H. Langmaack, W.-P. de Roever, J. Vytopil, ed., *Formal Techniques in Real-Time and Fault-Tolerant Systems*, Lübeck, Germany, 1994. Lect. Notes Comput. Sci., vol. 863, pp. 694–715.

Specifying Timed State Sequences in Powerful Decidable Logics and Timed Automata

(Extended Abstract)

Thomas Wilke

Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, D-24098 Kiel, Germany*

Abstract. A monadic second-order language, denoted by $\mathcal{L}\mathrm{d}$, is introduced for the specification of sets of timed state sequences. A fragment of $\mathcal{L}\mathrm{d}$, denoted by $\overleftrightarrow{\mathcal{L}\mathrm{d}}$, is proved to be expressively complete for timed automata (Alur and Dill), i. e., every timed regular language is definable by a $\overleftrightarrow{\mathcal{L}\mathrm{d}}$-formula and every $\overleftrightarrow{\mathcal{L}\mathrm{d}}$-formula defines a timed regular language. As a consequence the satisfiability problem for $\overleftrightarrow{\mathcal{L}\mathrm{d}}$ is decidable.
Timed temporal logics are shown to be effectively embeddable into $\overleftrightarrow{\mathcal{L}\mathrm{d}}$ and hence turn out to have a decidable theory. This applies to $\mathrm{TL}_\star$ (Manna and Pnueli) and $\mathrm{EMITL}_\mathrm{p}$, which is obtained by extending the logic MITL_p (Alur and Henzinger) by automata operators (Sistla, Vardi, and Wolper).
For every positive natural number k the full monadic second-order logic $\mathcal{L}\mathrm{d}$ and $\overleftrightarrow{\mathcal{L}\mathrm{d}}$ are equally expressive modulo the set of timed state sequences of variability $\leq k$. Therefore the $\mathcal{L}\mathrm{d}$-theory of the set of timed state sequences of variability $\leq k$ is decidable.

Introduction

Timed state sequences are widely accepted as a model of real-time computations of finite state systems, and sets of timed state sequences are used to model the behaviour of such systems [18] [9].

There are two main formalisms commonly used to define sets of timed state sequences: timed automata and variants [5] [19] [17] on the one hand, and temporal logics [6] [10] [15] on the other hand. The former are modelled on physical systems, the latter are used for specification. Timed automata have been extensively used to show that certain theories of the set of timed state sequences are decidable and to prove the decidability of model checking problems [6] [2] [1] [10] [4]. These results are based on the decidability of the emptiness problem for timed automata [4].

In this paper we investigate the possibilities of defining sets of timed state sequences by monadic second-order logics. We present a monadic second-order logic

* This paper was written during the author's stay at Laboratoire Bordelais de Recherche en Informatique (LaBRI). The presented results are part of the author's dissertation [24].

which is expressively complete for timed automata. The first result of this kind—an equivalence between recognition by automata and definability by monadic second-order formulas—was obtained by Büchi in 1960 when he showed that a set of finite words is regular (recognized by a Rabin-Scott automaton) if and only if it is defined by a monadic second-order formula in a suitable signature [12]. Similar results for infinite words, trees, (Mazurkiewicz-)traces, etc. followed [13] [23] [16].

The main obstacle for obtaining an analogous result in the case of timed automata is that the class of sets of timed state sequences defined by timed automata is not closed under complementation; thus there is no hope to find a full monadic second-order logic being complete for timed automata. Instead we use a fragment of a suitable monadic second-order language. It is called the monadic logic of relative distance, and is denoted by $\mathcal{L}\overleftrightarrow{\mathrm{d}}$; the full monadic logic is denoted by $\mathcal{L}\mathrm{d}$.

Since the transformation between timed automata and $\mathcal{L}\overleftrightarrow{\mathrm{d}}$-formulas is effective, we can use the decidability of the emptiness problem for timed automata to show the decidability of the satisfiability problem for $\mathcal{L}\overleftrightarrow{\mathrm{d}}$.

This result is useful from the following point of view. If we are given a (temporal) logic $\mathcal{L}$ for the specification of sets of timed state sequences and we want to show that the $\mathcal{L}$-theory of the set of timed state sequences is decidable we only need to establish an effective translation of $\mathcal{L}$ into $\mathcal{L}\overleftrightarrow{\mathrm{d}}$ that preserves logical equivalence. We give two examples of this kind: the logic TL_{Γ} introduced by Manna and Pnueli [20] and an extension of the metric temporal interval logic $\mathrm{MITL}_{\mathrm{p}}$ with past operators [10]. The latter is denoted by $\mathrm{EMITL}_{\mathrm{p}}$ (extended metric interval temporal logic with past operators) and obtained from $\mathrm{MITL}_{\mathrm{p}}$ by adding automata operators (cf. the work by Wolper [25] and Sistla, Vardi, and Wolper [26]). Automata operators allow, for instance, to express a property such as "every stimulus p is followed by a response q and, then, by another response r within 5 time units of the stimulus p" which is not known to be expressible in $\mathrm{MITL}_{\mathrm{p}}$. For other reasons, it is clear that $\mathrm{EMITL}_{\mathrm{p}}$ subsumes properly both $\mathrm{MITL}_{\mathrm{p}}$ and TL_{Γ}.

We are also interested in what happens when $\mathcal{L}\overleftrightarrow{\mathrm{d}}$ is only interpreted in timed state sequences with a fixed bounded variability. We show that one obtains closure under negation (complementation), which implies that in this case $\mathcal{L}\overleftrightarrow{\mathrm{d}}$ and $\mathcal{L}\mathrm{d}$ are of the same expressive power and that the corresponding $\mathcal{L}\mathrm{d}$-theory is decidable.

In the first three sections we present the basic definitions and main results. In Sect. 4–7 proof sketches are given, Sect. 8 discusses alternative models of real-time computations.

We conclude this introduction with a short review of the notions of time sequence and timed state sequence. A *time sequence* $\mathbf{t}$ is a strictly increasing, divergent sequence $\mathbf{t} = t_0, t_1, \ldots$ of real numbers starting with 0. Every t_i is called a *position.* A *set P of propositions* is a non empty finite set. A subset of a set P of propositions is called a *P-state.*

Throughout the paper we fix a set P of propositions.

A *timed state sequence* τ over P is a pair $(\mathbf{s}, \mathbf{t})$ consisting of an infinite sequence $\mathbf{s} = s_0, s_1, s_2, \ldots$ of P-states and a time sequence $\mathbf{t}$. The set of positions of $\mathbf{t}$ is denoted by $\mathrm{pos}(\tau)$. The set of all timed state sequences over P is denoted by $\mathrm{TSS}(P)$.

1 Monadic Logic and Timed Automata

The signature $\Sigma(P)$ contains the symbol $<$, for every proposition p a unary predicate Q_p, and for every natural number c and every relation symbol $\sim \in \{=, \neq, <, >, \leq, \geq\}$ a binary *distance predicate* $\mathrm{d}(.,.) \sim c$.

The language of the *monadic logic of distance* over P is the monadic second-order language in the signature $\Sigma(P)$. It is denoted by $\mathcal{L}\mathrm{d}(P)$. Variables for elements of the universe of a structure are denoted by small letters and variables for subsets of the universe by capital letters. As usual, we write Xx for "x is element of X".

An atomic formula of the form $\mathrm{d}(x, y) \sim c$ is called *distance formula.*

With every timed state sequence $\tau = (\mathbf{s}, \mathbf{t})$ over P we associate a $\Sigma(P)$-structure $\tau^{\Sigma(P)}$. Its universe is the set of positions of $\mathbf{t}$, the symbol $<$ is interpreted by the natural order of the reals (restricted to the universe), for $p \in \mathcal{P}$ the predicate Q_p is interpreted by $\{t_i \mid p \in s_i\}$, and for every natural number c and every relation symbol $\sim$ as above the interpretation of $\mathrm{d}(.,.) \sim c$ contains the pair (t_i, t_j) if $|t_i - t_j| \sim c$ and $i \leq j$ hold.

For notational simplicity we identify τ with $\tau^{\Sigma(P)}$ and $\mathrm{TSS}(P)$ with the set $\{\tau^{\Sigma(P)} \mid \tau \in \mathrm{TSS}(P)\}$.

We adopt the convention that 'equivalence' stands for equivalence modulo $\mathrm{TSS}(P)$ and 'satisfiability' means satisfiability in $\mathrm{TSS}(P)$.

Unrestricted use of distance predicates leads to an undecidable theory:

Theorem 1 (Alur & Henzinger [11][2]). *The theory of* $\mathrm{TSS}(P)$ *in the first-order fragment of* $\mathcal{L}\mathrm{d}(P)$ *is undecidable.*

The halting problem for Turing machines (or two counter machines) can be reduced to the validity problem of $\mathcal{L}\mathrm{d}(P)$ for two reasons: first, there is no bound on the number of positions that can be put into intervals of length one, i. e., an arbitrary amount of information can be stored in an interval of length one, such as a configuration of a Turing machine. Second, the distance predicate $\mathrm{d}(x, y) = 1$ allows to describe correspondences between positions in adjacent intervals of length one, in particular, one can describe that the positions in an interval of length one encode a Turing machine configuration that is the predecessor of the configuration encoded by the interval that borders on the left.

In this paper we are interested in a fragment of $\mathcal{L}\mathrm{d}(P)$ that has a decidable satisfiability problem. In this fragment the use of distance formulas is restricted (which, in fact, will rule out the possibility of describing correspondences between adjacent intervals).

[2] In [11] the result is phrased for a two-sorted logic, but it transfers immediately to our situation.

For every relation symbol $\sim$ as before and every natural number c consider the formulas $\overleftarrow{\mathrm{d}}(X,x) \sim c$ and $\overrightarrow{\mathrm{d}}(x,X) \sim c$ defined by

$$\begin{aligned} &\overleftarrow{\mathrm{d}}(X,x) \sim c \;\triangleq \\ &\quad \exists y\,(y < x \wedge Xy \wedge \neg\exists y'(y < y' < x \wedge Xy') \wedge \mathrm{d}(x,y) \sim c) \end{aligned} \tag{1}$$

and

$$\begin{aligned} &\overrightarrow{\mathrm{d}}(x,X) \sim c \;\triangleq \\ &\quad \exists y\,(x < y \wedge Xy \wedge \neg\exists x'(x < x' < y \wedge Xx') \wedge \mathrm{d}(x,y) \sim c)\ . \end{aligned} \tag{2}$$

The so-called *past formula* (1) is read as follows: "there exists a position less than x and belonging to X and the distance between the greatest such position and x satisfies $\sim c$." Similarly, the *future formula* (2) is read: "there exists a position greater than x and belonging to X and the distance between the least such position and x satisfies $\sim c$."

The formulas (1) and (2) are called *relative distance formulas.*

Now, the set $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$ of the *monadic logic of relative distance over* P contains the $\mathcal{L}\mathrm{d}(P)$-formulas of the form $\exists X_0 \ldots \exists X_{m-1}\psi$ where ψ is built from $\mathrm{Q}_p x$, $x < y$, Xx, $\overleftarrow{\mathrm{d}}(X_i,x) \sim c$, and $\overrightarrow{\mathrm{d}}(x,X_i) \sim c$ with $i < m$ using boolean connectives, quantification of first-order and set variables except for $X_0, \ldots, X_{m-1}$.

Note that universal quantification of set variables is still allowed in $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$, provided the quantified variable is not used as an argument of a relative distance formula.

If ϕ is a $\mathcal{L}\mathrm{d}(P)$-sentence, then ϕ defines a set $\mathrm{L}_P(\phi) \subseteq \mathrm{TSS}(P)$ as follows:

$$\mathrm{L}_P(\phi) = \{\tau \in \mathrm{TSS}(P) \mid \tau \models \phi\}\ .$$

We say that $\mathrm{L}_P(\phi)$ *is defined by* ϕ.

The fundamental result of the paper is a characterization of the sets definable by $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formulas in terms of timed automata.

In the following we refer to timed automata as introduced in [4]. For the reader's convenience, we briefly review the basic notions concerning timed automata.

Let A be a finite alphabet. A timed word over A is a pair $(\mathbf{a},\mathbf{t})$ where $\mathbf{t}$ is a time sequence and $\mathbf{a}$ an infinite sequence of letters of A. A timed language over A is a set of timed words over A. Notice that if $A = \wp(P)$, then $\mathrm{TSS}(P)$ is the set of all timed words over A and a timed language over A is a subset of $\mathrm{TSS}(P)$.

A timed automaton $\mathfrak{A}$ over A is a tuple (Q, C, q_0, Δ, F) consisting of a finite set Q of locations[3], a finite set C of clocks, an initial location $q_0 \in Q$, a finite transition relation Δ, and a set $F \subseteq Q$ of final locations. An element of the transition relation is of the form (q, a, ϕ, M, q'), where q and q' are locations, a is a letter, ϕ is a boolean combination of atomic formulas of the form $c \sim d$ for some clock $c \in C$ and some natural number d, and M is a subset of C.

[3] Traditionally, this set would be called a 'state set', but 'states' already occur in the definition of a timed state sequence.

A run of $\mathfrak{A}$ on a timed word $(\mathbf{a}, \mathbf{t})$ is given by an infinite sequence $\sigma = \sigma_0, \sigma_1, \sigma_2, \ldots$ of transitions of Δ with $\sigma_i = (q_i, a_i, \phi_i, M_i, q_{i+1})$. The sequence σ defines sequences $\nu_0, \nu_1, \nu_2, \ldots$ and $\nu'_0, \nu'_1, \nu'_2, \ldots$ of functions $C \to \mathbf{R}$ by the following rules: $\nu_0(c) = 0$ for every $c \in C$, $\nu'_i(c) = \nu_i(c) + (t_{i+1} - t_i)$ for every $c \in C$, $\nu_{i+1}(c) = 0$ for every $c \in M_i$, and $\nu_{i+1}(c) = \nu'_i$ for $c \in C \setminus M_i$. The run is consistent if ν'_i satisfies ϕ_i for every i, and it is accepting if it is consistent and if there exist an infinite number of i with $q_i \in F$.

If $\mathfrak{A}$ is a timed automaton over A, then $\mathrm{L}(\mathfrak{A})$ denotes the timed language recognized by $\mathfrak{A}$, i.e., the set of all timed words over A for which there exists an accepting run of $\mathfrak{A}$. A $\mathcal{L}\mathrm{d}(P)$-sentence ϕ and a timed automaton $\mathfrak{A}$ over $\wp(P)$ are called *equivalent,* if $\mathfrak{A}$ recognizes the same language as defined by ϕ.

A set of timed words over A is called a regular timed language if it is recognized by a timed automaton.

Theorem 2. *A subset of* $\mathrm{TSS}(P)$ *is defined by a* $\overleftrightarrow{\mathcal{L}\mathrm{d}}(P)$*-sentence if and only if it is recognized by a timed automaton over* $\wp(P)$*, i.e., if it is a timed regular language over* $\wp(P)$*.*

In addition, there exist effective procedures that transform a $\overleftrightarrow{\mathcal{L}\mathrm{d}}(P)$*-sentence into an equivalent timed automaton over* $\wp(P)$ *and a timed automaton over* $\wp(P)$ *into an equivalent* $\overleftrightarrow{\mathcal{L}\mathrm{d}}(P)$*-sentence.*

In other words, timed automata and the monadic logic of relative distance have the same expressive power.

From [4] it is known that the emptiness problem for timed automata is decidable. This allows us to check whether a formula $\phi \triangleq \exists \bar{X} \psi(\bar{X}, \bar{Y}, \bar{x})$ of the monadic logic of relative distance is satisfiable in a timed state sequence: first take $\chi \triangleq \exists \bar{X} \exists \bar{Y} \exists \bar{x} \psi$, then build a timed automaton $\mathfrak{A}$ with $\mathrm{L}(\mathfrak{A}) = \mathrm{L}_P(\chi)$ (using the above theorem), and finally check emptiness for $\mathfrak{A}$ (using the result of [4]). If the answer to the test is "no", ϕ is satisfiable, if the answer is "yes", ϕ is not satisfiable.

Corollary 3. *For* $\overleftrightarrow{\mathcal{L}\mathrm{d}}(P)$ *the satisfiability problem is decidable.*

Recall that $\overleftrightarrow{\mathcal{L}\mathrm{d}}(P)$-formulas are only interpreted in structures associated with timed state sequences. Therefore the satisfiability problem for $\overleftrightarrow{\mathcal{L}\mathrm{d}}(P)$ is the problem of deciding whether for a given $\overleftrightarrow{\mathcal{L}\mathrm{d}}(P)$-formula ϕ there exists $\tau \in \mathrm{TSS}(P)$ and a variable assignment ν such that $(\tau, \nu) \models \phi$.

2 Monadic Logic and Bounded Variability

The variability of a time sequence measures the maximal number of positions in a unit interval. Formally, the *variability* of a time sequence $\mathbf{t}$ is given by the following expression:

$$\mathrm{var}(\mathbf{t}) = \sup\{k + 1 \mid \exists i (t_{i+k} - t_i < 1)\} .$$

We say that a timed state sequence is *of bounded variability* k if the variability of its time sequence is less than or equal to k. The set of all timed state sequences of bounded variability k is denoted by $\mathrm{TSS}_k(P)$, i.e.,

$$\mathrm{TSS}_k(P) = \{(\mathbf{s}, \mathbf{t}) \in \mathrm{TSS}(P) \mid \mathrm{var}(\mathbf{t}) \leq k\}.$$

It should be noted that the set $\mathrm{TSS}_k(P)$ is defined by the following $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formula:

$$\mathsf{bvar}_k \triangleq \exists X_0 \ldots \exists X_{k-1} \big(X_0 0 \wedge \forall X_1 1 \wedge \ldots \wedge X_{k-1}(k-1) \\ \wedge \bigwedge_{i<k} \forall x (X_i x \leftrightarrow X(x+k)) \wedge \bigwedge_{i<k} \forall x (X_i x \rightarrow \overrightarrow{\mathrm{d}}(x, X_i) \geq 1) \big) .$$

Here, 0, 1, 2, ... stand for the first, second, third, etc. position of a timed state sequence and $x + k$ stands for the k-th position after x. These terms can easily be eliminated using the formula

$$\mathrm{suc}(x, y) \triangleq x < y \wedge \neg \exists x' (x < x' \wedge x' < y) .$$

For instance, if $k = 3$, then $X_i(x + k)$ is equivalent to

$$\exists x_1 \exists x_2 \exists x_3 (\mathrm{suc}(x, x_1) \wedge \mathrm{suc}(x_1, x_2) \wedge \mathrm{suc}(x_2, x_3) \wedge X_i x_3 .$$

In Sect. 7 we will prove that $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$ is closed under complementation when interpreted in timed state sequences of bounded variability k:

Theorem 4. *Let k be a natural number.*
For every $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formula ϕ one can effectively construct a formula ϕ' such that

$$\mathrm{TSS}_k(P) \models \neg\phi \leftrightarrow \phi' .$$

As an immediate consequence of this result and Theorem 2 we have:

Corollary 5. *Let k be a natural number.*
The class of timed regular languages $\subseteq \mathrm{TSS}_k(P)$ is an effective boolean algebra (with complementation with respect to $\mathrm{TSS}_k(P)$).

As another consequence of Theorem 4, we will obtain the following:

Corollary 6. *The $\mathcal{L}\mathrm{d}(P)$-theory of $\mathrm{TSS}_k(P)$ is decidable.*

3 Temporal Logics

The logics we consider are interpreted in pairs (τ, i), so-called *interpretations,* consisting of a timed state sequence $\tau = (\mathbf{s}, \mathbf{t})$ and a natural number. To explain the semantic of one of the logics we define whether a formula is true in an interpretation or not; we write $(\tau, i) \models \phi$ or $(\tau, i) \not\models \phi$. The *model set* of a formula ϕ is the set of all timed state sequences τ over the given set of propositions such that $(\tau, 0) \models \phi$. We say that a formula ϕ *defines* a set L of timed state sequences if L is the model set of ϕ.

3.1 The Logic TL_Γ by Manna and Pnueli

We start with a definition of TL_Γ as given in [20].

The syntax diagram of $\mathrm{TL}_\Gamma(P)$ is depicted in Fig. 1, where p stands for a proposition, $\sim$ for a relation symbol as usual, and c for a natural number. (We omit rules for grouping formulas with parentheses.)

$$\begin{aligned}\langle\text{formula}\rangle ::= \ & \mathtt{tt} \mid p \mid \neg\langle\text{formula}\rangle \mid \langle\text{formula}\rangle \wedge \langle\text{formula}\rangle \mid \\ & \bigcirc\langle\text{formula}\rangle \mid \ominus\langle\text{formula}\rangle \mid \\ & \langle\text{formula}\rangle \,\mathcal{U}\, \langle\text{formula}\rangle \mid \langle\text{formula}\rangle \,\mathcal{S}\, \langle\text{formula}\rangle \mid \Gamma(\langle\text{formula}\rangle) \sim c\end{aligned}$$

Fig. 1. Syntax of $\mathrm{TL}_\star(P)$

The semantic of $\mathrm{TL}_\Gamma(P)$ is given by the following rules:

$(\tau, i) \models \mathtt{tt}$	for every i,
$(\tau, i) \models p$	iff $p \in s_i$,
$(\tau, i) \models \neg\phi$	iff $(\tau, i) \not\models \phi$,
$(\tau, i) \models \phi \wedge \psi$	iff $(\tau, i) \models \phi$ and $(\tau, i) \models \psi$,
$(\tau, i) \models \bigcirc\phi$	iff $(\tau, i+1) \models \phi$,
$(\tau, i) \models \phi \,\mathcal{U}\, \psi$	iff there exists $j \geq i$ such that $(\tau, i') \models \phi$ for every i' with $i \leq i' < j$ and $(\tau, j) \models \phi$,
$(\tau, i) \models \ominus\phi$	iff $i > 0$ and $(\tau, i-1) \models \phi$,
$(\tau, i) \models \phi \,\mathcal{S}\, \psi$	iff there exists $j \leq i$ such that $(\tau, j') \models \phi$ for every j' with $j < j' \leq i$ and $(\tau, j) \models \psi$,
$(\tau, i) \models \Gamma(\phi) \sim c$	iff $(\tau, i) \not\models \phi$ and $0 \sim c$, or $(\tau, i) \models \phi$ and $t_i - t_j \sim c$ for the smallest j with $(\tau, j') \models \phi$ for every j' with $j \leq j' \leq i$.

For instance, the set of all timed state sequences $\tau = (\mathbf{s}, \mathbf{t})$ such that $t_2 = 1$ is defined by

$$\bigcirc\bigcirc\,(\Gamma(\mathtt{tt}) = 1)\ .$$

This formula is interesting because there is no equivalent $\mathrm{MITL}_{\mathrm{p}}(P)$-formula for it.

The semantic of $\mathrm{TL}_\Gamma(P)$ introduced above leads us to read $\bigcirc$ as "nexttime", $\ominus$ as "previoustime", $\mathcal{U}$ as "until", $\mathcal{S}$ as "since", and Γ as "the age of ... is".

In Section 5 we will give a proof sketch of the following result:

Theorem 7. *Every formula of* $\mathrm{TL}_\Gamma(P)$ *can be effectively transformed into an equivalent* $\mathcal{L}^{\leftrightarrow}_{\mathrm{d}}(P)$*-formula.*

Since the satisfiability problem for $\mathcal{L}^{\leftrightarrow}_{\mathrm{d}}(P)$ is decidable (Corollary 3) and timed automata can be effectively converted into equivalent $\mathcal{L}\mathrm{d}(P)$-formulas (Theorem 2) we obtain as a consequence:

Corollary 8. *1. For* $\mathrm{TL}_{\mathcal{I}}(P)$ *the satisfiability problem is decidable.*
2. The $\mathrm{TL}_{\mathcal{I}}(P)$*-theory of* $\mathrm{TSS}(P)$ *is decidable.*
3. The model-checking problem is decidable for timed automata over $\wp(P)$ *and* $\mathrm{TL}_{\mathcal{I}}(P)$*.*

3.2 The Metric Temporal Logics $\mathrm{MITL_p}$ and $\mathrm{EMITL_p}$

We start with the definition of $\mathrm{MITL_p}$ as given in [8].

A non-singular interval is an interval of the real line with bounds in $\mathbf{N}_\infty$ (the set of natural numbers augmented by ∞) and with infinitely many elements, e. g., $[1,\infty)$ and $(3,4]$ are such intervals but $[5,5]$ is not.

A syntax diagram of $\mathrm{MITL_p}(P)$ is given in Fig. 2, where I stands for a non-singular interval and p for a proposition.

$$\langle \text{formula} \rangle ::= \mathtt{tt} \mid p \mid \neg\langle \text{formula} \rangle \mid \langle \text{formula} \rangle \wedge \langle \text{formula} \rangle \mid \langle \text{formula} \rangle \, \mathcal{U}_I \, \langle \text{formula} \rangle \mid \langle \text{formula} \rangle \, \mathcal{S}_I \, \langle \text{formula} \rangle$$

Fig. 2. Syntax of $\mathrm{MITL_p}(P)$

The semantic of $\mathtt{tt}$, p, $\neg$, and $\wedge$ is the same as in $\mathrm{TL}_{\mathcal{I}}(P)$. To $\mathcal{U}$ and $\mathcal{S}$ the following rules apply:

$(\tau, i) \models \phi \, \mathcal{U}_I \, \psi$ iff there exists $j \geq i$ such that $t_j - t_i \in I$ and $(\tau, i') \models \phi$ for every i' with $i \leq i' < j$ and $(\tau, j) \models \phi$,

$(\tau, i) \models \phi \, \mathcal{S}_I \, \psi$ iff there exists $j \leq i$ such that $t_i - t_j \in I$ and $(\tau, j') \models \phi$ for every j' with $j < j' \leq i$ and $(\tau, j) \models \psi$.

In [10] it was shown that the $\mathrm{MITL_p}(P)$-theory of $\mathrm{TSS}(P)$ is decidable. This was proven by showing that every $\mathrm{MITL_p}(P)$-formula can effectively be transformed into an equivalent bounded two-way deterministic timed automaton (which were introduced for this purpose in that paper).

The restriction to non-singular intervals is essential. If it is dropped the theory becomes undecidable. This is already true for the version without the since operator [6].

The operators (also called 'modalities') $\mathcal{S}$ and $\mathcal{U}$ can be viewed as special instances of automata operators. We now extend the idea of automata operators to the real-time framework.

We use classical Rabin-Scott automata over alphabets of the form 2^n. (Recall that 2^n denotes the set of all functions from $\{0, \ldots, n-1\}$ to $\{0,1\}$.) We will write an element of 2^n as a column vector with n components of $\{0,1\}$.

The vocabulary of $\mathrm{EMITL_p}(P)$ contains for every Rabin-Scott automaton $\mathfrak{A}$ over the alphabet 2^n and for every non-singular interval I two special symbols of arity n, which are denoted by $\overleftarrow{\mathfrak{A}}_I$ and $\overrightarrow{\mathfrak{A}}_I$, respectively. The syntax diagram

of $\mathrm{EMITL_p}(P)$ is given in Fig. 3, where $\mathfrak{A}$ stands for a Rabin-Scott automaton and I, again, for a non-singular interval. The formulas in the second line are called *automata formulas*, the semantic of which is explained in the following paragraphs.

$$\langle\text{formula}\rangle ::= \mathtt{tt} \mid p \mid \neg\langle\text{formula}\rangle \mid \langle\text{formula}\rangle \wedge \langle\text{formula}\rangle \mid \overleftarrow{\mathfrak{A}}_I(\langle\text{formula}\rangle, \ldots, \langle\text{formula}\rangle) \mid \overrightarrow{\mathfrak{A}}_I(\langle\text{formula}\rangle, \ldots, \langle\text{formula}\rangle)$$

Fig. 3. Syntax of $\mathrm{EMITL_p}(P)$

Let $\phi_0, \ldots, \phi_{n-1}$ be a sequence of temporal formulas and let $\tau = (\mathbf{s}, \mathbf{t})$ be a timed state sequence. With both (the sequence of formulas and the timed state sequence) we associate an infinite word $a_0 a_1 a_2 \ldots$ over the alphabet 2^n defined by

$$a_i(j) = \begin{cases} 1\,, & \text{if } (\tau, i) \models \phi_j\,, \\ 0\,, & \text{if } (\tau, i) \not\models \phi_j\,. \end{cases}$$

The word is denoted by $[\phi_0, \ldots, \phi_{n-1}, \tau]$. It stores the truth values of the formulas $\phi_0, \ldots, \phi_{n-1}$.

If $u = a_0 a_1 a_2 \ldots$ is an infinite word and if i and j are natural numbers with $i \leq j$ then $u(i,j)$ denotes the segment $a_i \ldots a_j$ of u. If $i \geq j$ then $u(i,j)$ denotes $u(j,i)$ in reversed order, i. e., $u(i,j) = a_i \ldots a_j$.

Using these notations we define:

$(\tau, i) \models \overleftarrow{\mathfrak{A}}_I(\phi_0, \ldots, \phi_{n-1})$ iff there exists $j \leq i$ such that $[\phi_0, \ldots, \phi_{n-1}, \tau](i,j)$ is accepted by the automaton $\mathfrak{A}$ and $t_i - t_j \in I$ holds,

$(\tau, i) \models \overrightarrow{\mathfrak{A}}_I(\phi_0, \ldots, \phi_{n-1})$ iff there exists $j \geq i$ such that $[\phi_0, \ldots, \phi_{n-1}, \tau](i,j)$ is accepted by the automaton $\mathfrak{A}$ and $t_j - t_i \in I$ holds.

For instance, if $\mathfrak{A}$ is a Rabin-Scott automaton over 2^2 that accepts the language $\left(\binom{1}{0} + \binom{1}{1}\right)^* \left(\binom{0}{1} + \binom{1}{1}\right)$, we have:

$$\mathrm{TSS}(P) \models \overrightarrow{\mathfrak{A}}_I(\phi, \psi) \leftrightarrow \phi\, \mathcal{U}_I\, \psi\,,$$
$$\mathrm{TSS}(P) \models \overleftarrow{\mathfrak{A}}_I(\phi, \psi) \leftrightarrow \phi\, \mathcal{S}_I\, \psi\,.$$

So, in fact, $\mathcal{S}$ and $\mathcal{U}$ can be expressed in terms of automata operators.

Using an automaton for the language 1^*0 (and automata for simulating $\bigcirc$ and $\ominus$) one can show that every $\mathrm{TL}_I(P)$-formula can effectively be translated into an equivalent $\mathrm{EMITL_p}(P)$-formula. This is even true for $\mathrm{TL}_I(P)$-formulas if extended by automata operators in the natural way.

In Sect. 6 we will give a proof sketch of the following result:

Theorem 9. *Every* $\mathrm{EMITL}_{\mathrm{p}}(P)$*-formula can be effectively transformed into an equivalent* $\overleftrightarrow{\mathcal{L}\mathrm{d}}(P)$*-formula.*

And as in the case of $\mathrm{TL}_{\mathcal{I}}(P)$ we obtain:

Corollary 10. *1. For* $\mathrm{EMITL}_{\mathrm{p}}(P)$ *the satisfiability problem is decidable.*
2. The $\mathrm{EMITL}_{\mathrm{p}}(P)$*-theory of* $\mathrm{TSS}(P)$ *is decidable.*
3. The model-checking problem is decidable for timed automata over $\wp(P)$ *and* $\mathrm{EMITL}_{\mathrm{p}}(P)$*.*

Since $\mathrm{EMITL}_{\mathrm{p}}(P)$ allows "counting" it is clear that $\mathrm{EMITL}_{\mathrm{p}}(P)$ is strictly more powerful than $\mathrm{MITL}_{\mathrm{p}}(P)$.

The property "every stimulus p is followed by a response q and, then, by another response r within 5 time units of the stimulus p" has been considered in several papers and is assumed to be not expressible in $\mathrm{MITL}_{\mathrm{p}}(\{p,q,r\})$, see [7] and [9]. Using an automaton $\mathfrak{A}$ for the language

$$2^{2^*}\left(\binom{1}{0}+\binom{1}{1}\right)2^{2^*}\left(\binom{0}{1}+\binom{1}{1}\right)$$

the property is expressed by

$$\neg\left(\mathtt{tt}\,\mathcal{U}_{[0,\infty)}\ \left(p\wedge\neg\overrightarrow{\mathfrak{A}}_{[0,5]}(q,r)\right)\right).$$

4 Expressive Completeness of $\mathcal{L}^{\overleftrightarrow{\mathrm{d}}}$

This section is dedicated to a sketch of the proof of Theorem 2. We first show how a given $\overleftrightarrow{\mathcal{L}\mathrm{d}}(P)$-sentence can be transformed into an equivalent timed automaton. The transformation is based on an adaptation of the model theoretic notion of interpretation to existential monadic second-order logic; timed state sequences are interpreted in ordinary infinite words. This allows us to use Büchi's theorem on the equivalence between S1S and finite automata for infinite words.

We start with some simple properties of $\overleftrightarrow{\mathcal{L}\mathrm{d}}(P)$. The set of $\overleftrightarrow{\mathcal{L}\mathrm{d}}(P)$-formulas is closed under existential quantification, conjunction and disjunction. (To obtain a formula of $\mathcal{L}\mathrm{d}(P)$, rearrangement of the prefixes of the existential set quantifiers and renaming of the quantified set variables is necessary in general.)

4.1 Eliminating Future Formulas

Transitions in timed automata are constraint to tests of the current values of the clocks. In $\overleftrightarrow{\mathcal{L}\mathrm{d}}(P)$ these are represented by past distance formulas $\overleftarrow{\mathrm{d}}(X,x)\sim c$, where one should think of X as the set of positions where a clock is reset. The dual distance formulas $\overrightarrow{\mathrm{d}}(x,X)\sim c$ are not easily representable in timed automata (but will be very useful later). We therefore show that they can be avoided.

The set of $\overleftrightarrow{\mathcal{L}\mathrm{d}}(P)$-formulas without future distance formulas as subformulas is denoted by $\overleftarrow{\mathcal{L}\mathrm{d}}(P)$.

Lemma 11. *Every $\mathcal{L}^{\overleftrightarrow{\mathrm{d}}}(P)$-formula can be effectively transformed into an equivalent $\mathcal{L}^{\overleftarrow{\mathrm{d}}}(P)$-formula.*

Sketch of proof. Given an $\mathcal{L}^{\overleftrightarrow{\mathrm{d}}}(P)$-formula the number of future subformulas is reduced stepwise. We outline one of these steps.

Suppose that $\phi \triangleq \exists X_0 \ldots \exists X_{m-1}\psi$ is an $\mathcal{L}^{\overleftrightarrow{\mathrm{d}}}(P)$-formula with a subformula of the form $\overrightarrow{\mathrm{d}}(v,X) \sim l$. We translate ϕ into an equivalent formula ϕ' where $\overrightarrow{\mathrm{d}}(v,X) \sim l$ does not occur any more. The formula ϕ' is designed as follows:

$$\phi' \triangleq \exists \bar{X} \exists \bar{E} \exists \bar{G} \exists \bar{G}' \exists K\,(\psi' \wedge \mathsf{succ}_l \wedge \mathsf{after} \wedge \mathsf{fut\text{-}enc}_l)$$

with l-tuples $\bar{E}$, $\bar{G}$, and $\bar{G}'$ of distinct set variables,

$$\mathsf{succ}_l \triangleq \bigwedge_{c \le l} \forall x (G_c x \leftrightarrow G'_c(x{+}1))$$

and

$$\mathsf{after} \triangleq \forall x (K x \leftrightarrow (\neg \exists y (x \le y \wedge X y))\,.$$

Thus succ_l makes each of the variables G'_c to contain the successors of the elements of G_c, and after forces K to contain all positions that are *after* every position of X. Note that by definition these positions can never make true a future formula with X as argument.

The formula $\mathsf{fut\text{-}enc}_l$ will have a unique solution in every timed state sequence. Its defining property is the following:

$$\mathsf{fut\text{-}enc}_l \leftrightarrow \bigwedge_{c \le l} \forall v ((E_c v \leftrightarrow \overleftarrow{\mathrm{d}}(v,X) = c) \wedge (G_c v \leftrightarrow \overleftarrow{\mathrm{d}}(v,X) > c))\,.$$

In other words, $\mathsf{fut\text{-}enc}_l$ makes the variables E_c and G_c to contain the set of positions for which $\overrightarrow{\mathrm{d}}(v,X) = c$ resp. $\overrightarrow{\mathrm{d}}(v,X) > c$ holds. If $\mathsf{fut\text{-}enc}_l$ can be constructed without future formulas, then ϕ' will have the desired property (namely, to be equivalent to ϕ and to have one future subformula less than ϕ) if we use for ψ' the formula that is obtained from ψ by replacing the future subformula $\overrightarrow{\mathrm{d}}(v,X) \sim l$ by α according to the following table:

$\sim$	$=$	$>$	$<$	$\ge$	$\le$	$\neq$
α	$E_l v$	$G_l v$	$\neg E_l v \wedge \neg G_l v \wedge \neg K v$	$E_l v \vee G_l v$	$\neg G_l v \wedge \neg K v$	$\neg E_l v \wedge \neg K v$

A construction of $\mathsf{fut\text{-}enc}_l$ can be carried out using induction on l. The induction base is straightforward. For the induction step we assume that $\mathsf{fut\text{-}enc}_{l-1}$ has been constructed and we construct $\mathsf{fut\text{-}enc}_l$.

Since we can add $\mathsf{fut\text{-}enc}_{l-1}$ as a conjunct, it is sufficient to deal only with the set variables E_l and G_l. For every position x of X we look at the interval extending from x into the past till the previous position in X or the first position of the timed state sequence. There are three different cases to distinguish:

1. the length of the interval is at least l and there is a position in the interval which has distance exactly l from x,
2. the length of the interval is greater than l and there is no position in the interval which has distance exactly l from x,
3. the length of the interval is less than l.

Each of these cases is represented by a disjunction in the formula $\mathsf{fut\text{-}enc}_l$:

$$\begin{aligned}\mathsf{fut\text{-}enc}_l \triangleq\ & \mathsf{fut\text{-}enc}_{l-1} \wedge \forall x(Kx \to \neg E_l x \wedge \neg G_l x) \\ & \wedge \forall x(Xx \to (\text{“case 1”} \vee \text{“case 2”} \vee \text{“case 3”}))\,.\end{aligned}$$

For instance, in case 2 one searches for the greatest position y with distance $> l$ from x and verifies that no position of the interval is contained in E_l, that all positions before y and y itself belong to G_l, and that no position after y belongs to G_l. In order to find y one uses the primed variable G'_l. The full formula for the second case is as follows:

$$\begin{aligned}\text{“case 2”} \triangleq\ & \exists y\big(\mathrm{bf}(y,x,X) \wedge G_l y \wedge \overleftarrow{\mathrm{d}}(G_l,x) > l \\ & \wedge (y{+}1 = x \vee \neg\overleftarrow{\mathrm{d}}(G'_l,x) \le l) \\ & \wedge \neg\exists y'(\mathrm{bf}(y',x,X) \wedge (E_l y' \vee (G_l y' \wedge y < y'))) \\ & \wedge \forall y'(\mathrm{bf}(y',x,X) \wedge y' < y \to G_l y')\big)\end{aligned}$$

where

$$\mathrm{bf}(x,y,X) \triangleq x < y \wedge \neg\exists x'(x < x' < y \wedge Xx')\,.$$

The other cases are easier and do not need the primed variables. □

As a consequence of the lemma we have:

Corollary 12. *$\mathcal{L}^{\overleftrightarrow{\mathrm{d}}}(P)$ and $\mathcal{L}^{\overleftarrow{\mathrm{d}}}(P)$ are equally expressive.*

4.2 Interpretation in Infinite Words

Let v be a fixed first-order variable.

Let $\mathcal{T}_0$ be a finite set of past formulas of the form $\overleftarrow{\mathrm{d}}(X,v) \sim c$. Let $\mathcal{V}$ be the set of set variables occurring in $\mathcal{T}_0$ and define $\mathcal{T}$ by

$$\mathcal{T} = \{Xv \mid X \in \mathcal{V}\} \cup \{\mathrm{Q}_p v \mid p \in P\} \cup \mathcal{T}_0\,.$$

A $\mathcal{T}$-formula is a $\mathcal{L}^{\overleftarrow{\mathrm{d}}}(P)$-formula where variables of $\mathcal{V}$ are quantified only in the existential prefix and where every relative distance formula that is a subformula belongs to $\mathcal{T}_0$.

Obviously every $\mathcal{L}^{\overleftarrow{\mathrm{d}}}(P)$-formula is equivalent to a $\mathcal{T}$-formula for some $\mathcal{T}$.

With every $\mathcal{T}$-formula ϕ we associate a monadic second-order formula ϕ^* in the signature $\{<\} \cup \{\mathrm{Q}_\alpha \mid \alpha \in \mathcal{T}\}$. It is obtained from ϕ by substituting Q_α for every subformula α belonging to $\mathcal{T}$.

Now let ν be a variable assignment and τ a timed state sequence. The *encoding of τ and ν with respect to $\mathcal{T}$* is the infinite word $u = a_0 a_1 a_2 \ldots$ over the alphabet $\wp(\mathcal{T})$ defined as follows:

$$a_i = \{\alpha \in \mathcal{T} \mid (\tau, \nu\left[\tfrac{i}{v}\right]) \models \alpha\}$$

where $\nu\left[\frac{i}{v}\right]$ is the assignment that coincides with ν at all places except for v where it is m. The word u is denoted by $\mathrm{code}(\tau, \nu, \mathcal{T})$.

By induction on the structure of $\mathcal{T}$-formulas one shows the following:

Lemma 13. *Let ϕ be a $\mathcal{T}$-formula, $\tau \in \mathrm{TSS}(P)$, and ν a variable assignment. Then*

$$(\tau, \nu) \models \phi \quad \textit{iff} \quad (\mathrm{code}(\tau, \nu, \mathcal{T}), \nu) \models \phi^* \,. \tag{3}$$

4.3 From a Discrete to a Timed Automaton

By Büchi's theorem we know that for every monadic second-order formula in the signature $\{\mathrm{Q}_\alpha \mid \alpha \in \mathcal{T}\} \cup \{<\}$ interpreted in infinite words we can construct an equivalent Büchi automaton. In particular, we find an automaton $\mathfrak{A}$ that is equivalent to the formula ϕ^* constructed in the previous subsection. Our aim is to transform such an automaton into a timed automaton recognizing the set of timed state sequences defined by ϕ.

Let $\mathfrak{A} = (Q, q_0, \Delta, F)$ be an arbitrary Büchi automaton over $\wp(\mathcal{T})$. We construct a timed Büchi automaton $\hat{\mathfrak{A}} = (Q, C, q_0, \Delta', F)$ over $\wp(P)$.

First, the set C of clocks contains for every set variable $X \in \mathcal{V}$ a clock denoted by c_X. Second, for every transition (q, a, q') in Δ a transition (q, s, ϕ, M, q') with

$$\begin{aligned} s &= \{p \mid \mathrm{Q}_p v \in a\} \,, \\ \phi &\triangleq \bigwedge_{\overleftarrow{\mathrm{d}}(X,v) \sim c \in \mathcal{T}} c_X \sim c \,, \\ M &= \{c_X \mid Xv \in a\} \end{aligned}$$

is added to Δ', which is initialized by the empty set at the beginning.

By induction on the length of a computation one can prove:

Lemma 14. *Let $\mathfrak{A}$ be an arbitrary Büchi automaton over $\mathcal{T}$ and $\hat{\mathfrak{A}}$ as above. Let $\tau \in \mathrm{TSS}(P)$ and ν a variable assignment. Then*

$$\hat{\mathfrak{A}} \textit{ accepts } \tau \quad \textit{iff} \quad \mathfrak{A} \textit{ accepts } \mathrm{code}(\tau, \nu, \mathcal{T}) \,. \tag{4}$$

Now it is clear how the direction from left to right in Theorem 2 can be proved: given a $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-sentence ψ, this is first transformed into a $\mathcal{L}\overleftarrow{\mathrm{d}}(P)$-sentence ϕ using Lemma 11. Then ϕ^* is constructed as described in Subsect. 4.2 using an appropriate set $\mathcal{T}_0$ of past formulas. Next a Büchi automaton equivalent to ϕ^* is built using Büchi's theorem [13]. Finally, $\hat{\mathfrak{A}}$ is constructed according to the rules above. For every timed state sequence τ we then have:

$\tau \models \psi$

iff $\tau \models \phi$ (by construction, Lemma 11)

iff $\exists u \exists \nu (u = \mathrm{code}(\tau, \nu, \mathcal{T}) \wedge u \models \phi^*)$, (by Lemma 13)

iff $\exists u \exists \nu (u = \mathrm{code}(\tau, \nu, \mathcal{T}) \wedge u \in \mathrm{L}(\mathfrak{A}))$, (by Büchi's theorem)

iff $u \in \mathrm{L}(\hat{\mathfrak{A}})$, (by Lemma 14)

Therefore $\hat{\mathfrak{A}}$ is the automaton we are looking for: it is equivalent to ψ. □

In the author's dissertation [24] also for some classes of linear hybrid automata expressively complete monadic second-order logics are obtained along the same lines.

4.4 Describing the Run of a Timed Automaton

The other direction of Theorem 2 is much more simpler and classical in the sense that it follows the same idea which has also been used in [12], [13], or [22] in order to show that sets recognizable by automata are definable by existential monadic second-order sentences: one describes the existence of an accepting run.

If $\mathfrak{A}$ is a timed automaton with s transitions and t clocks, a formula is constructed in the form

$$\mathsf{accepted} \triangleq \exists X_0 \ldots \exists X_{s-1} \exists Y_0 \ldots \exists Y_{t-1} \mathsf{successful} .$$

Here X_i encodes the positions where the ith transition of $\mathfrak{A}$ is taken and Y_j contains the positions where the jth clock is reset. Using formulas of the form $\overleftarrow{\mathrm{d}}(Y_j, v) \sim c$ one can check whether a clock constraint is satisfied or not.

Since $\mathsf{successful}$ can be constructed without quantification of set variables, it follows:

Corollary 15. *The existential monadic second-order fragment of* $\mathcal{L}\overleftarrow{\mathrm{d}}(P)$ *and* $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$ *are equally expressive.*

5 Embedding $\mathrm{TL}_{\boldsymbol{\Gamma}}$

The proof of Theorem 7 uses the fact that $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$ can be extended by so-called set terms.

In general a *set term* is of the form

$$\{x \mid \alpha\} \tag{5}$$

where x is a first-order variable and α is an arbitrary formula. The free variables of (5) are the free variables of α except x.

Given an interpretation $\mathfrak{I} = (\mathfrak{M}, \beta)$, the interpretation of (5) in $\mathfrak{I}$ is the set

$$\{m \in M \mid (\mathfrak{M}, \beta[\tfrac{m}{x}]) \models \alpha\}$$

where M is the universe of $\mathfrak{M}$.

Set terms can be used everywhere in a formula where a set variable may appear.

One can show that the expressive power of monadic second-order logics is not increased when set terms are introduced. In the case of $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$ the use of set terms has to be restricted in order to ensure equal expressive power.

Lemma 16. *Every $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formula ϕ with set terms satisfying*

(#) the set terms have no free first-order variable and all free set variables occurring in set terms of ϕ are either free in the entire formula or bound in the existential prefix

can effectively be transformed into an equivalent $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formula.

Proof. If $\phi \triangleq \exists X_0 \ldots \exists X_{m-1}\psi$ is a formula with (5) as subterm and if (#) is satisfied, then ϕ is equivalent to

$$\exists X_0 \ldots \exists X_{m-1} \exists Y((\forall x(Yx \leftrightarrow \alpha) \wedge \psi')$$

where ψ' is obtained from ψ by substituting (5) by Y. Continued application of this substitution process finally leads to a $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formula without set terms. □

Therefore, if we speak of an $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$*-formula with set terms* we always require (#).

In order to prove Theorem 7 we show that for every $\mathrm{TL}_{\mathcal{I}}(P)$-formula ϕ there exists a $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formula $\phi^* \triangleq \phi^*(x)$ with set terms such that for every timed state sequence and for every natural number i the following holds:

$$(\tau, i) \models \phi \text{ iff } \tau \models \phi^*(i)\,.$$

Then ϕ will be equivalent to $\phi^*(0)$, and this can be transformed into a $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formula by Lemma 16.

The proof of the assertion proceeds by induction on ϕ. Apart from the induction step for the age operator it is straightforward. In this case it is convenient to have set terms, because one can use the formula

$$\overleftarrow{\mathrm{d}}(\{y \mid \neg\psi^*(y-1) \wedge y > 0\}, x) \sim c$$

in order to check the age of a formula ψ at a position x. □

6 Embedding $\mathrm{EMITL_p}$

In this section we outline the proof of Theorem 9.

The proof proceeds by induction on the structure of the $\mathrm{EMITL_p}(P)$-formulas using the following assertion:

For every $\mathrm{EMITL_p}(P)$-formula ϕ there exists a $\overleftrightarrow{\mathcal{LTL}}(P)$-formula $\phi^* \triangleq \phi^*(X)$ with set terms such that in every timed state sequence τ over P the formula $\phi^*(X)$ has a unique solution M and for this solution

$$(\tau, i) \models \phi \text{ iff } i \in M$$

holds for every i.

So we claim that for every $\mathrm{EMITL_p}(P)$-formula ϕ we can construct a formula defining the set of positions where ϕ is true.

The induction base and the induction step for conjunction and negation are straightforward, while the induction step for automata formulas is involved.

We sketch how to construct ϕ^* when

$$\phi \triangleq \overrightarrow{\mathfrak{A}}_I(\phi_0, \ldots, \phi_{n-1}).$$

The construction for the corresponding past formula is similar.

First notice that every non-singular interval is a union of at most two non-singular intervals with lower and upper bound a and b such that either

- $a = 0$ and $b = 1$,
- $a = 1$ and $b \geq 2$ is finite, or
- $a = 1$ and $b = \infty$.

So it is sufficient to consider only these types of intervals as subscripts for automata operators. We deal only with the second case, the others can be treated similarly.

With every timed state sequence $\tau = (\mathbf{s}, \mathbf{t})$ we associate a subset of the set of its positions, called its *grid* and denoted by $\mathrm{grid}(\tau)$. It is the smallest subset of $\mathrm{pos}(\tau)$ containing 0 and satisfying the following two conditions:

1. If $t_i \in \mathrm{grid}(\tau)$ and if $t_{i+1} - t_i \geq 1$, then $t_{i+1} \in \mathrm{grid}(\tau)$.
2. If $t_i \in \mathrm{grid}(\tau)$ and if $t_{i+1} - t_i < 1$, then $\max\{t_j \mid t_j - t_i < 1\} \in \mathrm{grid}(\tau)$. (Since $\mathbf{t}$ is assumed to be divergent the maximum exists.)

Obviously, $\mathrm{grid}(\tau)$ is an infinite set.

In the following we assume that $r_0, r_1, r_2, \ldots$ with $r_0 < r_1 < r_2 < \ldots$ is an enumeration of the elements of the grid of a fixed timed state sequence $\tau = (\mathbf{s}, \mathbf{t})$.

Remark. Let τ and $r_0, r_1, \ldots$ as defined above.

1. If k is a positive natural number, then $r_{i+2k} - r_i > k$ for every i.
2. Let J be a non-singular interval, let $M \subseteq \mathrm{pos}(\tau)$, and $t \in \mathrm{pos}(\tau)$. Define sets M_f and M_l as follows:

$$M_f = \{\min(N) \mid \exists i(N = M \cap [r_i, r_{i+1}] \wedge N \neq \emptyset)\},$$
$$M_l = \{\max(N) \mid \exists i(N = M \cap [r_i, r_{i+1}] \wedge N \neq \emptyset)\}.$$

Then the following conditions are equivalent:

(a) There exists a position $t' \in M$ such that $t' - t \in J$.

(b) There exists a position in $t' \in M_f \cup M_l$ such that $t' - t \in J$.

The first claim asserts that the grid is "coarse". On the contrary, the second claim asserts that the grid is "fine": for every position t_i of τ there exists a position in M with distance from t_i restricted by a non singular interval only if such a position exists in one of the sets M_f or M_l which are at most as "dense" as the grid. (Note that the claim is not true for singular intervals!)

In the following we explain how Remark 6 can be used for our purposes.

Let us assume that the automaton $\mathfrak{A}$ is given as $\mathfrak{A} = (Q, q_0, \delta, F)$. Then for every $q \in Q$ we define automata $\mathfrak{A}_q$ and $\mathfrak{A}^q$:

$$\mathfrak{A}_q = (Q, q, \delta, F) ,$$
$$\mathfrak{A}^q = (Q, q_0, \delta, \{q\}) .$$

Let $u = a_0 a_1 a_2 \ldots$ be the word $[\phi_0, \ldots, \phi_{n-1}, \tau]$ (as defined in Subsect. 3.2). For every $q \in Q$ we define sets M_f^q and M_l^q. For every i let j be the index such that $r_i = t_j$ and let

$$N_i = \{t_l \mid t_l \in [r_i, r_{i+1}] \wedge u(j,l) \in \mathrm{L}(\mathfrak{A}_q)\} .$$

Then let

$$M_f^q = \{\min(N_i) \mid N_i \neq \emptyset\} ,$$
$$M_l^q = \{\max(N_i) \mid N_i \neq \emptyset\} .$$

Now let t_i be an arbitrary position of τ. Then, using Remark 6, one can prove that the following conditions 1 and 2 are equivalent:

1. $(\tau, i) \models \phi$.
2. There exists a natural number l with $l \leq 2b$ (recall that b is the upper bound of I), a state $q \in Q$, and a position $t_j \in M_f^q \cup M_l^q$ such that the following holds for $k = \max\{k' \mid t_{k'} \in \mathrm{grid}(\tau) \wedge t_{k'} \leq t_j\}$:
   (a) $i \leq k$,
   (b) $u(i,k) \in \mathrm{L}(\mathfrak{A}^q)$,
   (c) $|\mathrm{grid}(\tau) \cap [t_i, t_k]| = l$,
   (d) $t_j - t_i \in I$.

Thus ϕ^* needs only express condition 2. That a suitable formula can be constructed is explained in the rest of the subsection.

We need further notation. Write $\mathrm{suc}(x, y, X)$ for

$$x < y \wedge Xy \wedge \neg\exists x'(x < x' < y \wedge Xx').$$

and define inductively $\mathrm{suc}^k(x, y, X)$ by

$$\mathrm{suc}^1(x, y, X) \triangleq \mathrm{suc}(x, y, X) ,$$
$$\mathrm{suc}^{k+1}(x, y, X) \triangleq \exists x'(\mathrm{suc}(x, x', X) \wedge \mathrm{suc}^k(x', y, X)) .$$

So $\mathrm{suc}^k(x,y,X)$ is true if y is the k-th position after x belonging to X. Now $\overrightarrow{\mathrm{d}_k}(x,X) \sim c$ is defined as follows:

$$\overrightarrow{\mathrm{d}_k}(x,X) \sim c \triangleq \exists y(\mathrm{suc}^k(x,y,X) \wedge \mathrm{d}(x,y) \sim c)\,.$$

So $\overrightarrow{\mathrm{d}_k}(x,X) \sim c$ is true if the distance from x to the k-th position after x belonging to X satisfies c. Symmetrically, $\mathrm{suc}^k(X,x,y)$ and $\overleftarrow{\mathrm{d}_k}(X,x) \sim c$ are defined. The formulas $\overrightarrow{\mathrm{d}_k}(x,X) \sim c$ and $\overleftarrow{\mathrm{d}_k}(X,x) \sim c$ are called *generalized relative distance formulas.*

Using a partition of X into k sets (counting modulo k) one can show that the expressive power of $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$ is not increased, if the new formulas are added as admissible atomic subformulas involving distance formulas.

In order to see that condition 2 is expressible, we first show that the grid of a timed state sequence is definable by a $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formula:

Lemma 17. *There is a $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formula* $\mathsf{grid}(G)$ *such that in every timed state sequence τ the set* $\mathrm{grid}(\tau)$ *is the unique solution of* $\mathsf{grid}(G)$.

Proof. Using generalized relative distance formulas we may simply write $\mathsf{grid}(G)$ in the following form:

$$\begin{aligned}\mathsf{grid} \triangleq\ & G0 \wedge \forall x \exists y(x < y \wedge Gy) \\ & \wedge \forall x\big(Gx \wedge x > 0 \rightarrow \overleftarrow{\mathrm{d}_2}(G, x{+}1) \geq 1\big) \\ & \wedge \forall x \forall y\big(Gx \wedge \mathrm{suc}(x,y,G) \rightarrow \big(\overleftarrow{\mathrm{d}}(G,y) < 1 \vee y = x+1\big)\big)\end{aligned}$$

□

From Büchi's theorem [12] we know that, for every $q \in Q$, one can construct a monadic second-order formula $\psi_q \triangleq \psi_q(x, y, X_0, \ldots, X_{n-1})$ in the signature $\{<\} \cup \{\mathrm{Q}_p \mid p \in P\}$ such that for every timed state sequence τ and every variable assignment ν with $\nu(x) \leq \nu(y)$ we have $(\tau,\nu) \models \psi_q$ if and only if $\mathfrak{A}_q$ accepts the word $a_0 \ldots a_{\nu(y)-\nu(x)}$ with a_i defined by

$$a_i(j) = \begin{cases} 1\,, & \text{if } \nu(x){+}i \in \nu(X_j), \\ 0\,, & \text{otherwise.}\end{cases}$$

Similarly, one can construct a formula $\psi^q(x, y, X_0, \ldots, X_{n-1})$ for the automaton $\mathfrak{A}^q$.

The formulas ψ^q are used to express condition 2(b), whereas the formulas ψ_q are used to define the sets M_f^q and M_l^q by $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formulas. For conditions 2(c) and 2(d) the generalized relative distance formulas $\overrightarrow{\mathrm{d}_k}(x,X) \sim c$ are used. (Note that t_j in 2(d) is a position of M_f^q or M_l^q; so relative distance formulas are appropriate to measure the distance.)

It is also possible to extend $\mathrm{EMITL}_{\mathrm{p}}(P)$ by future Büchi automata operators (but, of course, without interval constraints). From McNaughton's result on the determinization of Muller automata it is clear that such an extension will not add to the expressiveness of $\mathrm{EMITL}_{\mathrm{p}}(P)$.

7 Closure under Complementation for Bounded Variability

Let k be a positive natural number.

Let τ be a timed state sequence of bounded variability k and let c be a natural number. Then $t_{i+ck+1} - t_i > c$ holds for every i. Thus for every $\sim \in \{=, \neq, <, \leq\}$ the formula $\overleftarrow{\mathrm{d}}(X, y) \sim c$ is equivalent to

$$\exists x \Big(x < y \wedge Xx \wedge \neg \exists x' (x < x' < y \wedge Xx') \\ \wedge \bigvee_{i \leq kc} \big(y = x + i + 1 \wedge \overleftarrow{\mathrm{d}_{i+1}}(\{z \mid \mathtt{tt}\}, y) \sim c \big) \Big) . \tag{6}$$

In case of $\sim \in \{>, \geq\}$, the formula equivalent to $\overleftarrow{\mathrm{d}}(X, y) \sim c$ is a bit more complicated:

$$\exists x \Big(x < y \wedge Xx \wedge \neg \exists x' (x < x' < y \wedge Xx') \\ \wedge (x + kc < y \vee \bigvee_{i \leq kc} \big(y = x + i + 1 \wedge \overleftarrow{\mathrm{d}_{i+1}}(\{z \mid \mathtt{tt}\}, y) \sim c \big) \Big) . \tag{7}$$

After substitution of every past formula by (6) or (7) in a given $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formula, one obtains a $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formula with set terms where no quantified or free set variable occurs in a distance formula. Thus the negation of this formula is also a $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formula with set terms. By Lemma 11 we know that set terms can be eliminated. This proves Theorem 4. □

Every $\mathcal{L}\mathrm{d}(P)$-formula is equivalent (with respect to satisfiability) to a formula that is built from

$$\exists x (Xx \wedge \neg \exists y (\neg x = y \wedge Xy)) \tag{8}$$
$$\exists x (Xx \wedge \mathrm{Q}_p x) \tag{9}$$
$$\exists x (Xx \wedge \mathrm{Y} x) \tag{10}$$
$$\exists x \exists y (Xx \wedge Yy \wedge x < y) \tag{11}$$
$$\exists x \exists y (Xx \wedge Yy \wedge \mathrm{d}(x, y) \sim c) \tag{12}$$

using boolean connectives and quantification of set variables. (In order to see this one replaces first-order variables by set variables for singleton sets, cf. (8).) The distance formula in (12) is equivalent to

$$\exists X \left(Xy \wedge \forall y' (Xy' \rightarrow y = y') \wedge \overleftarrow{\mathrm{d}}(X, x) \sim c \right) . \tag{13}$$

Since $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$ is closed under existential quantification of set variables, conjunction and disjunction, since it is also closed under negation in the case of bounded variability k, and since (8) – (11) and (13) are $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formulas, every $\mathcal{L}\mathrm{d}(P)$-formula is equivalent (with respect to satisfiability) to a $\mathcal{L}\overleftrightarrow{\mathrm{d}}(P)$-formula. This implies Corollary 6. □

It is also possible to proof the results concerning bounded variability using automata theoretic methods (determinization of timed automata).

8 Alternative Models of Computation

The results of this paper do not depend on the model of real-time computation—here timed state sequences—one chooses; they transfer to other standard models: timed state sequences with non-decreasing time [21] (rather than strictly increasing time), trajectories as finitely variable functions from real-numbered time to states, and observation sequences [9].

Only minor modifications of the logical setting are necessary in these cases. Often there are several ways to adjust to the situation. We mention the following possibilities: For non-decreasing time the universe of the relational structures is replaced by the natural numbers as index set for the elements of a (non-decreasing) time sequence; a distance formula is then true for two natural numbers if the difference between the corresponding positions satisfies the given condition. Dealing with trajectories one allows only quantification over positions where state changes occur, and observation sequences can simply be treated as an alternative presentation of trajectories of finite variability.

Concluding Remark

It seems that $\mathcal{L}^{\leftrightarrow}_{\mathrm{d}}$ is a powerful decidable logic for the specification of timed state sequences; 'decidable' here means that the satisfiability problem is decidable. This view is supported by two main arguments:

- As we have seen $\mathcal{L}^{\leftrightarrow}_{\mathrm{d}}$ is strictly more expressive than $\mathrm{EMITL}_{\mathrm{p}}$, and $\mathrm{EMITL}_{\mathrm{p}}$ is strictly more expressive than $\mathrm{MITL}_{\mathrm{p}}$, and $\mathrm{MITL}_{\mathrm{p}}$ was the most expressive decidable logic for specifying timed state sequences previously known. Also $\mathrm{TL}_{\mathcal{F}}$ is subsumed by $\mathcal{L}^{\leftrightarrow}_{\mathrm{d}}$, even when extended by automata operators.
- From automata theory it is known that already small extensions of the model of timed automaton lead to classes of automata with an undecidable emptiness problem [3] [14] [24]. Since, as we have shown, $\mathcal{L}^{\leftrightarrow}_{\mathrm{d}}$ is expressively complete for timed automata extensions of $\mathcal{L}^{\leftrightarrow}_{\mathrm{d}}$ capturing larger reasonable classes of automata cannot have a decidable satisfiability problem.

References

1. R. Alur. "Techniques for Automatic Verification of Real-time Systems". PhD thesis, Stanford University, California (1991).
2. R. Alur, C. Courcoubetis, and D. Dill. Model-checking in dense real-time. *Inform. and Control* **104**(1), 1–34 (1993).
3. R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, "Hybrid Systems", vol. 736 of "Lecture Notes in Computer Science". Springer-Verlag, Berlin (1993).
4. R. Alur and D. Dill. A theory of timed automata. *Theoret. Comput. Sci.* **126**(2), 183–235 (1994).

5. R. Alur and D. L. Dill. Automata for modeling real-time systems. In M. S. Paterson, editor, "Automata, Languages and Programming: 17th International Colloquium", vol. 443 of "Lecture Notes in Computer Science", pp. 322–335, Warwick, England (1990). European Assoc. Theoret. Comput. Sci., Springer-Verlag.
6. R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. In "Proceedings of the 10th ACM Symposium on Principles of Distributed Computing", pp. 139–152, Montreal, Qu ebec, Canada (1991). ACM Press.
7. R. Alur and T. Henzinger. A really temporal logic. In "Proceedings of the 30th Annual Symposium on Foundations of Computer Science", pp. 164–169. IEEE Computer Society Press (1989).
8. R. Alur and T. Henzinger. Real-time logics: complexity and expressiveness. In "Fifth Annual IEEE Symposium on Logic in Computer Science", pp. 390–401, Philadelphia, Pennsylvania (1990). IEEE Computer Society Press.
9. R. Alur and T. A. Henzinger. Logics and models of real-time: a survey. In J. W. de Bakker, C. Huizing, W. de Roever, and G. Rozenberg, editors, "Real-Time: Theory in Practice", vol. 600 of "Lecture Notes in Computer Science", pp. 74–106, Mook, The Netherlands (1991). Springer-Verlag.
10. R. Alur and T. A. Henzinger. Back to the future: towards a theory of timed regular languages. In "33rd Annual Symposium on Foundations of Computer Science", pp. 177–186, Pittsburgh, Pennsylvania (1992). IEEE Computer Society Press.
11. R. Alur and T. A. Henzinger. Real-time logics: complexity and expressiveness. *Inform. and Control* **104**(1), 35–77 (1993).
12. J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* **6**, 66–92 (1960).
13. J. R. Büchi. On a decision method in restricted second-order arithmetic. In E. Nagel, P. Suppes, and A. Tarski, editors, "Logic, Methodology, and Philosophy of Science: Proc. of the 1960 International Congress", pp. 1–11. Stanford University Press (1962).
14. K. Čerāns. Decidability of bisimulation equivalences for parallel timer processes. In G. v. Bochmann and D. K. Probst, editors, "Computer Aided Verification, Fourth International Workshop, CAV '92", vol. 663 of "Lecture Notes in Computer Science", pp. 302–315, Montreal, Canada (1992). Springer-Verlag.
15. Z. Chaochen, C. Hoare, and A. Ravn. A calculus of durations. *Information Processing Letters* **40**(5), 269–276 (1991).
16. W. Ebinger and A. Muscholl. Logical definability on infinite traces. In A. Lingas, R. Karlsson, and S. Carlsson, editors, "Automata, Languages and Programming: 20th International Colloquium", vol. 700 of "Lecture Notes in Computer Science", pp. 335–346, Lund, Sweden (1993). European Assoc. Theoret. Comput. Sci., Springer-Verlag.
17. T. A. Henzinger, Z. Manna, and A. Pnueli. Timed transition systems. In J. W. de Bakker, C. Huizing, W. de Roever, and G. Rozenberg, editors, "Real-Time: Theory in Practice", vol. 600 of "Lecture Notes in Computer Science", pp. 226–251, Mook, The Netherlands (1991). Springer-Verlag.
18. R. Koymans, J. Vytopil, and W. P. de Roever. Real-time programming and asynchronous message passing. In "Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing", pp. 187–197. ACM Press (1983).
19. H. R. Lewis. A logic of concrete time intervals. In "Fifth Annual IEEE Symposium on Logic in Computer Science", pp. 380–389, Philadelphia, Pennsylvania (1990). IEEE Computer Society Press.

20. Z. Manna and A. Pnueli. Models for reactivity. *Acta Informatica* **30**(2), 609–678 (Oct. 1993).
21. A. Pnueli and Y. Kesten. Timed and hybrid statecharts and their textual representation. In J. Vytopil, editor, "Formal Techniques in Real-time and Fault-tolerant Systems: Second International Symposium, Nijmegen, The Netherlands", vol. 571 of "Lecture Notes in Computer Science", pp. 591–620, Nijmegen, The Netherlands (Jan. 1992). Springer-Verlag.
22. M. O. Rabin. Decidability of second-order theories and finite automata on infinite trees. *Trans. Amer. Math. Soc.* **141**, 1–35 (1969).
23. J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order arithmetic. *Math. Systems Theory* **2**(1), 57–81 (1968).
24. Th. Wilke. Automaten und Logiken zur Beschreibung zeitabhängiger Systeme. Technical Report 9407, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik (July 1994).
25. P. Wolper. Temporal logic can be more expressive. *Inform. and Control* **56**, 72–99 (1983).
26. P. Wolper, M. Y. Vardi, and A. P. Sistla. Reasoning about infinite computation paths. In "24th Annual Symposium on Foundations of Computer Science", pp. 185–194. IEEE Computer Society Press (1983).

This article was processed using the LaTeX macro package with LLNCS style