The Benefits of Relaxing Punctuality[1]

Rajeev Alur
AT&T Bell Laboratories
Murray Hill, NJ 07974
Email: alur@research.att.com

Tomás Feder
IBM Almaden Research Center
San Jose, CA 95120
Email: tomas@almaden.ibm.com

Thomas A. Henzinger[2]
Computer Science Department, Cornell University
Ithaca, NY 14853
Email: tah@cs.cornell.edu

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

Keywords: temporal logic, verification, real time.

---

[1]A preliminary version of this paper appeared in the *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing* (PODC 1991), pp. 139–152, and an extended version appeared in the *Journal of the ACM* **43**, 1996, pp. 116–146.

[2]Supported in part by the Office of Naval Research Young Investigator award N00014-95-1-0520, by the National Science Foundation CAREER award CCR-9501708, by the National Science Foundation grants CCR-9200794 and CCR-9504469, by the Air Force Office of Scientific Research contract F49620-93-1-0056, and by the Advanced Research Projects Agency grant NAG2-892.

1 Introduction

The formal study of reactive systems has recently led to a number of suggestions of how real-time requirements of 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 [Koy90, Lew90, ACD93, AD94], or they sacrifice continuity to obtain decision procedures [JM86, EMSS90, HLP90, Ost90, AH93, AH94] (for a discussion of this trade-off, see [AH92]). This paper shows how a slight relaxation of the notion of punctuality allows us to combine the best of both worlds.

We use a linear or trace semantics for reactive systems. The linear semantics of a system is 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. We represent the possible behaviors of a real-time system by such *timed state sequences*, each of which defines a function from the nonnegative reals to the system states.

Alas, even the satisfiability of a very simple class of real-time properties turns out to be undecidable in this model [AH94]. An inspection of the undecidability proof shows that the only timing constraints required are of the form

$$\Box(p \rightarrow \Diamond_{=5}\, q), \tag{†}$$

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 [AH93, AH94].

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. This is accomplished by prohibiting singular time intervals, of the form $[a, a]$, from constraining temporal operators. The resulting *Metric Interval Temporal Logic* MITL is shown to be decidable in EXPSPACE. The complexity is PSPACE for the fragment of MITL that employs only time intervals of the form $[a, \infty)$, (a, ∞), $[0, b)$, and $[0, b]$.

Properties of timed state sequences can, alternatively, be defined by *timed automata* [AD94]. While the emptiness problem for timed automata is solvable, they are not closed under complement. MITL identifies a fragment of the properties definable by timed automata that is closed under all boolean operations. The decision procedure for MITL leads to an algorithm for proving that a real-time system that is given as a timed automaton meets a requirements specification that is given in MITL. Thus the novelty of our results is that they provide a *logical* formalism with a *continuous* interpretation of time that is suitable for the automatic verification and synthesis of real-time systems.

The remainder of the paper is organized as follows. In Section 2, we introduce and motivate the logic MITL. In Section 3, we introduce a variant of timed automata as a model for real-time systems. In Section 4, we reduce the decision problem for MITL to the emptiness problem for timed automata. In the concluding section, we present a model-checking algorithm for verifying MITL-specifications of timed automata.

2 Metric Interval Temporal Logic

We define timed state sequences as formal representations of real-time behavior. Then we introduce a temporal language to define properties of timed state sequences.

2.1 Time intervals

We use the set $\mathbb{R}_{\geq 0}$ of the nonnegative real numbers as time domain. A (*time*) *interval* is a nonempty convex subset of $\mathbb{R}_{\geq 0}$. Intervals may be open, half-open, or closed; bounded or unbounded. Each interval has one of the following forms: $[a, b]$, $[a, b)$, $[a, \infty)$, $(a, b]$, (a, b), (a, ∞), where $a \leq b$ for $a, b \in \mathbb{R}_{\geq 0}$. For an interval 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 $\ell(I)$ and the right end-point, for bounded I, is denoted by $r(I)$.

The interval I is *singular* iff it is of the form $[a, a]$; that is, I is closed and $\ell(I) = r(I)$. The two intervals I and I' are *adjacent* iff (1) the right end-point of I is the same as the left end-point of I', and (2) either I is right-open and I' is left-closed, or I is right-closed and I' is left-open. For instance, the two intervals $(1, 2]$ and $(2, 2.5)$ are adjacent.

We freely use intuitive pseudo-arithmetic expressions to denote intervals. For example, the expressions $\leq b$ and $> a$ denote the intervals $[0, b]$ and (a, ∞), respectively, and the expression $< I$ denotes the interval $\{t \mid \text{for all } t' \in I, 0 \leq t < t'\}$. The expression $t + I$, for $t \in \mathbb{R}_{\geq 0}$, denotes the interval $\{t + t' \mid t' \in I\}$. Similarly, $I - t$ and $t \cdot I$ stand for the intervals $\{t' - t \mid t' \in I \text{ and } t' \geq t\}$ and $\{t \cdot t' \mid t' \in I\}$, respectively.

2.2 Timed state sequences

Let P be a finite set of *propositions*. We assume that, at any point in time, the observable state of a (finite-state) system can be modeled by a truth-value assignment for P. A *state*, then, is a subset of P. If $s \subseteq P$ is a state and $p \in s$ is a proposition in s, we write $s \models p$ and say that s is a p-state.

The execution of a system results in an infinite sequence of states. We model the timed execution of systems by associating a time interval with each state. We require that the intervals associated with consecutive states be adjacent, and that the union of all intervals partitions the nonnegative real line.

DEFINITION 2.1 A *state sequence* $\overline{s} = s_0 s_1 s_2 \ldots$ is an infinite sequence of states $s_i \subseteq P$. An *interval sequence* $\overline{I} = I_0 I_1 I_2 \ldots$ is an infinite sequence of time intervals such that

- [*Initiality*] I_0 is left-closed and $\ell(I_0) = 0$;
- [*Adjacency*] for all $i \geq 0$, the intervals I_i and I_{i+1} are adjacent;
- [*Progress*] every time $t \in \mathbb{R}_{\geq 0}$ belongs to some interval I_i.

A *timed state sequence* $\tau = (\overline{s}, \overline{I})$ is a pair that consists of a state sequence $\overline{s}$ and an interval sequence $\overline{I}$. For $i \geq 0$ and $t \in I_i$, the state $\tau^*(t)$ at time t is s_i. □

The timed state sequence $\tau = (\overline{s}, \overline{I})$ can thus be viewed as the function τ^* from the time domain $\mathbb{R}_{\geq 0}$ to the states 2^P, which provides a system state at every time instant. (There is the alternative view that the timed execution of a system alternates state changes and time delays. That view can be modeled by pairing state sequences with sequences of *closed* intervals [HNSY94]. It is routine to transfer all our results into that model.)

DEFINITION 2.2 The two timed state sequences τ_1 and τ_2 are *equivalent* iff for all $t \in \mathbb{R}_{\geq 0}$, $\tau_1^*(t) = \tau_2^*(t)$. □

Sometimes we represent the timed state sequence $\tau = (\overline{s}, \overline{I})$ by the infinite sequence

$$(s_0, I_0) \rightarrow (s_1, I_1) \rightarrow (s_2, I_2) \rightarrow \cdots$$

of state-interval pairs. It is also convenient to represent an infinite sequence of state-interval pairs with identical state components by a single state-interval pair with an unbounded interval: if $s_j = s_i$ for all $j \geq i$, we write

$$(s_0, I_0) \rightarrow (s_1, I_1) \rightarrow \cdots \rightarrow (s_{i-1}, I_{i-1}) \rightarrow (s_i, \bigcup_{j \geq i} I_j)$$

for τ. The time $t \in \mathbb{R}_{\geq 0}$ is a *transition point* of τ iff t is the left end-point $\ell(I_i)$ of an interval in $\overline{I}$. The state s_i is singular in τ iff the associated interval I_i is singular. Notice that in this case neither s_{i-1} nor s_{i+1} can be singular, because the interval I_{i-1} must be right-open and the interval I_{i+1} must be left-open. Singular states are useful for modeling events by propositions that are true only at transition points.

DEFINITION 2.3 Let $\tau = (\overline{s}, \overline{I})$ be a timed state sequence. Given $t \in I_i$, the *suffix* τ^t at time t is the timed state sequence

$$(s_i, I_i - t) \rightarrow (s_{i+1}, I_{i+1} - t) \rightarrow (s_{i+2}, I_{i+2} - t) \rightarrow \cdots$$

□

In particular, $\tau^0 = \tau$ and for all $t, t' \in \mathbb{R}_{\geq 0}$, $(\tau^t)^*(t') = \tau^*(t + t')$.

2.3 MITL

MITL is a linear temporal logic that is interpreted over timed state sequences. A standard way of introducing real time into the syntax of temporal languages constrains the temporal operators with time intervals [EMSS90, Koy90, AH93]. For example, the constrained *eventually* operator $\Diamond_{[2,4]}$ is interpreted as "eventually within 2 to 4 time units." We adopt this approach for MITL, with the restriction that temporal operators cannot be constrained by singular intervals.

2.3.1 Syntax

The formulas of MITL are built from propositions using boolean connectives and a time-constrained version of the *until* operator $\mathcal{U}$. The *until* operator may be constrained by any nonsingular interval with integer end-points. The restriction to integer end-points simplifies the presentation; we later show that our results extend to the case of rational end-points.

DEFINITION 2.4 The formulas of MITL are inductively defined by the grammar

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

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

We say that the integer constant c *appears* in the MITL-formula ϕ iff c is an end-point of some interval that appears in ϕ, as a subscript of an *until* operator.

2.3.2 Semantics

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

DEFINITION 2.5 For an MITL-formula ϕ and a timed state sequence $\tau = (\overline{s}, \overline{I})$, the satisfaction relation $\tau \models \phi$ is defined inductively as follows:

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

The timed state sequence τ is a *model* of the formula ϕ, or τ *satisfies* ϕ, iff $\tau \models \phi$. We write $L(\phi)$ for the set of models of ϕ. The formula ϕ is *satisfiable* iff $L(\phi) \neq \emptyset$; the two formulas ϕ and ϕ' are *equivalent* iff $L(\phi) = L(\phi')$. The *satisfiability problem* for MITL is to decide whether or not a given MITL-formula is satisfiable. $\Box$

Note that MITL has no *next-time* operator, because the time domain is dense. Instead, the *until* operator is *strict* in its first argument: if $r(I) > 0$, then for $\phi_1 \mathcal{U}_I \, \phi_2$ to hold at time t of a timed state sequence, neither ϕ_1 nor ϕ_2 need to hold at time t.

Also observe that MITL cannot distinguish equivalent timed state sequences. This is because for a timed state sequence $\tau = (\overline{s}, \overline{I})$ and an MITL-formula ϕ, the satisfaction relation $\tau \models \phi$ depends only on the function τ^*, and not on the particular choice of the interval sequence $\overline{I}$. We remember this observation as the following remark.

REMARK 2.6 *Let ϕ be an* MITL*-formula. If the two timed state sequences τ_1 and τ_2 are equivalent, then $\tau_1 \models \phi$ iff $\tau_2 \models \phi$.*

2.3.3 Defined operators

We introduce some standard abbreviations for additional temporal operators. The defined operators $\Diamond_I \, \phi$ (time-constrained *eventually*) and $\Box_I \, \phi$ (time-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 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 linear temporal logic [MP92]. The corresponding nonstrict operators are definable in MITL as

$$\begin{aligned} \Diamond_{\geq 0}\,\phi &= \phi \vee \Diamond\phi, \\ \Box_{\geq 0}\,\phi &= \phi \wedge \Box\phi, \\ \phi_1 \mathcal{U}_{\geq 0}\,\phi_2 &= \phi_2 \vee (\phi_1 \wedge \phi_1 \mathcal{U}\phi_2). \end{aligned}$$

Note that, on the other hand, the MITL-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 choose the strict versions of the temporal operators to be primitive.

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

$$\phi_1\,{}_I\mathcal{W}\phi_2 \;=\; \neg((\neg\phi_2)\mathcal{U}_I\,(\neg\phi_1)).$$

It follows that the formula $\phi_1\,{}_I\mathcal{W}\phi_2$ holds at time t of a timed state sequence iff either ϕ_1 is true throughout the interval $t + I$, or there is a time $t' > t$ such that ϕ_2 is true at time t' and ϕ_1 holds throughout the interval $[t, t'] \cap I$. For example, $\tau \models p\,{}_{[1,2]}\mathcal{W}q$ iff either $\tau^t \models p$ for all $t \in [1, 2]$, or $\tau^t \models q$ for some $t \in (0, 1)$, or $\tau^t \models q$ for some $t \in [1, 2]$ and for all $t' \in [1, t]$, $\tau^{t'} \models p$. Note that the unconstrained version $\phi_1\mathcal{W}\phi_2$ of the *unless* operator of MITL differs slightly from the conventional *strict unless* operator [MP92], which can be defined as $\phi_1\mathcal{W}(\phi_1 \wedge \phi_2)$.

We can apply the definition of the constrained *unless* operator to move negations through constrained *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 propositions.

2.3.4 Examples

Let us consider a few examples of MITL-formulas.

EXAMPLE 2.7 The typical bounded-response requirement that "every p-state is followed by a q-state within 3 time units," is expressed by the MITL-formula

$$\Box_{\geq 0}\,(p \;\rightarrow\; \Diamond_{(0,3]}\,q).$$

□

EXAMPLE 2.8 The following MITL-formula asserts that the proposition p is true in infinitely many singular states and nowhere else:

$$\Diamond_{\geq 0}\,p \wedge \Box_{\geq 0}\,(p \;\rightarrow\; (\neg p)\mathcal{U}p).$$

Such a requirement can be used to characterize events, and it cannot be expressed with an *until* operator that is nonstrict in its first argument. □

EXAMPLE 2.9 Now we consider a time-out requirement. Suppose that p is a state constraint, and q is the time-out event. We wish to specify the requirement that "whenever p ceases to hold, either p becomes true again within less than 5 time units, or at time 5 the time-out event q happens." This requirement is expressed by the MITL-formula

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

□

Additional examples of real-time requirements that are specifiable using time-constrained temporal operators can be found in [Koy90].

2.3.5 Model refinement

All timed state sequences obey the so-called *finite-variability* condition: between any two points in time there are only finitely many state changes. This assumption is adequate for modeling discrete systems. We now show that the timed state sequences satisfy the finite-variability condition not only with respect to the truth of propositions, but also with respect to the truth of all MITL-formulas. That is, we show that the truth value of every MITL-formula ϕ does not change more than ω times along a given timed state sequence τ. This is done by splitting the intervals of τ finitely often until the truth value of ϕ stays invariant over every interval. This process is called the *refinement* of τ.

DEFINITION 2.10 Given an MITL-formula ϕ, the timed state sequence $\tau = (\overline{s}, \overline{I})$ is *ϕ-fine* iff for all subformulas ψ of ϕ, for all $i \geq 0$, and for all $t, t' \in I_i$, $\tau^t \models \psi$ iff $\tau^{t'} \models \psi$. □

For an MITL-formula ϕ, then, the truth value of each subformula of ϕ stays invariant over every interval of a ϕ-fine timed state sequence.

LEMMA 2.11 *Let ϕ be an* MITL*-formula. For every timed state sequence τ, there exists a ϕ-fine timed state sequence that is equivalent to τ.*

PROOF. The proof proceeds by induction on the structure of ϕ. Let τ be a timed state sequence. For each subformula ψ of ϕ, we construct a ψ-fine timed state sequence τ_ψ that is equivalent to τ. For the proposition $\psi = p$, let $\tau_\psi = \tau$. For the negation $\psi = \neg\psi'$, let $\tau_\psi = \tau_{\psi'}$. In case of the conjunction $\psi = \psi_1 \wedge \psi_2$, the timed state sequence τ_ψ is constructed by refining the timed state sequence sequence τ_{ψ_1}. We split each interval of τ_{ψ_1} into a finite sequence of intervals, each of which is fully contained in an interval of τ_{ψ_2}. In other words, the interval sequence of $\tau_{\psi_1 \wedge \psi_2}$ is obtained by intersecting the two interval sequences of τ_{ψ_1} and τ_{ψ_2}.

Now consider the case $\psi = \psi_1 \mathcal{U}_I \psi_2$. First we construct the refined timed state sequence $\tau_{\psi_1 \wedge \psi_2}$. Then we construct inductively a diverging countable sequence $t_0 < t_1 < t_2 < \cdots$ of times $t_i \in \mathbb{R}_{\geq 0}$: let $t_0 = 0$; for all $i \geq 0$, let t_{i+1} be the least $t > t_i$ such that either t is a transition point of $\tau_{\psi_1 \wedge \psi_2}$, or $t = t_j + \ell(I)$ or $t = t_j + r(I)$ for some $0 \leq j \leq i$. We choose $\tau_\psi = (\overline{s}, \overline{I})$ to be equivalent to τ with the interval sequence $\overline{I} = [t_0, t_0], (t_0, t_1), [t_1, t_1], (t_1, t_2), \ldots$

Consider two times t and t' that belong to the same interval of τ_ψ, say, $t_i < t' < t < t_{i+1}$. Suppose that $\tau^t \models \psi$. Then there exists a time $u \in (t + I)$ such that $\tau^u \models \psi_2$ and for all $u' \in (t, u)$, $\tau^{u'} \models \psi_1$. Since $t < t_{i+1}$, there exists a time $u' \in (t_i, t_{i+1})$ such that $\tau^{u'} \models \psi_1$. Since the truth value of ψ_1 stays invariant throughout the interval (t_i, t_{i+1}), we have $\tau^{u'} \models \psi_1$ for all $u' \in (t_i, t_{i+1})$ and, hence, for all $u' \in (t', u)$. From the construction of the time sequence $t_0, t_1, \ldots$ it follows that $u \in (t' + I)$. Therefore $\tau^{t'} \models \psi$. It follows similarly that $\tau^{t'} \models \psi$ implies $\tau^t \models \psi$. □

2.4 Variations of MITL

We consider three variations of MITL. Recall that all time intervals that appear in timed state sequences have real end-points, and all time intervals that appear in MITL-formulas have integer end-points. First, a rational semantics does not change the satisfiability problem for MITL and, second, neither does a rational syntax. Third, the admission of singular intervals in MITL-formulas renders the satisfiability problem undecidable.

2.4.1 Real time versus rational time

While timed state sequences are defined over the real numbers, with respect to interpreting MITL-formulas, the crucial property of the time domain is not its continuity, but only its denseness. We show that replacing the time domain $\mathbb{R}_{\geq 0}$ with the set $\mathbb{Q}_{\geq 0}$ of the nonnegative rational numbers does not change the satisfiability of any MITL-formula. In other words, MITL cannot distinguish the time domain $\mathbb{R}_{\geq 0}$ from the time domain $\mathbb{Q}_{\geq 0}$.

DEFINITION 2.12 The timed state sequence τ is *rational* iff all transition points of τ are rational. The rational timed state sequence τ $\mathbb{Q}$-*satisfies* the MITL-formula ϕ iff $\tau \models \phi$, where the satisfaction relation $\models$ of Definition 2.5 is redefined so that all time quantifiers range over $\mathbb{Q}_{\geq 0}$ only. The MITL-formula ϕ is $\mathbb{Q}$-*satisfiable* iff there is a rational timed state sequence that $\mathbb{Q}$-satisfies ϕ. $\square$

The equivalence of the real and the rational semantics for MITL follows from two lemmas.

LEMMA 2.13 *Let ϕ be an* MITL-*formula and let τ a rational ϕ-fine timed state sequence. Then τ $\mathbb{Q}$-satisfies ϕ iff τ satisfies ϕ.*

PROOF. Let τ be a rational and ϕ-fine timed state sequence. We use induction on the structure of ϕ. We consider only the interesting case, for a subformula ψ of ϕ of the form $\psi_1 \mathcal{U}_I \psi_2$. Suppose that τ $\mathbb{Q}$-satisfies ψ; that is, τ^t $\mathbb{Q}$-satisfies ψ_2 for some rational $t \in I$, and $\tau^{t'}$ $\mathbb{Q}$-satisfies ψ_1 for all rationals $t' \in (0,t)$. By the induction hypothesis, we conclude that $\tau^t \models \psi_2$ and for all $t' \in (0,t)$, $\tau^{t'} \models \psi_1$. Hence, to show that $\tau \models \psi$, it suffices to show that $\tau^{t''} \models \psi_1$ for all reals $t'' \in (0,t)$. Consider an arbitrary real $t'' \in (0,t)$, and assume that $t'' \in I_i$, for an interval I_i of τ. If I_i is singular then, since τ is rational, t'' must be rational. Otherwise, I_i is nonsingular, and there is also a rational $t' \in I_i$ with $t' \in (0,t)$. We know that $\tau^{t'} \models \psi_1$ and, since τ is ϕ-fine, it follows that $\tau^{t''} \models \psi_1$. Therefore $\tau \models \psi$. It follows similarly that $\tau \models \psi$ implies that τ $\mathbb{Q}$-satisfies ψ. $\square$

The following lemma partitions the timed state sequences into blocks such that the members of a block cannot be distinguished by MITL-formulas. Two timed state sequences fall into the same block iff they agree on the state components, on the integral parts of all transition points, and on the ordering of the fractional parts of all transition points. For $t \in \mathbb{R}_{\geq 0}$, let $\langle t \rangle = t - \lfloor t \rfloor$ denote the fractional part of t.

LEMMA 2.14 *Let $\tau = (\overline{s}, \overline{I})$ and $\tau' = (\overline{s}, \overline{I}')$ be two timed state sequences such that for all $i, j \geq 0$, (1) $\lfloor \ell(I_i) \rfloor = \lfloor \ell(I'_i) \rfloor$ and (2) $\langle \ell(I_i) \rangle \leq \langle \ell(I_j) \rangle$ iff $\langle \ell(I'_i) \rangle \leq \langle \ell(I'_j) \rangle$. For all* MITL-*formulas ϕ, $\tau \models \phi$ iff $\tau' \models \phi$.*

PROOF. We write $\tau \sim \tau'$ iff the two timed state sequences τ and τ' satisfy the premise of the lemma. The proof is by induction on the structure of ϕ: we show that for all subformulas ψ of ϕ, and all timed state sequences τ and τ', if $\tau \sim \tau'$ and $\tau \models \psi$, then $\tau' \models \psi$. The interesting case is again that of a subformula ψ of the form $\psi_1 \mathcal{U}_I \psi_2$. Suppose that $\tau \sim \tau'$ for $\tau = (\overline{s}, \overline{I})$ and $\tau' = (\overline{s}, \overline{I}')$, and that $\tau \models \psi$. Let $t \in I$ be such that $\tau^t \models \psi_2$ and $\tau^{t''} \models \psi_1$ for all $t'' \in (0,t)$. Choose t' such that (1) $\lfloor t' \rfloor = \lfloor t \rfloor$ and (2) for all i, $\langle t \rangle \leq \langle \ell(I_i) \rangle$ iff $\langle t' \rangle \leq \langle \ell(I'_i) \rangle$, and $\langle t \rangle \geq \langle \ell(I_i) \rangle$ iff $\langle t' \rangle \geq \langle \ell(I'_i) \rangle$. Then $t' \in I$ and $\tau^t \sim \tau'^{t'}$. From the induction hypothesis applied to τ^t and $\tau'^{t'}$, we conclude that $\tau'^{t'} \models \psi_2$. By a similar argument, it follows that for all $t'' \in (0,t')$, $\tau'^{t''} \models \psi_1$. Therefore $\tau' \models \psi$. $\square$

THEOREM 2.15 *The* MITL-*formula ϕ is $\mathbb{Q}$-satisfiable iff ϕ is satisfiable.*

PROOF. Suppose that the rational timed state sequence τ $\mathbb{Q}$-satisfies ϕ. First we observe that Lemma 2.11 and Remark 2.6 apply to rational timed state sequences also. It follows that there is a rational ϕ-fine timed state sequence τ' that is equivalent to τ and $\mathbb{Q}$-satisfies ϕ. By Lemma 2.13, τ' is a (real) model of ϕ.

For the other direction of the theorem, consider a (real) model $\tau = (\overline{s}, \overline{I})$ of ϕ. We construct a rational timed state sequence $\tau' = (\overline{s}, \overline{I}')$ as follows. We choose inductively a diverging sequence $t_0 < t_1 < t_2 < \cdots$ of rational times $t_i \in \mathbb{Q}_{\geq 0}$: let $t_0 = 0$; for all $i \geq 0$, let (1) $\lfloor t_{i+1} \rfloor = \lfloor \ell(I_{i+1}) \rfloor$ and (2) for all $j \leq i$, $\langle t_{i+1} \rangle \leq \langle t_j \rangle$ iff $\langle \ell(I_{i+1}) \rangle \leq \langle \ell(I_j) \rangle$, and $\langle t_{i+1} \rangle \geq \langle t_j \rangle$ iff $\langle \ell(I_{i+1}) \rangle \geq \langle \ell(I_j) \rangle$. The denseness of $\mathbb{Q}_{\geq 0}$ allows us to choose such rational numbers t_i, say, by choosing the fractional part of each t_i to be a multiple of $1/2^i$. Then let the interval $\overline{I}'_i$ have the left end-point t_i, the right end-point t_{i+1}, and the type of I_i (i.e., $\overline{I}'_i$ is left-open iff I_i is left-open, and right-open iff I_i is right-open). The timed state sequences τ and τ' satisfy the premise of Lemma 2.14, and hence $\tau' \models \phi$. By the rational version of Lemma 2.11 and by Remark 2.6, there is a rational ϕ-fine timed state sequence τ'' that is equivalent to τ' and satisfies ϕ. By Lemma 2.13, τ'' $\mathbb{Q}$-satisfies ϕ. $\square$

2.4.2 Rational intervals

While defining the syntax of MITL we required the end-points of all time intervals that constrain *until* operators to be integers or infinite. This restriction can be relaxed, without affecting the satisfiability problem for MITL, by admitting intervals with rational end-points in MITL-formulas. For example, we may allow the formula $\Box_{(0.5,0.6)}\, p$, which asserts that p holds throughout the time interval $(0.5, 0.6)$.

LEMMA 2.16 *Let ϕ be a formula of* MITL *that contains intervals with rational end-points, and let $\tau = (\overline{s}, \overline{I})$ be a timed state sequence. For $c \in \mathbb{Q}_{\geq 0}$, let ϕ_c denote the formula obtained by replacing every interval in ϕ with the interval $c \cdot I$, and let $\tau_c = (\overline{s}, \overline{I}')$ denote the timed state sequence with $I'_i = c \cdot I_i$ for all $i \geq 0$. Then $\tau \models \phi$ iff $\tau_c \models \phi_c$.*

PROOF. The proof proceeds by a straightforward induction on the structure of ϕ. $\square$

Given a formula ϕ with rational constants, let c be the least common multiple of all denominators appearing in ϕ. In order to check the satisfiability of ϕ, then, it suffices to check the satisfiability of the MITL-formula ϕ_c, which contains only integer constants. Notice that the size $|\phi_c|$ of the description of ϕ_c is bounded by $|\phi|^2$.

2.4.3 Singular intervals

MITL prohibits the use of singular intervals. For example, the formula

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

which expresses the requirement that "every p-state is followed by a q-state after precisely 5 time units," is not an MITL-formula. There is, in fact, no MITL-formula that expresses this requirement, and the restriction of MITL to nonsingular intervals is essential for the decidability of the satisfiability problem. Before we prove this, we note that some forms of equality *are* expressible in MITL. Let $(\neg\phi)\mathcal{U}_{=c}\,\phi$ stand for the MITL-formula $(\Box_{(0,c)}\,\neg\phi) \wedge (\Diamond_{(0,c]}\,\phi)$. Thus the stronger requirement that "for every p-state the *next* following q-state occurs after precisely 5 time units,"

$$\Box(p \;\rightarrow\; (\neg q)\mathcal{U}_{=5}\, q),$$

can be expressed in MITL.

DEFINITION 2.17 $\text{MITL}_=$ is the extension of MITL that admits singular intervals as subscripts of the *until* operator. □

We show that the satisfiability problem for $\text{MITL}_=$ is complete for the complexity class Σ_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 recursively axiomatizable. The undecidability result depends on the denseness of the time domain. If the formulas of $\text{MITL}_=$ are interpreted over a discrete time domain, the resulting logic MTL has a decidable satisfiability problem [AH93].

THEOREM 2.18 *The satisfiability problem for* $\text{MITL}_=$ *is* Σ_1^1*-complete.*

PROOF. We prove Σ_1^1-hardness by reduction from the problem of deciding whether a given nondeterministic 2-counter machine M has a recurring computation (i.e., a computation in which a specified state repeats infinitely often), which is Σ_1^1-hard [HPS83]. In [AH93], it is shown how to construct a formula ϕ of the discrete-time logic MTL such that ϕ is satisfiable over dense-time models iff M has a recurring computation. The construction, with trivial modifications, applies to $\text{MITL}_=$. Indeed, only one temporal operator with a singular subscript, $\Diamond_{=1}$, is needed in the construction.

Now we prove containment in Σ_1^1. Let ϕ be a formula of $\text{MITL}_=$. First observe that Theorem 2.15 holds even in the presence of singular intervals in formulas. Thus, if ϕ has a model, then there is a rational timed state sequence that $\mathbb{Q}$-satisfies ϕ. The $\mathbb{Q}$-satisfiability of ϕ can be phrased as a Σ_1^1-sentence asserting that some timed state sequence with rational transition points $\mathbb{Q}$-satisfies ϕ. It is routine to encode a rational timed state sequence by a set of natural numbers, and to express the $\mathbb{Q}$-satisfaction relation in first-order arithmetic. □

Another possible variation of the syntax of MITL would permit interval constraints on *both* arguments of the *until* operator. The intended meaning of the formula $\phi_1 \,{}_{I'}\mathcal{U}_I\, \phi_2$ at time t of a timed state sequence is that there is a later time $t' \in (t+I)$ such that ϕ_2 holds at time t' and ϕ_1 holds throughout the interval $(t+I') \cap (t,t')$. This requirement can be expressed in temporal logics that admit explicit references to time through variables (say, in the style of the discrete-time logic TPTL of [AH94]). This extension, however, leads again to undecidability over a dense time domain. This is because the role of the formula $\Diamond_{=1}\,\phi$ in the undecidability argument for $\text{MITL}_=$ can be replaced by the formula *false* ${}_{\geq 1}\mathcal{U}_{\geq 1}\,\phi$.

3 Timed Automata

We use a variant of timed automata as defined in [AD94] to model (finite-state) real-time systems. Timed automata generalize nondeterministic finite automata over infinite strings. While ω-automata accept infinite state sequences (see, for example, [Tho90]), timed automata are additionally constrained by timing requirements and accept *timed* state sequences.

3.1 Definition of timed automata

A timed automaton operates with finite control—a finite set of control locations 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 start (or reset) some of the clocks. Each control location of the automaton puts constraints on the values of the propositions and on the values of the clocks: the control of the automaton can reside in a particular location iff the values of the propositions and clocks satisfy the corresponding constraints.

3.1.1 Syntax

We permit arbitrary constraints on the values of propositions. A *propositional constraint*, then, is a set of states. We usually denote propositional constraints as boolean combinations of propositions. For instance, we write $p \wedge \neg q$ for the set of states that contain p but not q.

We permit only simple constraints on the clock values. A *clock constraint* $\mathcal{I} \subseteq \mathbb{R}_{\geq 0}$ is a finite union of (possibly unbounded) intervals with integer end-points. The value $\sigma(x) \in \mathbb{R}_{\geq 0}$ of the clock x satisfies the clock constraint $\mathcal{I}$ iff $\sigma(x) \in \mathcal{I}$. We usually denote clock constraints for the clock x as boolean combinations of arithmetic expressions containing x. For instance, we write $1 \leq x < 3 \vee x = 4 \vee x > 5$ for the clock constraint $[1, 3) \cup [4, 4] \cup (5, \infty)$ that restricts the value of x.

Definition 3.1 A *timed automaton* A is a tuple $(V, V^0, \alpha, X, \beta, E)$ with the following components:

- V is a finite set of (control) locations.
- $V^0 \subseteq V$ is a set of initial locations.
- α is a location labeling function that assigns to each location v a propositional constraint $\alpha(v) \subseteq 2^P$.
- X is a finite set of clocks.
- β is a location labeling function that assigns to each location v and each clock x a clock constraint $\beta(v, x) \subseteq \mathbb{R}_{\geq 0}$. A clock valuation σ for the clocks in X *satisfies* the family $\beta(v)$ of clock constraints for the location $v \in V$ iff for all clocks $x \in X$, $\sigma(x) \in \beta(v, x)$.
- $E \subseteq V \times V \times 2^X$ is a set of transitions. Each transition $(v, v', \gamma) \in E$, also denoted $v \xrightarrow{\gamma} v'$, consists of a source location v, a target location v', and a set γ of clocks that are reset with the transition.

□

3.1.2 Semantics

At every time instant during a run of the timed automaton A, the configuration of A is completely determined by the location in which the control resides and by the values of all clocks. The clock values are given by *clock interpretations*, which are functions from X to $\mathbb{R}_{\geq 0}$: the value of the clock x under the clock interpretation σ is $\sigma(x) \in \mathbb{R}_{\geq 0}$. Given a clock interpretation σ and $t \in \mathbb{R}_{\geq 0}$, we write $\sigma + t$ for the clock interpretation that assigns to each clock $x \in X$ the value $\sigma(x) + t$. For $\gamma \subseteq X$, by $\sigma[\gamma := 0]$ we denote the clock interpretation that assigns 0 to all clocks in γ, and $\sigma(x)$ to all other clocks $x \notin \gamma$. We write Σ for the set of clock interpretations for the automaton A.

Assume that, at time $t \in \mathbb{R}_{\geq 0}$, the control of A resides in the location v and the clock values are given by the clock interpretation σ. Suppose that the control location of the automaton remains unchanged during the time interval I with $\ell(I) = t$. All clocks proceed at the rate at which time elapses. At all times $t' \in I$, the value of each clock x is $\sigma(x) + (t' - t)$; so the clock interpretation at time t' is $\sigma + (t' - t)$. Throughout the interval I the clock interpretation satisfies the clock constraint that is associated with the location v; that is, for each clock x, $\sigma(x) + (t' - t) \in \beta(v, x)$. Now suppose that the automaton changes its control location at time $r(I) = t''$ via the transition $v \xrightarrow{\gamma} v'$. This location change happens in one of two possible ways. If I is right-closed, then the location at time t'' is still v; otherwise the location at time t'' is already v'. The clocks in γ are reset at time t''. Let σ'' be the clock interpretation $(\sigma + (t'' - t))[\gamma := 0]$. The clock interpretation at the transition time t'' depends on whether the location at time t'' is v or v'. If I is right-closed, then the clock interpretation at time t'' is $\sigma + (t'' - t)$ and must satisfy $\beta(v)$. If I is right-open, then the clock values at time t'' are given by σ'' and must satisfy $\beta(v')$. The new location v' stays unchanged during some time interval adjacent to I, and the same cycle repeats.

DEFINITION 3.2 A *run* ρ of the timed automaton A is an infinite sequence

$$\xrightarrow[\sigma_0]{} (v_0, I_0) \xrightarrow[\sigma_1]{\gamma_1} (v_1, I_1) \xrightarrow[\sigma_2]{\gamma_2} (v_2, I_2) \xrightarrow[\sigma_3]{\gamma_3} \cdots$$

of locations $v_i \in V$, intervals I_i that form an interval sequence $\overline{I}_\rho = I_0 I_1 I_2 \ldots$, clock sets $\gamma_i \subseteq X$, and clock interpretations $\sigma_i \in \Sigma$ satisfying the following constraints:

- [*Initiality*] $v_0 \in V^0$.
- [*Consecution*] For all $i \geq 0$, $(v_i, v_{i+1}, \gamma_{i+1}) \in E$ and $\sigma_{i+1} = (\sigma_i + (r(I_i) - \ell(I_i)))[\gamma_{i+1} := 0]$. For $i \geq 0$ and $t \in I_i$, the location $v_\rho(t)$ at time t is v_i, and the clock interpretation $\sigma_\rho(t)$ at time t is $\sigma_i + (t - \ell(I_i))$.
- [*Timing*] For all $t \in \mathbb{R}_{\geq 0}$, $\sigma_\rho(t)$ satisfies $\beta(v_\rho(t))$.

□

According to this definition, in a run the clocks may start at any real values that satisfy the clock constraints of an initial location. The run ρ defines a function v_ρ from the time domain $\mathbb{R}_{\geq 0}$ to the control locations V, and a function σ_ρ from $\mathbb{R}_{\geq 0}$ to the clock interpretations Σ, providing both a control location and clock values at every time instant. The location v is *reachable* in A iff $v = v_\rho(t)$ for some run ρ of A and some $t \in \mathbb{R}_{\geq 0}$.

Every run of a timed automaton generates timed state sequences. Singular states can be enforced by clock constraints of the form $x = c$, for a constant c.

DEFINITION 3.3 The run ρ of the timed automaton A *generates* all timed state sequences τ of the form $(\overline{s}, \overline{I}_\rho)$ such that for all $t \in \mathbb{R}_{\geq 0}$, $\tau^*(t) \in \alpha(v_\rho(t))$. The timed automaton A *accepts* the timed state sequence τ iff τ is equivalent to a timed state sequence that is generated by a run of A. We write $L(A)$ for the set of timed state sequences accepted by A. The *emptiness problem* for timed automata is to decide whether or not a given timed automaton accepts any timed state sequence. □

3.1.3 Examples

EXAMPLE 3.4 The timed automaton A_1 of Figure 1 has four control locations, v_0 to v_3, and one clock, x. In figures, we suppress the trivial propositional constraint *true* and trivial clock constraints denoting $\mathbb{R}_{\geq 0}$.

The automaton A_1 starts in the initial location v_0 with the clock x set to 0 and the proposition p true. During the time interval $(0, 3)$, the automaton loops finitely, but arbitrarily, often between the locations v_1 and v_2 and, thus, the proposition p may change its value finitely often in the interval $(0, 3)$. At time 3 the automaton moves to the location v_3 to check that p is true. The clock x is reset at this point and the whole cycle repeats. Thus the automaton A_1 requires p to hold at all time instants that are integer multiples of 3. A sample prefix of a run of A_1 is

$$\xrightarrow[0]{} (v_0, [0, 0]) \xrightarrow[0]{\emptyset} (v_1, (0, 1.1)) \xrightarrow[1.1]{\emptyset} (v_2, [1.1, 3)) \xrightarrow[3]{\emptyset} (v_3, [3, 3]) \xrightarrow[0]{\{x\}} (v_2, (3, 4]) \rightarrow \cdots$$

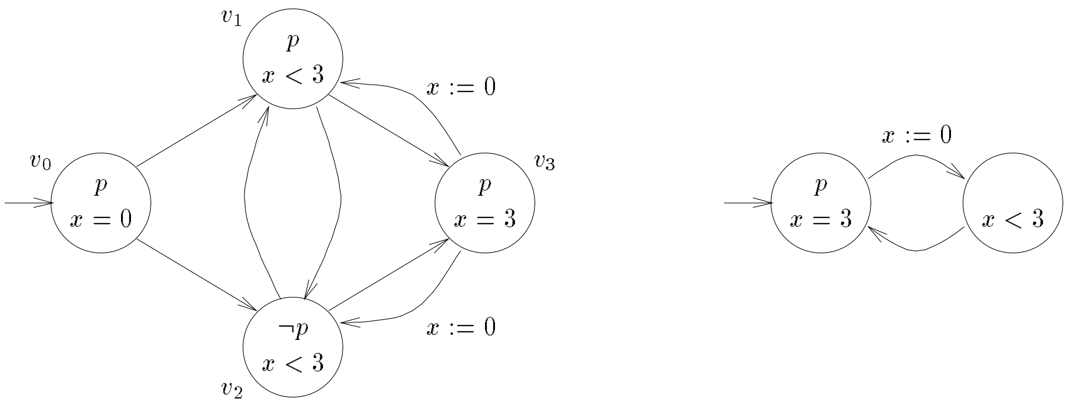

Figure 1: The timed automata A_1 and A'_1

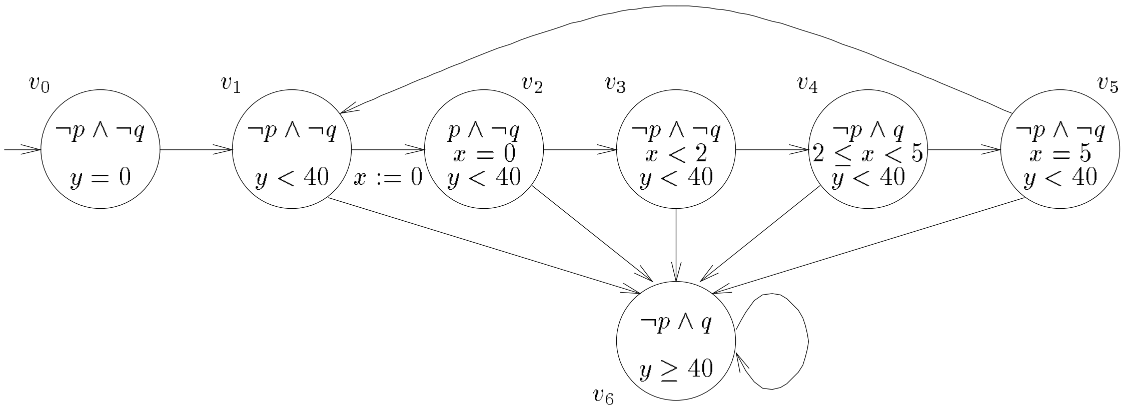

Figure 2: The timed automaton A_2

(since there is only one clock, clock interpretations are given as values for x). The set of timed state sequences accepted by A_1 is

$$L(A_1) \;=\; \{\tau \mid \forall n \in \mathbb{N}.\, p \in \tau^*(3 \cdot n)\}.$$

The timed automaton A'_1 of Figure 1 accepts the same timed state sequences as A_1. $\square$

EXAMPLE 3.5 The timed automaton A_2 of Figure 2 has seven control locations, v_0 to v_6, and two clocks, x and y.

The automaton A_2 starts in the initial location v_0 with the clock y set to 0. At time 40 the automaton moves to the location v_6 and stays there. The proposition p denotes an external event that is true only at instantaneous points $t < 40$ in time (and no more than once every 5 time units), namely, whenever the automaton is in the location v_2. The automaton responds to p by resetting the clock x, and then it requires that the proposition q holds over the interval $t + [2, 5)$. Thus the automaton A_2 models a system that responds, until time 40, to the event p by setting q to true for the interval $[2, 5)$ following p. A sample timed state sequence accepted by A_2 is

$$(\emptyset, [0, 13)) \;\to\; (\{p\}, [13, 13]) \;\to\; (\emptyset, (13, 15)) \;\to\; (\{q\}, [15, 20)) \;\to\; (\emptyset, [20, 40)) \;\to\; (\{q\}, [40, \infty)).$$

All timed state sequences that are accepted by A_2 satisfy the MITL-formula

$$\Box_{<40}\,(p \;\rightarrow\; \Box_{[2,5)}\,q) \;\wedge\; \Box_{\geq 40}\,q.$$

□

3.2 Checking emptiness

The emptiness problem for timed automata is solved in [AD94]. The algorithm given there can be adapted in a straightforward way to our variant of timed automata. We only sketch the basic idea behind the construction, and refer to [AD94] for the details.

Consider the timed automaton $A = (V, V^0, \alpha, X, \beta, E)$. With A we associate a transition relation $\Rightarrow$ over the space $V \times \Sigma$ of automaton configurations: $(v, \sigma) \Rightarrow (v', \sigma')$ iff there exist two times $t, t' \in \mathbb{R}_{\geq 0}$, a clock interpretation $\sigma'' \in \Sigma$, and a transition $v \xrightarrow{\gamma} v'$ such that

— $\sigma'' = (\sigma + t)[\gamma := 0]$;
— $\sigma' = \sigma'' + t'$;
— for all $t'' \in [0, t)$, $\sigma + t''$ satisfies $\beta(v)$;
— for all $t'' \in (0, t']$, $\sigma'' + t''$ satisfies $\beta(v')$;
— either $\sigma + t$ satisfies $\beta(v)$ or σ'' satisfies $\beta(v')$.

Now the emptiness problem for A can be reduced to a search problem over the infinite graph $(V \times \Sigma, \Rightarrow)$, whose solution rests on the observation that the uncountable configuration space $V \times \Sigma$ can be partitioned into finitely many equivalence classes as follows. Informally, two automaton configurations are equivalent iff they agree on the location components, on the integral parts of all clock values, and on the ordering of the fractional parts of all clock values. For each clock $x \in X$, let c_x be the largest constant such that c_x is an end-point of an interval in a clock constraint for x at any location of A. We define $(v, \sigma) \approx (v', \sigma')$ iff for all clocks x and y,

(1) $v = v'$;
(2) either $\lfloor \sigma(x) \rfloor = \lfloor \sigma'(x) \rfloor$, or both $\sigma(x) > c_x$ and $\sigma'(x) > c_x$;
(3) $\langle \sigma(x) \rangle \leq \langle \sigma(y) \rangle$ iff $\langle \sigma'(x) \rangle \leq \langle \sigma'(y) \rangle$ (recall that $\langle t \rangle = t - \lfloor t \rfloor$);
(4) $\langle \sigma(x) \rangle = 0$ iff $\langle \sigma'(x) \rangle = 0$.

The equivalence relation $\approx$ has two key properties:

1. If $(v_1, \sigma_1) \approx (v_1', \sigma_1')$ and $(v_1, \sigma_1) \Rightarrow (v_2, \sigma_2)$, then there is a configuration (v_2', σ_2') such that $(v_2, \sigma_2) \approx (v_2', \sigma_2')$ and $(v_1', \sigma_1') \Rightarrow (v_2', \sigma_2')$ (that is, $\approx$ is a bisimulation).

2. The number of equivalence classes of $\approx$ is finite, namely, $O(|V| \cdot |X|! \cdot \Pi_{x \in X} c_x)$.

The *region graph* for the timed automaton A is the quotient of the infinite graph $(V \times \Sigma, \Rightarrow)$ with respect to the equivalence $\approx$. The first property allows us to reduce reachability problems over $(V \times \Sigma, \Rightarrow)$ to reachability problems over the region graph. The second property ensures that the region graph is finite.

It follows that the emptiness problem for timed automata can be solved in time $O((|V| + |E|) \cdot |X|! \cdot \Pi_{x \in X} c_x)$. That is, the complexity is linear in the size of the location-transition graph, exponential in the number of clocks, and exponential in the (binary) encoding of the largest constant appearing in clock constraints. For containment in PSPACE, the emptiness of the region graph can be checked nondeterministically while storing only a constant number of vertices. Each vertex of the region graph is described by listing a control location and a set of clock constraints. The description of each vertex requires space logarithmic in $|V|$, polynomial in $|X|$, and polynomial in the encoding of the largest constant. It follows that the emptiness problem for timed automata is in PSPACE. The PSPACE-hardness follows from the corresponding result proved in [AD94].

THEOREM 3.6 *The emptiness problem for timed automata is PSPACE-complete.*

3.3 Parallel composition

For describing real-time systems it is useful to describe individual system components separately. Timed automata that describe system components can be put together using the following product construction.

THEOREM 3.7 *Let* $A_1 = (V_1, V_1^0, \alpha_1, X_1, \beta_1, E_1)$ *and* $A_2 = (V_2, V_2^0, \alpha_2, X_2, \beta_2, E_2)$ *be two timed automata. There exists a timed automaton* $A_1 \times A_2$ *such that* $L(A_1 \times A_2) = L(A_1) \cap L(A_2)$.

PROOF. Assuming that the clock sets X_1 and X_2 are disjoint (this can always be achieved by renaming clocks), we define the product automaton $A_1 \times A_2 = (V, V^0, \alpha, X, \beta, E)$ as follows. The location set V is the set $V_1 \times V_2$ of location pairs. The set V^0 of initial locations is the set $V_1^0 \times V_2^0$ of pairs of initial locations. The propositional constraint $\alpha(v_1, v_2)$ is the intersection $\alpha_1(v_1) \cap \alpha_2(v_2)$ of the component constraints. The clock set X is the (disjoint) union $X_1 \cup X_2$ of clock sets. For each clock $x_1 \in X_1$, the clock constraint $\beta((v_1, v_2), x_1)$ is $\beta_1(v_1, x_1)$; and for each clock $x_2 \in X_2$, the clock constraint $\beta((v_1, v_2), x_2)$ is $\beta_2(v_2, x_2)$. For every pair of transitions $v_1 \stackrel{\gamma_1}{\rightarrow} v_1'$ and $v_2 \stackrel{\gamma_2}{\rightarrow} v_2'$ of A_1 and A_2, respectively, the product automaton has *three* transitions: $(v_1, v_2) \stackrel{\gamma_1 \cup \gamma_2}{\longrightarrow} (v_1', v_2')$, $(v_1, v_2) \stackrel{\gamma_1}{\rightarrow} (v_1', v_2)$, and $(v_1, v_2) \stackrel{\gamma_2}{\rightarrow} (v_1, v_2')$. Thus the transitions in E simulate the joint execution of the two component automata. □

3.4 Fairness requirements

When verifying reactive systems, we are generally interested only in properties of the *fair* executions [MP92]. For example, for a system with two processes, we may wish to consider only those behaviors in which each process executes infinitely often. While concrete timing can usually replace abstract fairness, we need to consider fair timed automata for solving the satisfiability problem for MITL. We add fairness to timed automata using *generalized Büchi conditions*.

DEFINITION 3.8 A *fairness requirement* for the timed automaton A is a set of locations of A. A *fairness condition* for A is a set of fairness requirements for A. A *fair timed automaton* B consists of a timed automaton A and a fairness condition $\mathcal{F}$ for A. The run

$$\underset{\sigma_0}{\rightarrow} \ (v_0, I_0) \ \underset{\sigma_1}{\stackrel{\gamma_1}{\rightarrow}} \ (v_1, I_1) \ \underset{\sigma_2}{\stackrel{\gamma_2}{\rightarrow}} \ (v_2, I_2) \ \underset{\sigma_3}{\stackrel{\gamma_3}{\rightarrow}} \ \cdots$$

of A is a *fair run* of B iff for all fairness requirements $F \in \mathcal{F}$ there are infinitely many $i \geq 0$ with $v_i \in F$. The fair timed automaton B *accepts* the timed state sequence τ iff τ is equivalent to a timed state sequence that is generated by a fair run of A. By $L(B)$ we denote the set of timed state sequences accepted by B. □

The algorithm for checking the emptiness of timed automata can be extended to handle fairness conditions in the standard way [AD94]. In particular, it can be decided in PSPACE whether or not a given fair timed automaton accepts any timed state sequence. Similarly, the product construction for timed automata is easily extended to fair timed automata.

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 construct a fair timed automaton B_ϕ that accepts precisely the models of ϕ.

In the following, let $K \in \mathbb{N}$ be such that $K - 1$ is the largest integer constant that appears in the given formula ϕ, and let $N \in \mathbb{N}$ be the number of propositions, boolean connectives, and temporal operators in ϕ.

4.1 Preliminary transformations

We begin with making some assumptions that can be made without loss of generality, and without extra cost. First, we assume that the given formula ϕ is in normal form. Second, it suffices that all runs of B_ϕ generate timed state sequences that are in ϕ-normal form.

4.1.1 Normal-form formulas

DEFINITION 4.1 The MITL-formula ϕ is in *normal form* iff it is built from propositions and negated propositions using conjunction, disjunction, and temporal subformulas of the following six types:

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

□

Using a series of four transformations, every MITL-formula ϕ can be transformed into an equivalent formula ϕ^* in normal form. First, every interval must not contain 0. This can be achieved by applying the following equivalence: if $0 \in I$, then

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

Second, all unbounded intervals are of the form $(0, \infty)$. This can be achieved by applying the following two equivalences: if $a > 0$, then

$$\begin{aligned} \psi_1 \mathcal{U}_{(a,\infty)} \psi_2 \quad &\leftrightarrow \quad \Box_{(0,a]} (\psi_1 \wedge \psi_1 \mathcal{U} \psi_2), \\ \psi_1 \mathcal{U}_{[a,\infty)} \psi_2 \quad &\leftrightarrow \quad \Box_{(0,a)} \psi_1 \wedge \Box_{(0,a]} (\psi_2 \vee (\psi_1 \wedge \psi_1 \mathcal{U} \psi_2)). \end{aligned}$$

Third, only *eventually* and *always* operators are constrained with bounded intervals I such that $\ell(I) = 0$. This can be achieved by applying the following equivalence: if $\ell(I) = 0$, then

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

Fourth, we push all negations to the inside (see Section 2.3.3) and use the following equivalence to eliminate each subformula of the form $\psi_1 \mathcal{W} \psi_2$:

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

It is easy to check that the resulting formula ϕ^* is in normal form.

Observe that the number of distinct subformulas of ϕ^* is linear in the length of ϕ. This is because with each transformation step, only a constant number of new subformulas is created. Therefore, if formulas are represented as directed acyclic graphs, thus avoiding the duplication of shared subformulas, then the representation of ϕ^* is only a constant factor larger than the representation of ϕ.

LEMMA 4.2 *For every* MITL*-formula ϕ there exists an equivalent formula ϕ^* in normal form such that*

- *the largest constant in ϕ^* is the same as the largest constant $K-1$ in ϕ, and*
- *if N is the number of propositions, boolean connectives, and temporal operators in ϕ, then the number of distinct syntactic subformulas of ϕ^* is $O(N)$.*

Henceforth, we assume that all MITL-formulas under consideration are in normal form.

4.1.2 Normal-form models

DEFINITION 4.3 The timed state sequence τ is in *ϕ-normal form*, for the MITL-formula ϕ, iff (1) τ is ϕ-fine and (2) all intervals of τ are either singular or open. $\Box$

To check the satisfiability of ϕ, it suffices to consider timed state sequences in ϕ-normal form. First, by Lemma 2.11 and Remark 2.6, for every model of ϕ there is an equivalent ϕ-fine model. Second, by Remark 2.6, every ϕ-fine model of ϕ can be refined into an equivalent model in ϕ-normal form; for instance, the interval $[a,b)$ can be split into the two intervals $[a,a]$ and (a,b). It follows that ϕ is satisfiable iff ϕ has a model in ϕ-normal form.

Henceforth, we assume that all timed state sequences under consideration are in ϕ-normal form. It follows that, if $\tau = (\overline{s}, \overline{I})$, and ψ is a subformula of ϕ, we may write $\tau^i \models \psi$ for "$\tau^t \models \psi$ for all $t \in I_i$." We also introduce a new proposition p_{sing} such that $\tau^i \models p_{sing}$ iff the interval I_i is singular; that is, iff i is even. Then:

- $\tau \models \psi_1 \mathcal{U}_I \psi_2$ iff for some i with $I_i \cap I \neq \emptyset$, (1) both $\tau^i \models \psi_2$ and $\tau^i \models \psi_1 \vee p_{sing}$, and (2) $\tau^j \models \psi_1$ for all $0 < j < i$, and (3) $\tau^0 \models \psi_1 \vee p_{sing}$.
- $\tau \models \psi_1 \,{}_I\mathcal{W} \psi_2$ iff $\tau^0 \models \psi_1$ if $I_0 \cap I \neq \emptyset$, and either (1) $\tau^0 \models \psi_2 \wedge \neg p_{sing}$, or (2) $\tau^i \models \psi_2$ for some $i > 0$, and $\tau^j \models \psi_1$ for all $0 < j \leq i$ with $I_j \cap I \neq \emptyset$, or (3) $\tau^j \models \psi_1$ for all $j > 0$ with $I_j \cap I \neq \emptyset$.

4.2 Checking subformulas

The six types of temporal subformulas of ϕ are handled differently. 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 involved cases are those of type-3 and type-4 formulas. We begin with the simpler cases of type-1 and type-2 formulas.

4.2.1 Type-1 and type-2 formulas

Type 1. Consider the type-1 formula $\psi = \Diamond_I \psi'$, where $I = (0, b)$ or $I = (0, b]$. Whenever the automaton B_ϕ needs to check that ψ holds, say at time t, it starts a clock x and writes a proof obligation into its memory—namely, the obligation to verify that ψ' holds at some later location 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 $\tau^t \models \Diamond_I \psi'$ and $\tau^{t'} \models \Diamond_I \psi'$ (recall that $\ell(I) = 0$). Once the proof obligation is discharged, the clock x can be reused. Thus one clock suffices to check the formula ψ as often as necessary.

This strategy works for checking the truth of ψ at singular intervals. There is, however, a subtle complication 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 in the open interval $(0, 1)$. To check the truth of $\Diamond_{(0,1)} p$ during the interval $(0, 1)$, the automaton starts a clock x upon entry, at time 0. However, the proof obligation that p holds at some later location with the clock constraint $x \in (0, 1)$ 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 $(0, 1)$, 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 truth of $\Diamond_{(0,1]} p$ throughout the open interval $(0, 1)$ implies that $\Diamond_{(0,1)} p$ is also true throughout the interval $(0, 1)$.

Lemma 4.4 *Let ψ and $\hat{\psi}$ be the type-1 formulas $\Diamond_I \psi'$ and $\Diamond_{I \cup \{r(I)\}} \psi'$, respectively. For every timed state sequence $\tau = (\overline{s}, \overline{I})$ and every open interval I_i in $\overline{I}$, $\tau^i \models \psi$ iff $\tau^i \models \hat{\psi}$.*

Proof. First note that, for all $t \geq 0$, if $\tau^t \models \psi$ then $\tau^t \models \hat{\psi}$. This is because $I \subseteq I \cup \{r(I)\}$.

Now consider an open interval I_i and assume that $\tau^i \models \hat{\psi}$. If I is right-closed, then $\psi = \hat{\psi}$. So suppose that I is right-open, and let $t \in I_i$. Since I_i is open, there exists some $t' \in I_i$ with $t' < t$. Since $\tau^{t'} \models \hat{\psi}$, there exists some $j \geq i$ such that $I_j \cap (t' + (I \cup \{r(I)\})) \neq \emptyset$ and $\tau^j \models \psi'$. It follows that $I_j \cap (t + I) \neq \emptyset$ and, hence, that $\tau^t \models \psi$. $\Box$

Consequently, to check the truth of a type-1 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.

Type 2. For checking the type-2 formula $\psi = \Box_I \psi'$, where $I = (0, b)$ or $I = (0, b]$, the situation is symmetric. The automaton uses again 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 proof 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 $\tau^{t'} \models \Box_I \psi'$, then also $\tau^t \models \Box_I \psi'$. Once the proof obligation is discharged, the clock x can be reused to check ψ again whenever necessary.

As in the case of type-1 formulas, we need to be more careful when checking ψ during open intervals. For the type-2 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 $\tau = (\overline{s}, \overline{I})$, if I_i is open, then $\tau^i \models \psi$ iff $\tau^i \models \hat{\psi}$. 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.

4.2.2 Type-3 and type-4 formulas

A sample construction. Consider the MITL-formula

$$\phi_0 \;=\; \Box_{(0,1)}\,(p \;\rightarrow\; \Diamond_{[1,2]}\,q).$$

The subformula $\Diamond_{[1,2]}\,q$ is a type-3 formula, because the left end-point of the interval $[1,2]$ is greater than 0. Let us assume, for simplicity, that both p and q are true in singular intervals only. Furthermore, we assume that there is at least one q-state in the time interval $(1,2)$, at least one q-state in the time interval $(2,3)$, and q is false at time 2. Let us try to build a timed automaton B_{ϕ_0} that accepts, under these assumptions, precisely the models of ϕ_0.

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 starting a clock x when the p-state is visited, and demanding that some q-state is visited later in a location with the clock constraint $1 \leq x \leq 2$. This strategy requires one clock per visit to a p-state within the interval $(0,1)$. The number of such visits, however, is potentially unbounded and, hence, there is no automaton with a fixed number of clocks that can start a new clock with every visit: this simple strategy cannot be made to work.

Instead, we have the automaton guess the times of future q-states in advance. The automaton nondeterministically guesses two times t_1 and t_2 within the interval $(0,1)$; this is done by starting 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 (both such states exist by assumption). 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). The resulting automaton is shown in Figure 3. The clock z is used to count the global time. If the guessed value of t_1 is smaller than t_2—i.e., the clock x is started before the clock y—then the automaton requires that there are no p states in the interval (t_1,t_2). Later the automaton checks that its guesses are correct: if $x=1$ or $y=2$, then q is required to hold; and between $x=1$ and $y=2$, q is required not to hold.

The strategy of guessing times of future q-states 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, it is necessary to have witnesses that may be open intervals, instead of singular intervals. To see this, let us relax the assumption that q holds only in singular intervals. Let $0 < t_1 < t_1' < 1$ be such that q is true during the interval $I_1 = (t_1+1, t_1'+1)$, and false during the interval $[t_1'+1, 2]$. Let $0 < t_2 < t_2' < 1$ be such that q is false during $[2, t_2+2]$ and true during $I_2 = (t_2+2, t_2'+2)$. Thus I_1 is the last q-interval within $(1,2)$, and I_2 is the first q-interval within $(2,3)$. The formula $\Diamond_{[1,2]}\,q$ holds during the intervals $(0,t_1')$ and $(t_2,1)$, and does not hold during the interval $[t_1',t_2]$. To check the formula ϕ_0, then, the automaton B_{ϕ_0} must nondeterministically guess *four* times, t_1, t_1', t_2, and t_2': it requires that no p-state lies within $[t_1',t_2]$, and it checks that the guesses are correct. In this case we say that the intervals I_1 and I_2 witness the formula $\Diamond_{[1,2]}\,q$ throughout the intervals $(0,t_1')$ and $(t_2,1)$, respectively. Notice that we cannot choose a particular time instant from I_1 as a witness for $(0,t_1')$; only if I_1 is right-closed can we choose its right end-point as the witness.

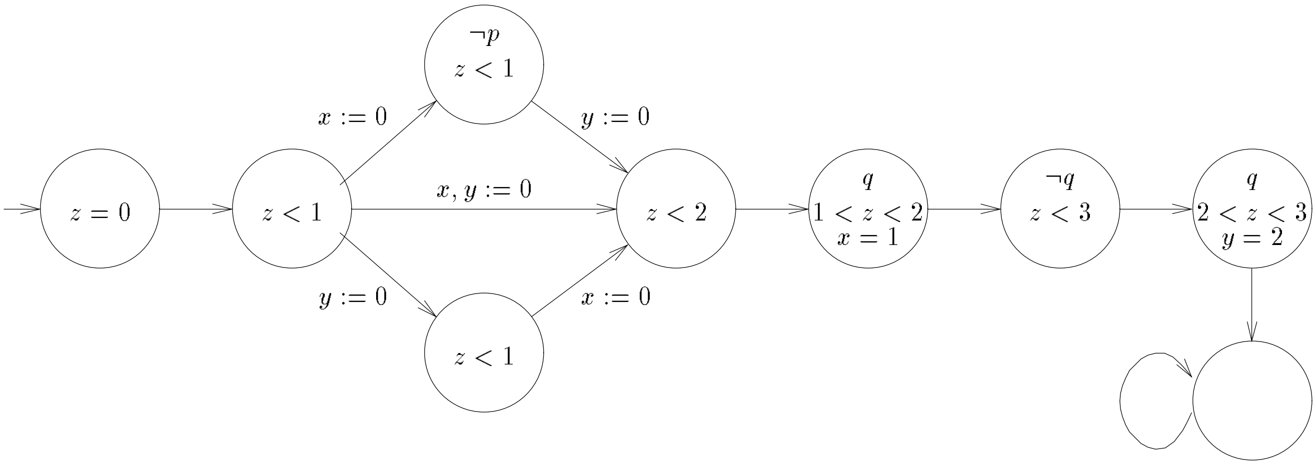

Figure 3: The timed automaton B_{ϕ_0}

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. In particular, the strategy fails if we were to allow singular intervals in formulas. Consider, for example, the formula

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

There, for each p-state at time t in the interval $(0,1)$, the automaton needs to check that there is a q-state at time $t+1$. The strategy of guessing witnesses is not helpful: a q-state at time t in the interval $(1,2)$ serves as a witness only for the single time instant $t-1$; hence there is a potentially unbounded number of witnesses.

Witnessing intervals.

Definition 4.5 Let τ be a timed state sequence and let $t \in \mathbb{R}_{\geq 0}$. The interval I' *witnesses* the type-3 formula $\psi_1 \mathcal{U}_I\, \psi_2$ under τ^t iff $I' \cap (t+I) \neq \emptyset$ and $\tau^t \models \psi_1 \mathcal{U}_{J-t}\, \psi_2$ for every nonempty interval $J \subseteq I'$. The interval I' *witnesses* the type-4 formula $\psi_1\, {}_I\mathcal{W} \psi_2$ under τ^t iff $(t+I) \subseteq I'$ and $\tau^t \models \psi_1\, {}_{I'-t}\mathcal{W} \psi_2$. $\Box$

Observe that if I' witnesses $\psi_1 \mathcal{U}_I\, \psi_2$ under τ^t, then ψ_1 holds throughout the interval $(t, r(I'))$, and ψ_2 holds throughout the interval I'. Witnessing intervals are defined to have the following property.

Lemma 4.6 *Let ψ be a type-3 or type-4 formula, let τ be a timed state sequence, and let $t \in \mathbb{R}_{\geq 0}$. There is a witnessing interval for ψ under τ^t iff $\tau^t \models \psi$.*

Proof. If $\tau^t \models \psi$ for the type-3 formula $\psi = \psi_1 \mathcal{U}_I\, \psi_2$, then $\tau^{t'} \models \psi_2$ for some $t' \in t+I$ and the singular interval $[t', t']$ witnesses ψ under τ^t. If $\tau^t \models \psi$ for the type-4 formula $\psi = \psi_1\, {}_I\mathcal{W} \psi_2$, then the interval $t+I$ witnesses ψ under τ^t.

The other direction of the lemma follows from the semantic clauses for the *until* and *unless* operators. $\Box$

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

Type 3.

EXAMPLE 4.7 Consider the following timed state sequence τ over the two propositions p and q:

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

Along τ the proposition p is always true, and 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 τ^t for every time $t \in [0, 0.6)$. On the other hand, the interval $[1.6, 3]$ witnesses the formula $\Box_{(1,2)} \neg q$ under τ^t for every time $t \in [0.6, 1]$. $\Box$

LEMMA 4.8 *Let ψ be a type-3 formula. For every timed state sequence τ, there are two bounded (singular or open) intervals I' and I'' such that, for all $t \in [0, 1)$, $\tau^t \models \psi$ iff either I' or I'' witnesses ψ under τ^t. Furthermore, $r(I') \leq r(I) + 1$ and $r(I'') \leq r(I) + 1$.*

PROOF. Let $\tau = (\overline{s}, \overline{I})$ 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). Recall that the $\ell(I)$ and $r(I)$ are integers such that $r(I) > \ell(I) > 0$. We choose the two intervals I' and I'' as follows:

- Let $\hat{\imath}$ be the *maximal* $i \geq 0$ such that $I_i \cap I \neq \emptyset$, both $\tau^i \models \psi_2$ and $\tau^i \models \psi_1 \vee p_{sing}$, and $\tau^k \models \psi_1$ for all $0 \leq k < i$. If no such i exists, let $I' = \emptyset$; otherwise, let $I' = I_{\hat{\imath}}$.
- Let $\hat{\jmath}$ be the *minimal* $j \geq 0$ such that $I_j \cap (1 + I) \neq \emptyset$, both $\tau^j \models \psi_2$ and $\tau^j \models \psi_1 \vee p_{sing}$, and $\tau^k \models \psi_1$ for all $0 \leq k < j$ with $I_k \cap (< (1 + I)) \neq \emptyset$. If no such j exists, let $I'' = \emptyset$; otherwise, let $I'' = I_{\hat{\jmath}}$.

Assume that $\tau^t \models \psi$ for some $t \in [0, 1)$. Then $\tau^{t'} \models \psi_1$ for all $t' < I$. If $I' \cap (t + I) \neq \emptyset$, then I' witnesses ψ under τ^t. Otherwise, let $t' \in (t + I)$ be such that $\tau^{t'} \models \psi_2$ and $\tau^{t''} \models \psi_1$ for all $t'' \in (t, t')$. In this case I'' is nonempty, and if $t' \in I_k$ then $\hat{\jmath} \leq k$. Hence $I_{\hat{\jmath}} \cap (t + I) \neq \emptyset$, and I'' witnesses ψ under τ^t.

Conversely, if either I' or I'' witnesses ψ under τ^t, then $\tau^t \models \psi$ by Lemma 4.6. $\Box$

Now we can be more precise about how we construct the timed automaton B_ϕ that accepts the models of ϕ. To check the truth of type-3 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 σ, the clock interval $C = [x, y]$, for two clocks x and y, represents the closed witnessing interval $[t + K - \sigma(x), t + K - \sigma(y)]$; the clock interval $C = [x, y)$ represents the corresponding half-open interval, etc. (recall that $K - 1$ is the largest constant appearing in ϕ). We write $K - C$ for the interval $\{K - \sigma(x), K - \sigma(y)\}$, for any type of clock interval $C = \{x, y\}$.

For simplicity, let us consider a type-3 formula ψ of the form $\Diamond_I \psi'$ (with $\ell(I) > 0$). The automaton starts, 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 is started at time t_1, and y is started at time $t_2 \geq t_1$, then the guess is that the interval $I' = \{t_1 + K, t_2 + K\}$ witnesses ψ. To check the truth of ψ at time $t \geq t_2$, the automaton needs to check 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 true throughout the interval I'; that is, the automaton needs to verify that ψ' holds at all states for which the clock constraint $0 \in (K - C)$ is true.

Lemma 4.8 is the key to constructing an automaton that needs only *finitely* many clocks. For each type-3 formula $\psi = \psi_1 \mathcal{U}_I \psi_2$, at most two witnessing intervals need to be guessed per time 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, to check the formula ψ everywhere, we need, at any point in time, at most $2r(I) + 2$ active clock intervals; that is, clock intervals that represent guesses of witnessing intervals and, therefore, have to be verified later. Consequently, $4K$ clocks suffice to check any type-3 subformula of ϕ.

Type 4.

LEMMA 4.9 *Let ψ be a type-4 formula. For every timed state sequence τ, there is a bounded interval I' such that, for all $t \in [0, 1)$, $\tau^t \models \psi$ iff either τ^t satisfies the type-1 formula $\Diamond_{(0,\infty)\cap(<I)} \psi_2$ or I' witnesses ψ under τ^t. Furthermore, $r(I') \leq r(I) + 1$.*

PROOF. Let $\tau = (\overline{s}, \overline{I})$ be a ψ-fine timed state sequence with only singular and open intervals, including the singular interval $I_n = [r(I) + 1, r(I) + 1]$. We choose the interval I' as follows:

- Let $\hat{\imath}$ be the minimal $i \geq 0$ such that $I_i \cap I \neq \emptyset$ and either
  1. $\tau^k \models \psi_1$ for all $k \geq i$ with $I_k \cap I \neq \emptyset$, or
  2. there is some $i \leq j \leq n$ such that $\tau^j \models \psi_1 \wedge \psi_2$, and $\tau^k \models \psi_1$ for all $i \leq k < j$.
- Given $\hat{\imath}$, let $\hat{\jmath}$ be the maximal $\hat{\imath} \leq j \leq n$ such that either $\tau^k \models \psi_1$ for all $\hat{\imath} \leq k \leq j$, or $\tau^k \models \psi_1 \wedge \psi_2$ for some $\hat{\imath} \leq k \leq j$. Note that if $\hat{\imath}$ exists, then so does $\hat{\jmath}$; in particular, if $\hat{\imath}$ exists because of clause 2, then $\hat{\jmath} = n$.

If no appropriate $\hat{\imath}$ exists, let $I' = \emptyset$; otherwise, let I' be the union of all I_k for $\hat{\imath} \leq k \leq \hat{\jmath}$.

Assume that $t \in [0, 1)$; then $\tau^t \models \psi$ iff either (1) $\tau^i \models \psi_1$ for all i with $I_i \cap (t + I) \neq \emptyset$, or (2) $\tau^i \models \psi_1 \wedge \psi_2$ for some i with $I_i \cap (t + I) \neq \emptyset$, and $\tau^j \models \psi_1$ for all $j < i$ with $I_j \cap (t + I) \neq \emptyset$, or (3) $\tau^{t'} \models \psi_2$ for some $t < t' < t + I$. In either of the first two cases, I' witnesses ψ under τ^t; the third case is equivalent to τ^t satisfying the formula $\Diamond_{(0,\infty)\cap(<I)} \psi_2$.

Conversely, if τ^t satisfies $\Diamond_{(0,\infty)\cap(<I)} \psi_2$, then $\tau^t \models \psi$. If I' witnesses ψ under τ^t, then $\tau^t \models \psi$ by Lemma 4.6. $\square$

It follows that for each type-4 formula, a single witness per unit interval suffices. Thus, to check the type-4 formula $\psi_1 \, {}_I\mathcal{W} \psi_2$, we need, at any point in time, no more than $r(I) + 1$ active clock intervals. Consequently, $2K$ clocks suffice to check any type-4 subformula of ϕ.

4.3 Constructing the timed automaton

Now we define the fair timed automaton B_ϕ. For type-1 and type-2 subformulas ψ of ϕ, the automaton uses one clock, x_ψ, per formula. For each type-3 subformula, the automaton uses $2K$ 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 $[x, y]$. By Lemma 4.8, for checking type-3 formulas we need only open and singular witnessing intervals. Thus, associated with each type-3 subformula ψ of ϕ, the automaton uses $4K$ clock intervals; they are denoted $C_1(\psi), \ldots, C_{4K}(\psi)$. For each type-4 subformula of ϕ, the automaton uses K clock pairs giving $4K$ clock intervals. In addition to these clocks, the automaton uses the clock x_{sing} to enforce that all runs alternate singular and open intervals.

4.3.1 Closure set

DEFINITION 4.10 The *closure set* $Closure(\phi)$ of the MITL-formula ϕ consists of the following elements:

- All subformulas of ϕ; for each proposition p appearing in ϕ, the negation $\neg p$; for each type-1 subformula $\psi = \Diamond_I \psi'$ of ϕ, the type-1 formula $\hat{\psi} = \Diamond_{I \cup \{r(I)\}} \psi'$; for each type-2 subformula $\psi = \Box_I \psi'$ of ϕ, the type-2 formula $\hat{\psi} = \Box_{I - \{r(I)\}} \psi'$; and for each type-4 subformula $\psi_1 \,{}_I\mathcal{W} \psi_2$ of ϕ, the type-1 formula $\Diamond_{(0,\infty) \cap (<I)} \psi_2$.

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

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

  It should be noticed that these conditions are indeed clock constraints. For instance, the condition $0 \in (K - [x, y))$ stands for the clock constraint $x \leq K \wedge y > K$; the condition $0 = (K - [x, y))$ is false.

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

□

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 $O(N \cdot K)$.

4.3.2 Automaton locations

The control locations of B_ϕ are the subsets of $Closure(\phi)$. A location $v \subseteq Closure(\phi)$ is initial iff both ϕ and $x_{sing} = 0$ are in v. For each location v, the propositional constraint $\alpha(v)$ is the conjunction of all propositions and negated propositions in v. The clock constraint $\beta(v)$ is the conjunction of all clock constraints in v.

Notice that the propositional constraint of each location contains a single state. Hence every run of B_ϕ generates, up to equivalence, a unique timed state sequence. For each location v, the temporal formulas in v represent temporal conditions on the future of all runs through v. The clocks in v indicate which clocks are currently active and represent proof obligations for type-1 and type-2 formulas. The clock intervals in v indicate which clock intervals are currently active and represent witnessing intervals for type-3 and type-4 formulas.

4.3.3 Automaton transitions

The transitions of B_ϕ are the triples $v \xrightarrow{\gamma} v'$ that satisfy the following catalog of consistency criteria.

Logical consistency.

- For each proposition p in $Closure(\phi)$, precisely one of p and $\neg p$ is in v.

- If the formula $\psi_1 \wedge \psi_2$ is in v, then both ψ_1 and ψ_2 are in v.

- If the formula $\psi_1 \vee \psi_2$ is in v, then either ψ_1 or ψ_2 is in v.

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

Timing consistency.

- For each type-1 and type-2 formula ψ in $Closure(\phi)$, v contains at most one of the clock constraints $x_\psi \in I$ and $x_\psi > I$.
- For each clock interval C in $Closure(\phi)$, v contains at most one of the clock constraints $0 < (K - C)$, $0 \in (K - C)$, $0 = (K - C)$, and $(K - C) = \emptyset$. Furthermore, no two clock intervals in v share clocks; for instance, v does not contain both the clock intervals (x, y) and $[x, y]$.
- If v contains $x_{sing} = 0$, then $x_{sing} \notin \gamma$. If v does not contain $x_{sing} = 0$, then $x_{sing} \in \gamma$ and v' contains $x_{sing} = 0$.

These conditions ensure that no reachable location contains clock constraints that are mutually inconsistent. The location v is *singular* iff it contains the clock constraint $x_{sing} = 0$; otherwise we say that v is *open*. The third condition ensures that singular and open locations alternate along all runs.

Type-1 formulas. Let $\psi = \Diamond_I \psi'$ be a type-1 formula in $Closure(\phi)$.
First, if ψ is in v, then either

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

These conditions activate a clock to represent a proof obligation. Lemma 4.4 justifies the decision to start a clock corresponding to the weaker formula $\hat{\psi}$ when v is open.
Second, if x_ψ is in v, then

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

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

Type-2 formulas. Let $\psi = \Box_I \psi'$ be a type-2 formula in $Closure(\phi)$.
First, if ψ is in v, then either

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

These conditions activate a clock to represent a proof obligation, and reset it, as is justified in Section 4.2.1. Recall that if v is open, then instead of checking ψ it suffices to check $\hat{\psi}$.
Second, if x_ψ is in v then

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

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-3 formulas. Let $\psi = \psi_1 \mathcal{U}_I \psi_2$ be a type-3 formula in *Closure*(ϕ).

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

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

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. Note that if v is singular, the corresponding clock interval is activated only in the following open location. This is because, to check that the interval $K - C$ is indeed a witness, no conditions are required of a singular state.

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

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

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.

Type-4 formulas. Let $\psi = \psi_1 \, {}_I\mathcal{W} \psi_2$ be a type-4 formula in *Closure*(ϕ).

First, if ψ is in v, then either

1. $\Diamond_{(0,\infty)\cap(<I)} \psi_2$ is in v, or
2. there is some clock interval $C = C_j(\psi)$ such that
   - $I \subseteq (K - C)$ is in v, and
   - either C is in v, or v is singular and C is in v' and the clocks associated with C are not in γ.

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 this clock interval C to represent a witnessing interval for ψ.

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

- either $0 < (K - C)$ or $0 \in (K - C)$ or $0 = (K - C)$ is in v, and
- if either $0 \in (K - C)$ or $0 = (K - C)$ is in v, then ψ_1 is in v, and

- either ψ_2 is in v, or the clocks associated with C are not in γ and either C or $(K - C) = \emptyset$ is in v'.

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.

Type-5 formulas. Let $\psi = \psi_1 \mathcal{U} \psi_2$ be a type-5 formula in *Closure*(ϕ). Whenever ψ is in v, then either

- v is singular and ψ is in v', or
- v is open and ψ_1 is in v, and either ψ_2 is in v or ψ_2 is in v' or both ψ_1 and ψ are in v'.

These conditions ensure that unconstrained *until* formulas are propagated correctly (remember that singular and open intervals alternate). These conditions, however, admit the possibility that a run consists of locations containing ψ and ψ_1 without ever visiting a location containing ψ_2. We use a fairness requirement to ensure that whenever a run ρ visits a location v containing the type-5 formula ψ, then some later location v'' along ρ contains ψ_2.

Type-6 formulas. Let $\psi = \Box \psi'$ be a type-6 formula in *Closure*(ϕ). Whenever ψ is in v, then either

- v is singular and ψ is in v', or
- v is open and ψ' is in v and both ψ' and ψ are in v'.

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

4.3.4 Fairness requirements

For each type-5 formula $\psi = \psi_1 \mathcal{U} \psi_2$ in *Closure*(ϕ), we define the fairness requirement

$$F_\psi \;=\; \{v \subseteq \mathit{Closure}(\phi) \mid \psi_2 \in v \text{ or } \psi \notin v\}.$$

The fairness condition of B_ϕ consists of the fairness requirements F_ψ, one for each type-5 formula ψ in *Closure*(ϕ).

This concludes the definition of the fair timed automaton B_ϕ.

4.3.5 Correctness

The following main lemma states the correctness of our construction by relating the fair runs of the automaton B_ϕ to the models of the formula ϕ.

LEMMA 4.11 *For every* MITL*-formula ϕ in normal form,* $L(B_\phi) = L(\phi)$.

COROLLARY 4.12 *The* MITL*-formula ϕ is satisfiable iff* $L(B_\phi) \neq \emptyset$.

PROOF. Let ρ be a fair run of B_ϕ, and let τ_ρ be a timed state sequence that is generated by ρ. We first prove, by induction on the structure of ϕ, that for all formulas ψ in *Closure*(ϕ) and all $t \in \mathbb{R}_{\geq 0}$, if ψ is contained in the location $v_\rho(t)$, then $\tau_\rho^t \models \psi$. By Remark 2.6, it follows that the timed state sequences accepted by B_ϕ are models of ϕ.

We consider only the case that ψ is the type-3 formula $\psi_1 \mathcal{U}_I \psi_2$. Let $t \in \mathbb{R}_{\geq 0}$ and assume that ψ is contained in $v_\rho(t)$. Also assume that the clock interval $C = C_j(\psi)$ satisfies the consistency conditions for type-3 formulas in $v_\rho(t)$. By Lemma 4.6, it suffices to show that the interval $I' = t + (K - C)$ is a witness for ψ under τ_ρ^t. The clock constraint $(K - C) \cap I \neq \emptyset$ is in $v_\rho(t)$ and, therefore, $I' \cap (t+I) \neq \emptyset$. If the location $v_\rho(t)$ is open, then $v_\rho(t)$ contains C, and if $v_\rho(t)$ is singular, then the successor location contains C. All following locations contain C until a location with the clock constraint $(K - C) = \emptyset$ is reached, marking the end of I'. Since the clocks associated with C are not reset, they continue to represent the same witness I'. Since $I' \cap (t + I) \neq \emptyset$, each location $v_\rho(t')$ with $t < t' < I$ contains C. The consistency conditions, then, require that $v_\rho(t')$ contains the clock constraint $0 < (K - C)$ and, hence, the formula ψ_1. Therefore, by the induction hypothesis, $\tau_\rho^{t'} \models \psi_1$. Similarly, each location $v_\rho(t'')$ with $t'' \in I'$ contains the clock constraint $0 \in (K - C)$ or $0 = (K - C)$ and, hence, the formula ψ_2. Furthermore, if $t'' \neq r(I')$, then $v_\rho(t'')$ contains $0 \in (K - C)$ and ψ_1. Therefore, by the induction hypothesis that $\tau_\rho^{t''} \models \psi_2$ and, if $t'' \neq r(I')$, then $\tau_\rho^{t''} \models \psi_1$. Thus I' satisfies all criteria to be a witness for ψ under τ_ρ^t.

Conversely, let τ be a timed state sequence in ϕ-normal form. We construct a fair run ρ of B_ϕ such that for all formulas ψ in *Closure*(ϕ) and all $t \in \mathbb{R}_{\geq 0}$, if $\tau^t \models \psi$, then $v_\rho(t)$ contains ψ. It follows that B_ϕ accepts all models of ϕ.

We consider again the type-3 case of $\psi = \psi_1 \mathcal{U}_I \psi_2$. Let $t \in \mathbb{R}_{\geq 0}$ such that $\tau^t \models \psi$. By Lemma 4.8, the automaton B_ϕ can, at time t, either share an already activated clock interval $C_j(\psi)$, or it has enough clocks to activate an unused clock interval $C_j(\psi)$. If C is the activated clock interval and $K - C$ stands for the guessed witness, then all the consistency conditions for type-3 formulas are satisfied. In the first location that contains the clock constraint $(K - C) = \emptyset$, the automaton discards the clock interval C from the location, and the associated clocks may be reused later. $\Box$

We therefore have an algorithm for checking the satisfiability of a given MITL-formula ϕ: first, we construct the fair timed automaton B_ϕ, and then we check if $L(B_\phi)$ is nonempty.

4.4 Complexity of MITL

We show that the time complexity of our algorithm for checking the satisfiability of ϕ is doubly exponential in the length $\log K$ of the integer constants that appear in ϕ, and singly exponential in the number N of logical and temporal operators in ϕ. Moreover, the algorithm also implies an upper bound of EXPSPACE for deciding MITL. A matching lower bound of EXPSPACE for MITL can be obtained along the lines of the proof that the discrete-time logic MTL is EXPSPACE-hard [AH93].

THEOREM 4.13 *The satisfiability problem for* MITL *is EXPSPACE-complete. In particular, the proposed algorithm checks the satisfiability of the* MITL*-formula ϕ in time $O(2^{N \cdot K \cdot \log(N \cdot K)})$, where $K - 1$ is the largest integer constant appearing in ϕ, and N is the number of propositions, boolean connectives, and temporal operators in ϕ.*

PROOF. The first step of the algorithm transforms the given formula ϕ into the equivalent formula ϕ^* in normal form. By Lemma 4.2, the number of subformulas of ϕ^* is $O(N)$, and the size of the closure set *Closure*(ϕ^*) is $O(N \cdot K)$. Hence the number of locations of the automaton B_{ϕ^*} is $O(2^{N \cdot K})$. The number of clocks of B_{ϕ^*} is $O(N \cdot K)$. Furthermore, for every clock x, the largest integer constant appearing in a clock constraint for x is bounded by K. Consequently, the size of the region graph for B_{ϕ^*} is $O(2^{N \cdot K} \cdot (N \cdot K)! \cdot (N \cdot K)^K)$ (see Section 3.2). Hence the algorithm that checks the emptiness of $L(B_{\phi^*})$ runs in time $O((N \cdot K)^{N \cdot K})$.

For containment in EXPSPACE, observe that the automaton B_{ϕ^*} need not be constructed explicitly. The emptiness of $L(B_{\phi^*})$ can be checked nondeterministically by repeated testing that there is an edge between two vertices of region graph for B_{ϕ^*}, while only a constant number of vertices needs to be stored [AD94]. Recall that a vertex of the region graph is described using space logarithmic in the number of locations of B_{ϕ^*}, polynomial in the number of clocks of B_{ϕ^*}, and polynomial in the length of the largest constant appearing in the clock constraints of B_{ϕ^*}. It follows that a vertex of the region graph is described using space polynomial in $N \cdot K$. The transitions of the automaton B_{ϕ^*} are defined locally, and all consistency conditions are easy to check. Consequently, given the descriptions of two vertices of the region graph for B_{ϕ^*}, it can be tested in polynomial time if there is an edge between the two vertices. It follows that the satisfiability of ϕ can be decided in space polynomial in $N \cdot K$, that is, in EXPSPACE. □

4.5 A PSPACE-fragment of MITL

The main source of complexity for the construction of the automaton B_ϕ are the type-3 and type-4 formulas. Disallowing these formulas reduces the complexity by one exponential.

DEFINITION 4.14 $\text{MITL}_{0,\infty}$ is the fragment of MITL that consists of all formulas ϕ such that for each interval I appearing in ϕ, either $\ell(I) = 0$ or $r(I) = \infty$. □

Equivalently, $\text{MITL}_{0,\infty}$ is the fragment of MITL where all interval subscripts are of the form $\geq a$, $> a$, $< b$, or $\leq b$.

THEOREM 4.15 *The satisfiability problem for* $\text{MITL}_{0,\infty}$ *is PSPACE-complete. In particular, the proposed algorithm checks the satisfiability of the* $\text{MITL}_{0,\infty}$*-formula* ϕ *in time* $O(2^{N \cdot \log(N \cdot K)})$, *where* $K - 1$ *is the largest integer constant appearing in* ϕ, *and* N *is the number of propositions, boolean connectives, and temporal operators in* ϕ.

PROOF. By transforming the $\text{MITL}_{0,\infty}$-formula ϕ into normal form, we obtain an equivalent formula ϕ^* that does not contain type-3 and type-4 subformulas. Each type-1 and type-2 subformula introduces only one clock and two clock constraints in the closure set $Closure(\phi^*)$; the size of $Closure(\phi^*)$ is therefore bounded by N. Consequently, the automaton B_{ϕ^*} has $O(2^N)$ locations and $O(N)$ clocks. The size of the largest integer constant that appears in the clock constraints of B_{ϕ^*} is K. From the region-graph construction, it follows that the emptiness of $L(B_{\phi^*})$ can be checked in time $O(2^N \cdot N! \cdot K^N)$. The PSPACE upper bound follows, as before, by the observation that the search in the region graph can be performed without explicitly constructing the automaton B_{ϕ^*}.

The PSPACE-hardness of $\text{MITL}_{0,\infty}$ follows from the PSPACE-hardness of propositional temporal logic with *until* [SC85]. □

Thus, the complexity of MITL decreases from EXPSPACE to PSPACE if we prohibit bounded intervals with nonzero left end-points. This phenomenon has been observed also by [EMSS90] for discrete-time logics.

5 MITL-based Real-time Verification

Model checking is a powerful and well-established technique for the automatic verification of finite-state systems; it compares a temporal-logic specification of a system against a state-transition description of the system. In the untimed case, the system is modeled by its state-transition graph, and the specification may be presented either as a branching-time formula [CES86] or as a linear-time formula [LP85, SC85]. In the discrete-time case, the untimed model-checking algorithms can be extended to real-time logics using a special *tick* transition [EMSS90, AH93, AH94]. In the continuous-time case, model-checking algorithms are known for branching-time specifications of timed automata [ACD93]. We present the first model-checking algorithm for a linear-time logic with a continuous-time semantics, by comparing MITL-specifications against system descriptions given as timed automata.

We model a real-time system by a timed automaton A and write the requirements specification as a formula ϕ of MITL.

DEFINITION 5.1 The *model-checking problem* for MITL is to decide whether or not all timed state sequences that are accepted by a given timed automaton A satisfy a given MITL-formula ϕ:

$$L(A) \stackrel{?}{\subseteq} L(\phi)$$

□

We use our construction for testing the satisfiability of MITL-formulas to solve the model-checking problem. First, we construct the fair timed automaton $B_{\neg\phi}$ that accepts precisely the models of the negated formula $\neg\phi$. Hence the model-checking problem can be reformulated as follows:

$$L(A) \subseteq L(\phi) \ \text{ iff } \ L(A) \cap L(B_{\neg\phi}) = \emptyset.$$

Second, we construct the product automaton $A \times B_{\neg\phi}$ and check it for emptiness (see Sections 3.3 and 3.2). The size of the product automaton is polynomial in the sizes of A and $B_{\neg\phi}$; that is, the description of $A \times B_{\neg\phi}$ is exponential in the length of ϕ, and polynomial in the length of the description of A. Since the emptiness problem for fair timed automata can be solved in PSPACE, the model-checking problem for MITL can be solved in EXPSPACE.

THEOREM 5.2 *The model-checking problem for* MITL *is EXPSPACE-complete.*

PROOF. We have already outlined how the model-checking problem can be solved in EXPSPACE. To prove EXPSPACE-hardness, we observe that, as with all linear-time logics, the satisfiability problem for MITL can be reduced to the model-checking problem: the MITL-formula ϕ is unsatisfiable iff $L(A_U) \subseteq L(\neg\phi)$ for the universal timed automaton A_U, which accepts all possible timed state sequences. □

The time complexity of the model-checking algorithm for MITL is polynomial in the qualitative part of the system description, exponential in the qualitative part of the specification, exponential in the timing part of the system description, and doubly exponential in the timing part of the specification (this double exponential disappears for $\text{MITL}_{0,\infty}$-specifications). Compared to this the model-checking algorithm for propositional linear temporal logic is polynomial in the size of the system description and exponential in the size of the specification. Thus, in the general case the move to real time adds an exponential. This blow-up seems, however, unavoidable for formalisms for quantitative reasoning about time; it occurs already in the simplest, discrete-time, case of synchronous systems that proceed at the rate of one transition per time unit [EMSS90, AH93, AH94].

Acknowledgment. We wish to thank an anonymous referee for pointing out the PSPACE-fragment of Section 4.5.

References

[ACD93] R. Alur, C. Courcoubetis, and D.L. Dill. Model checking in dense real time. *Information and Computation*, 104(1):2–34, 1993.

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

[AH92] R. Alur and T.A. Henzinger. Logics and models of real time: a survey. In J.W. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, editors, *Real Time: Theory in Practice*, Lecture Notes in Computer Science 600, pages 74–106. Springer-Verlag, 1992.

[AH93] R. Alur and T.A. Henzinger. Real-time logics: complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.

[AH94] R. Alur and T.A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–204, 1994.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal-logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[EMSS90] E.A. Emerson, A.K. Mok, A.P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. In R.P. Kurshan and E.M. Clarke, editors, *CAV 90: Computer-aided Verification*, Lecture Notes in Computer Science 531, pages 136–145. Springer-Verlag, 1990.

[HLP90] E. Harel, O. Lichtenstein, and A. Pnueli. Explicit-clock temporal logic. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science*, pages 402–413. IEEE Computer Society Press, 1990.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[HPS83] D. Harel, A. Pnueli, and J. Stavi. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 26(2):222–243, 1983.

[JM86] F. Jahanian and A.K. Mok. Safety analysis of timing properties in real-time systems. *IEEE Transactions on Software Engineering*, SE-12(9):890–904, 1986.

[Koy90] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-time Systems*, 2(4):255–299, 1990.

[Lew90] H.R. Lewis. A logic of concrete time intervals. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science*, pages 380–389. IEEE Computer Society Press, 1990.

[LP85] O. Lichtenstein and A. Pnueli. Checking that finite-state concurrent programs satisfy their linear specification. In *Proceedings of the 12th Annual Symposium on Principles of Programming Languages*, pages 97–107. ACM Press, 1985.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, 1992.

[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 Book Company, 1967.

[SC85] A.P. Sistla and E.M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[Tho90] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 133–191. Elsevier Science Publishers (North-Holland), 1990.