The Regular Real-Time Languages***

T.A. Henzinger[1], J.-F. Raskin[2], and P.-Y. Schobbens[2]

[1] Electrical Engineering and Computer Sciences, University of California, Berkeley
[2] Computer Science Institute, University of Namur, Belgium

Abstract. A specification formalism for reactive systems defines a class of ω-languages. We call a specification formalism *fully decidable* if it is constructively closed under boolean operations and has a decidable satisfiability (nonemptiness) problem. There are two important, robust classes of ω-languages that are definable by fully decidable formalisms. The *ω-regular languages* are definable by finite automata, or equivalently, by the Sequential Calculus. The *counter-free ω-regular languages* are definable by temporal logic, or equivalently, by the first-order fragment of the Sequential Calculus. The gap between both classes can be closed by finite counting (using automata connectives), or equivalently, by projection (existential second-order quantification over letters).

A specification formalism for real-time systems defines a class of *timed* ω-languages, whose letters have real-numbered time stamps. Two popular ways of specifying timing constraints rely on the use of clocks, and on the use of time bounds for temporal operators. However, temporal logics with clocks or time bounds have undecidable satisfiability problems, and finite automata with clocks (so-called *timed automata*) are not closed under complement. Therefore, two fully decidable restrictions of these formalisms have been proposed. In the first case, clocks are restricted to *event clocks*, which measure distances to immediately preceding or succeeding events only. In the second case, time bounds are restricted to *nonsingular intervals*, which cannot specify the exact punctuality of events. We show that the resulting classes of timed ω-languages are robust, and we explain their relationship.

First, we show that temporal logic with event clocks defines the same class of timed ω-languages as temporal logic with nonsingular time bounds, and we identify a first-order monadic theory that also defines this class. Second, we show that if the ability of finite counting is added to these formalisms, we obtain the class of timed ω-languages that are definable by finite automata with event clocks, or equivalently, by a restricted second-order extension of the monadic theory. Third, we show that if projection is added, we obtain the class of timed ω-languages that are definable by timed automata, or equivalently, by a richer second-order extension of the monadic theory. These results identify three robust classes of timed ω-languages, of which the third, while popular, is not definable by a fully decidable formalism. By contrast, the first two classes are definable by fully decidable formalisms from temporal logic, from automata theory, and from monadic logic. Since the gap between these two classes can be closed by finite counting, we dub them the *timed ω-regular languages* and the *timed counter-free ω-regular languages*, respectively.

---

* An abbreviated version of this paper appeared in the *Proceedings of the 25th International Colloquium on Automata, Languages, and Programming* (ICALP), Lecture Notes in Computer Science 1443, Springer-Verlag, 1998, pp. 580–591.

** This work is supported in part by the ONR YIP award N00014-95-1-0520, the NSF CAREER award CCR-9501708, the NSF grant CCR-9504469, the ARO MURI grant DAAH-04-96-1-0341, the SRC contract 97-DC-324.041, the Belgian National Fund for Scientific Research (FNRS), the European Commission under WGs Aspire and Fireworks, the Portuguese FCT, and by Belgacom.

1 Introduction

A run of a reactive system produces an infinite sequence of events. A property of a reactive system, then, is an ω-language containing the infinite event sequences that satisfy the property. There is a very pleasant expressive equivalence between modal logics, classical logics, and finite automata for defining ω-languages [Büc62,Kam68,GPSS80,Wol82]. Let TL stand for the propositional linear temporal logic with next and until operators, and let Q-TL and E-TL stand for the extensions of TL with propositional quantifiers and grammar (or automata) connectives, respectively. Let ML_1 and ML_2 stand for the first-order and second-order monadic theories of the natural numbers with successor and comparison (also called S1S or the Sequential Calculus). Let BA stand for Büchi automata. Then we obtain the following two levels of expressiveness:

	Languages	Temporal logics	Monadic theories	Finite automata
1	*counter-free ω-regular*	TL	ML_1	
2	*ω-regular*	Q-TL = E-TL	ML_2	BA

For example, the TL formula $\Box(p \rightarrow \Diamond q)$, which specifies that every p event is followed by a q event, is equivalent to the ML_1 formula $(\forall i)(p(i) \rightarrow (\exists j \geq i)q(j))$ and to a Büchi automaton with two states. The difference between the first and second levels of expressiveness is the ability of automata to count. A counting requirement, for example, may assert that all even events are p events, which can be specified by the Q-TL formula $(\exists q)(q \wedge \Box(q \leftrightarrow \bigcirc\neg q) \wedge \Box(q \rightarrow p))$.

We say that a formalism is *positively decidable* if it is constructively closed under positive boolean operations, and satisfiability (emptiness) is decidable. A formalism is *fully decidable* if it is positively decidable and also constructively closed under negation (complement). All of the formalisms in the above table are fully decidable. The temporal logics and Büchi automata are less succinct formalisms than the monadic theories, because only the former satisfiability problems are elementarily decidable.

A run of a real-time system produces an infinite sequence of time-stamped events. A property of a real-time system, then, is a set of infinite time-stamped event sequences. We call such sets *timed ω-languages*. If all time stamps are natural numbers, then there is again a very pleasant expressive equivalence between modal logics, classical logics, and finite automata [AH93]. Specifically, there are two natural ways of extending temporal logics with timing constraints. The Metric Temporal Logic MetricTL (also called MTL [AH93]) adds time bounds to temporal operators; for example, the MetricTL formula $\Box(p \rightarrow \Diamond_{=5} q)$ specifies that every p event is followed by a q event such that the difference between the two time stamps is exactly 5. The Clock Temporal Logic ClockTL (also called TPTL [AH94]) adds clock variables to TL; for example, the time-bounded response requirement from above can be specified by the ClockTL formula $\Box(p \rightarrow (x := 0)\Diamond(q \wedge x = 5))$, where x is a variable representing a clock that is started by the quantifier $(x := 0)$. Interestingly, over natural-numbered time, both ways of expressing timing constraints are equally expressive. Moreover, by adding the ability to count, we obtain again a canonical second level of expressiveness. Let TimeFunctionML stand for the monadic theory of the natural numbers extended with a unary function symbol that maps event numbers to time stamps, and let TA (Timed Automata) be finite automata with clock variables. In the following table, the formalisms are annotated with the superscript $\mathbb{N}$ to emphasize the fact that all time stamps are natural numbers:

	Languages	Temporal logics	Monadic theories	Finite automata
1	*$\mathbb{N}$-timed counter-free ω-regular*	$\mathsf{MetricTL}^{\mathbb{N}} = \mathsf{ClockTL}^{\mathbb{N}}$	$\mathsf{TimeFunctionML}_1^{\mathbb{N}}$	
2	*$\mathbb{N}$-timed ω-regular*	$\mathsf{Q\text{-}MetricTL}^{\mathbb{N}} = \mathsf{Q\text{-}ClockTL}^{\mathbb{N}} = \mathsf{E\text{-}MetricTL}^{\mathbb{N}} = \mathsf{E\text{-}ClockTL}^{\mathbb{N}}$	$\mathsf{TimeFunctionML}_2^{\mathbb{N}}$	$\mathsf{TA}^{\mathbb{N}}$

Once again, all these formalisms are fully decidable, and the temporal logics and finite automata with timing constraints are elementarily decidable.

If time stamps are real instead of natural numbers, then the situation seems much less satisfactory. Several positively and fully decidable formalisms have been proposed, but no expressive equivalence results were known for fully decidable formalisms [AH92]. The previously known results are listed in the following table, where the omission of superscripts indicates that time stamps are real numbers:

Temporal logics	Monadic theories	Finite automata
Fully decidable		
MetricIntervalTL [AFH96] EventClockTL [RS97]		
		EventClockTA [AFH94]
Positively decidable		
$\mathsf{TL}^+ + \mathsf{TA}$ [Wil94]	$\mathcal{L}d^{\leftarrow}$ [Wil94]	TA [AD94]
Fully undecidable		
MetricTL [AH93]		
ClockTL [AH94]	TimeFunctionML$_1$ [AH93]	
	TimeFunctionML$_2$	

On one hand, the class of Timed Automata is unsatisfactory, because over real-numbered time it is only positively decidable: $\mathbb{R}$-timed automata are not closed under complement, and the corresponding temporal and monadic logics (and regular expressions [ACM97]) have no negation operator. On the other hand, the classes of Metric and Clock Temporal Logics (as well as monadic logic with a time function), which include negation, are unsatisfactory, because over real-numbered time their satisfiability problems are undecidable. Hence several restrictions of these classes have been studied.

1. The first restriction concerns the style of specifying timing constraints using time-bounded temporal operators. The Metric-Interval Logic MetricIntervalTL (also called MITL [AFH96]) is obtained from MetricTL by restricting the time bounds on temporal operators to nonsingular intervals. For example, the MetricIntervalTL formula $\Box(p \rightarrow \Diamond_{[4,6]} q)$ specifies that every p event is followed by a q event such that the difference between the two time stamps is at least 4 and at most 6. The restriction to nonsingularity prevents the specification of the exact real-numbered time difference 5 between events.
2. The second restriction concerns the style of specifying timing constraints using clock variables. The Event-Clock Logic EventClockTL (also called SCL [RS97]) and Event-Clock Automata EventClockTA are obtained from ClockTL and TA, respectively, by restricting the use of clocks to refer to the times of previous and next occurrences of events only. For example, if y_q is a clock that always refers to the time difference between now and the next q event, then the EventClockTL formula $\Box(p \rightarrow y_q = 5)$ specifies that every p event is followed by a q event such that the difference between time stamps of the p event and the first subsequent q event is exactly 5. A clock such as y_q, which is permanently linked to the next q event, does not need to be started explicitly, and is called an *event clock*. The restriction to event clocks prevents the specification of time differences between a p event and any subsequent (rather than the first subsequent) q event.

Both restrictions lead to pleasing formalisms that are fully (elementarily) decidable and have been shown sufficient in practical applications. However, nothing was known about

the relative expressive powers of these two independent approaches, and so the question which sets of timed ω-languages deserve the labels "$\mathbb{R}$-timed counter-free ω-regular" and "$\mathbb{R}$-timed ω-regular" remained open.

In this paper, we show that MetricIntervalTL and EventClockTL are equally expressive, and by adding the ability to count, as expressive as EventClockTA. This result is quite surprising, because (1) over real-numbered time, unrestricted MetricTL is known to be strictly less expressive than unrestricted ClockTL [AH93], and (2) the nonsingularity restriction (which prohibits exact time differences but allows the comparison of unrelated events) is very different in flavor from the event-clock restriction (which allows exact time differences but prohibits the comparison of unrelated events). Moreover, the expressive equivalence of Metric-Interval and Event-Clock logics reveals a robust picture of canonical specification formalisms for real-numbered time that parallels the untimed case and the case of natural-numbered time. We complete this picture by characterizing both the counter-free and the counting levels of expressiveness also by fully decidable monadic theories, called $\mathsf{MinMaxML}_1$ and $\mathsf{MinMaxML}_2$. These are first-order and second-order monadic theories of the real numbers with integer addition, comparison, and (besides universal and existential quantification) two first-order quantifiers that determine the first time and the last time at which a formula is true. Our results, which are summarized in the following table, suggest that we have identified two classes of ω-languages with real-numbered time stamps that may justly be called "$\mathbb{R}$-timed counter-free ω-regular" and "$\mathbb{R}$-timed ω-regular":

	Languages	Temporal logics	Monadic theories	Finite automata
		Fully decidable		
1	$\mathbb{R}$-*timed counter-free* ω-*regular*	MetricIntervalTL = EventClockTL	$\mathsf{MinMaxML}_1$	
2	$\mathbb{R}$-*timed* ω-*regular*	Q-MetricIntervalTL = Q-EventClockTL = E-MetricIntervalTL = E-EventClockTL	$\mathsf{MinMaxML}_2$	EventClockTA

Finally, we explain the gap between the $\mathbb{R}$-timed ω-regular languages and the languages definable by positively decidable formalisms such as timed automata. We show that the richer class of languages is obtained by closing the $\mathbb{R}$-timed ω-regular languages under projection. (It is unfortunate, but well-known [AFH94] that we cannot nontrivially have both full decidability and closure under projection in the case of real-numbered time.) The complete picture, then, results from adding the following line to the previous table (projection, or outermost existential quantification, is indicated by P-):

		Positively decidable		
3	*projection-closed* $\mathbb{R}$-*timed* ω-*regular*	P-EventClockTL	P-$\mathsf{MinMaxML}_2$ = $\mathcal{L}d^{-}$	P-EventClockTA = TA

2 The Regular Timed ω-Languages

An *interval* $I \subseteq \mathbb{R}^+$ is a convex nonempty subset of the nonnegative reals. Given $t \in \mathbb{R}^+$, we freely use notation such as $t + I$ for the interval $\{t' \mid \text{exists } t'' \in I \text{ with } t' = t + t''\}$, and $t > I$ for the constraint "$t > t'$ for all $t' \in I$." Two intervals I and J are *adjacent* if the right endpoint of I is equal to the left endpoint of J, and either I is right-open and J is left-closed or I is right-closed and J is left-open. An *interval sequence* $\bar{I} = I_0, I_1, \ldots$ is a finite or infinite sequence of bounded intervals so that for all $i \geq 0$, the intervals I_i and I_{i+1} are adjacent. We say that the interval sequence $\bar{I}$ *covers* the interval $\bigcup_{i \geq 0} I_i$. If $\bar{I}$ covers $[0, \infty)$, then $\bar{I}$ partitions the nonnegative real line so that every bounded subset of $\mathbb{R}^+$ is contained within a finite union of elements from the partition.

Let P be a finite set of propositional symbols. A *state* $s \subseteq P$ is a set of propositions. Given a state s and a proposition p, we write $s \models p$ instead of $p \in s$. A *timed state sequence* $\tau = (\bar{s}, \bar{I})$ is a pair that consists of an infinite sequence $\bar{s}$ of states and an infinite interval sequence $\bar{I}$ that covers $[0, \infty)$. Equivalently, the timed state sequence τ can be viewed as a function from $\mathbb{R}^+$ to 2^P, indicating for each time $t \in \mathbb{R}^+$ a state $\tau(t) \subseteq P$. In the introduction, we spoke of events rather than states. We do not formalize this distinction; it suffices to say that an event can be viewed as a change in state. A *timed ω-language* is a set of timed state sequences.

2.1 Recursive Event-Clock Automata

An event-clock automaton is a special case of a timed automaton [AD94], where the starting of clocks is determined by the input instead of by the transition relation. We generalize the definition of event-clock automata from [AFH94]. There, an automaton accepts (or rejects) a given timed state sequence τ. The automaton views the input sequence τ starting from time 0 by executing a transition relation that is constrained by conditions on clocks. There are two clocks for each proposition p. The history clock x_p always shows the amount of time that has expired since the proposition p has become false in τ, and the prophecy clock y_p always shows the amount of time that will expire until the proposition p will become true in τ. More precisely, if p has never been true, then the history clock has the undefined value $\perp$; if p was last true v time units ago during a right-closed interval, then the value of x_p is v; and if p was last true during a right-open interval whose right endpoint was v time units ago, then the value of x_p is the nonstandard real v^+, which is larger than v but smaller than all reals that are larger than v. Similar conventions apply to the prophecy clock y_p.

We have an automaton A accept (or reject) a given pair (τ, t) that consists of a timed state sequence τ and a time $t \in \mathbb{R}^+$. The automaton is started at time t and views the "past" of the input sequence τ by executing a *backward* transition relation, and the "future" by executing to a *forward* transition relation. If A accepts the pair (τ, t), we say that A accepts τ at time t. This allows us to associate a history clock and a prophecy clock with each automaton. The history clock x_A always shows the amount of time that has expired since the last time at which A accepted τ, and the prophecy clock y_A always shows the amount of time that will expire until the next time at which A will accept τ. This definition of event-clock automata is recursive. The base automata, whose transition relations are not constrained by clocks, are called floating automata. Formally, a *floating automaton* is a tuple $A = (Q, Q_0, \delta, \delta', P, \lambda, Q_F, Q'_F)$ such that

- Q is a finite set of locations,
- $Q_0 \subseteq Q$ is the set of starting locations,
- $\delta \subseteq Q \times Q$ is the forward transition relation,
- $\delta' \subseteq Q \times Q$ is the backward transition relation,
- P is a finite set of propositional symbols,
- λ: $Q \rightarrow \mathsf{BoolComb}(P)$ is a function that labels each location with a boolean combination of propositions,
- $Q_F \subseteq Q$ is a set of forward accepting locations, and
- $Q'_F \subseteq Q$ is a set of backward accepting locations.

Let τ be a timed state sequence whose propositional symbols contain all propositions in P. The floating automaton A *accepts* τ at time $t \in \mathbb{R}^+$, denoted $\mathsf{Accept}_\tau(A, t)$, iff there exist an infinite forward run $\rho = (\bar{q}, \bar{I})$ and an finite backward run $\rho' = (\bar{q}', \bar{I}')$ such that the following conditions are met.

Covering The forward run ρ consists of an infinite sequence $\bar{q}$ of locations from Q, and an infinite interval sequence $\bar{I}$ that covers $[t, \infty)$. The backward run ρ' consists of a finite sequence $\bar{q}'$ of locations and a finite interval sequence $\bar{I}'$, of the same length as $\bar{q}'$, which covers $[0, t]$.

Starting The forward and backward runs start in the same starting location; that is, $\rho(t) = \rho'(t)$ and $\rho(t) \in Q_0$.

Consecution The forward and backward runs respect the corresponding transition relations; that is, $(q_i, q_{i+1}) \in \delta$ for all $i \geq 0$, and $(q'_i, q'_{i-1}) \in \delta'$ for all $0 < i < |\bar{q}'|$.

Singularities We use the transition relations to encode the fact that a location may be transient, in the sense that the automaton control cannot remain in the location for any positive amount of time. For all $i \geq 0$, if $(q_i, q_i) \notin \delta$, then I_i is singular, and for all $0 \leq i < |\bar{q}'|$, if $(q'_i, q'_i) \notin \delta'$, then I'_i is singular.

Constraints The timed state sequence respects the constraints that are induced by the forward and backward runs; that is, $\tau(t') \models \lambda(\rho(t'))$ for all real times $t' \in [t, \infty)$, and $\tau(t') \models \lambda(\rho'(t'))$ for all real times $t' \in [0, t]$.

Accepting The forward run is Büchi accepting and the backward run ends in an accepting location; that is, there exist infinitely many $i \geq 0$ such that $q_i \in Q_F$, and $q'_0 \in Q'_F$.

A *recursive event-clock automaton of level* $i \in \mathbb{N}$ is a tuple $A = (Q, Q_0, \delta, \delta', P, \lambda, Q_F, Q'_F)$ that has the same components as a floating automaton, except that the labeling function $\lambda\colon Q \to \mathsf{BoolComb}(P \cup \Gamma_i)$ assigns to each location a boolean combination of propositions and level-i clock constraints. The set Γ_i of *level-i clock constraints* contains all atomic formulas of the form $x_B \sim c$ and $y_B \sim c$, where B is a recursive event-clock automaton of level less than i whose propositions are contained in P, where c is a nonnegative integer constant, and where $\sim \in \{<, \leq, =, \geq, >\}$. The clock x_B is called a *history clock*, and the clock y_B, a *prophecy clock*. In particular, the set of level-0 clock constraints is empty, and thus the level-0 event-clock automata are the floating automata. The level-1 event-clock automata have a history clock and a prophecy clock for each floating automaton, etc. If A contains a constraint on x_B or y_B, we say that B is a *subautomaton* of A.

The definition of when the recursive event-clock automaton A of level i accepts a timed state sequence τ at time t is as for floating automata, only that we need to define the satisfaction relation $\tau(t') \models (z \sim c)$ for every time $t' \in \mathbb{R}^+$ and every level-i clock constraint $(z \sim c) \in \Gamma_i$. This is done as follows. The timed state sequence τ *satisfies* the clock constraint $z \sim c$ at time t, written $\tau(t) \models (z \sim c)$, iff $\mathsf{Val}|_\tau(z, t) \neq \bot$ and $\mathsf{Val}|_\tau(z, t) \sim c$ for the following value of clock z at time t of τ (throughout this paper, v, v', and v'' range over the nonnegative reals):

$$\mathsf{Val}|_\tau(x_B, t) = \begin{cases} v & \text{if } \mathsf{Accept}_\tau(B, t - v),\ v > 0, \\ & \quad \text{and for all } v',\ 0 < v' < v, \text{ not } \mathsf{Accept}_\tau(B, t - v') \\ v^+ & \text{if for all } v' > v, \text{ exists } v'',\ v < v'' < v' \text{ with } \mathsf{Accept}_\tau(B, t - v''), \\ & \quad \text{and for all } v',\ 0 < v' \leq v, \text{ not } \mathsf{Accept}_\tau(B, t - v') \\ \bot & \text{if for all } v,\ 0 < v \leq t, \text{ not } \mathsf{Accept}_\tau(B, t - v) \end{cases}$$

$$\mathsf{Val}|_\tau(y_B, t) = \begin{cases} v & \text{if } \mathsf{Accept}_\tau(B, t + v),\ v > 0, \\ & \quad \text{and for all } v',\ 0 < v' < v, \text{ not } \mathsf{Accept}_\tau(B, t + v') \\ v^+ & \text{if for all } v' > v, \text{ exists } v'',\ v < v'' < v' \text{ with } \mathsf{Accept}_\tau(B, t + v''), \\ & \quad \text{and for all } v',\ 0 < v' \leq v, \text{ not } \mathsf{Accept}_\tau(B, t + v') \\ \bot & \text{if for all } v > 0, \text{ not } \mathsf{Accept}_\tau(B, t + v) \end{cases}$$

The recursive event-clock automaton A *defines* a timed ω-language, namely, the set of timed state sequences τ such that $\mathsf{Accept}_\tau(A, 0)$.

The following two theorems were proved in [AFH94] for the special case of level-1 event-clock automata. There, it was shown that every level-1 event-clock automaton can be determinized, because at all times during the run of an automaton, the value of each clock is determined solely by the input sequence and does not depend on nondeterministic choices made during the run. This crucial property, which fails for arbitrary timed automata, holds for all recursive event-clock automata.

Theorem 1. *The timed ω-languages definable by recursive event-clock automata are constructively closed under all boolean operations. In particular, for every recursive event-clock automaton we can construct a recursive event-clock automaton that defines the complementary timed ω-language.*

Theorem 2. *The emptiness problem for recursive event-clock automata is complete for* PSPACE.

It can be shown that level-$(i+1)$ event-clock automata, for all $i \in \mathbb{N}$, are strictly more expressive than level-i event-clock automata. It can also be shown that recursive event-clock automata are only partially closed under projection: only propositions that do not appear in subautomata can be projected. This partial-projection result will be used later to define decidable real-time logics with second-order quantification.

2.2 The Real-Time Sequential Calculus

In the sequel, we use p, q, and r for monadic predicates (second-order variables) over the nonnegative reals, and t, t_1, and t_2 for first-order variables over $\mathbb{R}^+$. The formulas of the *Second-Order Real-Time Sequential Calculus* $\mathsf{MinMaxML}_2$ are generated by the following grammar:

$$\begin{aligned}\Phi \ ::= \ & p(t) \mid t_1 \sim t_2 \mid \\ & (\mathsf{Min}\ t_1)(t_1 > t_2 \wedge \Psi(t_1)) \sim (t_2 + c) \mid (\mathsf{Max}\ t_1)(t_1 < t_2 \wedge \Psi(t_1)) \sim (t_2 - c) \mid \\ & \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid (\exists t)\Phi \mid (\exists p)\Theta\end{aligned}$$

where $\Psi(t_1)$ is a $\mathsf{MinMaxML}_2$ formula that contains no free occurrences of first-order variables other than t_1, where Θ is a $\mathsf{MinMaxML}_2$ formula that contains no occurrences of p within the scope of a Min or Max quantifier, where c is a nonnegative integer constant, and where $\sim \in \{<, \leq, =, \geq, >\}$. The formulas of the *First-Order Real-Time Sequential Calculus* $\mathsf{MinMaxML}_1$ are defined as above, without the clause $(\exists p)\Theta$.

The truth value of a $\mathsf{MinMaxML}_2$ formula Φ is evaluated over a pair (τ, α) that consists of a timed state sequence τ whose propositional symbols contain all second-order variables of Φ, and a valuation α that maps each free first-order variable of Φ to a nonnegative real. By $\alpha_{[t \mapsto v]}$ we denote the valuation that agrees with α on all variables except t, which is mapped to the value v. We first define for each $\mathsf{MinMaxML}_2$ term θ a value $\mathsf{Val}_{\tau,\alpha}(\theta)$, which is either a nonstandard real or undefined. Intuitively, the term $(\mathsf{Min}\ t_1)(t_1 > t_2 \wedge \Psi(t_1))$ denotes the smallest value greater than t_2 that satisfies the formula Ψ. If there is no value greater than t_2 that satisfies Ψ, then the term $(\mathsf{Min}\ t_1)(t_1 > t_2 \wedge \Psi(t_1))$ denotes the undefined value $\perp$. If Ψ is satisfied throughout a left-open interval with left endpoint $v > t_2$, then the term $(\mathsf{Min}\ t_1)(t_1 > t_2 \wedge \Psi(t_1))$ denotes the nonstandard real v^+. Similarly, the term $(\mathsf{Max}\ t_1)(t_1 < t_2 \wedge \Psi(t_1))$ denotes the greatest value smaller than t_2 that satisfies Ψ:

$$\begin{aligned}&\mathsf{Val}_{\tau,\alpha}(t) = \alpha(t) \\ &\mathsf{Val}_{\tau,\alpha}(t + c) = \alpha(t) + c\end{aligned}$$

$$\mathsf{Val}_{\tau,\alpha}(t-c) = \begin{cases} \alpha(t)-c & \text{if } \alpha(t) \geq c \\ \bot & \text{otherwise} \end{cases}$$

$$\mathsf{Val}_{\tau,\alpha}((\mathsf{Min}\ t_1)(t_1 > t_2 \wedge \Psi(t_1))) =$$
$$\begin{cases} v & \text{if } (\tau, \alpha_{[t_1 \mapsto v]}) \models (t_1 > t_2 \wedge \Psi(t_1)), \\ & \quad \text{and for all } v' < v, \text{ not } (\tau, \alpha_{[t_1 \mapsto v']}) \models (t_1 > t_2 \wedge \Psi(t_1)) \\ v^+ & \text{if for all } v' > v, \text{ exists } v'' < v' \text{ with } (\tau, \alpha_{[t_1 \mapsto v'']}) \models (t_1 > t_2 \wedge \Psi(t_1)), \\ & \quad \text{and for all } v' \leq v, \text{ not } (\tau, \alpha_{[t_1 \mapsto v']}) \models (t_1 > t_2 \wedge \Psi(t_1)) \\ \bot & \text{if for all } v \geq 0, \text{ not } (\tau, \alpha_{[t_1 \mapsto v]}) \models (t_1 > t_2 \wedge \Psi(t_1)) \end{cases}$$

$$\mathsf{Val}_{\tau,\alpha}((\mathsf{Max}\ t_1)(t_1 < t_2 \wedge \Psi(t_1))) =$$
$$\begin{cases} v & \text{if } (\tau, \alpha_{[t_1 \mapsto v]}) \models (t_1 < t_2 \wedge \Psi(t_1)), \\ & \quad \text{and for all } v' > v, \text{ not } (\tau, \alpha_{[t_1 \mapsto v']}) \models (t_1 < t_2 \wedge \Psi(t_1)) \\ v^- & \text{if for all } v' < v, \text{ exists } v'' > v' \text{ with } (\tau, \alpha_{[t_1 \mapsto v'']}) \models (t_1 < t_2 \wedge \Psi(t_1)), \\ & \quad \text{and for all } v' \geq v, \text{ not } (\tau, \alpha_{[t_1 \mapsto v']}) \models (t_1 < t_2 \wedge \Psi(t_1)) \\ \bot & \text{if for all } v \geq 0, \text{ not } (\tau, \alpha_{[t_1 \mapsto v]}) \models (t_1 < t_2 \wedge \Psi(t_1)) \end{cases}$$

Now we can define the satisfaction relation for $\mathsf{MinMaxML}_2$ formulas:

$(\tau, \alpha) \models p(t)$ iff $p \in \tau(\alpha(t))$
$(\tau, \alpha) \models t_1 \sim t_2$ iff $\mathsf{Val}_{\tau,\alpha}(t_1) \sim \mathsf{Val}_{\tau,\alpha}(t_2)$
$(\tau, \alpha) \models \Phi_1 \wedge \Phi_2$ iff $(\tau, \alpha) \models \Phi_1$ and $(\tau, \alpha) \models \Phi_2$
$(\tau, \alpha) \models \neg\Phi$ iff not $(\tau, \alpha) \models \Phi$
$(\tau, \alpha) \models (\exists t)\Phi$ iff exists $v \geq 0$ with $(\tau, \alpha_{[t \mapsto v]})\Phi$
$(\tau, \alpha) \models (\exists p)\Theta$ iff $(\tau^p, \alpha) \models \Theta$ for some timed state sequence τ^p that agrees with τ on all propositions except p

A $\mathsf{MinMaxML}_2$ formula is *closed* iff it contains no free occurrences of first-order variables. Every closed $\mathsf{MinMaxML}_2$ formula Φ *defines* a timed ω-language, namely, the set of timed state sequences τ such that $(\tau, \emptyset) \models \Phi$.

Example 1. The $\mathsf{MinMaxML}_1$ formula

$$(\forall t_1)(p(t_1) \rightarrow (\exists t_2)(t_2 > t_1 \wedge q(t_2) \wedge (\mathsf{Min}\ t_3)(t_3 > t_2 \wedge r(t_3)) = t_2 + 5))$$

asserts that every p-state is followed by a q-state that is followed by an r-state after, but no sooner than, 5 time units. □

The following theorem shows that the second-order real-time sequential calculus defines the same class of timed ω-languages as the recursive event-clock automata. We call this class the *ω-regular real-time languages*.

Theorem 3. *For every* $\mathsf{MinMaxML}_2$ *formula we can construct a recursive event-clock automaton that defines the same timed ω-language, and vice versa.*

Corollary 1. *The satisfiability problem for* $\mathsf{MinMaxML}_2$ *is decidable.*

It is not difficult to embed the first-order fragment ML_1 of the sequential calculus into $\mathsf{MinMaxML}_1$ in linear time, which implies that the satisfiability problem for $\mathsf{MinMaxML}_1$ is nonelementary [Sto74].

3 The Counter-Free Regular Timed ω-Languages

3.1 Two Decidable Real-Time Temporal Logics

We recall the definitions of two real-time temporal logics that are known to have decidable satisfiability problems.

Event-Clock Temporal Logic The formulas of EventClockTL [RS97] are built from propositional symbols, boolean connectives, the temporal "until" and "since" operators, and two real-time operators: at any time t, the *history operator* $\lhd_I \phi$ asserts that ϕ was true last in the interval $t - I$, and the *prophecy operator* $\rhd_I \phi$ asserts that ϕ will be true next in the interval $t + I$. The formulas of EventClockTL are generated by the following grammar:

$$\phi \ ::= \ p \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \phi_1 \mathcal{U} \phi_2 \mid \phi_1 \mathcal{S} \phi_2 \mid \lhd_I \phi \mid \rhd_I \phi$$

where p is a proposition and I is an interval whose finite endpoints are nonnegative integers. Let ϕ be an EventClockTL formula and let τ be a timed state sequence whose propositional symbols contain all propositions that occur in ϕ. The formula ϕ *holds* at time $t \in \mathbb{R}^+$ of τ, denoted $(\tau, t) \models \phi$, according to the following definition:

- $(\tau, t) \models p$ iff $p \in \tau(t)$
- $(\tau, t) \models \phi_1 \wedge \phi_2$ iff $(\tau, t) \models \phi_1$ and $(\tau, t) \models \phi_2$
- $(\tau, t) \models \neg\phi$ iff not $(\tau, t) \models \phi$
- $(\tau, t) \models \phi_1 \mathcal{U} \phi_2$ iff exists a real $t' > t$ with $(\tau, t') \models \phi_2$, and for all reals $t'' \in (t, t')$, we have $(\tau, t'') \models \phi_1 \vee \phi_2$
- $(\tau, t) \models \phi_1 \mathcal{S} \phi_2$ iff exists a real $t' < t$ with $(\tau, t') \models \phi_2$, and for all reals $t'' \in (t', t)$, we have $(\tau, t'') \models \phi_1 \vee \phi_2$
- $(\tau, t) \models \lhd_I \phi$ iff exists a real $t' < t$ with $t' \in (t - I)$ and $(\tau, t') \models \phi$, and for all reals $t'' < t$ with $t'' > (t - I)$, not $(\tau, t'') \models \phi$
- $(\tau, t) \models \rhd_I \phi$ iff exists a real $t' > t$ with $t' \in (t + I)$ and $(\tau, t') \models \phi$, and for all reals $t'' > t$ with $t'' < (t + I)$, not $(\tau, t'') \models \phi$

Note that the temporal and real-time operators are defined in a strict manner; that is, they do not constrain the current state. Nonstrict operators are easily defined from their strict counterparts. We use the following abbreviations for additional operators: $\Diamond\phi \equiv \top \mathcal{U} \phi$, $\Box\phi \equiv \neg\Diamond\neg\phi$, and $\mathcal{J}\phi \equiv \bot \mathcal{U} \phi$. The EventClockTL formula ϕ *defines* the timed ω-language that contains all timed state sequences τ with $(\tau, 0) \models \phi$.

Example 2. The EventClockTL formula $\Box(p \rightarrow \Diamond(q \wedge \rhd_{[5,5]} r))$ defines the timed ω-language from Example 1. □

Theorem 4. [RS97] *The satisfiability problem for* EventClockTL *is complete for* PSPACE.

Metric-Interval Temporal Logic The formulas of MetricIntervalTL [AFH96] are built from propositional symbols, boolean connectives, and time-bounded "until" and "since" operators:

$$\phi \ ::= \ p \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \phi_1 \mathcal{U}_I \phi_2 \mid \phi_1 \mathcal{S}_I \phi_2$$

where p is a proposition and I is a *nonsingular* interval whose finite endpoints are nonnegative integers. The formulas of the fragment $\mathsf{MetricIntervalTL}_{0,\infty}$ are defined as above, except that the interval I must either have the left endpoint 0, or be unbounded; in these cases I can be replaced by an expression of the form $\sim c$, for a nonnegative integer constant c and $\sim \in \{<, \leq, \geq, >\}$. The MetricIntervalTL formula ϕ *holds* at time $t \in \mathbb{R}^+$ of the timed state sequence τ, denoted $(\tau, t) \models \phi$, according to the following definition (the propositional and boolean cases are as for EventClockTL):

- $(\tau, t) \models \phi_1 \mathcal{U}_I \phi_2$ iff exists a real $t' \in (t + I)$ with $(\tau, t') \models \phi_2$, and for all reals $t'' \in (t, t')$, we have $(\tau, t'') \models \phi_1$
- $(\tau, t) \models \phi_1 \mathcal{S}_I \phi_2$ iff exists a real $t' \in (t - I)$ with $(\tau, t') \models \phi_2$, and for all reals $t'' \in (t', t)$, we have $(\tau, t'') \models \phi_1$

The MetricIntervalTL formula ϕ *defines* the timed ω-language that contains all timed state sequences τ with $(\tau, 0) \models \phi$.

Example 3. The MetricIntervalTL formula $\Box(p \rightarrow \Diamond_{[4,6]}\, q)$ asserts that every p-state is followed by a q-state at a time difference of at least 4 and at most 6 time units. Note that, by contrast, the EventClockTL formula $\Box(p \rightarrow \rhd_{[4,6]}\, q)$ asserts the stronger requirement that after every p-state, the *next* subsequent q-state occurs at a time difference of at least 4 and at most 6 time units. □

Theorem 5. [AFH96] *The satisfiability problem for* MetricIntervalTL *is complete for* EXPSPACE. *The satisfiability problem for* $\mathsf{MetricIntervalTL}_{0,\infty}$ *is complete for* PSPACE.

It is not difficult to see that the union of EventClockTL and MetricIntervalTL is still decidable in EXPSPACE, and that the union of EventClockTL and $\mathsf{MetricIntervalTL}_{0,\infty}$ is still decidable in PSPACE. These union logics are of practical interest, because the event-clock style and the metric-interval style of specification are orthogonal: for each style, there are properties that are more naturally specified in that style. Next we shall see that while the union enhances convenience in practical specification, it does not enhance the theoretical expressiveness of either logic.

3.2 Expressive-Equivalence Results

We show that all of EventClockTL, MetricIntervalTL, and $\mathsf{MetricIntervalTL}_{0,\infty}$ are expressively equivalent to the first-order real-time sequential calculus; that is, they all define the same class of timed ω-languages. We call this class the *counter-free ω-regular real-time languages*.

Theorem 6. *For every* $\mathsf{MinMaxML}_1$ *formula we can construct an* EventClockTL *formula that defines the same timed ω-language. For every* EventClockTL *formula we can construct an* $\mathsf{MetricIntervalTL}_{0,\infty}$ *formula that defines the same timed ω-language. For every* MetricIntervalTL *formula we can construct a* $\mathsf{MinMaxML}_1$ *formula that defines the same timed ω-language.*

Proof sketch. The following two examples present some of the main ideas behind the proof. First, the EventClockTL formula $\rhd_{[c,c]}\, p$ is equivalent to the MetricIntervalTL formula $\Box_{(0,c)} \neg p \wedge \Diamond_{(0,c]}\, p$. Second, the MetricIntervalTL formula $\Diamond_{(1,2)}\, p$ is equivalent to the disjunction of the three EventClockTL formulas (1) $\rhd_{[1,1]} \mathcal{J} p \vee \rhd_{(0,1)} \rhd_{[1,1]} \mathcal{J} p$, (2) $\rhd_{(0,1)} \rhd_{[1,1]}\, p$, and (3) $\neg \rhd_{(0,1]} \neg \rhd_{(0,1)}\, p$. Disjunct (1) corresponds to the case that the first p-state in interval $(1, 2)$ occurs at least 1 time unit after the previous p-state, and is left-open; disjunct (2) corresponds to the left-closed case; and disjunct (3) corresponds to the case that the first p-state in interval $(1, 2)$ occurs less than 1 time unit after the previous p-state. The translation of the more general MetricIntervalTL formula $\Diamond_{(c,c+1)}\, p$ is inductive and requires an exponential blow-up, which is unavoidable due to the lower complexity of EventClockTL satisfiability. □

3.3 From Counter-Free to Counting Regular Timed ω-Languages

The difference between the counter-free ω-regular real-time languages and the ω-regular real-time languages is the counting ability of automata (and of second-order quantification). This is demonstrated by the fact that if we add either automata connectives (in the style of [WVS83]) or quantification over propositions (in the style of [Sis83]) to either

EventClockTL or MetricIntervalTL (or MetricIntervalTL$_{0,\infty}$), we obtain a formalism that defines exactly the full class of the ω-regular real-time languages. These results complete the pleasant analogy to the untimed case.

Automata Connectives We define the *extended temporal logics* E-EventClockTL and E-MetricIntervalTL by adding to the definition of formulas the syntactic clause $\phi ::= A(\phi_1, \ldots, \phi_n)$, where A is a floating automaton whose propositions are replaced by the set $\{\phi_1, \ldots, \phi_n\}$ of formulas. In the definition of the satisfaction relation, add $(\tau, t) \models A(\phi_1, \ldots, \phi_n)$ iff $\mathsf{Accept}_\tau(A, t)$, using formula satisfaction $(\tau, t) \models \phi$ instead of proposition satisfaction $\tau(t) \models p$.

Theorem 7. *The satisfiability problems for* E-EventClockTL *and* E-MetricIntervalTL$_{0,\infty}$ *are complete for* PSPACE. *The satisfiability problem for* E-MetricIntervalTL *is complete for* EXPSPACE.

Theorem 8. *For every* MinMaxML$_2$ *formula we can construct an* E-EventClockTL *formula that defines the same timed ω-language. For every* E-EventClockTL *formula we can construct an* E-MetricIntervalTL$_{0,\infty}$ *formula that defines the same timed ω-language. For every* E-MetricIntervalTL *formula we can construct a* MinMaxML$_2$ *formula that defines the same timed ω-language.*

Propositional Quantification The *quantified temporal logics* Q-EventClockTL and Q-MetricIntervalTL are defined by adding to the definition of formulas the syntactic clause $\phi ::= (\exists p)\psi$, where p is a proposition which, inside the formula ψ, does not occur within the scope of a history or prophecy operator. Then, $(\tau, t) \models (\exists p)\psi$ iff $(\tau^p, t) \models \psi$ for some timed state sequence τ^p that agrees with τ on all propositions except p.

Theorem 9. *The satisfiability problems for* Q-EventClockTL *and* Q-MetricIntervalTL *are decidable.*

Since already the untimed quantified temporal logic Q-TL is nonelementary [Sis83], so are the satisfiability problems for Q-EventClockTL and Q-MetricIntervalTL$_{0,\infty}$.

Theorem 10. *For every* MinMaxML$_2$ *formula we can construct an* Q-EventClockTL *formula that defines the same timed ω-language. For every* Q-EventClockTL *formula we can construct an* Q-MetricIntervalTL$_{0,\infty}$ *formula that defines the same timed ω-language. For every* Q-MetricIntervalTL *formula we can construct a* MinMaxML$_2$ *formula that defines the same timed ω-language.*

4 Beyond Full Decidability

In our second-order formalisms MinMaxML$_2$ and Q-EventClockTL, we have prohibited quantified monadic predicates (propositions) from occurring within the scope of Min or Max quantifiers (history or prophecy operators). This restriction is necessary for decidability. If we admit only *outermost existential* quantification (projection) over monadic predicates (propositions) that occur within the scope of real-time operators, we obtain a positively decidable formalism (satisfiability is decidable, but validity is not) which is expressively equivalent to timed automata. Consequently, if we admit full quantification over monadic predicates (propositions) that occur within the scope of real-time operators, then both satisfiability and validity are undecidable, and the formalism is expressively equivalent to boolean combinations of timed automata.

A formula of the *Projection-Closed Real-Time Sequential Calculus* P-MinMaxML$_2$ has the form $(\exists q_1, \ldots, q_n)\Phi$, where $q_1, \ldots, q_n$ are monadic predicates and Φ is a formula of MinMaxML$_2$. A formula of the *Projection-Closed Event-Clock Logic* P-EventClockTL has the form $(\exists q_1, \ldots, q_n)\phi$, where $q_1, \ldots, q_n$ are propositions and ϕ is a formula of EventClockTL. A *projection-closed event-clock automaton* has the form $(\exists q_1, \ldots, q_n)A$, where $q_1, \ldots, q_n$ are propositions and A is a recursive event-clock automaton; it accepts all timed state sequences that agree with timed state sequences accepted by A on all propositions except $q_1, \ldots, q_n$. From Theorems 2, 3, and 4 it follows immediately that the satisfiability problem for P-MinMaxML$_2$ is decidable, and that the satisfiability problem for P-EventClockTL and the emptiness problem for projection-closed event-clock automata are complete for PSPACE. The dual problems of validity and universality, however, cannot be decided. This follows from the following theorem, which shows that by closing event-clock automata or event-clock logic under projection, we obtain timed automata, whose universality problem is undecidable [AD94].

Theorem 11. *For every timed automaton* [AD94] *we can construct a projection-closed event-clock automaton that defines the same timed ω-language, and vice versa.*

It follows that the formalisms of timed automata, projection-closed event-clock automata, P-EventClockTL, P-MinMaxML$_2$, and $\mathcal{L}d^{\leftrightarrow}$ [Wil94] define the same class of timed ω-languages. This class is closed under positive boolean operations, but not under complement [AD94].

Corollary 2. *The validity problems for* P-MinMaxML$_2$ *and* P-EventClockTL *and the universality problem for projection-closed event-clock automata are undecidable.*

A fully undecidable extension of MinMaxML$_1$ is obtained by relaxing the restriction that in every formula of the form $(\mathsf{Min}\ t_1)(t_1 > t_2 \wedge \Psi(t_1)) \sim (t_2 + c)$ or $(\mathsf{Max}\ t_1)(t_1 < t_2 \wedge \Psi(t_1)) \sim (t_2 - c)$, the subformula $\Psi(t_1)$ contains no free occurrences of first-order variables other than t_1. If we suppress this restriction, it can be shown that the real-time temporal logic MetricTL, which is MetricIntervalTL without the prohibition of singular intervals, can be embedded in MinMaxML$_1$. Since MetricTL is undecidable [AH93], so is the satisfiabilty problem for unrestricted MinMaxML$_1$.

References

[ACM97] E. Asarin, P. Caspi, and O. Maler. A Kleene theorem for timed automata. In *Proc. 12th IEEE Symp. Logic in Computer Science*, pp. 160–171, 1997.

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AFH94] R. Alur, L. Fix, and T.A. Henzinger. A determinizable class of timed automata. In *Computer-aided Verification*, LNCS 818:1–13. Springer-Verlag, 1994.

[AFH96] R. Alur, T. Feder, and T.A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43:116–146, 1996.

[AH92] R. Alur and T.A. Henzinger. Back to the future: towards a theory of timed regular languages. In *Proc. 33rd IEEE Symp. Foundations of Computer Science*, pp. 177–186, 1992.

[AH93] R. Alur and T.A. Henzinger. Real-time logics: complexity and expressiveness. *Information and Computation*, 104:35–77, 1993.

[AH94] R. Alur and T.A. Henzinger. A really temporal logic. *J. ACM*, 41:181–204, 1994.

[Büc62] J.R. Büchi. On a decision method in restricted second-order arithmetic. In *Proc. First Congress on Logic, Methodology, and Philosophy of Science* (1960), pp. 1–11. Stanford University Press, 1962.

[GPSS80] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proc. 7th ACM Symp. Principles of Programming Languages*, pp. 163–173, 1980.

[Kam68] J.A.W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California at Los Angeles, 1968.

[RS97] J.-F. Raskin and P.-Y. Schobbens. State clock logic: a decidable real-time logic. In *Hybrid and Real-time Systems*, LNCS 1201:33–47. Springer-Verlag, 1997.

[Sis83] A.P. Sistla. *Theoretical Issues in the Design and Verification of Distributed Systems*. PhD thesis, Harvard University, 1983.

[Sto74] L.J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD thesis, Massachusetts Institute of Technology, 1974.

[Wil94] T. Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In *Formal Techniques in Real-time and Fault-tolerant Systems*, LNCS 863:694–715. Springer-Verlag, 1994.

[Wol82] P.L. Wolper. *Synthesis of Communicating Processes from Temporal Logic Specifications*. PhD thesis, Stanford University, 1982.

[WVS83] P.L. Wolper, M.Y. Vardi, and A.P. Sistla. Reasoning about infinite computation paths. In *Proc. 24th IEEE Symp. Foundations of Computer Science*, pp. 185–194, 1983.