The Benefits of Relaxing Punctuality*†

Rajeev Alur‡ Tomás Feder§ Thomas A. Henzinger‡

Abstract. The most natural, compositional way of modeling real-time systems uses a dense domain for time. The satisfiability of real-time constraints that are capable of expressing punctuality in this model is, however, known to be undecidable.

We introduce a temporal language that can constrain the time difference between events only with *finite* (yet arbitrary) precision and show the resulting logic to be EXPSPACE-complete. This result allows us to develop an algorithm for the verification of timing properties of real-time systems with a dense semantics.

1 Introduction

The formal study of reactive systems has led recently to a number of suggestions of how real-time requirements of such systems ought to be modeled, specified, and verified. Most of these approaches are situated at either extreme of the trade-off between *realistic modeling* of time and *feasible verification* of timing properties. Typically, they either use a continuous model of time at the expense of decidability [ACD90, Koy90, Lew90], or they sacrifice continuity to obtain decision procedures [JM86, AH89, AH90, EMSS89, HLP90, Ost90]. This paper shows how a slight relaxation of the notion of punctuality allows us to combine the best of both worlds.

Let us be more specific. The linear (trace) semantics of a reactive system is defined as a set of possible behaviors, each of which is represented by a sequence of system states. This model is most naturally extended to incorporate real time by associating, with every state, an interval of the real line, which indicates the period of time during which the system is in that state. That is, we represent the possible behaviors of a real-time system by *timed state sequences.*

Alas, even the satisfiability of a very simple class of real-time properties turns out to be undecidable in this model [AH89]. An inspection of the proof shows that the only timing constraints required are of the form

$$\Box\,(p \;\rightarrow\; \Diamond_{=5}\, q), \qquad (\dagger)$$

predicting that every p-state is followed by a q-state *precisely* 5 time units later.

This negative result has led us, at first, to weaken the expressiveness of the model by adopting the *semantic* abstraction that, at every state change, we may record only a discrete approximation — the number of ticks of a digital clock — to the real time. Thus we have interpreted the formula (†) to require only that the p-state and the corresponding q-state are separated by exactly 5 clock ticks; their actual difference in time may be as much as (say) 5.9 time units or as small as 4.1 time units. We have shown that several interesting real-time logics are decidable under this weaker, *digital-clock*, interpretation [AH89, AH90].

In this paper we pursue an alternative, *syntactic*, concession. Instead of digitizing the meaning of a sentence, we prohibit timing constraints that predict the time difference between two states with infinite accuracy. In particular, we may not state the property given above, but only an approximation such as

$$\Box\,(p \;\rightarrow\; \Diamond_{(4.9,5.1)}\, q),$$

requiring that the p-state and the corresponding q-state are separated by more than 4.9 time units and less than 5.1 time units.

We define a language that can constrain the time difference between events only with finite (yet arbitrary) precision. The resulting *metric interval temporal logic* MITL is shown to be decidable in EXPSPACE. Furthermore, we show how to verify a real-time system with respect to a specification in MITL.

Properties of timed state sequences can, alternatively, be defined by *timed automata* [AD90]. While the emptiness problem for these automata is solvable, they are not closed under complement. MITL identifies a fragment of the properties definable by timed

*All proofs are omitted from this paper and can be found in a technical report with the same title issued by the Department of Computer Science of Stanford University (1991).

†This research was supported in part by an IBM graduate fellowship, by the National Science Foundation grants CCR-89-11512, CCR-89-13641, and MIP-88-588807, by the Defense Advanced Research Projects Agency under contract N00039-84-C-0211, and by the United States Air Force Office of Scientific Research under contract AFOSR-90-0057.

‡Department of Computer Science, Stanford University, Stanford, CA 94305.

§Bell Communications Research, Morristown, NJ 07962.

automata that is closed under all boolean operations. Thus the novelty of our results is that they give a *logical* formalism with a *continuous* interpretation of time that is suitable for the automatic verification and synthesis of finite-state real-time systems.

Both the semantic abstraction of digitizing models as well as the syntactic restriction of excluding equality in timing constraints limit the real-time properties that are definable in a similar way: they rule out the notion of absolute punctuality and replace it by a looser concept of *almost*-on-time behavior. This sacrifice is viable because, by choosing the clock tick of the digital clock small enough, we can still achieve arbitrary precision in either approach; moreover, the corresponding costs for achieving the desired accuracy are the same.

Yet the introduction of a mandatory slack through the syntax (rather than through the semantics) turns out to be the more powerful technique: we show that the properties of timed state sequences that can be defined in MITL are a proper superset of those definable with equality under a digital-clock interpretation. Also, many of the practically interesting forms of punctuality are still expressible in MITL, such as the requirement that every p-state is separated from the *closest* subsequent q-state by precisely 5 time units.

The remainder of the paper is organized in four parts. In Section 2, we introduce and motivate the logic MITL, and show it to be more expressive than digitization. In Section 3, we introduce a variant of timed automata as a model for finite-state real-time systems. In Section 4, we reduce the decision problem for MITL to the emptiness problem of timed automata. In the concluding section, we show how the results of this paper lead to an algorithm that verifies MITL-specifications of real-time systems that are given as timed automata.

We remark that in this paper we introduce MITL with future temporal operators only. All of our results, in particular EXPSPACE-completeness, generalize to MITL with both future and *past* temporal operators.

2 Metric Interval Temporal Logic

We define timed state sequences as formal models of real-time behavior. Then we introduce a temporal language to define properties of timed state sequences and study its expressive power.

2.1 Intervals and interval sequences

An interval is a convex subset of the nonnegative real numbers R^+. Intervals may be open, halfopen, or closed; bounded or unbounded. More precisely, each interval is of one of the following forms: $[a,b]$, $[a,b)$, $[a,\infty)$, $(a,b]$, (a,b), (a,∞), where $a \leq b$ and $a,b \in \mathsf{R}^+$. For an interval I of the above form, a is its left end-point, and b is its right end-point; the left end-point of I is denoted by $l(I)$ and the right end-point, for bounded I, is denoted by $r(I)$.

An interval I is *singular* iff it is of the form $[a,a]$; that is, I is closed and $l(I) = r(I)$.

Two intervals I and I' are *adjacent* iff (1) either I is right-open and I' is left-closed, or I is right-closed and I' is left-open, and (2) $r(I) = l(I')$. For instance, the intervals $(1,2]$ and $(2,2.5)$ are adjacent.

An *interval sequence* $\tau = I_0 I_1 I_2 I_3 \ldots$ is a finite or infinite sequence of intervals that partitions R^+:

1. Any two neighboring intervals I_i and I_{i+1} are adjacent.
2. For all $t \in \mathsf{R}^+$, there is some interval I_i with $t \in I_i$.

In particular, I_0 is left-closed and $l(I_0) = 0$; if τ is finite, then its last interval must be unbounded.

We will freely use intuitive pseudo-arithmetic expressions to denote intervals. For example, the expressions $\leq b$ and $> a$ stand for the intervals $[0,b]$ and (a,∞), respectively; by $< I$ we denote the interval $\{t' \mid 0 \leq t' < t \text{ for all } t \in I\}$. The expression $t + I$, where I is an interval and $t \in \mathsf{R}^+$, denotes the interval $\{t + t' \mid t' \in I\}$; similarly, $I - t$ and tI stand for the intervals $\{t' - t \mid t' \in I \text{ and } t' \geq t\}$ and $\{tt' \mid t' \in I\}$, respectively.

2.2 Timed state sequences

Let P be a finite set of atomic propositions. We assume that, at any point in time, the global state of a (finite-state) system can be modeled by an interpretation (or truth-value assignment) for P. We therefore identify states s with subsets of P; that is, $s \models p$ iff $p \in s$ (for $p \in P$).

A behavior of a discrete system over time can, consequently, be modeled by a finite or infinite sequence

$$\rho: \quad (s_0, I_0) \rightarrow (s_1, I_1) \rightarrow (s_2, I_2) \rightarrow (s_3, I_3) \rightarrow \cdots$$

of states $s_i \in 2^P$ and corresponding time intervals $I_i \subseteq \mathsf{R}^+$. A *timed state sequence* $\rho = (\sigma, \tau)$ consists of a sequence $\sigma: s_0 s_1 s_2 \ldots$ of states and an interval sequence $\tau: I_0 I_1 I_2 \ldots$ of the same length.

A timed state sequence $\rho = (\sigma, \tau)$ can be viewed as a map ρ^* from the time domain R^+ to the states 2^P (let $\rho^*(t) = s_i$ if $t \in I_i$). Thus a timed state sequence provides complete information about the global state of a system at each time instant: at time $t \in I_i$, the system is in state $\rho^*(t) = s_i$. Timed state sequences obey the

finite-variability condition: between any two points in time there are only finitely many state changes. This assumption is adequate for modeling *discrete* systems.

Given a timed state sequence (σ, τ), the i-th transition point, denoted by t_i, is defined to be the left end-point of the interval I_i; that is, $t_i = l(I_i)$. Note that the state at time t_i is s_{i-1} if I_i is left-open, and is s_i if I_i is left-closed.

Our definition allows *transient* states, which occur only a single point in time. If I_i is a singular interval $[t_i, t_i]$, then the state at time t_i is s_i, but the state just before t_i is s_{i-1}, and the state just after t_i is s_{i+1}. Observe that in such a case neither s_{i-1} nor s_{i+1} can be transient, because the interval I_{i-1} must be right-open and the interval I_{i+1} must be left-open. Transient states are useful for modeling the truth of propositions that represent instantaneous events and, thus, are true only at isolated points in time.

We will also need the concept of a *suffix* of a timed state sequence. For a timed state sequence $\rho = (\sigma, \tau)$ and time $t \in I_i$, let $\rho^t = (\sigma^i, \tau^t)$ be the timed state sequence with the state component $\sigma^i : s_i s_{i+1} s_{i+2} \cdots$ and the time component

$$\tau^t: \quad (I_i - t)(I_{i+1} - t)(I_{i+2} - t) \cdots$$

Note that the suffix operator is defined such that $(\rho^t)^*(t') = \rho^*(t + t')$ for all $t' \in \mathsf{R}^+$. In particular, $\rho^0 = \rho$.

2.3 Syntax and semantics of MITL

We introduce an extension of linear temporal logic, *metric interval temporal logic* (or MITL), that is interpreted over *timed* state sequences. A standard way of adding timing requirements to temporal languages is to replace the temporal operators with time-constrained versions, such as the constrained *eventually* operator $\Diamond_{[2,4]}$ meaning "eventually within 2 to 4 time units" [EMSS89, AH90, Koy90]. We adopt this approach for MITL, with the restriction that operators cannot be constrained by singular time intervals.

The formulas of MITL are built from atomic propositions by boolean connectives and time-constrained versions of the *until* operator $\mathcal{U}$; they are defined inductively as follows:

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

where $p \in P$ and I is a *nonsingular* interval with rational end-points (I may be unbounded).

The formulas of MITL are interpreted over timed state sequences, which provide an interpretation for the atomic propositions at each time instant. Informally, the formula $\phi_1 \mathcal{U}_I \phi_2$ holds at time $t \in \mathsf{R}^+$ of a timed state sequence iff there is a later time instant $t' \in t + I$ such that ϕ_2 holds at time t' and ϕ_1 holds throughout the time interval (t, t').

Given an MITL-formula ϕ and a timed state sequence $\rho = (\sigma, \tau)$, the satisfaction relation $\rho \models \phi$ is defined inductively as follows:

$\rho \models p$ iff $p \in s_0$.
$\rho \models \neg\phi$ iff $\rho \not\models \phi$.
$\rho \models \phi_1 \wedge \phi_2$ iff $\rho \models \phi_1$ and $\rho \models \phi_2$.
$\rho \models \phi_1 \mathcal{U}_I \phi_2$ iff $\rho^t \models \phi_2$ for some $t \in I$, and $\rho^{t'} \models \phi_1$ for all $t' \in (0, t)$.

The MITL-formula ϕ is *satisfiable* (*valid*) iff $\rho \models \phi$ for some timed state sequence ρ (all timed state sequences ρ, respectively).

Observe that the logic MITL is insensitive to *stuttering*. Given two timed state sequence $\rho = (\sigma, \tau)$ and $\rho' = (\sigma', \tau')$ such that ρ' has a subsequence of the form

$$(s_{i-1}, I_{i-1}) \rightarrow (s_i, I) \rightarrow (s_i, I') \rightarrow (s_{i+1}, I_{i+1})$$

and $I \cup I' = I_i$, then $\rho^* = \rho'^*$, and $\rho \models \phi$ iff $\rho' \models \phi$ for every MITL-formula ϕ.

The satisfaction relation has another desirable property: the truth value of any MITL-formula does not change more than ω times along a timed state sequence. Thus timed state sequences satisfy the finite-variability condition not only with respect to the truth of atomic propositions, but also with respect to arbitrarily complex MITL-formulas. The following lemma states this property formally:

Lemma 2.1 (Model refinement)
Let ϕ be an MITL*-formula and $\rho = (\sigma, \tau)$ be a timed state sequence. There exists an interval sequence $\tau_\phi : J_0 J_1 \ldots$ such that whenever t and t' belong to the same interval J_i, we have $\rho^t \models \psi$ iff $\rho^{t'} \models \psi$ for each subformula ψ of ϕ. Moreover, if all interval end-points in τ are rational numbers, then so are all interval end-points in τ_ϕ.*

For any MITL-formula ϕ, we say that the timed state sequence $\rho = (\sigma, \tau_\phi)$ is *ϕ-fine*. Clearly, ϕ is satisfiable iff it has a ϕ-fine model.

2.4 Defined operators

Now let us introduce some standard abbreviations for additional temporal operators. The defined operators $\Diamond_I \phi$ (constrained *eventually*) and $\Box_I \phi$ (constrained *always*) stand for $true\, \mathcal{U}_I \phi$ and $\neg\Diamond_I \neg\phi$, respectively. It follows that the formula $\Box_I \phi$ (or $\Diamond_I \phi$) holds at time $t \in \mathsf{R}^+$ of a timed state sequence iff ϕ holds at all times (at some time, respectively) within the interval $t + I$.

We usually suppress the interval $(0, \infty)$ as a subscript. Thus the MITL-operators $\Diamond$, $\Box$, and $\mathcal{U}$ coincide with the conventional unconstrained *strict eventually*, *strict always*, and *strict until* operators of temporal logic. This is because the *until* operator of MITL is implicitly strict in its first argument. The corresponding non-strict operators are definable in MITL as $\Diamond_{[0,\infty)}$ (also written $\Diamond_{\geq 0}$), $\Box_{\geq 0}$, and

$$\phi_2 \vee (\phi_1 \wedge \phi_1 \mathcal{U} \phi_2)$$

for $\phi_2 \mathcal{U}^{=} \phi_1$ (where $\mathcal{U}^{=}$ denotes the unconstrained non-strict *until* operator). Note that, on the other hand, the operator $\mathcal{U}_I$ cannot be defined in terms of an *until* operator that is not strict in its first argument; this is why we have chosen the *strict* versions of temporal operators to be primitive.

Using these abbreviations, the typical bounded response property that "every p-state is followed by a q-state within 5 time units," can be expressed by the MITL-formula

$$\Box_{\geq 0}(p \rightarrow \Diamond_{(0,5]} q).$$

We also define a constrained *unless* operator as the dual of the *until* operator:

$$\phi_1 \, {}_I\mathsf{U} \, \phi_2 \text{ stands for } \neg((\neg\phi_2)\mathcal{U}_I(\neg\phi_1)).$$

It follows that the formula $\phi_1 \, {}_I\mathsf{U} \, \phi_2$ holds at time $t \in \mathsf{R}^+$ of a timed state sequence iff either ϕ_1 is true throughout the interval $t + I$, or there is a time instant $t' > t$ such that ϕ_2 is true at time t' and ϕ_1 holds at all instants $t'' \leq t'$ within the interval $t + I$. Note that the unconstrained version $\phi_1 \, \mathsf{U} \, \phi_2$ of the *unless* operator of MITL differs slightly from the conventional strict *unless* operator, which can be defined as $\phi_1 \, \mathsf{U} \, (\phi_1 \wedge \phi_2)$.

We can apply the definition of the *unless* operator to move negations through *until* operators. Thus we may obtain, from any MITL-formula, an equivalent formula, containing both *until* and *unless* operators, in which all negations are in front of atomic propositions.

2.5 Avoiding undecidability

A few comments on our choice of syntax are in order. First, MITL has no *next-time* operator, because due to the density of the time domain there is no unique next time. Also, MITL is, syntactically viewed, essentially the restriction of *metric temporal logic* (MTL [AH90]) that prohibits the use of equality in time bounds. For example, in MITL we cannot directly express the punctuality condition that "every p-state is followed by a q-state after exactly 5 time units,"

$$\Box_{\geq 0}(p \rightarrow \Diamond_{=5} q),$$

because the singular interval $[5, 5]$ is not allowed as a subscript. We will show that there is, in fact, no MITL-formula that expresses this condition, and that the restriction of MITL to *nonsingular* intervals is essential for decidability.

Note that some practically important forms of equality are expressible in MITL; we define $(\neg\phi)\mathcal{U}_{=n}\,\phi$, for $n > 0$, as an abbreviation for $\Box_{(0,n)}\,\neg\phi \wedge \Diamond_{(0,n]}\,\phi$. Thus the stronger condition that "for every p-state the closest subsequent q-state is after exactly 5 time units,"

$$\Box_{\geq 0}(p \rightarrow (\neg q)\mathcal{U}_{=5}\, q),$$

is expressible in MITL.

Let $\text{MITL}_{=}$ be the extension of MITL that admits singular intervals as time bounds on the temporal operators. We show that the decision problem of $\text{MITL}_{=}$ is complete for the complexity class Π^1_1, which is situated in the analytical hierarchy strictly above all recursively enumerable sets (see, for example, [Rog67]). It follows that $\text{MITL}_{=}$ is not even axiomatizable.

Theorem 2.1 (MITL **with equality**)
The decision problem for $\text{MITL}_{=}$ *is* Π^1_1*-complete.*

Another possible extension of the syntax of MITL is to permit time bounds on *both* arguments of the *until* operator, as is the case for all logics that admit explicit references to time in atomic formulas (such as TPTL [AH89]). The intended meaning of the formula $\phi_1 \, {}_I\mathcal{U}_J \, \phi_2$ at time $t \in \mathsf{R}^+$ of a timed state sequence is that there is a later time instant $t' \in t + I$ such that ϕ_2 holds at time t' and ϕ_1 holds throughout the time interval $(t + I') \cap [t, t']$. Such an extension leads, however, again to undecidability. This is because the role of $\Diamond_{=n}\,\phi$ in the undecidability argument for $\text{MITL}_{=}$ can be replaced by the formula $\mathit{false}\,{}_{\geq n}\mathcal{U}_{\geq n}\,\phi$.

2.6 Real versus rational time

Having justified our choice of syntax, let us look at other options for defining the semantics of MITL. While timed state sequences are defined by choosing the set of (nonnegative) reals to model time, for interpreting formulas of MITL, the crucial property of the time domain R^+ is not its continuity, but only its denseness. In particular, we show that replacing the time domain R^+ with the nonnegative rational numbers Q^+ when defining the semantics of MITL does not change the satisfiability (and validity) of any MITL-formula.

We call a timed state sequence (σ, τ) *rational* iff the end-points of all intervals in τ are rational. A formula ϕ of $\text{MITL}_{=}$ is said to be Q*-satisfiable* iff $\rho \models \phi$ for some rational timed state sequence ρ, where the satisfaction relation $\models$ is redefined so that all time quantifiers range over Q^+ only.

We show that this new notion of satisfiability is the same as the old one. In other words, MITL-formulas cannot distinguish the time domain R^+ from the time domain Q^+. This equivalence of real and rational models follows from the following two lemmas.

Lemma 2.2 (Rational models)
Let ϕ be an MITL*-formula and ρ a rational ϕ-fine timed state sequence. Then ρ Q-satisfies ϕ iff $\rho \models \phi$.*

For any MITL-formula ϕ, let n_ϕ be the least common denominator of all (rational) interval end-points in ϕ; that is, all constants in ϕ are multiples of $1/n_\phi$.

Lemma 2.3 (Model equivalence)
Let $\rho = (\sigma, \tau)$ and $\rho' = (\sigma, \tau')$ be two timed state sequences, and ϕ be a formula of $\mathrm{MITL}_=$*. Suppose that for all $t \in \mathsf{R}^+$, if $t = t_i + m/n_\phi$ for some left end-point t_i of an interval in τ and some nonnegative integer $m \in \mathsf{N}$, then $t \in I_j$ iff $t \in I'_j$. Then $\rho \models \phi$ iff $\rho' \models \phi$.*

Lemma 2.3 classifies timed state sequences into equivalence classes such that the members of a class cannot be distinguished by formulas of $\mathrm{MITL}_=$. Together, the Lemmas 2.2 and 2.3 imply the following theorem:

Theorem 2.2 (Rational time)
A formula ϕ of $\mathrm{MITL}_=$ *is* Q*-satisfiable iff it is satisfiable.*

2.7 Expressive power of MITL

We define the semantics of a system as a set of timed state sequences; such a set is called a *real-time property*. Every formula ϕ of a real-time logic (say, MITL) specifies a real-time property — the set of models of ϕ. The expressive power of a logic is measured by the real-time properties that can be specified by formulas of the logic.

We compare the expressive power of MITL to the use of a digital clock and MTL, which admits singular intervals as time bounds on temporal operators. More precisely, we show that the analog-clock model without equality (MITL) is more expressive than any digital-clock model with equality (MTL).

First let us review the definition of the logic MTL [AH90]. The syntax of MTL is the same as that of $\mathrm{MITL}_=$. The formulas of MTL are interpreted over observation sequences. An *observation sequence* ϱ is an infinite sequence

$$(s_0, T_0) \rightarrow (s_1, T_1) \rightarrow (s_2, T_2) \rightarrow (s_3, T_3) \rightarrow \cdots$$

of observations. Each *observation* consists of a state $s_i \in 2^P$ and a time stamp $T_i \in \mathsf{N}$. The observation sequence ϱ satisfies the *initiality* condition that $T_0 = 0$, the *monotonicity* condition that $T_i \leq T_{i+1}$ for all $i \geq 0$, and the *progress* condition that, for all $n \in \mathsf{N}$, there is some $i \geq 0$ such that $T_i > n$.

For an observation sequence ϱ and an MTL-formula ϕ, the satisfaction relation $\varrho \models \phi$ is defined as usual by induction on the structure of ϕ. The following clause considers the case of the (strict) *until* operator:

$$\varrho \models \phi_1 \,\mathcal{U}_I\, \phi_2 \text{ iff } \varrho^i \models \phi_2 \text{ for some } i \geq 0, \text{ and } \varrho^j \models \phi_1 \text{ for all } 0 < j < i.$$

(For an observation sequence ϱ and $i \in \mathsf{N}$, the observation sequence ϱ^i is the suffix of ϱ that begins with the observation (s_i, T_i).) We consider only the fragment of MTL without the *next-state* operator; this restriction makes MTL-formulas insensitive to stuttering.

We need to formalize which real-time properties can be specified in MTL. To this end, let us consider how to extract an observation sequence from a timed state sequence ρ that describes the actual behavior of a real-time system. Observations are made with respect to a digital clock; the observation at time t records the state $\rho^*(t)$ and the value of the clock at time t. Clearly the observations depend on how fast the clock ticks, and at what time the clock is started.

Consequently, we define a *digital clock* $D = (\delta, \epsilon)$ to be a pair consisting of the distance $\delta \in \mathsf{R}^+$ between two successive clock ticks and the time $\epsilon \in \mathsf{R}^+$ of the first clock tick; that is, $0 \leq \epsilon < \delta$. At time $t \in \mathsf{R}^+$ the clock D shows the integer value $t_D = \lceil (t - \epsilon)/\delta \rceil$. The clock D is called *rational* iff both δ and ϵ are rational numbers.

The D-observation of the timed state sequence ρ at time t is $O_t = (\rho^*(t), t_D)$. As time increases, the D-observation stays the same until either the clock ticks or the state changes along ρ. All possible D-observations along ρ can be described by an ω-sequence: the D-observed behavior of ρ is the observation sequence

$$\rho_D : \quad O_{t_0} \rightarrow O_{t_1} \rightarrow O_{t_2} \rightarrow \cdots,$$

such that for all $i \geq 0$, (1) $t_i < t_{i+1}$, and (2) for all $t \in (t_i, t_{i+1})$, O_t equals either O_{t_i} or $O_{t_{i+1}}$. These properties define ρ_D uniquely modulo stuttering (i.e., duplication of neighboring observations). Furthermore, the state component of ρ_D is the state component of ρ (modulo stuttering) with, if ρ is finite, infinite repetition of the final state.

For instance, consider the timed state sequence ρ:

$$(s_0, [0, 1)) \rightarrow (s_1, [1, 1]) \rightarrow (s_2, (1, 1.5]) \rightarrow (s_3, (1.5, \infty)).$$

Then the digital clock $(1, 0.5)$ observes the observation sequence $\rho_{(1,0.5)}$:

$$(s_0,0) \to (s_0,1) \to (s_1,1) \to (s_2,1) \to (s_3,2) \to (s_3,3) \to (s_3,4) \to \cdots$$

For every digital clock D, every formula ϕ of MTL specifies a real-time property Π_ϕ^D — the set of timed state sequences ρ such that $\rho_D \models \phi$. We say that the MTL-formula ϕ *D-specifies* the real-time property Π_ϕ^D.

Now we can be specific about the sense in which the analog-clock model is, even without equality, more expressive than the digital-clock model, for any choice of digital clock. In particular, we can construct a satisfiable MITL-formula for which every model contains arbitrarily close state changes to show the following theorem:

Theorem 2.3 (Expressiveness of MITL)
(a) Every real-time property that can be D-specified by an MTL*-formula for some rational digital clock D, can also be specified in* MITL. *(b) There is a real-time property that can be specified in* MITL *but not D-specified by any* MTL*-formula for any digital clock D.*

3 Timed Automata

We use a variant of timed automata defined in [AD90] to model finite-state real-time systems. This formalism is a generalization of (nondeterministic) finite-state machines over infinite strings. While ω-automata generate (or accept) infinite sequences of states [Tho90], timed automata are additionally constrained by timing requirements and produce *timed* state sequences.

A timed automaton operates with finite control — a finite set of states and a finite set of real-valued clocks. All clocks proceed at the same rate and measure the amount of time that has elapsed since they were started (or reset). Each transition of the automaton may reset some of the clocks; each state of the automaton puts certain constraints on the values of the atomic propositions as well as on the values of the clocks: the control of the automaton can reside in a particular state only if the values of the propositions and clocks satisfy the corresponding constraints.

We permit only simple constraints on the clock values. A *clock constraint* $\mathcal{I} \subseteq \mathsf{R}^+$ is a finite union of (possibly unbounded) intervals with rational end-points; the value $\gamma(x) \in \mathsf{R}^+$ of a clock x satisfies the constraint $\mathcal{I}$ iff $\gamma(c) \in \mathcal{I}$. We usually denote the clock constraints for a clock x as boolean combination of arithmetic expressions containing x; for instance,

$$1 \le x < 3 \vee x = 4 \vee x > 5$$

stands for the clock constraint $[1,3) \cup [4,4] \cup (5,\infty)$ that restricts the value of x. Let $\mathcal{R}$ be the set of clock constraints.

Formally, a *timed automaton* is a six-tuple $\mathcal{M} = \langle S, C, \mu, \nu, S_0, E \rangle$, where

- S is a finite set of states,
- C is a finite set of clocks,
- $\mu\colon S \to 2^P$ assigns to each state and proposition a truth value,
- $\nu\colon S \to \mathcal{R}^C$ assigns to each state and clock a clock constraint,
- $S_0 \subseteq S$ is a set of initial states,
- $E \subseteq S^2 \times 2^C$ is a set of transitions. Each transition $\langle s, s', \lambda \rangle$ identifies a source state s, a target state s', and a set $\lambda \subseteq C$ of clocks to be reset; we usually denote this transition by $s \xrightarrow{\lambda} s'$.

The runs of a timed automaton define timed state sequences. At any time instant during a run, the configuration of the automaton is completely determined by the state in which the control resides and the values of all clocks. The values of all clocks are given by a *clock interpretation* γ, which is a map from C to R^+: for any clock $x \in C$, the value of x under the interpretation γ is $\gamma(x) \in \mathsf{R}^+$.

Assume that, at time $t \in \mathsf{R}^+$, a timed automaton is in state s and the clock values are given by the clock interpretation γ. Suppose that the state of the automaton remains unchanged during the time interval I with $l(I) = t$. All clocks proceed at the same rate as time elapses; at any time $t' \in I$ the value of any clock x is $\gamma(x) + t' - t$. During all this time the value of x satisfies the clock constraint that is associated with s and x:

$$(\gamma(x) + t' - t) \in \nu(s, x).$$

Now suppose that the automaton changes its state at time $r(I) = t''$ via the transition $s \xrightarrow{\lambda} s'$. This state change happens in one of two ways. If I is right-closed, then the state at time t'' is still s and

$$(\gamma(x) + t'' - t) \in \nu(s, x)$$

for all clocks x; otherwise the state at time t'' is s' and $0 \in \nu(s', x)$ for all clocks $x \in \lambda$, which are reset, and

$$(\gamma(x) + t'' - t) \in \nu(s', x)$$

for all other clocks.

Let us formalize this intuition. A *run* of a timed automaton $\mathcal{M} = \langle S, C, \mu, \nu, S_0, E \rangle$, is a finite or infinite sequence

$$r\colon \quad \underset{\gamma_0}{\longmapsto} (s_0, I_0) \xrightarrow[\gamma_1]{\lambda_1} (s_1, I_1) \xrightarrow[\gamma_2]{\lambda_2} (s_2, I_2) \xrightarrow[\gamma_3]{\lambda_3} \cdots$$

of states $s_i \in S$, intervals I_i, clock sets $\lambda_i \subseteq C$, and clock interpretations $\gamma_i\colon C \to \mathsf{R}^+$ such that

- $s_0 \in S_0$,
- $\langle s_i, s_{i+1}, \lambda_i \rangle \in E$ for all $i \geq 0$,
- $I_0 I_1 I_2 \ldots$ is an interval sequence,
- for all $x \in C$ and $i \geq 0$, $\gamma_{i+1}(x) = 0$ if $x \in \lambda_{i+1}$, and $\gamma_{i+1}(x) = \gamma_i(x) + r(I_i) - l(I_i)$ otherwise.
- $(\gamma_i(x) + t - l(I_i)) \in \nu(s_i, x)$ for all $x \in C$, $i \geq 0$, and $t \in I_i$.

Note that, according to this definition, the clocks may start at any real values that satisfy the clock constraints of an initial state.

The run r uniquely determines the timed state sequence

$$\rho_r: \quad (\mu(s_0), I_0) \rightarrow (\mu(s_1), I_1) \rightarrow (\mu(s_2), I_2) \rightarrow \cdots$$

By $\Pi(\mathcal{M})$ we denote the set of all timed state sequences ρ_r that correspond to runs of the timed automaton $\mathcal{M}$. We say that $\mathcal{M}$ *generates* (or *accepts*) the timed state sequences in $\Pi(\mathcal{M})$.

We will use timed automata to model real-time systems. A real-time system is represented by the timed automaton $\mathcal{M}$ iff its possible behaviors are exactly the timed state sequences in $\Pi(\mathcal{M})$. Accordingly, the system modeled by $\mathcal{M}$ satisfies its MITL-specification ϕ, denoted by $\mathcal{M} \models \phi$, iff $\rho_r \models \phi$ for all runs r of $\mathcal{M}$.

We point out that a run may contain transient states. Such states allow us to model instantaneous conditions during the execution of a real-time system, like the occurrence of events. Their times can be enforced accurately by using singular intervals as clock constraints.

The emptiness problem for timed automata is solved in [AD90]: the problem of whether a timed automaton has any run is PSPACE-complete. Our definition of timed automata is somewhat more general than the one in [AD90]; it can also enforce transient states. But the decision procedure for checking emptiness can be easily adapted to prove the following result:

Theorem 3.1 (Emptiness of timed automata) *The problem of deciding if $\Pi(\mathcal{M}) = \emptyset$ for a timed automaton $\mathcal{M} = \langle S, C, \mu, \nu, S_0, E \rangle$ is PSPACE-complete. There is an algorithm that decides this problem in time $O((|S| + |E|) \cdot 2^{|\nu|})$.*

To enforce *fairness* constraints on the legal behaviors of a real-time system, we add standard liveness conditions to timed automata, such as *Büchi* acceptance criteria or *Muller* acceptance criteria for ω-automata (see [AD90] for details). Theorem 3.1 carries over to either case.

4 Deciding MITL

We solve the satisfiability problem for MITL by reducing it to the emptiness problem for timed automata. Our main result is that, given an MITL-formula ϕ, we can construct a timed automaton $\mathcal{M}_\phi$ such that the runs of $\mathcal{M}_\phi$ that meet certain fairness requirements correspond precisely to the timed state sequences that satisfy ϕ.

4.1 Restricting the problem

To simplify the exposition of the decision procedure, we restrict the satisfiability question for MITL to formulas and models of a specific form and show that this can be done without loss of generality.

Given an MITL-formula ϕ, a timed state sequence ρ, and a constant $a \in \mathbb{Q}$, let $a\phi$ and $a\rho$ be the MITL-formula and the timed state sequence that result from ϕ and ρ, respectively, by replacing each interval I by the interval aI. Clearly, $\rho \models \phi$ iff $a\rho \models a\phi$. Thus, for the purpose of checking the satisfiability of ϕ, we may assume that all interval end-points in ϕ are integers; for if they are not, then consider $n_\phi \phi$ for the least common denominator n_ϕ of all (rational) interval end-points in ϕ. This translation causes at most a quadratic blow-up in the size of the formula.

Next we give a series of transformations that allow us to rewrite any formula ϕ into an equivalent formula ϕ^* that contains only temporal operators of very specific forms.

First, we require that no interval in ϕ contains 0. This can be achieved by applying the following equivalence:

$$\psi_1 \mathcal{U}_I \psi_2 \leftrightarrow (\psi_2 \vee \psi_1 \mathcal{U}_{I \cap (0,\infty)} \psi_2)$$

provided that $0 \in I$.

Secondly, we require that the only unbounded intervals in ϕ are of the form $(0, \infty)$. This can be achieved by applying the following two equivalences:

$$\psi_1 \mathcal{U}_{(n,\infty)} \psi_2 \leftrightarrow \Box_{(0,n]} (\psi_1 \wedge \psi_1 \mathcal{U} \psi_2)$$

$$\psi_1 \mathcal{U}_{[n,\infty)} \psi_2 \leftrightarrow \begin{pmatrix} \Box_{(0,n)} \psi_1 \wedge \\ \Box_{(0,n]} (\psi_2 \vee (\psi_1 \wedge \psi_1 \mathcal{U} \psi_2)) \end{pmatrix}$$

provided that $n > 0$.

Thirdly, we require that only the *eventually* and the *always* operators are constrained with bounded intervals I such that $l(I) = 0$. This can be achieved by applying the following equivalence:

$$\psi_1 \mathcal{U}_I \psi_2 \leftrightarrow \Diamond_I \psi_2 \wedge \psi_1 \mathcal{U} \psi_2$$

provided that $l(I) = 0$.

Finally, we push all negations in ϕ to the inside and use the following equivalence to eliminate each subformula of the form $\psi_1 \mathsf{U} \psi_2$:

$$\psi_1 \mathsf{U} \psi_2 \leftrightarrow \Box \psi_1 \vee \psi_1 \mathcal{U} (\psi_1 \wedge \psi_2).$$

The resulting formula ϕ^* is equivalent to ϕ and consists of atomic propositions, negated atomic propositions, conjunctions, disjunctions, and temporal subformulas ψ of the following six types:

1. $\psi_1 \mathcal{U}_I \psi_2$ with bounded I and $l(I) > 0$.
2. $\psi_1 {}_I\mathsf{U} \psi_2$ with bounded I and $l(I) > 0$.
3. $\Diamond_I \psi'$ with $I = (0, n)$ or $I = (0, n]$.
4. $\Box_I \psi'$ with $I = (0, n)$ or $I = (0, n]$.
5. $\psi_1 \mathcal{U} \psi_2$.
6. $\Box \psi'$.

Although these rewritings blow up the size of the formula ϕ, we can bound the size of the constants in ϕ^* and the number of subformulas in ϕ^* as follows:

- Let $K \in \mathsf{N}$ be such that $K - 1$ is the largest (integer) constant appearing as an interval end-point in ϕ. Then the largest constant that occurs as an end-point of an interval in ϕ^* is $K - 1$.
- Let $N \in \mathsf{N}$ be the number of atomic propositions, boolean connectives, and temporal operators in ϕ. Then the number of syntactic subformulas of ϕ^* is $O(N)$.

Thus we restrict ourselves to test the satisfiability of MITL-formulas each of whose temporal subformulas are, according to the above classification, of one of six types, *type-1* to *type-6*.

Moreover, to check the satisfiability of an MITL-formula ϕ, by Lemma 2.1 we can confine ourselves to the question if ϕ has a ϕ-fine model. Therefore we consider, throughout this section, only ϕ-fine timed state sequences $\rho = (\sigma, \tau)$. It follows that, if ψ is a subformula of ϕ, we may write $\rho^i \models \psi$ for "$\rho^t \models \psi$ for all $t \in I_i$." In addition, we assume that all intervals in τ are either singular or open. This is sufficient, because any model of ϕ can be brought into this form by splitting all nonsingular closed intervals; for instance, the interval $[a, b)$ can be split into the two intervals $[a, a]$ and (a, b). Clearly, singular and open intervals must alternate in any interval sequence.

The different types of temporal subformulas of ϕ are handled differently by our algorithm. The simplest case is that of type-5 and type-6 formulas; they are treated essentially in the same way in which tableau decision procedures for linear temporal logic handle unconstrained temporal operators. The most interesting case is that of type-1 and type-2 formulas. We concentrate first on this case. The case of type-3 and type-4 formulas will be considered later.

4.2 Outline of the algorithm

Consider the MITL-formula

$$\Box_{[0,1)} (p \rightarrow \Diamond_{[1,2]} q).$$

Let us assume that both p and q are true only in singular intervals and let us try to build a timed automaton that accepts precisely the models of this formula.

Whenever the automaton visits a p-state, it needs to make sure that within 1 to 2 time units a q-state is visited. This can be done by setting a clock x to 0 when the p-state is visited, and demanding that some q-state with the clock constraint $1 \leq x \leq 2$ is visited later. This strategy requires a clock per visit to a p-state within the interval $[0, 1)$. However, the number of such visits is potentially unbounded and, hence, any automaton with a fixed number of clocks cannot reset a new clock for every visit. That is why this simple strategy cannot be made to work.

An alternative approach is to guess the times for future q-states in advance. The automaton nondeterministically guesses two time values t_1 and t_2 within the interval $[0, 1)$; this is done by resetting a clock x at time t_1 and another clock y at time t_2. The guess is that the *last* q-state within the interval $[1, 2)$ is at time $t_1 + 1$, and that the *first* q-state within the interval $[2, 3)$ is at time $t_2 + 2$. If the guesses are correct, then the formula $\Diamond_{[1,2]} q$ holds during the intervals $[0, t_1]$ and $[t_2, 1)$, and does not hold during the interval (t_1, t_2). Consequently, the automaton requires that every p-state within the interval $[0, 1)$ lies either within $[0, t_1]$ or within $[t_2, 1)$. It also needs to make sure that the guesses are right; that is, whenever either $x = 1$ or $y = 2$, the automaton must be in a q-state. This strategy requires only two clocks for the interval $[0, 1)$ of length 1, irrespective of the number of p-states within $[0, 1)$.

We say that the guessed times t_1+1 and t_2+2 witness the formula $\Diamond_{[1,2]} q$ throughout the intervals $[0, t_1]$ and $[t_2, 1)$, respectively. In general, the witnesses need not be singular intervals, they can be open intervals. In the following we develop an algorithm based on this idea of guessing, in advance, time intervals that witness temporal formulas and, later, checking the correctness of these guesses. The crucial fact that makes this strategy work, with a finite number of clocks, is that the *same* interval may serve as a witness for many points in time.

4.3 Witnessing intervals

The interval I' is called a *witnessing interval* for the MITL-formula $\psi_1 \mathcal{U}_I \psi_2$ under ρ^t, for a timed state sequence ρ and $t \in \mathsf{R}^+$, iff $I' \cap (t + I) \neq \emptyset$ and $\rho^t \models \psi_1 \mathcal{U}_{J-t} \psi_2$ for every nonempty interval $J \subseteq I'$. Observe that if I' witnesses $\psi_1 \mathcal{U}_I \psi_2$ under ρ^t, then $\rho^{t'} \models \psi_1$ for all $t < t' < r(I')$ and $\rho^{t'} \models \psi_2$ for all $t' \in I'$. The interval I' is a witnessing interval for the MITL-formula $\psi_1 \, {}_I\mathsf{U} \, \psi_2$ under ρ^t iff $t + I \subseteq I'$ and $\rho^t \models \psi_1 \, {}_{I'-t}\mathsf{U} \, \psi_2$.

Witnessing intervals are defined such that the following property holds:

Lemma 4.1 (Witnessing intervals)
Let ψ be an MITL*-formula of the form $\psi_1 \mathcal{U}_I \psi_2$ or $\psi_1 \, {}_I\mathsf{U} \, \psi_2$, let ρ be a timed state sequence and $t \in \mathsf{R}^+$. There is a witnessing interval for ψ under ρ^t iff $\rho^t \models \psi$.*

Now we show that the same interval may serve as a witnessing interval for a temporal formula under (infinitely) many suffixes of a timed state sequence.

Consider, for example, the timed state sequence ρ over two propositions p and q:

$$(\{p\}, [0, 1.2]) \rightarrow (\{p, q\}, (1.2, 1.6)) \rightarrow (\{p\}, [1.6, \infty)).$$

Thus along ρ the proposition p is always true, but the proposition q is true only during the interval $I_q = (1.2, 1.6)$. The interval I_q witnesses the formula $p \mathcal{U}_{(1,2)} q$ under ρ^t for every $t \in [0, 0.6)$. On the other hand, the interval $[1.6, 3]$ witnesses the formula $\Box_{(1,2)}(\neg q)$ under ρ^t for every $t \in [0.6, 1]$.

Lemma 4.2 (Sharing type-1 witnesses)
Let ψ be the type-1 formula $\psi_1 \mathcal{U}_I \psi_2$. For every timed state sequence ρ, there are two bounded, open or closed, intervals I_1 and I_2 such that, for every $t \in [0, 1)$, the formula ψ is satisfied by ρ^t iff either I_1 or I_2 witnesses ψ under ρ^t. Furthermore, $r(I_i) \leq r(I) + 1$ for $i = 1, 2$.

Proof of Lemma 4.2 Let $\rho = (\sigma, \tau)$ be a ψ-fine timed state sequence with only singular and open intervals, including the singular interval $[r(I) + 1, r(I) + 1]$ (split intervals if necessary). We choose two witnessing intervals I_1 and I_2 as follows:

- Let i be the *maximal* $i \geq 0$ such that $I_i \cap I \neq \emptyset$, and $\rho^i \models \psi_2$, and $\rho^i \models \psi_1$ if I_i is open, and $\rho^k \models \psi_1$ for all $0 \leq k < i$ with $I_k \cap I \neq \emptyset$. If no such i exists, let $I_1 = \emptyset$; otherwise, let $I_1 = I_i$.
- Let j be the *minimal* $j \geq 0$ such that $I_j \cap (I + 1) \neq \emptyset$, and $\rho^j \models \psi_2$, and $\rho^j \models \psi_1$ if I_j is open, and $\rho^k \models \psi_1$ for all $0 \leq k < j$ with $I_k \cap (I \cup I + 1) \neq \emptyset$. If no such j exists, let $I_2 = \emptyset$; otherwise, let $I_2 = I_j$.

■

In the case of type-2 formulas, a single witness per unit interval suffices to reduce the problem to type 3:

Lemma 4.3 (Sharing type-2 witnesses)
Let ψ be the type-2 formula $\psi_1 \, {}_I\mathsf{U} \, \psi_2$. For every timed state sequence ρ, there are is a bounded interval I' such that, for every $t \in [0, 1)$, the formula ψ is satisfied by ρ^t iff either ρ^t satisfies the type-3 formula $\Diamond_{(0,\infty) \cap (<I)} \psi_2$ or I' witnesses ψ under ρ^t. Furthermore, $r(I') \leq r(I) + 1$.

4.4 Type-1 and type-2 formulas

Now we can be more precise about how we will construct the timed automaton $\mathcal{M}_\phi$ that accepts exactly the models of ϕ. To check the truth of type-1 and type-2 subformulas of ϕ, the automaton guesses corresponding witnessing intervals. The boundaries of a witnessing interval are marked by clocks: a *clock interval* is a bounded interval that is defined by its *type* (e.g., left-closed and right-open) and a pair of clocks. Given a time t and a clock interpretation γ, the clock interval $C = [x, y]$, for two clocks x and y, stands for the closed witnessing interval $[t + K - \gamma(x), t + K - \gamma(y)]$; the clock interval $C = [x, y)$ stands for the corresponding half-open interval, etc. We write $K - C$ for the interval $\{K - \gamma(x), K - \gamma(y)\}$, for any type of clock interval $C = \{x, y\}$.

For simplicity, let us consider a type-1 subformula ψ of the form $\Diamond_I \psi'$. The automaton resets, nondeterministically, any of its clocks at any time. When guessing a witnessing interval I', it writes the prediction that "the clock interval $C = \{x, y\}$ witnesses the formula ψ" into its memory. If the clock x was reset at time t_1, and y was reset at time $t_2 \geq t_1$, then the witnessing interval guessed is $I' = \{t_1 + K, t_2 + K\}$. To check the the truth of the temporal formula ψ at time $t \geq t_2$, the automaton needs to verify that its guess I' is indeed a witness. The condition $I' \cap (t + I) \neq \emptyset$ translates to verifying the clock constraint $(K - C) \cap I \neq \emptyset$. It remains to be checked that ψ' is satisfied throughout the witnessing interval I'; that is, the automaton needs to verify that ψ' holds at all states with the clock constraint $0 \in (K - C)$.

The Lemmas 4.2 and 4.3 are the key to constructing an automaton that needs only *finitely* many clocks. For the type-1 formula $\psi_1 \mathcal{U}_I \psi_2$, at most 2 witnessing intervals need to be guessed per interval of unit length. Furthermore, the fact that the right end-point of a witnessing interval is bounded allows the automaton to reuse every clock after a period of length $r(I) + 1$. Thus we need, at any point in time, at most $2r(I) + 2$ *active* clock intervals; that is, clock intervals that stand for a guess of a witnessing interval and, therefore, have to be verified later. Similarly, to check a type-2 formula $\psi_1 \, {}_I\mathsf{U} \, \psi_2$, we need, at any point in time, no more than $r(I) + 1$ active clock intervals. Consequently, $4K$

clocks suffice to check any type-1 subformula of ϕ, and $2K$ clocks suffice for any type-2 subformula of ϕ.

4.5 Type-3 and type-4 formulas

Now let us move to formulas of the form $\Diamond_I \psi'$ and $\Box_I \psi'$ with $I = (0, n)$ or $I = (0, n]$. Checking the truth of such a formula is much easier and can be done using a single clock.

Consider the type-3 formula $\psi = \Diamond_I \psi'$. Whenever the automaton needs to check that ψ holds, say at time t, it starts a clock x and writes the corresponding proof obligation into its memory — to verify that ψ' holds at some later state with the clock constraint $x \in I$. The obligation is discharged as soon as an appropriate ψ'-state is found. If the automaton encounters another ψ-state in the meantime, at time $t' > t$ before the obligation is discharged, it does not need to check the truth of ψ separately for this state. This is because if there is a ψ'-state after time t' within the interval $t + I$, then both $\rho^t \models \Diamond_I \psi'$ and $\rho^{t'} \models \Diamond_I \psi'$. Once the proof obligation is discharged, the clock x can be used again. Thus one clock suffices to check the formula ψ as often as necessary.

The described strategy works for checking the truth of ψ at singular intervals. There is, however, a subtle problem with this method when the truth of ψ during open intervals needs to be checked, as is illustrated by the following example. Consider the timed state sequence

$$(\{\}, [0, 0]) \rightarrow (\{\}, (0, 1)) \rightarrow (\{p\}, [1, \infty]);$$

it satisfies the formula $\Diamond_{(0,1)} p$ at all times $t \in (0, 1)$. To check the truth of $\Diamond_{(0.1)} p$ during the open interval $(0, 1)$, the automaton starts a clock x upon entry, at time 0. However, the proof obligation that p holds at some later state with the clock constraint $x \in I$ can never be verified. On the other hand, if the automaton were to check, instead, the truth of the formula $\Diamond_{(0,1]} p$ during the interval $(1, 0)$, then our strategy works and the corresponding proof obligation can be verified, because there is a p-state while $x \in (0, 1]$ holds. Furthermore, observe that the validity of $\Diamond_{(0,1]} p$ throughout the open interval $(0, 1)$ implies that $\Diamond_{(0,1)} p$ is also true throughout $(0, 1)$.

In general, the following lemma holds:

Lemma 4.4 (Weakening type-3 formulas) *Let ψ and $\hat{\psi}$ be the type-3 formulas $\Diamond_I \psi'$ and $\Diamond_{I \cup \{r(I)\}} \psi'$, respectively. For every timed state sequence $\rho = (\sigma, \tau)$ and open interval I_i in τ, $\rho^i \models \psi$ iff $\rho^i \models \hat{\psi}$.*

Consequently, to check the truth of a type-3 formula ψ during an open interval, it suffices to check the truth of the weaker formula $\hat{\psi}$. Accordingly, the automaton we construct writes only the proof obligation that corresponds to checking $\hat{\psi}$ into its memory.

For checking a type-4 formula of the form $\psi = \Box_I \psi'$, the situation is symmetric. The automaton uses also a single clock x to check this formula. Whenever the formula ψ needs to be verified, say at time t, the automaton starts the clock x with the proof obligation that as long as the clock constraint $x \in I$ holds, so does ψ'. The obligation is discharged as soon as $x > I$. If the automaton encounters another ψ-state within the interval $t + I$, say at time t', it simply resets the clock x, and thus overwrites the previous proof obligation. This strategy is justified by the observation that if ψ' holds throughout the interval $(t, t']$ and $\rho^{t'} \models \Box_I \psi'$, then also $\rho^t \models \Box_I \psi'$. Once the proof obligation is discharged, the clock x can be reused to check ψ again whenever necessary.

As in the case of type-3 formulas, we need to be more careful when checking ψ during open intervals. For the type-4 formula $\psi = \Box_I \psi'$, let $\hat{\psi}$ be the formula $\Box_{I - \{r(I)\}} \psi'$. From Lemma 4.4 and duality, it follows that for every timed state sequence $\rho = (\sigma, \tau)$, if I_i is open, then $\rho^i \models \psi$ iff $\rho^i \models \hat{\psi}$. Hence to check the truth of ψ during an open interval, it suffices again to check the truth of the weaker formula $\hat{\psi}$. Accordingly, only a proof obligation for $\hat{\psi}$ is set up. This is because the corresponding clock x is started at time $r(I_i)$, and for ψ to hold during the open interval I_i, ψ' need not hold at time $r(I_i) + r(I)$, even if I is right-closed.

4.6 Constructing the timed automaton

Now let us define the timed automaton $\mathcal{M}_\phi$ formally. For each temporal subformula of ϕ of type-1, the automaton $\mathcal{M}_\phi$ has $2K$ pairs of clocks. These clocks always appear in pairs, to form clock intervals. From any pair of clocks x and y, four different clock intervals can be formed: (x, y), $[x, y)$, $(x, y]$, and $[x, y]$. From Lemma 4.2 for checking type-1 formulas we need only open or closed witnessing intervals. Thus associated with each type-1 subformula ψ of ϕ we have $4K$ clock intervals; they are denoted by $C_1(\psi), \ldots C_{4K}(\psi)$. For each type-2 subformula of ϕ the automaton uses K clock pairs giving $4K$ clock intervals. For subformulas ψ of types 3 and 4, the automaton needs one clock x_ψ per formula.

In addition to these clocks, we use the clock x_{sing} to enforce that the runs of $\mathcal{M}_\phi$ have alternate singular and open intervals.

Given the MITL-formula ϕ, we define its *closure* set $Closure(\phi)$ to consist of the following items:

1. All subformulas of ϕ.

2. For each type-2 formula $\psi_1 \, {}_I\mathcal{U} \, \psi_2$ in the closure set, the type-3 formula $\Diamond_{(0,\infty)\cap(<I)} \psi_2$; for each type-3 formula $\psi = \Diamond_I \psi'$ in the closure set, the type-3 formula $\hat{\psi} = \Diamond_{I\cup\{r(I)\}} \psi'$; and for each type-4 formula $\psi = \Box_I \psi'$ in the closure set, the type-4 formula $\hat{\psi} = \Box_{I-\{r(I)\}} \psi'$.

3. For each type-1 formula ψ in the closure set, the clock intervals $C_1(\psi),\ldots C_{4K}(\psi)$; for each type-2 formula ψ in the closure set, the clock intervals $C_1(\psi),\ldots C_{4K}(\psi)$; and for each type-3 and type-4 formula ψ in the closure set, the clock x_ψ.

4. For each clock interval $C = C_j(\psi)$ in the closure set, where ψ is $\psi_1 \mathcal{U}_I \psi_2$ or $\psi_1 \, {}_I\mathcal{U} \, \psi_2$, all clock constraints of the form $0 < (K - C)$, $0 \subset (K - C)$, $0 = (K - C)$, $(K - C) = \emptyset$, $I \subseteq (K - C)$, and $(K - C) \cap I \neq \emptyset$; and for each clock x_ψ in the closure set, where ψ is $\Diamond_I \psi'$ or $\Box_I \psi'$, the clock constraints $x \in I$ and $x > I$.

   We write $0 \subset (K - C)$ short for $\{0\} \subset (K - C)$. It should be clear that all of these conditions are indeed clock constraints. For instance, the condition $0 \subset (K - [x, y))$ stands for the clock constraint $x \leq K \wedge y > K$; the condition $0 = (K - [x, y))$ is never satisfied.

5. The clock constraint $x_{sing} = 0$.

Note that the number of subformulas of ϕ is $O(N)$ and the number of clocks is $O(K)$ for each subformula of ϕ. Hence the size of the closure set $Closure(\phi)$ is $O(N{\cdot}K)$.

The states of the desired automaton $\mathcal{M}_\phi$ will be subsets of $Closure(\phi)$. We need to consider only those subsets of $Closure(\phi)$ that satisfy certain local consistency constraints. Whenever the automaton is in state s, the formulas in s indicate which subformulas of ϕ are true. Accordingly, a state $s \subseteq Closure(\phi)$ is initial iff both ϕ and $x_{sing} = 0$ are in s, and for each state s the propositional constraints $\mu(s)$ are defined such that $p \in \mu(s)$ iff $p \in s$ for all atomic propositions $p \in P$.

The clock constraints $\nu(s)$ are the conjunction of all clock constraints in s. The clock intervals in s indicate which clock intervals are currently active and represent witnessing intervals for type-1 and type-2 formulas; the clocks in s indicate which clocks are currently active and represent proof obligations for type-3 and type-4 formulas.

The transitions of $\mathcal{M}_\phi$ are all triples $s \xrightarrow{\lambda} s'$ that satisfy certain global consistency criteria. Both the local and the global consistency conditions are defined in the following catalog. For every state $s \subseteq Closure(\phi)$ and every transition $s \xrightarrow{\lambda} s'$ with source state s:

Logical consistency

- For each atomic proposition $p \in P$, precisely one of p and $\neg p$ is in s.
- If the formula $\psi_1 \wedge \psi_2$ is in s, then both ψ_1 and ψ_2 are in s.
- If the formula $\psi_1 \vee \psi_2$ is in s, then either ψ_1 or ψ_2 is in s.

These conditions ensure that no state contains subformulas of ϕ that are mutually inconsistent.

Timing consistency

- s contains at most one of the clock constraints $0 < (K - C)$, $0 \subset (K - C)$, $0 = (K - C)$, and $(K - C) = \emptyset$ for each clock interval C. Furthermore, no two clock intervals in s share clocks; for instance, s does not contain both the clock intervals (x, y) and $[x, y)$.
- s contains at most one of the clock constraints $x_\psi \in I$ and $x_\psi > I$ for each type-3 or type-4 formula ψ.
- If s contains $x_{sing} = 0$, then $x_{sing} \notin \lambda$. If s does not contain $x_{sing} = 0$, then $x_{sing} \in \lambda$ and s' contains $x_{sing} = 0$.

These conditions guarantee that no state contains clock constraints that are mutually inconsistent. We say that a state s is *singular* iff it contains $x_{sing} = 0$; otherwise s is *open*. The last clause of the above conditions ensures that singular and open states alternate along any run.

Type-1 formulas

Consider a type-1 formula $\psi = \psi_1 \mathcal{U}_I \psi_2$ in the closure set.

Firstly, if ψ is in s, then there is some clock interval $C = C_j(\psi)$ such that

- $(K - C) \cap I \neq \emptyset$ is in s, and
- either C is in s, or s is singular and C is in s' and the clocks associated with C are not in λ.

The first condition checks that the interval $K - C$ is an appropriate candidate for witnessing the formula ψ. The second condition activates the clock interval C to represent a witnessing interval for ψ.

Secondly, if some clock interval $C = C_j(\psi)$ is in s, then

- if either $0 = (K - C)$ or $0 \subset (K - C)$ is in s, then ψ_2 is in s, and

- if either $0 < (K - C)$ or $0 \subset (K - C)$ is in s, then ψ_1 is in s, and
- the clocks associated with C are not in λ and either C or $(K - C) = \emptyset$ is in s'.

The first two conditions verify that the active clock interval C represents indeed a witness for the formula ψ. The final condition keeps the clock interval C active as long as necessary.

Suppose that these conditions are satisfied along a run r and the formula ψ is in a state at time t. Also assume (the induction hypothesis) that, along the run r, whenever a state at time t' contains a subformula ψ' of ψ, then $\rho_r^{t'} \models \psi'$. A clock interval $C = C_j(\psi)$ is activated at time t. It is not hard to show that the interval $t + K - C$ is a witnessing interval for ψ under ρ_r^t. By Lemma 4.1, it follows that $\rho_r^t \models \psi$.

Conversely, if $\rho^t \models \psi$, then there is a run r that satisfies all conditions. This is because, by Lemma 4.2, the automaton can, at time t, either share an already activated clock interval $C_j(\psi)$ or has enough clocks to activate an unused clock interval $C_j(\psi)$.

Type-2 formulas

Consider a type-2 formula $\psi = \psi_1 \, {}_I\mathsf{U} \, \psi_2$ in the closure set.

Firstly, if ψ is in s, then either

- $\Diamond_{(0,\infty)\cap(<I)} \psi_2$ is in s

or there is some clock interval $C = C_j(\psi)$ such that

- $I \subseteq (K - C)$ is in s, and
- either C is in s, or s is singular and C is in s' and the clocks associated with C are not in λ.

If $\Diamond_{(0,\infty)\cap(<I)} \psi_2$ holds then so does ψ. The second clause corresponds to guessing a witness. The first condition checks that the interval $K - C$ is an appropriate candidate for witnessing the formula ψ. The second condition activates the clock interval C to represent a witnessing interval for ψ.

Secondly, if some clock interval $C = C_j(\psi)$ is in s, then

- if either $0 = (K - C)$ or $0 \subset (K - C)$ is in s, then ψ_1 is in s, and
- either ψ_2 is in s, or the clocks associated with C are not λ and either C or $(K - C) = \emptyset$ is in s'.

These conditions ensure that the active clock interval C represents indeed a witness for the formula ψ and that it is kept active as long as necessary.

Soundness and completeness of these conditions follow by the Lemmas 4.1 and 4.3.

Type-3 formulas

Consider a type-3 formula $\psi = \Diamond_I \psi'$ in the closure set.

Firstly, if ψ is in s, then either

- s is singular and $x_\psi \in s'$, or
- s is open and I is right-open and $\hat{\psi}$ is in s, or
- s is open and I is right-closed and x_ψ is in s.

These conditions activate a clock to represent a proof obligation. Lemma 4.4 justifies the decision to check, if s is open, instead of ψ the weaker type-3 formula $\hat{\psi}$.

Secondly, if x_ψ is in s, then

- $x_\psi \in I$ is in s, and
- either ψ' is in s, or x_ψ is in s' and $x_\psi \notin \lambda$.

These conditions verify the proof obligation that is represented by the clock x_ψ and keep it active as long as necessary.

Type-4 formulas

Consider a type-4 formula $\psi = \Box_I \psi'$ in the closure set.

Firstly, if ψ is in s, then either

- s is singular and $x_\psi \in s'$ and $x_\psi \in \lambda$, or
- s is open and I is right-closed and $\hat{\psi}$ is in s, or
- s is open and I is right-open and $x_\psi \in s$ and $x_\psi \in s'$ and $x_\psi \in \lambda$.

These conditions activate a clock to represent a proof obligation, and reset it, as was justified in the previous subsection. Recall that if s is open, then instead of checking ψ, it suffices to check the weaker type-4 formula $\hat{\psi}$.

Secondly, if x_ψ is in s then

- ψ' is in s, and
- either x_ψ or $x > I$ is in s'.

The first condition verifies the proof obligation that is represented by the clock x_ψ, and the second condition keeps it active as long as necessary.

Type-5 formulas

Consider a type-5 formula $\psi = \psi_1 \, \mathcal{U} \, \psi_2$ in the closure set. Whenever ψ is in s, then either

- s is singular and $\psi \in s'$, or
- s is open and ψ_1 is in s, and either ψ_2 is in s or ψ_2 is in s' or both ψ_1 and ψ are in s'.

These conditions ensure that unconstrained *until* formulas are propagated correctly (remember that singular and open intervals alternate).

Type-6 formulas

Consider a type-6 formula $\psi = \Box\,\psi'$ in the closure set. Whenever ψ is in s, then either

- s is singular and $\psi \in s'$, or
- s is open and $\psi' \in s$ and both ψ' and ψ are in s'.

These conditions guarantee that unconstrained *always* formulas are propagated forever.

This concludes the definition of the timed automaton $\mathcal{M}_\phi$. The runs of $\mathcal{M}_\phi$ are defined as before. We put, however, additional fairness requirements on the timed state sequences that are generated by $\mathcal{M}_\phi$. A run r is called *accepting* iff for every type-5 formula ψ of the form $\psi_1\,\mathcal{U}\,\psi_2$, if ψ is in some state s along r, then ψ_2 is in some later state s'.

The following main lemma states the correctness of our construction by relating the accepting runs of $\mathcal{M}_\phi$ to the models of ϕ.

Lemma 4.5 (Correctness of $\mathcal{M}_\phi$)
A timed state sequence ρ satisfies an MITL*-formula ϕ iff the timed automaton $\mathcal{M}_\phi$ has an accepting run r with $\rho = \rho_r$.*

This result yields algorithms for checking the satisfiability and validity of the given MITL-formula ϕ. To check satisfiability, we first construct the timed automaton $\mathcal{M}_\phi$, and then we use the algorithm that checks whether $\mathcal{M}_\phi$ has any accepting run to test if ϕ has a model. Similarly, ϕ is valid iff $\mathcal{M}_{\neg\phi}$ has no accepting run.

4.7 Complexity of MITL

We conclude this section by showing that our decision procedure for MITL is in EXPSPACE, and that this is optimal, because the decision problem for MITL is EXPSPACE-complete.

Recall that the size $|Closure(\phi)|$ of the closure set of ϕ is $O(N{\cdot}K)$, where N is the number of atomic propositions, boolean connectives, and temporal operators in ϕ, and $K-1$ is the product of the largest constant in ϕ and the least common denominator of all constants in ϕ. Clearly, $|Closure(\neg\phi)| = O(N \cdot K)$ as well.

Hence the number of states in $\mathcal{M}_\phi$ and $\mathcal{M}_{\neg\phi}$ is $O(2^{N \cdot K})$. Consequently, the description of $\mathcal{M}_{\neg\phi}$ can be given in space polynomial in $N{\cdot}K$; that is, in space exponential in the length of ϕ, assuming binary encoding of all interval end-points. The emptiness problem for a timed automaton $\mathcal{M}$ can be solved in space polynomial in the length of the description of $\mathcal{M}$. It follows that the validity of ϕ can be decided in space polynomial in $N{\cdot}K$, that is, in EXPSPACE.

The lower bound of EXPSPACE for MITL can be shown along the lines of the proof of the EXPSPACE-hardness of the real-time logic MTL [AH90].

Theorem 4.1 (Complexity of MITL)
The decision problem of MITL *is EXPSPACE-complete. Furthermore, we have an EXPSPACE algorithm that solves this problem.*

5 Model Checking

Model checking is a powerful and well-established technique for the automatic verification of finite-state systems (see, for example, [BCM+90]); it compares a temporal-logic specification of a system against a state-transition description of the system.

In the qualitative case, the system is modeled by its state-transition graph, also known as Kripke structure, and the specification may be presented as a formula of the propositional linear temporal logic PTL [LP84]. For real-time systems, model checking algorithms have been developed for linear temporal logics under a digital-clock interpretation of time [AH89, AH90, HLP90] as well as for branching-time logics under a continuous interpretation of time [ACD90, Lew90]. Using our results about MITL, we can present a real-time verification procedure that checks *linear* specifications under a *continuous* model of time.

We model a real-time system by a timed automaton $\mathcal{M}$ and give the specification as a formula ϕ of MITL. Hence the *model checking* problem is to decide whether or not the automaton $\mathcal{M}$ satisfies the specification ϕ:

$$\mathcal{M} \stackrel{?}{\models} \phi$$

Our construction for testing the satisfiability of MITL-formulas can be used to develop an algorithm for model checking. The first step is to construct a timed automaton $\mathcal{M}_{\neg\phi}$ such that its accepting runs precisely capture the models of the negated formula $\neg\phi$: for every timed state sequence ρ, $\mathcal{M}_{\neg\phi}$ has an accepting run r with $\rho_r = \rho$ iff $\rho \models \neg\phi$.

The model checking question can then be reformulated as follows: $\mathcal{M} \models \phi$ iff no timed state sequence is generated by both $\mathcal{M}$ and $\mathcal{M}_{\neg\phi}$. The next step in the model checking algorithm is to construct a timed automaton $\mathcal{M}'$ that is the product of $\mathcal{M}$ and $\mathcal{M}_{\neg\phi}$; a timed state sequence is generated by $\mathcal{M}'$ iff it is generated by both $\mathcal{M}$ and $\mathcal{M}_{\neg\phi}$. The *product* construction for timed automata presented in [AD90] can be easily modified to our version of timed automata.

Hence we have reduced the model checking problem to the emptiness question for timed automata: $\mathcal{M} \models \phi$

iff $\mathcal{M}'$ has no accepting runs. The size of $\mathcal{M}'$ is polynomial in the sizes of $\mathcal{M}$ and $\mathcal{M}_{\neg\phi}$. Consequently, the description of $\mathcal{M}'$ is exponential in the length of ϕ, and polynomial in the length of the description of $\mathcal{M}$. Since the emptiness for timed automata can be solved in PSPACE, it follows that the model checking problem can be solved in EXPSPACE.

As for all linear temporal logics, the model checking question for MITL is no simpler than the satisfiability question: a formula ϕ is unsatisfiable iff the universal timed automaton, which generates all possible timed state sequences, satisfies $\neg\phi$. Thus EXPSPACE-hardness of satisfiability implies EXPSPACE-hardness of model checking. The following theorem follows:

Theorem 5.1 (Model checking)
The problem of checking whether a timed automaton $\mathcal{M}$ satisfies an MITL*-formula ϕ is EXPSPACE-complete.*

The time complexity of the model checking algorithm is polynomial in the qualitative part of the system description, exponential in the qualitative part of the MITL-specification, exponential in the timing part of the system description, and doubly exponential in the timing part of the specification. Compared to this the model checking algorithm for PTL [LP84] is polynomial in the size of the Kripke structure and exponential in the size of the specification.

Thus moving to *real-time* gives an additional *exponential* blow-up. This blow-up seems, however, unavoidable for formalisms for quantitative reasoning about time. It occurs even in the simplest case — synchronous processes that are clocked by a digital clock — in which we can model time by a discrete domain and identify next-state with next-time [EMSS89, AH90].

Acknowledgements. We thank Moshe Vardi for encouraging us repeatedly to look for logics of continuous time, Ron Koymans for helpful suggestions, and David Dill and Zohar Manna for their guidance.

References

[ACD90] R. Alur, C. Courcoubetis, and D. L. Dill. Model-checking for real-time systems. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, 1990.

[AD90] R. Alur and D. L. Dill. Automata for modeling real-time systems. In *17th International Colloquium on Automata, Languages, and Programming*. Springer-Verlag Lecture Notes in Computer Science 443, 1990.

[AH89] R. Alur and T. A. Henzinger. A really temporal logic. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, 1989.

[AH90] R. Alur and T. A. Henzinger. Real-time logics: complexity and expressiveness. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, 1990.

[BCM+90] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, 1990.

[EMSS89] E. A. Emerson, A. K. Mok, A. P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. Presented at the First Annual Workshop on Computer-aided Verification, Grenoble, France, 1989.

[HLP90] E. Harel, O. Lichtenstein, and A. Pnueli. Explicit-clock temporal logic. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, 1990.

[JM86] F. Jahanian and A. K. Mok. Safety analysis of timing properties in real-time systems. *IEEE Transactions on Software Engineering*, SE-12, 1986.

[Koy90] R. Koymans. Specifying real-time properties with metric temporal logic. *Journal of Real-time Systems*, 2, 1990.

[Lew90] H. R. Lewis. A logic of concrete time intervals. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, 1990.

[LP84] O. Lichtenstein and A. Pnueli. Checking that finite-state concurrent programs satisfy their linear specification. In *Proceedings of the 11th Annual ACM Symposium on Principles of Programming Languages*, 1984.

[Ost90] J. S. Ostroff. *Temporal Logic of Real-time Systems.* Research Studies Press, 1990.

[Rog67] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, 1967.

[Tho90] W. Thomas. Automata on infinite objects. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B. Elsevier, 1990.