Quantifying Similarities Between Timed Systems*

Thomas A. Henzinger[1], Rupak Majumdar[2], and Vinayak S. Prabhu[3]

[1]Department of Computer and Communication Sciences, EPFL
tah@epfl.ch
[2]Department of Computer Science, UC Los Angeles
rupak@cs.ucla.edu
[3]Department of Electrical Engineering and Computer Sciences, UC Berkeley
vinayak@eecs.berkeley.edu

Abstract. We define quantitative similarity functions between timed transition systems that measure the degree of closeness of two systems as a real, in contrast to the traditional boolean yes/no approach to timed simulation and language inclusion. Two systems are close if for each timed trace of one system, there exists a corresponding timed trace in the other system with the same sequence of events and closely corresponding event timings. We show that timed CTL is robust with respect to our quantitative version of bisimilarity, in particular, if a system satisfies a formula, then every close system satisfies a close formula. We also define a discounted version of CTL over timed systems, which assigns to every CTL formula a real value that is obtained by discounting real time. We prove the robustness of discounted CTL by establishing that close states in the bisimilarity metric have close values for all discounted CTL formulas.

1 Introduction

Timed systems model not only the sequence of system events but the timing information as well. Unfortunately, most formal models for timed systems are too precise: two states can be distinguished even if there is an arbitrarily small mismatch between the timings of an event. For example, traditional timed language inclusion requires that each trace in one system be matched *exactly* by a trace in the other system. Since formal models for timed systems are only approximations of the real world, and subject to estimation errors, this presents a serious shortcoming in the theory, and has been well noted in the literature [15, 21, 23, 13, 18, 16, 3, 19, 17].

We develop a theory of refinement for timed systems that is *robust* with respect to small timing mismatches. The robustness is achieved by generalizing timed refinement relations to metrics on timed systems that quantitatively estimate the closeness of two systems. That is, instead of looking at refinement

* This research was supported in part by the AFOSR MURI grant F49620-00-1-0327 and the NSF grants CCR-0208875, CCR-0225610, and CCR-0427202.

between systems as a boolean true/false relation, we assign a positive real number between zero and infinity to a pair of timed systems (T_r, T_s) which indicates how well T_r refines T_s. In the linear setting, we define the distance between two traces as ∞ if the untimed sequences differ, and as the supremum of the difference of corresponding time points otherwise. The distance between two systems is then taken to be the supremum of closest matching trace differences from the initial states. For example, the distance between the traces $a \xrightarrow{1} b$ and $a \xrightarrow{2} b$ is 1 unit, and occurs due to the second trace lagging the first by 1 unit at b. Similarly, the distance between the first trace and the trace $a \xrightarrow{100} b$ is 99. Intuitively, the first trace is "closer" to the second than the third; our metric makes this intuition precise.

Timed trace inclusion is undecidable on timed automata [2]. To compute a refinement distance between timed automata, we therefore take a branching view. We define quantitative notions of timed similarity and bisimilarity which generalize timed similarity and bisimilarity relations [6, 22] to metrics over timed systems. Given a positive real number ε, we define a state r to be ε-similar to another state s, if (1) the observations at the states match, and (2) if for every timed step from r there is a timed step from s such that the timing of events on the traces from r and s remain within ε. We provide algorithms to compute the similarity distance between two timed systems modeled as timed automata to within any given precision.

We show that bisimilarity metrics provide a robust refinement theory for timed systems by relating the metrics to timed computation tree logic (TCTL) specifications. We prove a *robustness theorem* that states close states in the metric satisfy TCTL specifications that have "close" timing requirements. For example, if the bisimilarity distance between states r and s is ε, and r satisfies the TCTL formula $\exists\Diamond_{\leq 5} a$ (i.e., r can get to a state where a holds within 5 time units), then s satisfies $\exists\Diamond_{\leq 5+2\varepsilon} a$. A similar robustness theorem for MITL was studied in [20]. However, they do not provide algorithms to compute distances between systems, relying on system execution to estimate the bound.

As an illustration, consider the two timed automata in Figure 1. Each automaton has four locations and two clocks x, y. Observations are the same as the locations. Let the initial states be $\langle a, x = 0, y = 0\rangle$ in both automata. The two automata seem close on inspection, but traditional language refinement of T_s by T_r does not hold. The trace $\langle a, x = 0, y = 0\rangle \xrightarrow{0} \langle b, 0, 0\rangle \xrightarrow{4} \langle c, 4, 4\rangle \ldots$ in T_r cannot be matched by a trace in T_s. The automaton T_s however, does have a similar trace, $\langle a, x = 0, y = 0\rangle \xrightarrow{0} \langle b, 0, 0\rangle \xrightarrow{3} \langle c, 3, 3\rangle \ldots$ (the trace difference is 1 time unit). We want to be able to quantify this notion of similar traces. Our metric gives a directed distance of 1 between T_r and T_s: for every (timed) move of T_r from the starting state, there is a move for T_s such that the trace difference is never more than 1 unit. The two automata do have the same untimed languages, but are not timed similar. Thus, the traditional theory does not tell us if the timed languages are close, or widely different. Looking at TCTL specifications, we note T_s satisfies $\exists\Diamond(c \wedge \exists\Diamond_{\geq 7} d)$, while T_r only satisfies

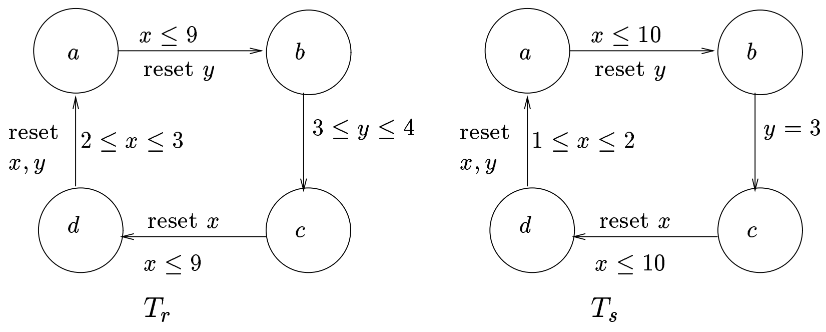

Fig. 1. Two similar timed automata

the more relaxed specification $\exists\Diamond(c \wedge \exists\Diamond_{\geq 5} d)$. Robustness guarantees a bound on the relaxation of timing requirements.

Once we generalize refinement to quantitative metrics, a natural progression is to look at logical formulae as functions on states, having real values in the interval $[0, 1]$. We use *discounting* [10, 12] for this quantification and define DCTL, a quantitative version of CTL for timed systems. Discounting gives more importance to near events than to those in the far future. For example, for the reachability query $\exists\Diamond a$, we would like to see a as soon as possible. If the shortest time to reach a from the state s is t_a, then we assign β^{t_a} to the value of $\exists\Diamond a$ at s, where β is a positive discount factor less than 1 in our multiplicative discounting. The subscript constraints in TCTL (e.g., ≤ 5 in $\exists\Diamond_{\leq 5} a$) may be viewed as another form of discounting, focusing only on events before 5 time units. Our discounting in DCTL takes a more uniform view; the discounting for a time interval depends only on the duration of the interval. We also show that the DCTL values are well behaved in the sense that close bisimilar states have close values for all DCTL specifications. For the discounted CTL formula $\exists\Diamond c$, the value in T_r is β^9 and β^{10} in T_s (shortest time to reach c on time diverging paths is 9 in T_r and 10 in T_s). They are again close (on the β scale).

The rest of the paper is organized as follows. In Section 2 we define quantitative notions of simulation and bisimilarity, and exhibit an algorithm to compute these functions to within any desired degree of accuracy for timed automata. In Section 3 we prove the robustness theorem for quantitative bisimilarity with respect to timed computation tree logic. In Section 4, we define DCTL, show its robustness, and give a model checking algorithm for a subset of DCTL over timed automata. Metrics have been studied before for discrete and probabilistic syetms in [14, 12, 17, 11], and for timed systems in [5, 17, 20]; this paper provides, to our knowledge, the first algorithms for computing refinement metrics on timed systems.

2 Quantitative Timed Simulation Functions

We define *quantitative* refinement functions on timed systems. These functions allow approximate matching of timed traces and generalize timed and untimed simulation relations.

2.1 Simulation Relations and Quantitative Extensions

A *timed transition system* (TTS) is a tuple $A = \langle Q, \Sigma, \rightarrow, \mu, Q_0 \rangle$ where

- Q is the set of states.
- Σ is a set of atomic propositions (the observations).
- $\rightarrow \subseteq Q \times \mathbb{R}^+ \times Q$ is the transition relation.
- $\mu : Q \mapsto 2^{\Sigma}$ is the observation map which assigns a truth value to atomic propositions true in a state.
- $Q_0 \subseteq Q$ is the set of initial states.

We write $q \xrightarrow{t} q'$ if $(q, t, q') \in \rightarrow$. A *state trajectory* is an infinite sequence $q_0 \xrightarrow{t_0} q_1 \xrightarrow{t_1} \ldots$, where for each $j \geq 0$, we have $q_j \xrightarrow{t_j} q_{j+1}$. The state trajectory is *initialized* if $q_0 \in Q_0$ is an initial state. A state trajectory $q_0 \xrightarrow{t_0} q_1 \ldots$ induces a *trace* given by the observation sequence $\mu(q_0) \xrightarrow{t_0} \mu(q_1) \xrightarrow{t_1} \ldots$. To emphasize the initial state, we say q_0-trace for a trace induced by a state trajectory starting from q_0. A trace is initialized if it is induced by an initialized state trajectory. A TTS A_i *refines* or *implements* a TTS A_s if every initialized trace of A_i is also an initialized trace of A_s. The general trace inclusion problem for timed systems is undecidable [2], simulation relations allow us to restrict our attention to a computable relation.

Let A be a TTS. A binary relation $\preceq \subseteq Q \times Q$ is a *timed simulation* if $q_1 \preceq q_2$ implies the following conditions:

1. $\mu(q_1) = \mu(q_2)$.
2. If $q_1 \xrightarrow{t} q_1'$, then there exists q_2' such that $q_2 \xrightarrow{t} q_2'$, and $q_1' \preceq q_2'$.

The state q is timed simulated by the state q' if there exists a timed simulation $\preceq$ such that $q \preceq q'$. A binary relation $\equiv$ is a *timed bisimulation* if it is a symmetric timed simulation. Two states q and q' are timed bisimilar if there exists a timed bisimulation $\equiv$ with $q \equiv q'$. Timed bisimulation is stronger than timed simulation which in turn is stronger than trace inclusion. If state q is timed simulated by state q', then every q-trace is also a q'-trace.

Untimed simulation and bisimulation relations are defined analogously by ignoring the duration of time steps. Formally, a binary relation $\preceq \subseteq Q \times Q$ is an (untimed) simulation if condition (2) above is replaced by

$(2)'$ If $q_1 \xrightarrow{t} q_1'$, then there exists q_2' and $t' \in \mathbb{R}^+$ such that $q_2 \xrightarrow{t'} q_2'$, and $q_1' \preceq q_2'$.
A symmetric untimed simulation relation is called an untimed bisimulation.

Timed simulation and bisimulation require that times be matched exactly. This is often too strict a requirement, especially since timed models are approximations of the real world. On the other hand, untimed simulation and bisimulation relations ignore the times on moves altogether. We now define *approximate* notions of refinement, simulation, and bisimulation that quantify if the behavior of an implementation TTS is "close enough" to a specification TTS. We begin by defining a metric on traces. Given two traces $\pi = r_0 \xrightarrow{t_0} r_1 \xrightarrow{t_1} r_2 \ldots$ and $\pi' = s_0 \xrightarrow{t'_0} s_1 \xrightarrow{t'_1} s_2 \ldots$, the distance $\mathcal{D}(\pi, \pi')$ is defined by

$$\mathcal{D}(\pi, \pi') = \begin{cases} \infty & : \quad \text{if } r_j \neq s_j \text{ for some } j \\ \sup_j \{ | \sum_{n=0}^{j} t_n - \sum_{n=0}^{j} t'_n | \} & : \quad \text{otherwise} \end{cases}$$

The trace metric $\mathcal{D}$ induces a *refinement distance* between two TTS. Given two timed transition systems A_r, A_s, with initial states Q_r, Q_s respectively, the *refinement distance* of A_r with respect to A_s is given by $\sup_{\pi_q} \inf_{\pi'_{q'}} \{\mathcal{D}(\pi_q, \pi'_{q'})\}$ where π_q (respectively, $\pi'_{q'}$) is a q-trace (respectively, q'-trace) for some $q \in Q_r$ (respectively, $q' \in Q_s$). Notice that the refinement distance is asymmetric: it is a *directed distance* [11].

We also generalize the simulation relation to a directed distance in the following way. For states r, s and $\delta \in \mathbb{R}$, the *simulation function* $\mathcal{S} : Q \times Q \times \mathbb{R} \to \mathbb{R}$ is the least fixpoint (in the absolute value sense) of the following equation:

$$\mathcal{S}(r, s, \delta) = \begin{cases} \infty & \text{if } \mu(r) \neq \mu(s) \\ \sup'_{t_r} \inf'_{t_s} \{\max' (\delta,\ \mathcal{S}(r', s', \delta + t_r - t_s)) \mid r \xrightarrow{t_r} r', s \xrightarrow{t_s} s'\} & \text{otherwise} \end{cases}$$

where $\sup', \inf', \max'$ consider only the modulus in the ordering, i.e., $x <' y$ iff $|x| < |y|$ in the standard real number ordering. We say r is ε*-simulated* by s if $|\mathcal{S}(r, s, 0)| \leq \varepsilon$. Note the ε-simulation is *not* transitive in the traditional sense. If r is ε-simulated by s, and s is ε-simulated by w, then r is (2ε)-simulated by w.

Given two states r, s, it is useful to think of the value of $\mathcal{S}(r, s, \delta)$ as being the outcome of a game. Enviroment plays on r (and its successors), and chooses a move at each round. We play on s and choose moves on its successors. Each round adds another step to both traces (from r and s). The goal of the environment is to maximise the trace difference, our goal is to minimize. The value of $\mathcal{S}(r, s, \delta)$ is the maximum lead of the r trace with respect to the s trace when the simulation game starts with the r trace starting with a lead of δ. If from r, s the environment can force the game into a configuration in which we cannot match its observation, we assign a value of ∞ to $\mathcal{S}(r, s, \cdot)$. Otherwise, we recursively compute the maximum trace difference for each step from the successor states r', s'. For the successors r', s', the lead at the first step is $(\delta + t_r - t_s)$. The lead from the first step onwards is then $\mathcal{S}(r', s', \delta + t_r - t_s)$. The maximum trace difference is either the starting trace difference (δ), or some difference after the first step ($\mathcal{S}(r', s', \delta + t_r - t_s)$).

Note that different accumulated differences in the times in the two traces may lead to different strategies, we need to keep track of the accumulated delay or lead. For example, suppose the environment is generating a trace and is currently at state r, and our matching trace has ended up at state s. Suppose r can only take a step of length 1, and s can take two steps of lengths 0 and 100. If the two traces ending at r and s have an accumulated difference of 0 (the times at which

r and s occur are exactly the same), then s should take the step of length 0. But if the r trace leads the s trace by say 70 time units, then s should take the step of length 100, the trace difference after the step will then be $|70+1-100| = 29$, if s took the 0 step, the trace difference would be $70 + 1 - 0 = 71$.

We also define the corresponding bisimulation function. For states $r, s \in Q$ and a real number δ, the *bisimulation function* $\mathcal{B} : Q \times Q \times \mathbb{R} \to \mathbb{R}$ is the least fixpoint (in the absolute value sense) of the equations $\mathcal{B}(r, s, \delta) = \infty$ if $\mu(r) \neq \mu(s)$, and

$$\mathcal{B}(r, s, \delta) = \max \left\{ \begin{array}{l} \sup'_{t_r} \inf'_{t_s} \{\max'(\delta,\ \mathcal{B}(r', s', \delta + t_r - t_s)) \mid r \xrightarrow{t_r} r', s \xrightarrow{t_s} s'\}, \\ \sup'_{t_s} \inf'_{t_r} \{\max'(\delta,\ \mathcal{B}(r', s', \delta + t_r - t_s)) \mid r \xrightarrow{t_r} r', s \xrightarrow{t_s} s'\} \end{array} \right\}$$

otherwise, where $\sup', \inf', \max'$ consider only the modulus in the ordering. The *bisimilarity distance* between two states r, s of a TTS is defined to be $\mathcal{B}(r, s, 0)$. States r, s are ε-bisimilar if $\mathcal{B}(r, s, 0) \leq \varepsilon$. Notice that $\mathcal{B}(r, s, 0) = 0$ iff r, s are timed bisimilar.

Proposition 1. *Let r and s be two states of a* TTS. *For every trace π_r from r, there is a trace π_s from s such that $\mathcal{D}(\pi_r, \pi_s) \leq |\mathcal{S}(r, s, 0)|$. The bisimilarity distance $\mathcal{B}(r, s, 0)$ is a pseudo-metric on the states of TTSs.*

Example 1. Consider the example in Fig. 2. The observations have been numbered for simplicity: $\mu(a_1) = a, \mu(b_i) = b, \mu(c_i) = c$. We want to compute $\mathcal{S}(a, a_1, 0)$. It can be checked that a is untimed similar to a_1 All paths have finite weights, so $\mathcal{S}(a, a_1, 0) < \infty$. Consider the first step, a takes a step of length 7 in A_r. A_s has two options, it can take a step to b_1 of length 5 or a step to b_2 of length 8, and to decide which one to take, it needs $\mathcal{S}(b, b_1, 2)$ and $\mathcal{S}(b, b_2, -1)$. $\mathcal{S}(b, b_2, -1)$ is $-1 + 10 - 4 = 5$. To compute $\mathcal{S}(b, b_1, 2)$, we look at b_1's options. In the next step, if we move to c_2, then the trace at the (c, c_2) configuration will be $2 + 10 - 9 = 3$. If we move to c_1, the trace difference will be $2 + 10 - 12 = 0$ (this is the better option). Thus $\mathcal{S}(b, b_1, 2) = 2$ (the 2 is due to the initial lead). Thus $\mathcal{S}(a, a_1, 0) = 2$. □

2.2 Algorithms for Simulation Functions

Finite Weighted Graphs. We first look at computing ε-simulation on a special case of timed transition systems. A *finite timed graph* $T = (Q, \Sigma, E, \mu, W)$ consists of a finite set of locations Q, a set Σ of atomic propositions, an edge relation $E \subseteq Q \times Q$, an observation function $\mu : V \to 2^{\Sigma}$ on the locations, and an integer weight function $W : E \to \mathbb{N}^{+}$ on the edges. For vertices $s, s' \in Q$, we write $s \xrightarrow{t} s'$ iff there is an edge $(s, s') \in E$ with $W(s, s') = t$. The following theorem provides a bound on simulation functions on a finite timed graph.

Theorem 1. *Let A be a finite timed graph and let $n = |Q|$ be the number of nodes and $W_{\max} = \max_{e \in E}\{W(e)\}$ the maximum weight of any edge. Let*

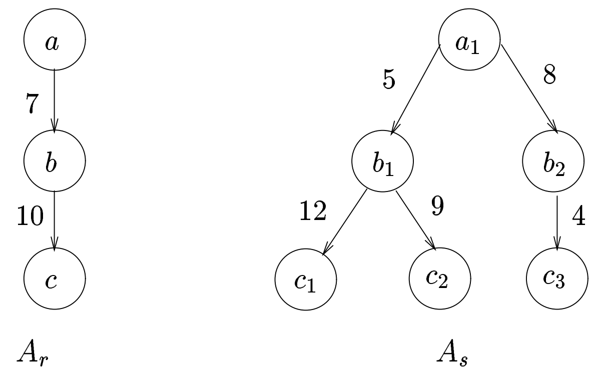

Fig. 2. A_r is 2-similar to A_s

$f \in \{\mathcal{S}, \mathcal{B}\}$. (1) For every pair of vertices $r, s \in Q$, if $|f(r,s,0)| < \infty$, then $|f(r,s,0)| \leq 2n^2 \cdot W_{\max}$. (2) The values $\mathcal{S}(r,s,0)$ and $\mathcal{B}(r,s,0)$ are computable over finite timed graphs in time polynomial in n and $W_{\max}$.

The proof of (1) is by contradiction, we give the argument for $\mathcal{S}(r,s,0)$. Since we are working on a finite graph, the sup-inf in the definiton of $\mathcal{S}$ can be replaced by a max-min. Consider the product graph $A \times A$ where if $r \xrightarrow{t_r} r'$ and $s \xrightarrow{t_s} s'$ in A, then $\langle r, s\rangle \xrightarrow{t_r - t_s} \langle r', s'\rangle$ in $A \times A$. The value of the max-min can be viewed as the outcome of a game, where the environment chooses a (maximising) move for the first vertex in the product graph, we choose a (minimising) move for the second vertex, and the game moves to the resulting vertex pair.

Suppose $n^2 W_{\max} < |\mathcal{S}(r,s,0)| < \infty$. Since there are only n^2 locations in the pair graph, and since each composite move can cause at most $W_{\max}$ lead or lag, there must be a cycle of composite locations in the game, with non-zero accumulative weight. When the game starts, we would do our best to not to get into such a cycle. If we cannot *avoid* getting into such a cycle because of observation matching of the environment moves, $|\mathcal{S}(v,s,0)|$ will be ∞, because the enviroment will force us to loop around that cycle forever. If $|\mathcal{S}(r,s,0)| < \infty$, and we *choose* to go into such a cycle, it must be the case that there is an alternative path/cycle that we can take which has accumated delay of the opposite sign. For example, it may happen that at some point in the game we have an option of going into two loops, loop 1 has total gain 10, loop 2 has total gain -1000. We will take loop 1 the first 500 times, then loop 2 once, then repeat with loop 1. The leads and lags cancel out in part keeping $\mathcal{S}(r,s,0)$ bounded. A finite value value of $\mathcal{S}(r,s,0)$ is then due to 1) some initial hard observation matching contraint steps, with the number of steps being less than n^2 (no cycles), and 2) presence of different weight cycles (note we never need to go around the maximum weight cycle more than once). A cycle in the pair graph can have weight at most $n^2 W_{\max}$. Hence the value of $|\mathcal{S}(r,s,0)|$ is bounded by $2n^2 W_{\max}$.

Given the upper bound, the value of $\mathcal{S}()$ can then be computed using dynamic programming (since all edges are integer valued, it suffices to restrict our attention to $\mathcal{S}(\cdot,\cdot,\delta)$ for integer valued δ). Further, this bound is tight: there is a finite timed graph A and two states r, s of A with $\mathcal{S}(r,s,0)$ in $\Theta(n^2 W_{\max})$.

Timed Automata. Timed automata provide a syntax for timed transition systems. A *timed automaton* A is a tuple $\langle L, \Sigma, C, \mu, \rightarrow, Q_0\rangle$, where

- L is the set of locations.
- Σ is the set of atomic propositions.
- C is a finite set of clocks. A *clock valuation* $v : C \mapsto \mathbb{R}^+$ for a set of clocks C assigns a real value to each clock in C.
- $\mu : L \mapsto 2^{\Sigma}$ is the observation map (it does not depend on clock values).
- $\rightarrow \subseteq L \times L \times 2^C \times \Phi(C)$ gives the set of transitions, where $\Phi(C)$ is the set of clock contraints generated by $\psi \; := \; x \leq d \mid d \leq x \mid \neg\psi \mid \psi_1 \wedge \psi_2$.
- $Q_0 \subseteq L \times \mathbb{R}^{+^{|C|}}$ is the set of initial states.

Each clock increases at rate 1 inside a location. A *state* is a location together with a clock valuation, the set of states is denoted $Q = L \times (\mathbb{R}^+)^{|C|}$. An edge $\langle l, l', \lambda, g\rangle$ represents a transition from location l to location l' when the clock values at l satisfy the constraint g. The set $\lambda \subseteq C$ gives the clocks to be reset with this transition. The semantics of timed automata are given as timed transition systems. This is standard [2], and omitted here.

For simplicity, we assume every clock of a timed automaton A stays within $M+1$, where M is the largest constant in the system. A *region* R of a timed automaton A is a tuple $\langle l, h, \mathcal{P}(C)\rangle$ where

- l is a location of A.
- h is a function which specifies the integer values of clocks $h : C \rightarrow (\mathbb{N} \cap [0, M])$ (M is the largest constant in A).
- $\mathcal{P}(C)$ is a disjoint partition of the clocks $\{X_0, \ldots X_n \mid \uplus X_i = C, X_i \neq \emptyset \text{ for } i > 0\}$.

We say a state s with clock valuation v is in the region R when,

1. The location of s corresponds to the location of R
2. For all clocks x with $v(x) < M+1$, $\lfloor v(x) \rfloor = h(x)$.
3. For $v(x) \geq M+1$, $h(x) = M$. (This is slightly more refined than the standard region partition, we have created more partitions in $[M, M+1)$, we map clock values which are greater than M into this interval. This is to simplify the proofs.)
4. Let $\text{frac}(r)$ denote the fractional value of r. For any pair of clocks (x, y), $\text{frac}(v(x)) < \text{frac}(v(y))$ iff $x \in X_i$ and $y \in X_j$ with $i < j$ (so, $x, y \in X_k$ implies $\text{frac}(v(x)) = \text{frac}(v(y))$).
5. $\text{frac}(v(x)) = 0$ iff $x \in X_0$.

We say two states s, s' to be *region equivalent* if they belong to the same region.

We now show that given states r, s in a timed automaton A, the values of $\mathcal{S}(r,s,0)$ and $\mathcal{B}(r,s,0)$ can be computed to within any desired degree of accuracy.

We use a *corner point abstraction* (similar to that in [4]) which can be viewed as a region graph augmented with additional timing information. We show that the corner points are at a close bisimilarity distance from the states inside the corresponding regions. Finally we use Theorem 1 to compute the approximation for $\mathcal{S}(\cdot)$ on the corner point graph.

A *corner point* is a tuple $\langle \alpha, R \rangle$, where $\alpha \in \mathbb{N}^{|C|}$ and R is a region. A region $R = \langle l, h, \{X_0, \dots X_n\} \rangle$ has $n+1$ corner points $\{\langle \alpha_i, R \rangle \mid 0 \leq i \leq n\}$:

$$\alpha_i(x) = \begin{cases} h(x) & : \quad x \in X_j \text{ with } j \leq i \\ h(x)+1 & : \quad x \in X_j \text{ with } j > i \end{cases}$$

Intuitively, corner points denote the boundary points of the region.

Using the corner points, we construct a finite timed graph as follows. The structure is similar to the region graph, only we use corner points, and weights on some of the edges to model the passage of time. For a timed automaton A, the *corner point abstraction* $\mathrm{CP}(A)$ has corner points p of A as states. The observation of the state $\langle \alpha, \langle l, h, \mathcal{P}(C) \rangle \rangle$ is $\mu(l)$. The abstraction has the following weighted transitions :

Discrete There is an edge $\langle \alpha, R \rangle \xrightarrow{0} \langle \alpha', R' \rangle$ if A has an edge $\langle l, l', \lambda, g \rangle$ (l, l' are locations of R, R' respectively) such that (1) R satisfies the constraint g, and (2) $R' = R[\lambda \mapsto 0]$, $\alpha' = \alpha[\lambda \mapsto 0]$ (note that corner points are closed under resets).

Timed For corner points $\langle \alpha, R \rangle, \langle \alpha', R \rangle$ such that $\forall x \in C,\ \alpha'(x) = \alpha(x)+1$, we have an edge $\langle \alpha, R \rangle \xrightarrow{1} \langle \alpha', R \rangle$. These are the edges which model the flow of time. Note that for each such edge, there are concrete states in A which are arbitrarily close to the corner points, such that there is a time flow of length arbitralily close to 1 in between those two states.

Region flow These transitions model the immediate flow transitions in between "adjacent" regions. Suppose $\langle \alpha, R \rangle, \langle \alpha, R' \rangle$ are such that R' is an immediate time successor of R, then we have an edge $\langle \alpha, R \rangle \xrightarrow{0} \langle \alpha, R' \rangle$. If $\langle \alpha+1, R' \rangle$ is also a corner point of R', then we also add the transition $\langle \alpha, R \rangle \xrightarrow{1} \langle \alpha+1, R' \rangle$.

Self loops Each state also has a self loop transition of weight 0.

Transitive closure We transitively close the timed, region flow, and the self loop transitions upto weight M (the subset of the full transitive closure where edges have weight less than or equal to M).

The number of states in the corner point abstraction of a timed automaton A is $O(|L| \cdot |C| \cdot (2M)^{|C|})$, where L is the set of locations in A, C the set of clocks, and M the largest constant in the system.

Lemma 1. *Let s be a state in a timed automaton A, and let p be a corner point of the region R corresponding to s in the corner point abstraction of A. Then s is ε-bisimilar to p for $\varepsilon = |C|+1$, that is, $\mathcal{S}(s, \mathsf{p}, 0) \leq |C|+1$, where C is the set of clocks in A.*

Informally, each clock can be the cause of at most 1 unit of time difference, as the time taken to hit a constraint is always of the form $d - v(x)$ for some clock x and integer d. Once a clock is reset, it collapses onto a corner point, and the time taken from that point to reach a constraint controlled by x is the same as that for the corresponding corner point in $\mathrm{CP}(A)$.

Using Lemma 1 and Theorem 1, we can "blow" up the time unit for a timed automaton to compute ε-simulation and ε-bisimilarity to within any given degree of accuracy. This gives an EXPTIME algorithm in the size of the timed automaton and the desired accuracy.

Theorem 2. *Given two states r, s in a timed automaton A, and a natural number m, we can compute numbers $\gamma_1, \gamma_2 \in \mathbb{R}$ such that $\mathcal{S}(r, s, 0) \in [\gamma_1 - \frac{1}{m}, \gamma_1 + \frac{1}{m}]$ and $\mathcal{B}(r, s, 0) \in [\gamma_2 - \frac{1}{m}, \gamma_2 + \frac{1}{m}]$ in time polynomial in the number of states of the corner point abstraction and in $m^{|C|}$, where C is the set of clocks of A.*

3 Robustness of Timed Computation Tree Logic

TCTL. Timed computation tree logic (TCTL) [1] is a real time extension of CTL [7]. TCTL adds time contraints such as ≤ 5 to CTL formulae for specifying timing requirements. For example, while the CTL formula $\forall \Diamond a$ only requires a to eventually hold on all paths, the TCTL formula $\forall \Diamond_{\leq 5} a$ requires a to hold on all paths before 5 time units.

We will use $\sim$ to mean one of the binary relations $<, \leq, >, \geq$. The formulae of TCTL are given inductively as follows:

$$\varphi \ := \ a \mid \text{FALSE} \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \exists(\varphi_1 \,\mathcal{U}_{\sim d} \varphi_2) \mid \forall(\varphi_1 \,\mathcal{U}_{\sim d} \varphi_2)$$

where $a \in \Sigma$ and $d \in \mathbb{N}$.

The semantics of TCTL formulas is given over states of timed transition systems. For a state s in a TTS

$s \models a$ iff $a \in \mu(s)$; $\quad s \not\models \text{FALSE}$; $\quad s \models \neg\varphi$ iff $s \not\models \varphi$.
$s \models \varphi_1 \vee \varphi_2$ iff $s \models \varphi_1$ or $s \models \varphi_2$.
$s \models \varphi_1 \wedge \varphi_2$ iff $s \models \varphi_1$ and $s \models \varphi_2$.
$s \models \exists(\varphi_1 \,\mathcal{U}_{\sim d} \varphi_2)$ iff for some run π_s starting from s, for some $t \sim d$, the state at time t, $\pi_s(t) \models \varphi_2$, and for all $0 \leq t' < t$, $\pi_s(t') \models \varphi_1$.
$s \models \forall(\varphi_1 \,\mathcal{U}_{\sim d} \varphi_2)$ iff for all (infinite) paths π_s starting from s, for some $t \sim d$, the state at time t, $\pi_s(t) \models \varphi_2$, and for all $0 \leq t' < t$, $\pi_s(t') \models \varphi_1$.

We define the *waiting-for* operator as $\exists(\varphi_1 \,\mathcal{W}_{\sim c} \varphi_2) = \neg\forall(\neg\varphi_2 \,\mathcal{U}_{\sim c} \neg(\varphi_1 \vee \varphi_2))$, $\forall(\varphi_1 \,\mathcal{W}_{\sim c} \varphi_2) = \neg\exists(\neg\varphi_2 \,\mathcal{U}_{\sim c} \neg(\varphi_1 \vee \varphi_2))$. The until operator in $\varphi_1 \,\mathcal{U}_{\sim d} \varphi_2$ requires that φ_2 become true at some time, the waiting-for formula $\varphi_1 \,\mathcal{W}_{\sim d} \varphi_2$ admits the possibility of φ_1 forever "waiting" for all times $t \sim d$ and φ_2 never being satisfied. Formally, $s \models \forall(\varphi \,\mathcal{W}_{\sim d} \theta)$ (respectively, $s \models \exists(\varphi \,\mathcal{W}_{\sim d} \theta)$) iff for all traces (respectively, for some trace) π_s from s, either 1) for all times $t \sim d$,

$\pi_s(t) \models \varphi$, or 2) at some time t, $\pi_s(t) \models \theta$, and for all $(t' < t) \wedge (t' \sim d)$, $\pi_s(t') \models \varphi$. Using the waiting-for operator and the identities $\neg(\varphi \exists \mathcal{U}_{\sim d} \theta) = (\neg\varphi) \forall \mathcal{W}_{\sim d}(\neg\varphi \wedge \neg\theta)$ and $\neg(\varphi \forall \mathcal{U}_{\sim d} \theta) = (\neg\varphi) \exists \mathcal{W}_{\sim d}(\neg\varphi \wedge \neg\theta)$, we can write each TCTL formula φ in negation normal form by pushing the negation to the atomic propositions.

δ-weakened TCTL. For each TCTL forumla φ in negation normal form, and $\delta \in \mathbb{R}^+$, a δ-weakening $\zeta^\delta(\varphi)$ of φ with respect to δ is defined as follows:

$$
\begin{aligned}
&\zeta^\delta(a) := a, \quad \zeta^\delta(\neg a) := \neg a, \quad \zeta^\delta(\text{FALSE}) := \text{FALSE} \\
&\zeta^\delta(\varphi_1 \vee \varphi_2) = \zeta^\delta(\varphi_1) \vee \zeta^\delta(\varphi_2), \quad \zeta^\delta(\varphi_1 \wedge \varphi_2) = \zeta^\delta(\varphi_1) \wedge \zeta^\delta(\varphi_2) \\
&\zeta^\delta(\ddagger(\varphi_1 \mathcal{U}_{\sim d} \varphi_2)) = \ddagger(\zeta^\delta(\varphi_1) \mathcal{U}_{\sim \delta(d,\sim)} \zeta^\delta(\varphi_2)), \\
&\zeta^\delta(\ddagger(\varphi_1 \mathcal{W}_{\sim d} \varphi_2)) = \ddagger(\zeta^\delta(\varphi_1) \mathcal{W}_{\sim \delta'(d,\sim)} \zeta^\delta(\varphi_2))
\end{aligned}
$$

where $\ddagger \in \{\exists, \forall\}$ and

$$
\delta(d,\sim) = \begin{cases} d+\delta & \text{if } \sim \in \{<, \leq\} \\ d-\delta & \text{if } \sim \in \{>, \geq\} \end{cases} \qquad \delta'(d,\sim) = \begin{cases} d-\delta & \text{if } \sim \in \{<, \leq\} \\ d+\delta & \text{if } \sim \in \{>, \geq\} \end{cases}
$$

The ζ^δ function relaxes the timing constraints by δ. The $\mathcal{U}$ and the $\mathcal{W}$ operators are weakened dually. Note that $\neg\zeta^\delta(\psi) \neq \zeta^\delta(\neg\psi)$. The discrepancy occurs because of the difference in how δ and δ' are defined.

Example 2. Let a and b be atomic propositions. We have $\zeta^2(\exists(a \mathcal{U}_{\leq 5} b)) = \exists(a \mathcal{U}_{\leq 7} b)$. Earlier, a state had to get to b within 5 time units, now it has 7 time units to satisfy the requirement. Similarly, $\zeta^2(\exists(a \mathcal{W}_{\leq 5} b)) = \exists(a \mathcal{W}_{\leq 3} b))$. The pre-weakened formula requires that either 1) for all $t \leq 5$ the proposition a must hold, or 2) at some time t, b must hold, and for all $(t' < t) \wedge (t' \leq 5)$ a must hold. The weakening operator relaxes the requirement on a holding for all times less than or equal to 5 to only being required to hold at times less than or equal to 3 (modulo the $(t' < t)$ clause in case 2). $\square$

Proposition 2. *For all reals $\delta \geq 0$,* TCTL *formulae φ, and states s of a* TTS*, if $s \models \varphi$, then $s \models \zeta^\delta(\varphi)$.*

We now connect the bisimilarity metric with satisfaction of TCTL specifications. Of course, close states may not satisfy the *same* TCTL specifications. Take $\varphi = \forall\Diamond_{=5} a$, it requires a to occur at exactly 5 time units. One state may have traces that satisfy a at exactly 5 time units, another state at $5+\varepsilon$ for an arbitrarily small ε. The first state will satisfy φ, the second will not. However, two states close in the bisimilarity metric does satisfy "close" TCTL specifications. Theorem 3 makes this precise.

Theorem 3. *Let $\varepsilon > 0$. Let r, s be two ε-bisimilar states of a timed transition system, and let φ a TCTL formula in negation normal form. If $r \models \varphi$, then $s \models \zeta^{2\varepsilon}(\varphi)$.*

The proof proceeds by induction on formulae. The crucial point is to note that if r, s are ε-bisimilar, and if, starting from r, s the bisimilarity game arrives at the configuration r_1, s_1, then r_1, s_1 are 2ε-bisimilar. So if $r \overset{t_1}{\rightsquigarrow} r_1 \overset{t_2}{\rightsquigarrow} r_2$, and $s \overset{t_1'}{\rightsquigarrow} s_1 \overset{t_2'}{\rightsquigarrow} s_2$ (with r_i, s_i being the corresponding states), then $|t_2 - t_2'| \leq 2\varepsilon$. The states r_1 and s_1 are *not* ε-bisimilar in general, but the traces originating from the two states are close and remain within 2ε.

4 Discounted CTL for Timed Systems

Our next step is to develop a quantitative specification formalism that assigns real numbers in the interval $[0, 1]$ to CTL formulas. A value close to 0 is "bad," a value close to 1 "good." We use time and *discounting* for this quantification. Discounting gives more weight to the near future than to the far away future. The resulting logic is called DCTL.

Syntax, Semantics, and Robustness. We look at a subset of standard boolean CTL, with $\diamond$ being the only temporal operator. The formulae of DCTL are inductively defined as follows:

$$\varphi \;:=\; a \mid \text{FALSE} \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \exists\diamond\varphi \mid \forall\diamond\varphi$$

where a ranges over atomic propositions. From this, we derive the formulas: $\exists\Box\varphi = \neg\forall\diamond\neg\varphi$ and $\forall\Box\varphi = \neg\exists\diamond\neg\varphi$.

The semantics of DCTL formulas are given as functions from states to the real interval $[0, 1]$. For a *discount parameter* $\beta \in [0, 1]$, and a timed transition system, the value of a formula φ at a state s is defined as follows:

$[\![a]\!](s) := 1$ if $s \models a$, 0 otherwise.
$[\![\text{FALSE}]\!](s) := 0; \qquad [\![\neg\varphi]\!](s) := 1 - [\![\varphi]\!](s)$
$[\![\varphi_1 \left\{\begin{smallmatrix}\vee\\ \wedge\end{smallmatrix}\right\} \varphi_2]\!](s) := \left\{\begin{smallmatrix}\max\\ \min\end{smallmatrix}\right\}\{[\![\varphi_1]\!](s), [\![\varphi_2]\!](s)\}$
$[\![\left\{\begin{smallmatrix}\exists\\ \forall\end{smallmatrix}\right\}\diamond\varphi]\!](s) := \left\{\begin{smallmatrix}\sup\\ \inf\end{smallmatrix}\right\}_{\pi_s} \sup_{t\in\mathbb{R}^+}\{\beta^t([\![\varphi]\!](\pi_s(t)))\}$

where π_s is an infinite time diverging path starting from state s, and $\pi_s(t)$ is the state on that path at time t. Intuitively, for the $\diamond$ operator, the quicker we can get to a good state, the better, and the discounted value reflects this fact. The temporal operators can again be seen as playing a game. Environment chooses the path π_s, and we choose the best value on that path. In $\exists\diamond$ the environment is cooperating and chooses the best path, in $\forall\diamond$, it plays adversially and takes the worst path. Note that $\beta = 1$ gives us the boolean case.

Example 3. Consider T_r in Figure 1. Assume we cannot stay at a location forever (location invariants can ensure this). The value of $\forall\diamond b$ at the state $\langle a, x = 0, y = 0\rangle$ is β^6. The automaton must move from a to b within 6 time units, for otherwise it will get stuck at c and not be able to take the transition to d. Similarly, the value at the starting state in T_s is β^7.

Consider now the formula $\forall\Box(b \Rightarrow \forall\diamond a) = \neg\exists\diamond\neg(\neg b \vee \forall\diamond a) = 1 - \exists\diamond(\min(b, (1 - \forall\diamond a)))$. What is its value at the starting state, $\langle a, 0, 0\rangle$, of T_r?

The value of $\min(b,\cdot)$ is 0 at states not satisfying b, so we only need look at the b location in the outermost $\exists\Diamond$ clause. T_r needs to move out of b within 9 time units (else it will get stuck at c). Thus we need to look at states $\langle b, 0 \le x \le 9, 0 \le y \le 4\rangle$. On those states, we need the value of $\forall\Diamond a$. Suppose we enter b at time t. Then the b states encountered are $\{\langle b, t+z, z\rangle \mid z \le 4, t+z \le 9\}$. The value of $\forall\Diamond a$ at a state $\langle b, t+z, z\rangle$ is $\beta^{3+9-(t+z)}$ (we exit c at time $9-(t+z)$, and can avoid a for 3 more time units). Thus the value of $\exists\Diamond(\min(b,(1-\forall\Diamond a)))$ at the initial state is $\sup_{t,z}\{\beta^{t+z}(1-\beta^{3+9-(t+z)}) \mid z \le 4, t+z \le 9\}$ (view $t+z$ as the elapsed time; the individual contributions of t and z in the sum depending on the choice of the path). The maximum value occurs when $t+z$ is 0. Thus the value of the sup is $1-\beta^{12}$. So finally we have the value of $\forall\Box(b \Rightarrow \forall\Diamond a)$ at the starting state to be β^{12}. It turns out that the initial state in T_s has the same value for $\forall\Box(b \Rightarrow \forall\Diamond a)$. Both systems have the same "response" times for an a following a b. $\square$

dCTL is robust with respect to ε-bisimilarity: close states in the bisimilarity metric have close dCTL values. Notice however that the closeness is not uniform and may depend on the nesting depth of temporal operators [10].

Theorem 4. *Let k be the number of nested temporal operators in a* dCTL *formula φ, and let β be a real discount factor in* $[0,1]$. *For all states r, s in a* TTS, *if $|\mathcal{B}(r,s,0)| \le \varepsilon$, then $|[\![\varphi]\!](r) - [\![\varphi]\!](s)| \le (k+1)(1-\beta^{2\varepsilon})$.*

Example 4. Consider $\forall\Diamond b$ at the starting states (which are 1-bisimilar) in T_r, T_s in Fig. 1. As shown in Ex. 3, the value in T_r is β^6, and β^7 in T_s. $\beta^6 - \beta^7 = \beta^6(1-\beta) \le 1-\beta \le 1-\beta^2$. $\square$

Model Checking dCTL over Timed Automata. We compute the value of $[\![\varphi]\!](s)$ as follows: for $\varphi = \exists\Diamond\theta$, first recursively obtain $[\![\theta]\!](v)$ for each state v in the TTS. The value of $[\![\varphi]\!](s)$ is then $\sup\{\beta^{t_v}([\![\theta]\!](v))\}$, where t_v is the shortest time to reach state v from state s. For $\varphi = \forall\Diamond\theta$, we need to be a bit more careful. We cannot simply take the longest time to reach states and then have an outermost inf (i.e., dual to the $\exists\Diamond$ case). The reason is that the $\exists\Diamond$ case had $\sup_{\pi_s}\sup_t$, and both the sups can be collapsed into one. The $\forall\Diamond$ case has $\inf_{\pi_s}\sup_t$, and the actual path taken to visit a state matters. For example, it may happen that on the longest path to visit a state v, we encounter a better value of θ before v say at u; and on some other path to v, we never get to see u, and hence get the true value of the inf. The value for a formula at a state in a finite timed graph can be computed using the algorithms in [10] (with trivial modifications). Timed automata involve real time and require a different approach. We show how to compute the values for a subset of dCTL on the states of a timed automaton.

Let $F_{\min}(s, Z)$ denote the set of times that must elapse in order for a timed automaton A to hit some configuration in the set of states Z starting from the state s. Then the minimum time to reach the set Z from state s (denoted by $t_{\min}(s,Z)$) is defined to be the inf of the set $F_{\min}(s,Z)$. The maximum time to reach a set of states Z from s for the first time ($t_{\max}(s,Z)$) can be defined dually.

Theorem 5 ([9]). *(1) For a timed automaton A, the minimum and maximum times required to reach a region R from a state s for the first time $(t_{\min}(s,R), t_{\max}(s,R))$ are computable in time $O(|C| \cdot |G|)$ where C is the set of clocks in A, and G is the region automaton of A. (2) For regions R and R', either there is an integer constant d such that for every state $s \in R'$, we have $t_{\min}(s,R) = d$, or there is an integer constant d and a clock x such that for every state $s \in R'$, we have $t_{\min}(s,R) = d - \mathrm{frac}(t_x)$, where t_x is the value of clock x in s; and similarly for $t_{\max}(s,R)$.*

We note that for any state s, we have $[\![P]\!](s)$ is 0 or 1 for a boolean combination of propositions P, and this value is constant over a region. Thus the value of $[\![\exists\Diamond P]\!](s)$ is $\beta^{t_{\min}}$ where $t_{\min}$ is the shortest time to reach a region satisfying P from s. For computing $\forall\Diamond P$, we look at the inf-sup game where the environment chooses a path π_s, and we pick a state $\pi_s(t)$ on that path. The value of the game resulting from these choices is $\beta^t P(\pi_s(t))$. Enviroment is trying to minimise this value, and we are trying to maximise. Given a path, we will pick the earliest state on that path satisfying P. Thus the environment will pick paths which avoid P the longest. Hence, the value of $[\![\forall\Diamond P]\!](s)$ is $\beta^{t_{\max}}$ where $t_{\max}$ is the maximum time that can be spent avoiding regions satisfying P. The next theorem generalizes Theorem 5 to pairs of states. A state is integer (resp., rational) valued if its clock valuation maps each clock to an integer (resp., rational).

Theorem 6. *(1) Let r be an integer valued state in a timed automaton A. Then $t_{\min}(r,s)$, the minimum time to reach the state s from r is computable in time $O(|C| \cdot |G|)$ where C is the set of clocks in A, and G is the region automaton of A. (2) For a region R', either there is an integer constant d such that for every state $s \in R'$, we have $t_{\min}(r,s) = d$; or there is an integer constant d and a clock x such that $t_{\min}(r,s) = d + \mathrm{frac}(t_x)$, where t_x is the value of clock x in s.*

Theorem 6 is based on the fact that if a timed automaton can take a transition from s to s', then 1) for every state w region equivalent to s, there is a transition $w \to w'$ where w' is region equivalent to s', and 2) for every state w' region equivalent to s', there is a transition $w \to w'$ where w is region equivalent to s. If π_r is a trajectory starting from r and ending at s with minimal delay, then for any other state s' region equivalent to s, there is a corresponding minimum delay trajectory π'_r from r which makes the same transitions as π_r, in the same order, going through the same regions of the region graph (only the timings may be different). Note that an integer valued state constitutes a seperate region by itself. Theorem 6 is easily generalised to rational valued initial states using the standard trick of multiplying automata guards with integers.

Theorem 7. *Let φ be a* DCTL *formula with no nested temporal operators. Then $[\![\exists\Diamond\varphi]\!](s)$ (and so $[\![\forall\Box\varphi]\!](s)$) can be computed for all rational-valued states s of a timed automaton.*

Let φ be a boolean combination of formulas of form $[\![\{{}^{\exists}_{\forall}\}\Diamond P]\!]$ (P a boolean combination of propositions). We have shown $[\![\varphi]\!](s')$ to be computable for all states (and moreover it to have a simple form over regions). The value of $[\![\exists\Diamond\varphi]\!](s)$ is then $\sup\{\beta^{t_{\min}(s,s')}[\![\varphi]\!](s')\}$. The sup as s' varies over a region is easily computable, as both $[\![\varphi]\!](s')$ and $t_{\min}(s,s')$ have uniform forms over regions. We can then take a max over the regions. Let $|G|$ be the size of the region graph, $|G(Q)|$ the number of regions. Then computation of φ over all regions takes time $O(|G(Q)| \cdot |C| \cdot |G|)$. The computation of the minimum time in Theorem 6 takes $O(|C| \cdot |G| \cdot m^{|C|})$, where m is the least common multiple of the denominators of the rational clock values. Thus, the value of the formula $\exists\Diamond\varphi$ can be computed in time $O(|C|^2 \cdot |G|^2 \cdot |G(Q)| \cdot m^{|C|})$, i.e., polynomial in the size of the region graph and in $m^{|C|}$.

We can also compute the maximum time that can elapse to go from a rational valued state to any (possibly irrational valued) state, but that does not help in the computation in the $\forall\Diamond\varphi$ case, as the actual path taken is important. We can do it for the first temporal operator since then φ is a boolean combination of propositions, and either 0 or 1 on regions. In the general case φ can have some real value in $[0,1]$, and this boolean approach does not work. Incidentally, note that its not known whether the maximum time problem between two general states is decidable. The minimum time problem *is* decidable for general states via a complicated reduction to the additive theory of real numbers [8]. Whether these techniques may be used to get a model checking algorithm for dCTL is open.

5 Conclusion

Quantitative simulation and bisimulation functions precisely characterize the degree of closeness between timed systems, generalizing the (boolean) notions of timed simulation and bisimulation. For formal models of systems where the exact (infinite-precision) values of numerical quantities may not be known, such quantitative metrics provide a natural theory of refinement. Further, the metrics quantify the robust satisfaction of TCTL specifications: states close under the metric satisfy specifications with close timing requirements.

We also presented a quantitative theory for discounted CTL for timed systems. Discounted theories have been presented before for discrete systems [12, 10] which discount each individual transition by some discount factor. We discount *time* rather than individual *transitions*. Equal times are discounted by an equal amount irrespective of the number of transitions involved. We showed dCTL is robust: the values for a formula converge as the states converge in the bisimilarity metric. Finally, we showed a subset of dCTL to be computable over timed automata, and indicated why the continuous nature introduces difficulty over the discrete case. The general model checking problem for dCTL remains open.

References

1. R. Alur, C. Courcoubetis, and D.L. Dill. Model checking in dense real time. *Inf. and Comp.*, 104:2–34, 1993.

2. R. Alur and D.L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126:183–235, 1994.
3. R. Alur, S. La Torre, and P. Madhusudan. Perturbed timed automata. In *HSCC 05*, LNCS 3414, pages 70–85. Springer, 2005.
4. P. Bouyer, E. Brinksma, and K.G. Larsen. Staying alive as cheaply as possible. In *HSCC 04*, LNCS 2993, pages 203–218. Springer, 2004.
5. P. Caspi and A. Benveniste. Toward an approximation theory for computerised control. In *EMSOFT 02*, LNCS 2491, pages 294–304. Springer, 2002.
6. K. Cerans. Decidability of bisimulation equivalences for parallel timer processes. In *CAV 92*, LNCS 663, pages 302–315. Springer, 1992.
7. E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8:244–263, 1986.
8. H. Comon and Y. Jurski. Timed automata and the theory of real numbers. In *CONCUR 99*, LNCS 1664, pages 242–257. Springer, 1999.
9. C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design*, 1:385–415, 1992.
10. L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, and M. Stoelinga. Model checking discounted temporal properties. In *TACAS 04*, LNCS 2988, pages 77–92. Springer, 2004.
11. L. de Alfaro, M. Faella, and M. Stoelinga. Linear and branching metrics for quantitative transition systems. In *ICALP 04*, LNCS 3142, pages 97–109, 2004.
12. L. de Alfaro, T.A. Henzinger, and R. Majumdar. Discounting the future in systems theory. In *ICALP 03*, LNCS 2719, pages 1022–1037. Springer, 2003.
13. M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. In *FORMATS 04*, LNCS 3253, pages 118–133. Springer, 2004.
14. J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labeled Markov processes. *Theor. Comput. Sci.*, 318:323–354, 2004.
15. M. Fränzle. Analysis of hybrid systems: An ounce of realism can save an infinity of states. In *CSL 99*, LNCS 1683, pages 126–140. Springer, 1999.
16. V. Gupta, T.A. Henzinger, and R. Jagadeesan. Robust timed automata. In *HART 97*, LNCS 1201, pages 331–345. Springer, 1997.
17. V. Gupta, R. Jagadeesan, and P. Panangaden. Approximate reasoning for real-time probabilistic processes. In *QEST 04*, pages 304–313. IEEE, 2004.
18. T.A. Henzinger and J.-F. Raskin. Robust undecidability of timed and hybrid systems. In *HSCC 00*, LNCS 1790, pages 145–159. Springer, 2000.
19. J. Huang, J. Voeten, and M. Geilen. Real-time property preservation in approximations of timed systems. In *MEMOCODE 03*, pages 163–171. IEEE, 2003.
20. J. Huang, J. Voeten, and M. Geilen. Real-time property preservation in concurrent real-time systems. In *RTCSA 04*. Springer, 2004.
21. A. Puri. Dynamical properties of timed automata. In *FTRTFT 98*, LNCS 1486, pages 210–227. Springer, 1998.
22. S. Tasiran. *Compositional and Hierarchical Techniques for the Formal Verification of Real-Time Systems*. Dissertation, UC Berkeley, 1998.
23. M. De Wulf, L. Doyen, and J.-F. Raskin. Almost ASAP semantics: From timed models to timed implementations. In *HSCC 04*, LNCS 2993, pages 296–310. Springer, 2004.