$ denoting "dont care" for the other regions. Let $\mathsf{BDSet}(s)$ denote the union of all BDs for a location s. For the initial location s_0, consider all $\mathsf{BD}_i\in\mathsf{BD}(s_0,R_j,R_k)$ that have a behaviour starting in R_j, and ends in an accepting configuration in R_k. Each LTL formula $\mathsf{BD}_i[R_i]$ is replaced with a star-free rational expression denoted $\mathsf{re}(\mathsf{BD}(s_0,R_j,R_k)[R_i])$. Then $\mathsf{BD}(s_0,R_j,R_k)$ is transformed into a SfrMTL formula $\varphi(s_0,R_j,R_k)=\bigvee_{BD_i\in\mathsf{BD}(s_0,R_j,R_k)}\bigwedge_{j\le g\le k}\mathsf{Rat}_{R_g}\mathsf{re}(\mathsf{BD}_i[R_g])$. The language accepted by the po-1-clock ATA $\mathcal{A}$ is then given by $\bigvee_{0\le j\le k\le 2K}\varphi(s_0,R_j,R_k)$.

Computing $\mathsf{BD}(s,R_i,R_j)$ for a location s and pair of regions $R_i\preceq R_j$. We first compute $\mathsf{BD}(s,R_i,R_j)$ for locations s which are lowest in the partial order, followed by computing $\mathsf{BD}(s',R_i,R_j)$ for locations s' which are higher in the order. For any location s, $\mathsf{Beh}(s)$ has the form φ or $\varphi_1\,\mathsf{W}\varphi_2$ or $\varphi_1\,\mathsf{U}^{\mathsf{ns}}\varphi_2$, where $\varphi,\varphi_1,\varphi_2$ are disjunctions of conjunctions over $\Phi(S\cup\Sigma\cup X)$, where S is the set of locations with or without the binding construct $x.$, and X is a set of clock constraints of the form $x\sim c$. Each conjunct has the form $\psi\wedge x\in R$ where $\psi\in\Phi(\Sigma\cup S)$ and $R\in\mathcal{R}$. Let $\varphi_1=\bigvee(P_i\wedge C_i),\varphi_2=\bigvee(Q_j\wedge E_j)$ where $P_i,Q_j\in\Phi(\Sigma\cup S)$ and $C_i,E_j\in\mathcal{R}$. Let $\mathcal{C}$ and $\mathcal{E}$ be shorthands for any C_k,E_l.

If $\mathsf{Beh}(s)$ is an expression without U,W (the case of φ above), then $\mathsf{BD}(s,R_i,R_i)$ is defined for a region R_i if $\varphi=\bigvee(Q_j\wedge E_j)$ and there is some E_l with $x\in R_i$. It is a $2K+1$ tuple with $\mathsf{BD}(s,R_i,R_i)[R_i]=Q_l$[3]we know that , and the rest of the entries are $\top$ (for dont care). If $\mathsf{Beh}(s)$ has the form $\varphi_1\,\mathsf{W}\varphi_2$ or $\varphi_1\,\mathsf{U}^{\mathsf{ns}}\varphi_2$, then for $R_i\preceq R_j$, and a location s, $\mathsf{BD}(s,R_i,R_j)=\{BD_1\}$ where BD_1 is a $2K+1$ tuple with (i) formula $\top$ in regions $R_0,\dots,R_{i-1},R_{j+1},\dots,R_K^+$, (ii) If $C_k=E_l=(x\in R_j)$ for some C_k,E_l, then the LTL formula in region R_j is $P_k\,\mathsf{U}Q_l$ if s is not accepting, and is $P_k\,\mathsf{W}Q_l$ if s is accepting, (iii) If no C_k is equal to any E_l, and if $E_l=(x\in R_j)$ for some l, then the formula in region R_j is Q_l. If $C_m=(x\in R_i)$ for some m, then the formula for region R_i is $\square^{\mathsf{ns}}P_m$. If there is some $C_h=(x\in R_w)$ for $i<w<j$, then the formula in region R_w is $\square^{\mathsf{ns}}P_h\vee\epsilon$, where ϵ signifies that there may be no points in regions R_w. If there are no C_m's such that $C_m=(x\in R_w)$ for $R_i\prec R_w\prec R_j$, then the formula in region R_w is ϵ. ϵ is used as a *special symbol* in LTL whenever there is no behaviour in a region.

$\mathsf{BD}(s,R_i,R_j)$ for location s lowest in po. Let s be a location that is lowest in the partial order. In general, if s is the lowest in the partial order, then $\mathsf{Beh}(s)$ has the form $\varphi_1\,\mathsf{W}\varphi_2$ or $\varphi_1\,\mathsf{U}^{\mathsf{ns}}\varphi_2$ or φ where $\varphi,\varphi_1,\varphi_2$ are disjunctions of conjunctions over $\Phi(\Sigma\cup X)$. Each conjunct has the form $\psi\wedge x\in R$ where $\psi\in\Phi(\Sigma)$ and $R\in\mathcal{R}$. See Figure 4, with regions

[3] We abuse the notation by indexing the $\mathsf{BD}(s,R_i,R_i)[R_i]$ instead of BD when it is a singleton set.

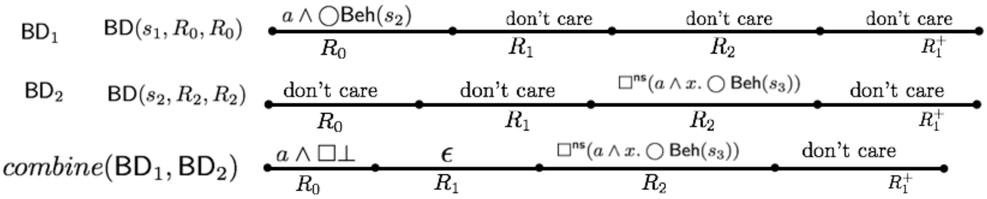

Figure 5 Combining BDs

R_0, R_1, R_2, R_1^+, and some example BDs. In Figure 4, using the BDs of the lowest location s_3, we write the SfrMTL formula for $\mathsf{Beh}(s_3)$: $\psi(s_3) = \varphi_{R_0}(s_3) \wedge \varphi_{R_1}(s_3) \wedge \varphi_{R_2}(s_3) \wedge \varphi_{R_1^+}(s_3)$, where each φ_R describes the behaviour of s_3 starting from region R. For a fixed region R_i, $\varphi_{R_i}(s_3)$ is $\bigwedge_{R_g \prec R_i} \mathsf{Rat}_{R_g}\epsilon \wedge \mathsf{Rat}_{R_i}\Sigma^+ \rightarrow \{\bigvee_{R_i \prec R_j} \varphi(s_3, R_i, R_j)\}$, where $\varphi(s_3, R_i, R_j)$ is described above. $\mathsf{Rat}_{R_g}\epsilon$ means that there is no behaviour in R_g. $\varphi_{R_0}(s_3)$ is given by $\mathsf{Rat}_{R_0}\Sigma^+ \rightarrow \{(\mathsf{Rat}_{R_0}a^* \wedge \mathsf{Rat}_{R_1}[a^* + \epsilon] \wedge \mathsf{Rat}_{R_2}[a^* + \epsilon] \wedge \mathsf{Rat}_{R_1^+}[a^* + a^*b])\}$.

BD(s, R_i, R_j) for a location s which is higher up . If s is not the lowest in the partial order, then $\mathsf{Beh}(s)$ can have locations $s' \in\downarrow s$. s' occurs as $\mathsf{O}(s')$ or $x.\mathsf{O}(s')$ in $\mathsf{Beh}(s)$. For $x.\mathsf{OBeh}(s_3)$ in $\mathsf{BD}(s, R_i, R_j)$, since the clock is frozen, we plug-in the SfrMTL formula $\psi(s_3)$ computed above for $x.\mathsf{OBeh}(s_3)$ in $\mathsf{BD}(s_1, R_i, R_j)$. For instance, in figure 4, $x.\mathsf{OBeh}(s_3)$ appears in $\mathsf{BD}(s_2, R_2, R_2)[R_2]$. We simply plug in the SfrMTL formula $\psi(s_3)$ in its place. Likewise, for locations s, t, if $\mathsf{OBeh}(t)$ occurs in $\mathsf{BD}(s, R_i, R_j)[R_k]$, we look up $\mathsf{BD}(t, R_k, R_l) \in \mathsf{BDSet}(t)$ for all $R_k \preceq R_l$ and *combine* $\mathsf{BD}(s, R_i, R_j), \mathsf{BD}(t, R_k, R_l)$ in a manner described below. This is done to detect if the "next point" for t has a behaviour in R_k or later.

(a) If the next point for t is in R_k itself, then we *combine* all $\mathsf{BD}_1 \in \mathsf{BD}(s, R_i, R_j)$ with every $\mathsf{BD}_2 \in \bigcup\limits_{R_k \preceq R_l} \mathsf{BD}(t, R_k, R_l) \subseteq \mathsf{BDSet}(t)$ as follows[4]. $\mathsf{combine}(\mathsf{BD}_1, \mathsf{BD}_2)$ results in BD_3 such that $\mathsf{BD}_3[R]$=$\mathsf{BD}_1[R]$ for $R \prec R_k$, $\mathsf{BD}_3[R]$=$\mathsf{BD}_1[R] \wedge \mathsf{BD}_2[R]$ for $R_k \prec R$, where $\wedge$ denotes component wise conjunction. $\mathsf{BD}_3[R_k]$ is obtained by replacing $\mathsf{OBeh}(s_2)$ in $\mathsf{BD}_1[R_k]$ with $\mathsf{BD}_2[R_k]$. Doing so enables the next point in R_k, emulating the behaviour of t in R_k.

(b) Assume the next point for t lies in R_b, $R_k \prec R_b$. The difference with case (a) is that we combine $\mathsf{BD}_1 \in \mathsf{BD}(s, R_i, R_j)$ with $\mathsf{BD}_2 \in \bigcup\limits_{R_k \preceq R_l} \mathsf{BD}(t, R_k, R_l) \subseteq \mathsf{BDSet}(t)$. Then $\mathsf{combine}(\mathsf{BD}_1, \mathsf{BD}_2)$ results in a BD, say BD_3 such that $\mathsf{BD}_3[R] = \mathsf{BD}_1[R]$ for $R \prec R_k$, $\mathsf{BD}_3[R] = \mathsf{BD}_1[R] \wedge \mathsf{BD}_2[R]$ for all $R_b \preceq R$, and $\mathsf{BD}_3[R] = \epsilon$ for $R_k \prec R \prec R_b$. The $\mathsf{OBeh}(t)$ in $\mathsf{BD}_1[R_k]$ is replaced with $\Box\bot$ to signify that the next point is not enabled for t. See Figure 5 where $R_b = R_2$. The conjunction with $\Box\bot$ in R_0 signifies that the next point for s_2 is not in R_0; the ϵ in R_1 signifies that there are no points in R_1 for s_2. Conjuncting $\Box\bot$ in a region signifies that the next point does not lie in this region.

We look at the "accepting" BDs in $\mathsf{BDSet}(s_0)$, viz., all $\mathsf{BD}(s_0, R_j, R_k)$, such that acceptance happens in R_k, and s_0 has a behaviour starting in R_j. The LTL formulae $\mathsf{BD}_i[R]$ [where $\mathsf{BD}_i \in \mathsf{BDSet}(s_0)$] is replaced with star-free expression $\mathsf{re}(\mathsf{BD}_i[R])$. $\mathsf{BDSet}(s_0)$ gives an SfrMTL formula φ= $\bigvee\limits_{\mathsf{BD}_i \in \mathsf{BDSet}(s_0)} \bigwedge_{R_j \preceq R \preceq R_k} \mathsf{Rat}_R\mathsf{re}(\mathsf{BD}_i[R])$ whose language is $L(\varphi)$=$L(\mathcal{A})$. ◀

[4] Take cross product of two sets and then applying combine operation

5 Discussion

We propose RatMTL which significantly increases the expressive power of MTL and yet retains decidability over pointwise finite words. The Rat operator added to MTL syntactically subsumes several other modalities in literature including threshold counting, modulo counting and the Pnueli modality. The reduction of RatMTL to equisatisfiable MTL has elementary complexity and allows us to identify two fragments of RatMTL with 2EXPSPACE and EXPSPACE satisfiability. In [11], oversampled temporal projections were used to reduce MTL with punctual future and non-punctual past to MTL. Our reduction can be combined with the one in [11] to obtain decidability of RatMTL and elementary decidability of MITL + URat + non-punctual past. These are amongst the most expressive decidable extensions of MTL known so far. The exact complexity class for satisfiability of MITL + URat is an interesting open question. We also show an exact logic-automaton correspondence between the fragment SfrMTL and po-1-clock ATA. It is not difficult to see that full RatMTL can be reduced to equivalent 1 clock ATA. This provides an alternative proof of decidability of RatMTL but the proof will not extend to decidability of RatMTL+ non-punctual past, nor prove elementary decidability of MITL + URat+non-punctual past. Hence, we believe that our proof technique has some advantages. An interesting related formalism of timed regular expressions was defined by Asarin, Maler, Caspi, and shown to be expressively equivalent to timed automata. Our RatMTL has orthogonal expressive power, and it is boolean closed (thus the decidability of universality checking comes for free). The exact expressive power of RatMTL which is between 1-clock ATA and po-1-clock ATA is open.

References

1 R. Alur, T. Feder, and T. Henzinger. The benefits of relaxing punctuality. *J.ACM*, 43(1):116–146, 1996.

2 Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. *Inf. Comput.*, 104(1):35–77, 1993. doi:10.1006/inco.1993.1025.

3 Eugene Asarin, Paul Caspi, and Oded Maler. Timed regular expressions. *J. ACM*, 49(2):172–206, 2002. doi:10.1145/506147.506151.

4 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. doi:10.1109/LICS.1999.782629.

5 Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the expressiveness of TPTL and MTL. In *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, 25th International Conference, Hyderabad, India, December 15-18, 2005, Proceedings*, pages 432–443, 2005. doi:10.1007/11590156_35.

6 Cindy Eisner and Dana Fisman. *A Practical Introduction to PSL*. Springer, 2006.

7 IEEE P1850-Standard for PSL-Property Specification Language, 2005.

8 Jesper G. Henriksen and P. S. Thiagarajan. Dynamic linear time temporal logic. *Ann. Pure Appl. Logic*, 96(1-3):187–207, 1999. doi:10.1016/S0168-0072(98)00039-6.

9 Philippe Herrmann. Renaming is necessary in timed regular expressions. In *Foundations of Software Technology and Theoretical Computer Science, 19th Conference, Chennai, India, December 13-15, 1999, Proceedings*, pages 47–59, 1999. doi:10.1007/3-540-46691-6_4.

10 P. Hunter. When is metric temporal logic expressively complete? In *CSL*, pages 380–394, 2013.

11 S. N. Krishna K. Madnani and P. K. Pandya. Partially punctual metric temporal logic is decidable. In *TIME*, pages 174–183, 2014.

12 Shankara Narayanan Krishna, Khushraj Madnani, and Paritosh K. Pandya. Metric temporal logic with counting. In *Foundations of Software Science and Computation Structures - 19th International Conference, FOSSACS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, pages 335–352, 2016.

13 F. Laroussinie, A. Meyer, and E. Petonnet. Counting ltl. In *TIME*, pages 51–58, 2010.

14 K. Lodaya and A. V. Sreejith. Ltl can be more succinct. In *ATVA*, pages 245–258, 2010.

15 J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *LICS*, pages 188–197, 2005.

16 A. Rabinovich. Complexity of metric temporal logic with counting and pnueli modalities. In *FORMATS*, pages 93–108, 2008.