What Good Are Digital Clocks?*

Thomas A. Henzinger† Zohar Manna‡ Amir Pnueli§

Abstract. Real-time systems operate in "real," continuous time and state changes may occur at any real-numbered time point. Yet many verification methods are based on the assumption that states are observed at integer time points only. What can we conclude if a real-time system has been shown "correct" for integral observations?

Integer time verification techniques suffice if the problem of whether all real-numbered behaviors of a system satisfy a property can be reduced to the question of whether the integral observations satisfy a (possibly modified) property. We show that this reduction is possible for a large and important class of systems and properties: the class of systems includes all systems that can be modeled as timed transition systems; the class of properties includes time-bounded invariance and time-bounded response.

1 Introduction

Over the past few years, we have seen a proliferation of formal methodologies for software and hardware design that emphasize the treatment of real-time requirements. Any formal approach to real-time systems typically

- assumes C, a mathematical *model of computation*;
- assumes T, a mathematical *model of time*;
- uses a formal language $\mathcal{L}_I$, the *implementation language*, for describing systems and their behavior over time;
- uses a formal language $\mathcal{L}_S$, the *specification language*, for describing timing requirements of systems;
- presents algorithms and/or proof rules that facilitate a formal argument that a particular system meets a particular specification under the assumed semantics of computation and time.

*A full version of this paper (including all proofs) is available as a technical report from Cornell University and Stanford University. The research was supported in part by the National Science Foundation under grants CCR-89-11512 and CCR-89-13641, by the Defense Advanced Research Projects Agency under contract NAG2-703, by the United States Air Force Office of Scientific Research under contract AFOSR-90-0057, and by the European Community ESPRIT Basic Research Action Project 3096 (SPEC).

†Department of Computer Science, Cornell University, Ithaca, NY 14853, U.S.A.

‡Department of Computer Science, Stanford University, Stanford, CA 94305, U.S.A.

§Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel.

The central question "Does the system S, given as an expression of $\mathcal{L}_I$, meet the specification ϕ, given as an expression of $\mathcal{L}_S$, with respect to the semantical assumption (C,T)?" is called the *real-time verification problem* for the parameters $(C,T,\mathcal{L}_I,\mathcal{L}_S)$ and written as

$$S \overset{?}{\models}_{(C,T)} \phi.$$

While several instances of the real-time verification problem have been studied extensively [10], comparatively little effort has been invested in (1) justifying the adequacy of various semantical assumptions and (2) exploring connections between different approaches. Both questions are intimately linked with each other. For example, the justification of an integer model of time must relate the model to real-numbered time.

We demonstrate that the comparative analysis of models can be used to justify timing assumptions and to broaden the scope of applicability of verification methods that have been developed for restrictive semantical assumptions. In particular, we contrast two important instances of the real-time verification problem — one with an *integral* model of time (let T be the integers Z) and one with a *dense* model of time (let T be the reals R). To avoid distraction from the issues that concern time, we consider, in both cases, a simple trace semantics C of computation.

Our results are twofold. First, we define a sufficient criterion, which we call *digitizability*, for the real-time verification problem to be invariant under the time model:

$$S \models_{\mathsf{R}} \phi \quad \text{iff} \quad S \models_{\mathsf{Z}} \phi$$

for digitizable systems S and digitizable specifications ϕ. Roughly speaking, a set of dense-time traces is digitizable iff it contains a trace ρ precisely when it contains all integral-time traces that result from observing ρ with fictitious digital clocks. Both general-purpose real-time programming languages $\mathcal{L}_I$ and real-time specification languages $\mathcal{L}_S$ all of whose expressions are digitizable have been proposed in the literature. We show that (1) every timed transition system [15] is digitizable and (2) every *qualitative* (time-independent) property and the most common *quantitative* (hard real-time) properties are digitizable. Timed transition systems can model sets of real-time processes that communicate through a shared memory or by message passing in multiprocessing and multiprogramming environments [16]. The class of digitizable properties includes all time-bounded invariance and time-bounded response requirements.

Second, we show that even nondigitizable dense-time properties may benefit from integer time verification. For any property ϕ that is definable in metric temporal logic [4, 19], we construct a (possibly somewhat weaker) property ϕ^W such that $S \models_{\mathsf{Z}} \phi$ implies $S \models_{\mathsf{R}} \phi^W$ for every timed transition system S. From this result we derive a scheme for approximating dense-time properties conservatively by integral-time properties.[1]

It follows that we can use verification techniques that have been developed for an integral-time model to prove (or approximate) properties of real-time systems under a dense-time model. This observation is of practical interest for two reasons:

- The dense-time model does not only correspond more closely to the physical reality than the integral-time model, but it is essential for composing systems [1, 6].

[1] Recently it has been brought to the authors' attention that a conservative approximation of real-numbered time by integer time was proposed independently by [8].

- Integral-time verification methods are not only significantly simpler than dense-time techniques, but apply to a wider range of real-time properties (because many instances of the dense-time verification problem are undecidable) [2, 4].

In addition, a wide variety of integral-time verification methods is readily available for the verification of timed transition systems (including model-checking algorithms [3, 4, 12, 25] and proof systems [13, 15, 25]). Our results prescribe the sound application of these techniques for proving how systems behave in real-numbered time.

The paper is organized as follows. In Section 2, we introduce both integral-time and dense-time models for real-time systems. We define the notion of digitizability and, for digitizable systems and specifications, we reduce the dense-time verification problem to an integral-time problem. In Section 3, we briefly review timed transition systems and metric temporal logic. We show that all timed transition systems and an important class of metric formulas are digitizable. We then proceed to determine which dense-time property of a timed transition system is established whenever a metric specification is checked under the assumption of integer time.

2 Semantics: Discrete Trace Models

This paper studies discrete trace models for real-time systems: *trace* models formalize a system as the set of its possible behaviors; *discrete* trace models formalize the behavior of a system as an infinite sequence of snapshots of the global system state at certain times. Discrete trace models have obvious limitations:

Trace (or *linear-time*) models abstract information about internal decisions of a system. More refined (*branching-time*) models are necessary to study the internal branching structure of a system [5].

Discrete trace models require the number of snapshots within a behavior to be countable. It follows that the state of a system cannot be observed at every real-numbered point in time. This restriction is adequate for modeling *discrete* processes, which change their state only finitely often between any two points in time. While the behavior of a discrete process can be described completely by an ω-sequence of state changes, more refined models are necessary to study *continuous* processes [21].

Despite (or, often, because of) these limitations, discrete trace models have been used successfully for the analysis of a wide variety of real-time systems and phenomena that arise in practice [1, 3, 15, 17, 20, 23, 25, 27].

Discrete trace models differ significantly in the kinds of systems and phenomena they can capture adequately [5]. These differences are caused by different models of time. We distinguish between *analog-clock* models, which record the precise real-numbered time of every snapshot (using a fictitious "analog clock"), and *digital-clock* models, which approximate the time of every snapshot by an integer (using a fictitious "digital clock").

2.1 Timed state sequences

Formally, discrete trace models define the behavior of a system as an infinite sequence of observations. An *observation* is a pair that consists of a state showing the result of a snapshot and a time-stamp recording the time of the snapshot. An infinite sequence

$$\rho: \quad (\sigma_0, \mathsf{T}_0) \longrightarrow (\sigma_1, \mathsf{T}_1) \longrightarrow (\sigma_2, \mathsf{T}_2) \longrightarrow (\sigma_3, \mathsf{T}_3) \longrightarrow \cdots$$

of observations is called a *timed state sequence* $\rho = (\sigma, \mathsf{T})$; it consists of an infinite sequence σ of states σ_i, for $i \geq 0$, and an infinite sequence T of corresponding time-stamps $\mathsf{T}_i \in \mathit{TIME}$. At this point, we do not commit to any particular time domain TIME; we only assume that there is a total ordering $<$ on TIME and demand that

Monotonicity (time does not decrease): $\mathsf{T}_i \leq \mathsf{T}_{i+1}$ for all $i \geq 0$. This condition ensures that the logical order of snapshots is consistent with the temporal order, while permitting adjacent observations with the same time-stamp.[2]

Progress (time progresses): for all $\mathsf{T} \in \mathit{TIME}$, there is some $i \geq 0$ such that $\mathsf{T}_i \geq \mathsf{T}$. This condition ensures that the time-stamps of observations do not converge, thus permitting only finitely many snapshots between any two points in time. This restriction is adequate for modeling discrete processes.

We will consider two types of fictitious clocks. *Analog-clock* (or *dense-time*) models are discrete trace models that take TIME to be the real numbers R; *digital-clock* (or *integral-time*) models take TIME to be the integers Z.[3] We call a timed state sequence of an analog-clock model *precisely timed*; of a digital-clock model, *digitally timed*. A set of precisely timed state sequences is called a *dense-time property*; a set of digitally timed state sequences, an *integral-time property*.

We routinely use the following operations on timed state sequences. For any timed state sequence $\rho = (\sigma, \mathsf{T})$, let ρ^- be its state component σ; applied to a set Π of timed state sequences, this "untime" operation yields the corresponding set Π^- of state components. By $\rho^i = (\sigma^i, \mathsf{T}^i)$, for $i \geq 0$, we denote the timed state sequence that results from the timed state sequence $\rho = (\sigma, \mathsf{T})$ by deleting the first i observations (in particular, $\rho^0 = \rho$). We write $\mathit{TSS}_{\mathit{TIME}}$ for the set of timed state sequences over the time domain TIME.

2.2 Dense time: Analog-clock models

Every real-time system S defines a dense-time property, namely $[\![S]\!]_{\mathsf{R}}$, the set of possible behaviors of S in real-numbered time. Similarly, every real-time specification ϕ defines a dense-time property, namely $[\![\phi]\!]_{\mathsf{R}}$, the set of real-time traces that meet the specification ϕ. The real-time system S satisfies the specification ϕ (denoted by $S \models_{\mathsf{R}} \phi$) iff $[\![S]\!]_{\mathsf{R}} \subseteq [\![\phi]\!]_{\mathsf{R}}$. Checking containment of dense-time properties is difficult, often undecidable, for many implementation and specification languages [1, 3, 4]. In comparison, containment of integral-time properties can be tested for transition-system based, automata based, and temporal-logic based languages [3, 4, 5, 12]. This is why many researchers have sacrificed the density of time and given their languages digital-clock interpretations [11, 14, 17, 24, 25, 27].

[2] The possibility of several contemporaneous, yet ordered, snapshots allows us to model, if desired, simultaneous parallel transitions of a system by nondeterministic interleaving [5].

[3] A third, extremely imprecise, type of fictitious clock results in *untimed* models. If time is immaterial, we can take TIME to be any trivial one-element domain. In this case, we may identify a timed state sequence with its state component.

We follow the same path, but rather than being content with solving the simpler integral-time verification problem, we would like to employ integer time techniques to solve the original dense-time verification problem (or some approximation thereof). Our aim is to interpret real-time systems and real-time specifications with the analog-clock semantics $\Pi_{\mathsf{R}} \subseteq TSS_{\mathsf{R}}$ as defining an integral-time property $\Pi_{\mathsf{Z}} \subseteq TSS_{\mathsf{Z}}$ such that

Amenability: The integral-time verification problem (denoted by $S \models_{\mathsf{Z}} \phi$), namely if $[\![S]\!]_{\mathsf{Z}} \subseteq [\![\phi]\!]_{\mathsf{Z}}$, can be solved.

Adequacy: The answer to the integral-time verification problem gives some insight about the dense-time verification problem; ideally, $S \models_{\mathsf{Z}} \phi$ iff $S \models_{\mathsf{R}} \phi$.

The remainder of this paper presents a digital-clock interpretation of nontrivial implementation and specification languages that satisfies both goals of amenability and adequacy.

2.3 Integer time: Digital-clock models

There are two deliberations we must take into account when defining an integral-time semantics for real-time systems. First, we want the interpretation of real-time languages to be independent of the clock model: if an expression denotes the set of timed state sequences that share a certain model-independent characteristics, then it ought to do so independently of the time domain. For example, a specification asserting in an analog-clock model the bounded-response property that "every request p is followed by a response q within 5 time units" should have the same meaning in the corresponding digital-clock model. So if an expression (of an implementation or specification language) defines in the analog-clock model the set Π_{R} of precisely timed state sequences, then its *clock-independent integral-time semantics*

$$\mathsf{Z}(\Pi_{\mathsf{R}}) = \Pi_{\mathsf{R}} \cap TSS_{\mathsf{Z}}$$

is the subset of digitally timed state sequences in Π_{R}.

Second, in order for a digital-clock model to be adequate, the integral-time semantics $[\![S]\!]_{\mathsf{Z}}$ of a system S must bear some resemblance to its dense-time semantics $[\![S]\!]_{\mathsf{R}}$. At the least, we want both $[\![S]\!]_{\mathsf{R}}$ and $[\![S]\!]_{\mathsf{Z}}$ to satisfy the same qualitative properties, such as the response property that "every request p is followed by a response q." Unfortunately, this is not the case for all real-time systems S and their clock-independent integral-time semantics $[\![S]\!]_{\mathsf{Z}} = \mathsf{Z}([\![S]\!]_{\mathsf{R}})$. Consider, for example, a system S that changes its state at the times $i + \delta_i$, for $i \in \mathsf{N}$ and a sequence of reals $0 < \delta_i < 1$ that converges toward 0. Every behavior in $[\![S]\!]_{\mathsf{R}}$ must contain a nonintegral time-stamp (for any choice of time unit). Consequently, $\mathsf{Z}([\![S]\!]_{\mathsf{R}}) = \emptyset$, which shows that the clock-independent integral-time semantics is a very poor representation for S. To preserve all qualitative properties, we cannot simply "throw away" behaviors with state changes between integral times, rather we must collect these state changes and record them at integral times. For this purpose, we introduce the notion of digitizing a dense-time behavior.

Digitization

Suppose we observe the behavior of a discrete process with a digital clock. The step-width of the clock (i.e., the time that elapses between successive clock ticks) is determined by the precision we are interested in; from now on, we normalize the stepwidth to 1. For example, if the system behavior

$$\rho: \quad (p, 0.2) \longrightarrow (q, 5.8) \longrightarrow (p, 5.9) \longrightarrow (q, 8) \longrightarrow (p, 8.4) \longrightarrow (q, 8.6) \longrightarrow \cdots$$

is observed by a clock that ticks at all integer times (and shows the integral "clock time" n throughout the real-time interval $[n, n+1)$ for all $n \in \mathsf{Z}$), we obtain the observation sequence

$$[\rho]_0: \quad (p, 0) \longrightarrow (q, 5) \longrightarrow (p, 5) \longrightarrow (q, 8) \longrightarrow (p, 8) \longrightarrow (q, 8) \longrightarrow \cdots.$$

The introduction of a fictitious digital clock suggests an integral-time semantics that identifies two systems iff they cannot be distinguished by the clock. While this integral-time semantics preserves all qualitative properties, such as response (since $[\rho]_0^- = \rho^-$), it does not preserve time-bounded response. To see this, consider the two timed state sequences ρ and $[\rho]_0$ given above; while every state p of the digitally timed sequence $[\rho]_0$ is followed by a state q within 5 time units, this is not the case for the precisely timed sequence ρ. It follows that integer time verification of bounded response would not be sound.

The suggested digital-clock semantics is not discriminative enough, because we have fixed the first tick of our digital clock, rather arbitrarily, at dense time 0. If our clock ticks, for example, at times $0.5 + n$ for all integers n, then the bounded-response property in question cannot distinguish between the system behavior ρ and the resulting observation sequence

$$[\rho]_{0.5}: \quad (p, 0) \longrightarrow (q, 6) \longrightarrow (p, 6) \longrightarrow (q, 8) \longrightarrow (p, 8) \longrightarrow (q, 9) \longrightarrow \cdots$$

(neither ρ nor $[\rho]_{0.5}$ satisfy the property that every p is followed by a q within 5 time units). So let us suppose we have infinitely many different fictitious clocks available for observation — one for every real number $0 \leq \epsilon < 1$. We will henceforth identify digital clocks with reals in the interval $[0, 1)$, assuming that the clock ϵ ticks at all times $\epsilon + n$, for $n \in \mathsf{Z}$. For any precisely timed state sequence $\rho = (\sigma, \mathsf{T})$ and digital clock $0 \leq \epsilon < 1$, let the *ϵ-digitization* $[\rho]_\epsilon = (\sigma, [\mathsf{T}]_\epsilon)$ of ρ be the digitally timed state sequence that results from recording the time-stamps in ρ with the digital clock ϵ:

$$[\rho]_\epsilon: \quad (\sigma_0, [\mathsf{T}_0]_\epsilon) \longrightarrow (\sigma_1, [\mathsf{T}_1]_\epsilon) \longrightarrow (\sigma_2, [\mathsf{T}_2]_\epsilon) \longrightarrow (\sigma_3, [\mathsf{T}_3]_\epsilon) \longrightarrow \cdots,$$

where $[x]_\epsilon = \lfloor x \rfloor$ if $x \leq \lfloor x \rfloor + \epsilon$, otherwise $[x]_\epsilon = \lceil x \rceil$. In words, the ϵ-digitization $[\rho]_\epsilon$ is obtained from ρ by rounding, with respect to ϵ, all time-stamps to integers. For any dense-time property Π, the *digitization* $[\Pi]$ of Π is the integral-time property that contains all ϵ-digitizations of sequences in Π:

$$[\Pi] \;=\; \{[\rho]_\epsilon \mid \rho \in \Pi \text{ and } 0 \leq \epsilon < 1\};$$

$[\{\rho\}]$ is abbreviated to $[\rho]$.

Now we can define an alternative integral-time semantics that identifies two systems iff they cannot be distinguished by *any* digital clock: the digitization $[[\![S]\!]_{\mathsf{R}}]$ is the *digitized integral-time semantics* of a real-time system S. We will prove that the digitized integral-time semantics preserves all qualitative properties as well as many important quantitative timing properties, including bounded response, for the real-time systems that are admissible in the following sense. Since we want both clock independence *and* adequacy, we must restrict ourselves to systems S whose clock-independent integral-time semantics $\mathsf{Z}([\![S]\!]_{\mathsf{R}})$ and digitized integral-time semantics $[[\![S]\!]_{\mathsf{R}}]$ coincide; that is, we are limited to real-time systems whose dense-time behaviors Π satisfy $\mathsf{Z}(\Pi) = [\Pi]$. We call these dense-time properties Π *closed under digitization*, because they are precisely the dense-time properties for which $[\Pi] \subseteq \Pi$ (i.e., $\rho \in \Pi$ implies $[\rho] \subseteq \Pi$). We will demonstrate that the restriction to systems whose possible behaviors are closed under digitization is not severe.

Inverse digitization

Further restrictions on the class of considered real-time systems can guarantee that the digital-clock model does not identify any two systems with distinct dense-time behaviors. We say that a dense-time property Π is *closed under inverse digitization* iff $[\rho] \subseteq \Pi$ implies $\rho \in \Pi$ for all $\rho \in TSS_{\mathsf{R}}$. A dense-time property Π is defined to be *digitizable* iff it is closed under both digitization and inverse digitization. Consequently, Π is digitizable precisely when

$$\rho \in \Pi \quad \text{iff} \quad [\rho] \subseteq \Pi$$

for all $\rho \in TSS_{\mathsf{R}}$. Digitization is a bijection between the set of digitizable properties and the set of integral-time properties. Hence our integral-time semantics is one-to-one for systems with digitizable dense-time behaviors.[4]

2.4 Semantic invariance of the verification problem

It is not difficult to check that for systems and specifications defining digitizable properties, the dense-time verification problem can be reduced to an integral-time verification problem. In fact, we have a stronger result:

Theorem 1 (Dense-time versus integral-time verification) *If the dense-time property Π is closed under digitization and the dense-time property Ψ is closed under inverse digitization, then $\Pi \subseteq \Psi$ iff $\mathsf{Z}(\Pi) \subseteq \mathsf{Z}(\Psi)$.*

For instance, assume that we are given a real-time system S whose dense-time semantics $[\![S]\!]_{\mathsf{R}}$ is digitizable, and a specification ϕ whose dense-time semantics $[\![\phi]\!]_{\mathsf{R}}$ is closed under inverse digitization. Then we may check if $S \models_{\mathsf{R}} \phi$ by attempting to prove, instead, that $S \models_{\mathsf{Z}} \phi$, which can be done using verification methods that have been developed for integer time. Either semantic techniques, such as model-checking algorithms, or syntactic techniques, using axioms and proof rules, can be employed to solve the integer time verification problem. Both soundness and completeness results for any integer time method carry over to the corresponding verification problem about real-numbered time.[5]

The following two propositions are helpful for showing that certain dense-time properties are closed under (inverse) digitization. We say that a dense-time property Π is

[4] An alternative restriction that allows the inversion of digitization is provided by the more restricted class of *synchronous* systems, which change states only at integer times [7, 9].

[5] In fact, the premises of Theorem 1 are needed for *soundness* only; *completeness* (i.e., $\Pi \subseteq \Psi$ implies $\mathsf{Z}(\Pi) \subseteq \mathsf{Z}(\Psi)$) holds for arbitrary dense-time properties Π and Ψ.

qualitative iff $\rho \in \Pi$ implies $\rho' \in \Pi$ for all precisely timed state sequences ρ and ρ' with identical state components (i.e., $\rho^- = \rho'^-$).

Proposition 1 (Qualitative properties) *Every qualitative property is digitizable.*

Proposition 2 (Boolean operations) *Closure under digitization is preserved by arbitrary unions and intersections of dense-time properties. Closure under inverse digitization is preserved by arbitrary intersections only. If a dense-time property Π is closed under digitization, then its complement $TSS_{\mathsf{R}} - \Pi$ is closed under inverse digitization.*

3 Syntax: Transition Systems, Temporal Logic

Let us now turn to a particular implementation language — timed transition systems — and a particular specification language — metric temporal logic. Both languages have been shown to allow a crisp description of a wide variety of real-time systems and timing requirements that are encountered in practice (see [16] for timed transition systems and [19] for metric temporal logic). Timed transition systems were introduced, originally, with an integral-time semantics [15]; metric temporal logic is known to be decidable under an integral-time interpretation and undecidable under a dense-time interpretation [4]. We use, nonetheless, both languages to define *dense-time* properties, by interpreting them over precisely timed state sequences.

3.1 Timed transition systems

Timed transition systems generalize the conventional model of discrete transition systems [18, 26] by imposing real-time constraints on the transitions: with each transition we associate a minimal and a maximal delay. We only summarize the formal definition of timed transition systems in this paper; for a more verbose introduction to timed transition systems and their applications consult [15, 16].

A *transition system* $S = \langle \Sigma, \Theta, \mathcal{T} \rangle$ consists of three components:

1. a (possibly infinite) set Σ of *states*;
2. a subset $\Theta \subseteq \Sigma$ of *initial states*;
3. a finite set $\mathcal{T}$ of *transitions*. Each transition $\tau \in \mathcal{T}$ is a binary relation on Σ; that is, it defines for every state $\sigma \in \Sigma$ a (possibly empty) set of τ-successors $\tau(\sigma) \subseteq \Sigma$. We say that the transition τ is *enabled* on a state σ iff $\tau(\sigma) \neq \emptyset$.

An infinite sequence σ of states is a *computation* (execution sequence) of the transition system $S = \langle \Sigma, \Theta, \mathcal{T} \rangle$ iff it satisfies the two requirements of

Initiality: $\sigma_0 \in \Theta$.

Consecution: For all $i \geq 0$ there is a transition $\tau \in \mathcal{T}$ such that $\sigma_{i+1} \in \tau(\sigma_i)$. We say that τ is *taken* at position i and *completed* at position $i+1$.

Let $[\![S]\!]$ be the set of computations of S.

We incorporate time into the transition system model by assuming that all transitions happen "instantaneously," while real-time constraints restrict the times at which transitions occur. These timing constraints are classified into two categories: *lower-bound* and *upper-bound* requirements. They ensure that transitions occur neither too early nor too late, respectively. All of our time bounds are natural numbers N; for notational convenience, we assume that $\infty \geq n$ for all $n \in \mathsf{N}$.

A *timed transition system* $S = \langle \Sigma, \Theta, \mathcal{T}, l, u \rangle$ consists of an underlying transition system $S^- = \langle \Sigma, \Theta, \mathcal{T} \rangle$ as well as

4. a *minimal delay* $l_\tau \in \mathsf{N}$ for every transition $\tau \in \mathcal{T}$;

5. a *maximal delay* $u_\tau \in \mathsf{N} \cup \{\infty\}$ for every transition $\tau \in \mathcal{T}$. We require that $u_\tau \geq l_\tau$ for all $\tau \in \mathcal{T}$.

A timed state sequence $\rho = (\sigma, \mathsf{T})$ is a *computation* of the timed transition system $S = \langle \Sigma, \Theta, \mathcal{T}, l, u \rangle$ iff the state sequence σ is a computation of the underlying transition system S^-, and

Lower bound: For every transition $\tau \in \mathcal{T}$ and all positions $i \geq 0$ and $j > i$ such that $\mathsf{T}_j < \mathsf{T}_{i+1} + l_\tau$, if τ is taken at position j, then τ is enabled on σ_i. In other words, once enabled, τ is delayed for at least l_τ time units; it can be taken only after being continuously enabled for l_τ time units.

Upper bound: For every transition $\tau \in \mathcal{T}$ and position $i > 0$ such that τ is enabled on σ_i, there is some position $j > i$ with $\mathsf{T}_j \leq \mathsf{T}_{i-1} + u_\tau$ such that either τ is completed at position j, or τ is not enabled on σ_j. In other words, once enabled, τ is delayed for at most u_τ time units; it cannot be continuously enabled for more than u_τ time units without being completed.[6]

The timed transition system S defines the set $[\![S]\!]_{TIME} \subseteq TSS_{TIME}$ of computations of S over the time domain $TIME$. Thus, S defines in the analog-clock model the dense-time property $[\![S]\!]_{\mathsf{R}}$; in the digital-clock model, S defines the integral-time property $[\![S]\!]_{\mathsf{Z}} = \mathsf{Z}([\![S]\!]_{\mathsf{R}})$. The timing constraints of S can be viewed as filters that prohibit certain execution sequences of the underlying untimed transition system S^-, because $[\![S]\!]_{TIME}^- \subseteq [\![S^-]\!]$. Special cases are a minimal delay 0 and a maximal delay ∞ for a transition τ. While the former does not rule out any computations of S^-, the latter adds to S^- a *weak-fairness* (justice) assumption [22]: τ cannot be continuously enabled without being taken.

Theorem 2 (Timed transition systems) *The set of dense-time computations of a timed transition system is digitizable.*

Proof. (1) To see that the set of computations of a timed transition system is closed under digitization, observe that $\mathsf{T}_j \geq \mathsf{T}_i + l$ implies $[\mathsf{T}_j]_\epsilon \geq [\mathsf{T}_i]_\epsilon + l$ and $\mathsf{T}_j \leq \mathsf{T}_i + u$ implies $[\mathsf{T}_j]_\epsilon \leq [\mathsf{T}_i]_\epsilon + u$ for all $\mathsf{T}_i, \mathsf{T}_j \in \mathsf{R}$, $l, u \in \mathsf{N} \cup \{\infty\}$, and $0 \leq \epsilon < 1$. These observations guarantee that whenever any ϵ-digitization $[\rho]_\epsilon$ of a timed state sequence ρ

[6] The reader familiar with [15] will notice that we have slightly modified the lower-bound and upper-bound requirements. This is because in the present paper we do not insist that the computations of a timed transition systems are deterministic (a timed state sequence is called *deterministic* iff with any transition either the state changes or the time increases, but not both). The lower-bound and upper-bound requirements are chosen so that the set of computations of any timed transition system is closed under a suitable notion of *timed stuttering*, which allows the refinement of timed transition systems by increasing the visible portion of the state space. Closure under timed stuttering, its applications, and its interaction with closure under digitization and inverse digitization is discussed in [14].

violates a lower-bound requirement for the positions $i < j$, then so does ρ; and whenever ρ satisfies an upper-bound requirement for position i at position $j > i$, then so does every ϵ-digitization $[\rho]_\epsilon$.

(2) Similarly, to see that the set of computations of a timed transition system is closed under inverse digitization, observe that there is, for all $\mathsf{T}_i, \mathsf{T}_j \in \mathsf{R}$ and $l, u \in \mathsf{N} \cup \{\infty\}$, some $0 \leq \epsilon < 1$ such that $\mathsf{T}_j < \mathsf{T}_i + l$ implies $[\mathsf{T}_j]_\epsilon < [\mathsf{T}_i]_\epsilon + l$, and some $0 \leq \epsilon < 1$ such that $\mathsf{T}_j > \mathsf{T}_i + u$ implies $[\mathsf{T}_j]_\epsilon > [\mathsf{T}_i]_\epsilon + u$. ■

It follows that every timed transition system S defines the same qualitative (untimed) property in either clock model (i.e., $[\![S]\!]^-_{\mathsf{R}} = [\![S]\!]^-_{\mathsf{Z}}$).

3.2 Metric temporal logic

Metric temporal logic (MTL for short) is an extension of linear temporal logic [26] that is interpreted over timed state sequences (rather than state sequences). To express timing requirements, the temporal operators are replaced with time-constrained versions, such as the constrained *eventually* operator $\Diamond_{[2,4]}$ meaning "eventually within 2 to 4 time units." We briefly review the syntax and semantics of MTL; for a more detailed introduction to metric temporal logic and its applications consult [4, 19].

Syntax and semantics

Let P be a set of atomic propositions. The *formulas* ϕ of MTL are built from atomic propositions by boolean connectives and time-constrained versions of the *until* operator $\mathcal{U}$; they are defined inductively as follows:

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

where $p \in P$ and I is an interval of the nonnegative real line both of whose end-points are natural-number constants.[7] Intervals may be open, halfopen, or closed; empty, bounded, or unbounded. We say that all intervals of the forms $[0, n)$ and (m, n), for $m \in \mathsf{N}$ and $n \in \mathsf{N} \cup \{\infty\}$, are *open*; all intervals of the forms $[m, n]$ and $[m, \infty)$, for $m, n \in \mathsf{N}$, are *closed*. We freely denote intervals by pseudo-arithmetic expressions. For example, the expressions $\leq m$ and $> n$ stand for the intervals $[0, m]$ and (n, ∞), respectively. The expression $x + I$, where I is an interval and $x \in \mathsf{R}$, denotes the interval $\{x + y \mid y \in I\}$.

The formulas of MTL are interpreted over timed state sequences whose states are taken from some set Σ such that each state in Σ provides an interpretation for the atomic propositions. Let $\rho = (\sigma, \mathsf{T})$ be a timed state sequence with $\sigma_i \in \Sigma$ for all $i \geq 0$. For any MTL-formula ϕ, the satisfaction relation $\rho \models \phi$ is defined inductively as follows:

$\rho \models p$ iff $\sigma_0 \models p$;
$\rho \models \neg\phi$ iff $\rho \not\models \phi$;
$\rho \models \phi_1 \wedge \phi_2$ iff $\rho \models \phi_1$ and $\rho \models \phi_2$;
$\rho \models \phi_1 \mathcal{U}_I\, \phi_2$ iff $\rho^i \models \phi_2$ for some $i \geq 0$ with $\mathsf{T}_i \in \mathsf{T}_0 + I$, and $\rho^j \models \phi_1$ for all $0 \leq j < i$.

[7] The restriction to *constant* time bounds on temporal operators distinguishes our version of MTL from [19]. This restriction is necessary for the results of this paper as well as the decidability results of [4]. The use of integer rather than rational time bounds is insignificant, because any formula with rational constraints can be scaled appropriately.

For an MTL-formula ϕ, let the set $[\![\phi]\!]_{TIME} \subseteq TSS_{TIME}$ of *TIME-models* contain all timed state sequences ρ over the time domain $TIME$ such that $\rho \models \phi$. Thus, ϕ defines in the analog-clock model the dense-time property $[\![\phi]\!]_{\mathsf{R}}$; in the digital-clock model, ϕ defines the integral-time property $[\![\phi]\!]_{\mathsf{Z}} = \mathsf{Z}([\![\phi]\!]_{\mathsf{R}})$.

Additional temporal operators can be defined as usual. The defined operators $\Diamond_I \phi$ (constrained *eventually*) and $\Box_I \phi$ (constrained *always*) stand for $true\, \mathcal{U}_I\, \phi$ and $\neg\Diamond_I \neg\phi$, respectively. It follows that the formula $\Box_I\, p$ (or $\Diamond_I\, p$) is satisfied by a timed state sequence $\rho = (\sigma, \mathsf{T})$ iff p holds at all states σ_i (at some state σ_i, respectively) with $\mathsf{T}_i \in \mathsf{T}_0 + I$. We also define a constrained *unless* operator as the dual of the *until* operator: $\phi_1\, {}_I\mathsf{U}\, \phi_2$ stands for $\neg((\neg\phi_2)\, \mathcal{U}_I\, (\neg\phi_1))$ (intuitively, "either ϕ_1 is true throughout the interval I or ϕ_2 is true at some point before the first time at which ϕ_1 is false in I"). We can apply the definition of the *unless* operator to move negations through *until* operators. Hence we may obtain, from any MTL-formula, an equivalent formula, containing both *until* and *unless* operators, in which all negations are in front of atomic propositions.

We usually suppress the universal interval $[0, \infty)$ as a subscript. Thus the MTL-operators $\Diamond$, $\Box$, and $\mathcal{U}$ coincide with the conventional unconstrained *eventually*, *always*, and *until* operators of linear temporal logic [26]. The standard unconstrained *unless* operator, ϕ_1 *unless* ϕ_2 [22], is definable by the MTL-formula $(\phi_1 \vee \phi_2)\, \mathsf{U}\, \phi_2$.

Digitizability

Not every MTL-formula defines a digitizable property. For instance, the R-models of the formula $\Diamond_{<1}\, p$ are not closed under digitization, and the R-models of the disjunction $\Diamond_{\leq 1}\, p \vee \Diamond_{\geq 2}\, p$ are not closed under inverse digitization. Yet the most common timing requirements of systems turn out to be digitizable.

Bounded invariance. A *bounded-invariance property* is a dense-time property that can be defined by an MTL-formula of the form

$$\Box(\phi_1 \;\rightarrow\; \Box_I\, \phi_2),$$

where I is open and contains at least one integer point, and both ϕ_1 and ϕ_2 define qualitative properties. A typical example of a bounded-invariance property is best-case performance, say, a lower bound of $l > 0$ time units on termination: $\Box(p \rightarrow \Box_{<l}\, \neg q)$ for propositions p and q that indicate the beginning and the end of execution, respectively.

Bounded response. A *bounded-response property* is a dense-time property that can be defined by an MTL-formula of the form

$$\Box(\phi_1 \;\rightarrow\; \Diamond_I\, \phi_2),$$

where I is closed and nonsingular (or $I = [0, 0]$), and both ϕ_1 and ϕ_2 define qualitative properties. A typical example of a bounded-response property is best-case performance, say, an upper bound of $u \geq 0$ time units on termination: $\Box(p \rightarrow \Diamond_{\leq u}\, q)$.

Clearly, MTL-formulas that contain only unconstrained temporal operators define qualitative properties, which are digitizable by Proposition 1. Furthermore:

Theorem 3 (Bounded invariance and bounded response) *Bounded-invariance properties and bounded-response properties are digitizable.*

Proof. Similar to the proof of Theorem 2. This is because each lower-bound requirement of a timed transition system is a bounded-invariance condition, and each upper-bound requirement is a bounded-response condition. ■

Now let us put Theorems 2, 3, and 1 together. These results imply that integer time verification techniques can be used to prove the dense-time correctness of timed transition systems with respect to conjunctions of unconstrained MTL-formulas, bounded-invariance formulas, and bounded-response formulas. An integer time model-checking algorithm for checking MTL-formulas over timed transition systems is given in [14]; an integer time proof system for bounded-invariance and bounded-response properties in [15].

In fact, Theorem 1 demands that the requirement specification of a timed transition system be closed under inverse digitization only (rather than being fully digitizable). Hence we can enlargen the class of MTL-formulas that can be established by integer time verification techniques. We say that an MTL-formula ϕ is *weakly constrained* iff it satisfies the following three conditions:

- Every negation in ϕ is in front of an atomic proposition.
- Every *until* operator in ϕ is constrained by an open interval.
- Every *unless* operator in ϕ is constrained by a closed interval.

All unconstrained MTL-formulas are weakly constrained (since the universal interval is both open and closed). But neither bounded-invariance nor bounded-response formulas are weakly constrained.

Theorem 4 (Weakly constrained formulas) *The* R*-models of a weakly constrained formula of* MTL *are closed under inverse digitization.*

Proof. Show by induction on the structure of a weakly constrained MTL-formula ϕ that $[\rho]_0 \models \phi$ implies $\rho \models \phi$ for all $\rho \in TSS_{\mathsf{R}}$. ■

Weakening and strengthening specifications

Theorem 4 can be applied in additional ways. First, from a proof that a timed transition system S satisfies an *arbitrary* MTL-formula ϕ in the digital-clock model, we can infer that S satisfies in the analog-clock model all weakly constrained formulas that are implied by ϕ. Second, to show that S satisfies a dense-time specification ϕ, it suffices to show that S satisfies in the digital-clock model any weakly constrained formula that implies ϕ.

Let ϕ^W (and ϕ^S) be the weakly constrained MTL-formula that results from the MTL-formula ϕ by the following process of *weakening* (*strengthening*, respectively):

1. push negations all the way to the inside;
2. replace the interval I of every *until* operator with the smallest open interval containing I (the largest open, possibly empty, interval contained in I);
3. replace the interval I of every *unless* operator with the largest closed interval contained in I (the smallest closed interval containing I).

Proposition 3 (Weakening and strengthening) *Let ϕ be an* MTL*-formula. Then ϕ implies ϕ^W, and ϕ^S implies ϕ (in either clock model).*

Thus, if a timed transition system S is shown to satisfy an MTL-specification ϕ in integer time, then we can infer that S satisfies the weaker formula ϕ^W in dense time. Similarly, to prove that S satisfies ϕ in dense time, it suffices to show that S satisfies the stronger formula ϕ^S in integer time. Naturally, we may not succeed in proving ϕ^S. In this case, it is often advisable to approximate ϕ with ϕ^W by refining the granularity of the digital-clock model. Suppose we need to show that S satisfies the specification $\Diamond_{[1,2]}\, p \vee \Diamond_{=3}\, p$ in dense time. By strengthening, we obtain $\Diamond_{(1,2)}\, p$, and suppose this formula is not satisfied by S. Using integer time, we can only establish the weaker property $\Diamond_{(0,3)}\, p \vee \Diamond_{(2,4)}\, p$ (i.e., $\Diamond_{(0,4)}\, p$). However, by refining the stepwidth of the digital-clock model by a factor of 10 (multiply all delays in S by 10), we can show that S satisfies $\Diamond_{(9,21)}\, p \vee \Diamond_{(29,31)}\, p$. In this fashion, by decreasing the time between two clock ticks further, we can arbitrarily increase our confidence that S indeed satisfies the dense-time specification $\Diamond_{[1,2]}\, p \vee \Diamond_{=3}\, p$. This is a typical example of how the digital-clock model may be used conservatively to reason about dense time.

Acknowledgements. We thank Rajeev Alur for uncountably many stimulating discussions, and Moshe Vardi for asking which dense-time properties are proved by integer time methods.

References

[1] Alur, R., and Dill, D. Automata for modeling real-time systems. In *ICALP 90: Automata, Languages, and Programming*, M. Paterson, Ed., Lecture Notes in Computer Science 443. Springer-Verlag, 1990, pp. 322–335.

[2] Alur, R., Feder, T., and Henzinger, T. The benefits of relaxing punctuality. In *Proceedings of the Tenth Annual Symposium on Principles of Distributed Computing* (1991), ACM Press, pp. 139–152.

[3] Alur, R., and Henzinger, T. A really temporal logic. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science* (1989), IEEE Computer Society Press, pp. 164–169.

[4] Alur, R., and Henzinger, T. Real-time logics: complexity and expressiveness. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science* (1990), IEEE Computer Society Press, pp. 390–401.

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

[6] Barringer, H., Kuiper, R., and Pnueli, A. A really abstract concurrent model and its temporal logic. In *Proceedings of the 13th Annual Symposium on Principles of Programming Languages* (1986), ACM Press, pp. 173–183.

[7] Berry, G., and Cosserat, L. The Esterel synchronous programming language and its mathematical semantics. In *CMU Seminar on Concurrency*, Lecture Notes in Computer Science 197. Springer-Verlag, 1985.

[8] Burch, J. Approximating continuous time. Presented at the IEEE Computer Society Workshop on VLSI, Orlando, Florida, 1991.

[9] Caspi, P., Pilaud, D., Halbwachs, N., and Plaice, J. Lustre: a declarative language for programming synchronous systems. In *Proceedings of the 14th Annual Symposium on Principles of Programming Languages* (1987), ACM Press.

[10] de Bakker, J., Huizing, K., de Roever, W.-P., and Rozenberg, G., Eds. *Real Time: Theory in Practice*. Lecture Notes in Computer Science 600. Springer-Verlag, 1992.

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

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

[13] Henzinger, T. Half-order modal logic: how to prove real-time properties. In *Proceedings of the Ninth Annual Symposium on Principles of Distributed Computing* (1990), ACM Press, pp. 281–296.

[14] Henzinger, T. *The Temporal Specification and Verification of Real-time Systems*. PhD thesis, Stanford University, 1991.

[15] Henzinger, T., Manna, Z., and Pnueli, A. Temporal proof methodologies for real-time systems. In *Proceedings of the 18th Annual Symposium on Principles of Programming Languages* (1991), ACM Press, pp. 353–366.

[16] Henzinger, T., Manna, Z., and Pnueli, A. Timed transition systems. In *Real Time: Theory in Practice*, J. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, Eds., Lecture Notes in Computer Science 600. Springer-Verlag, 1992.

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

[18] Keller, R. Formal verification of parallel programs. *Communications of the ACM 19*, 7 (1976), 371–384.

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

[20] Lynch, N., and Attiya, H. Using mappings to prove timing properties. In *Proceedings of the Ninth Annual Symposium on Principles of Distributed Computing* (1990), ACM Press, pp. 265–280.

[21] Maler, O., Manna, Z., and Pnueli, A. A formal approach to hybrid systems. In *Real Time: Theory in Practice*, J. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, Eds., Lecture Notes in Computer Science 600. Springer-Verlag, 1992.

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

[23] Merritt, M., Modugno, F., and Tuttle, M. Time-constrained automata. In *CONCUR 91: Theories of Concurrency*, Lecture Notes in Computer Science. Springer-Verlag, 1991.

[24] Nicollin, X., Richier, J.-L., Sifakis, J., and Voiron, J. ATP: an algebra for timed processes. In *Proceedings of the IFIP WG2.2/2.3 Working Conference on Programming Concepts and Methods* (1990), M. Broy and C. Jones, Eds., Elsevier Science Publishers (North-Holland), pp. 415–442.

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

[26] Pnueli, A. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science* (1977), IEEE Computer Society Press, pp. 46–57.

[27] Pnueli, A., and Harel, E. Applications of temporal logic to the specification of real-time systems. In *Formal Techniques in Real-time and Fault-tolerant Systems*, M. Joseph, Ed., Lecture Notes in Computer Science 331. Springer-Verlag, 1988, pp. 84–98.