State Equivalences for Rectangular Hybrid Automata*

Thomas A. Henzinger†
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720

Peter W. Kopke‡
Computer Science Department, Cornell University
Ithaca, NY 14853

Abstract. Three natural equivalence relations on the infinite state space of a hybrid automaton are language equivalence, simulation equivalence, and bisimulation equivalence. When one of these equivalence relations has a finite quotient, certain model checking and controller synthesis problems are decidable. When bounds on the number of equivalence classes are obtained, bounds on the running times of model checking and synthesis algorithms follow as corollaries.

We characterize the time-abstract versions of these equivalence relations on the state spaces of *rectangular hybrid automata* (RHA), in which each continuous variable is a clock with bounded drift. These automata are useful for modeling communications protocols with drifting local clocks, and for the conservative approximation of more complex hybrid systems. Of our two main results, one has positive implications for automatic verification, and the other has negative implications. On the positive side, we find that the (finite) language equivalence quotient for RHA is coarser than was previously known by a multiplicative exponential factor. On the negative side, we show that simulation equivalence for RHA is equality (which obviously has an infinite quotient).

Our main positive result is established by analyzing a subclass of timed automata, called *one-sided timed automata* (OTA), for which the language equivalence quotient is coarser than for the class all of timed automata. An exact characterization of language equivalence for OTA requires a distinction between synchronous and asynchronous definitions of (bi)simulation: if time actions are silent, then the induced quotient for OTA is coarser than if time actions are visible.

*A preliminary version of this paper appeared in the *Proceedings of the Seventh International Conference on Concurrency Theory* (CONCUR 96), *Lecture Notes in Computer Science* **1119**, Springer-Verlag, 1996, pp. 530–545.

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

‡Supported by the U.S. Army Research Office through the Mathematical Sciences Institute of Cornell University, Contract Number DAAL03-91-C-0027.

1 Introduction

A *hybrid automaton* consists of a finite automaton interacting with a continuous dynamical system on $\mathbb{R}^n$ [ACHH93, NOSY93]. Hybrid automata are used to model embedded controllers and other systems that consist of interacting discrete and continuous components. If the continuous dynamics are fixed, one can ask when two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ are distinguishable by some hybrid automaton from a given class. The purpose of this exercise is to find finite quotients of the infinite state space $\mathbb{R}^n$ so that algorithmic verification can be performed. For example, if every continuous variable moves uniformly at slope 1 (as in *timed automata*), then there is a finite bisimulation equivalence quotient on $\mathbb{R}^n$ [AD94]. It follows that verification and control problems for timed automata can be solved using finite-state methods.

We consider *time-abstract* distinguishability, in which the durations of time delays are not taken into account. The opposite approach, *timed* distinguishability, leads to a sterile theory in which, relative to any interesting class of hybrid automata, no two states are equivalent. Moreover, any specific timing constraint that is required of a system can (and should) be specified by the introduction of a clock variable that enforces the constraint [Hen95]. Thus it is unnecessary to burden the semantics by the introduction of infinitely many time actions.

We consider three types of time-abstract distinguishability, or, dually, three types of equivalence: language equivalence, simulation equivalence[1], and bisimulation equivalence. Each equivalence relation $\equiv$ has a direct connection to a temporal logic $\mathrm{TL}_{\equiv}$ (bisimulation equivalence corresponds to the full branching-time logic CTL* [BCG88]; simulation equivalence corresponds to the universal fragment of CTL* [BBLS92]; language equivalence corresponds to linear temporal logic). When the quotient of $\equiv$ is finite (and computable) for a class $\mathcal{C}$ of hybrid automata, the model checking problem for $\mathrm{TL}_{\equiv}$ and the controller synthesis problem for invariance are decidable on the class $\mathcal{C}$ [Hen96]. Moreover, when the number of equivalence classes is determined, upper bounds on the running times of the model checking and controller synthesis algorithms are obtained.

We distinguish between two models of each equivalence. Hybrid automata have time actions and discrete actions. In the *synchronous model*, time actions are visible, while in the *asynchronous model*, time actions are silent. Thus in the synchronous model of simulation, discrete actions must be matched by discrete actions and time actions must be matched by time actions, while in the asynchronous model, each discrete action may be matched by a silent time action followed by a matching discrete action. Language equivalence is inherently asynchronous. However, it is profitable to consider both models of (bi)simulation, because asynchronous (bi)simulation equivalence may be coarser than synchronous (bi)simulation equivalence; and so the subtle distinction is necessary for a precise analysis.[2] We therefore study five different equivalence relations.

We are interested in *rectangular hybrid automata (RHA)*, in which each continuous variable x is a clock with bounded drift, and therefore follows a nondeterministic differential equation of the form $\frac{dx}{dt} \in [a, b]$. Rectangular hybrid automata are useful for modeling distributed communication protocols [HW95], and for approximating nonlinear hybrid systems [HH95].

Two well-studied classes of hybrid automata are *timed automata* [AD94] and *linear hybrid automata (LHA)* [ACH$^+$95]. In a timed automaton, every continuous variable x is a precise clock, and therefore satisfies the differential equation $\frac{dx}{dt} = 1$. In a linear hybrid automaton, the first derivatives of the continuous variables satisfy linear relationships. Linearity allows many properties of LHA to be analyzed symbolically [AHH96]. Rectangular hybrid automata are an intermediate class, more general than timed automata, but less general than linear hybrid automata. For timed

[1] Two states are *simulation equivalent* if each simulates the other.

[2] It has come to our attention that the two models of bisimulation are introduced independently in [TY96].

automata, all five equivalence relations coincide (with a relation called *region equivalence*, which considers two vectors to be equivalent if their components (1) have the same integer parts, and (2) their fractional parts have the same relative ordering). For linear hybrid automata, each of the five equivalence relations degenerates to equality on $\mathbb{R}^n$. The intermediate class of rectangular hybrid automata is amenable to automatic verification because even though RHA are more general than timed automata, their language equivalence quotient is still finite [PV94]. On the other hand, the synchronous bisimulation quotient for RHA is known to be infinite [Hen95]. Our goal is to elucidate the relationships between the five equivalence relations for the class of rectangular hybrid automata: we determine which quotients are finite, and provide bounds on the number of equivalence classes.

We develop two main results, which we discuss throughout the remainder of this introduction. The first shows that language equivalence for RHA is coarser than was previously known by a multiplicative exponential factor. We therefore obtain an improved model checking algorithm for linear temporal logic on RHA. The second shows that, contrary to a previous conjecture [HHK95], both the synchronous and asynchronous simulation equivalence relations (and hence also both bisimulation equivalence relations) for RHA degenerate to equality. In fact, equality is the only synchronous simulation already in three dimensions (i.e., when $n = 3$). It follows that symbolic algorithms, such as those implemented in HyTech [HHWT95, AHH96], need not terminate when applied to universal CTL* properties of RHA.

Language Equivalence for Rectangular Hybrid Automata. Rectangular hybrid automata generalize timed automata, yet incur no complexity penalty for several decision problems [HKPV95]. In particular, the language emptiness problem for RHA is PSPACE-complete, due to the existence of a language-preserving translation from rectangular hybrid automata into timed automata. Examining this translation, we find that the clocks of the resulting timed automata have a special property: each clock is constrained in only one direction. We study these *one-sided timed automata (OTA)*, which have two sets of clocks: upper-bounded clocks are constrained by guards only from above, while lower-bounded clocks are constrained by guards only from below. We show that for OTA, language equivalence and asynchronous simulation equivalence coincide, and are coarser than region equivalence by a multiplicative factor of 2^n. Since two states of a hybrid automaton are language equivalent if they are translated to language equivalent states of a one-sided timed automaton, we obtain a new sufficient condition for language equivalence for RHA as a corollary. For symbolic model checkers, such as HyTech, Kronos [DY95], and Uppaal [LPY95], our results may be applied to obtain better bounds on performance. For enumerative model checking, we obtain better algorithms, because we reduce the number of equivalence classes that must be enumerated.

We illustrate by an example the increased compaction of the state space on rectangular hybrid automata given by our main positive result. Consider the class $\mathcal{C}$ of all rectangular hybrid automata with three continuous variables, each a clock drifting between slopes 1 and 2. Consider the 3-vectors $\mathbf{y} = (0.4, 0.3, 0.25)$ and $\mathbf{y}' = (0.25, 0.3, 0.4)$. The translation of RHA into OTA maps a 3-vector of drifting clock values into a 6-vector of three upper-bounded clock values and three lower-bounded clock values. The former are obtained by doubling the drifting clock values, while the latter coincide with the drifting clock values. Thus the 3-vector $\mathbf{y}$ translates into the 6-vector $\mathbf{z} = (0.8, 0.6, 0.5, 0.4, 0.3, 0.25)$, and the 3-vector $\mathbf{y}'$ translates into $\mathbf{z}' = (0.5, 0.6, 0.8, 0.25, 0.3, 0.4)$, where the first three components represent upper-bounded clock values, and the last three components represent lower-bounded clock values. See Figure 1. The two 6-vectors are not equivalent with respect to the class of all timed automata, because $z_1 > z_2$ while $z'_1 < z'_2$. To see this, consider a timed automaton T that has an edge e guarded by the predicate $x_1 \geq 1 \geq x_2$, where x_i denotes the value of the ith clock variable. After a delay of duration $\delta \in \mathbb{R}_{\geq 0}$, where $1 - z_2 \geq \delta \geq 1 - z_1$,

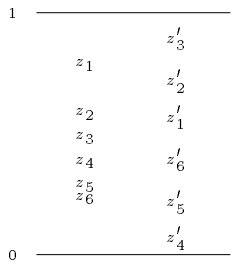

Figure 1: Vectors $\mathbf{z}$ and $\mathbf{z}'$ are one-sided region equivalent

the vector $\mathbf{z}$ is transformed into the vector $\mathbf{z} + \delta\mathbf{1}$, which satisfies $z_1 + \delta \geq 1 \geq z_2 + \delta$. But for every duration $\delta' \in \mathbb{R}_{\geq 0}$, vector $\mathbf{z}'$ does not satisfy $z'_1 + \delta' \geq 1 \geq z'_2 + \delta'$. Thus the edge e with its guard $x_1 \geq 1 \geq x_2$ distinguishes between $\mathbf{z}$ and $\mathbf{z}'$. However, no one-sided timed automaton has such a guard, which constrains clocks of the same type (in this case, upper-bounded) both from above and from below. Our results enable us to deduce that $\mathbf{y}$ and $\mathbf{y}'$ are language equivalent with respect to $\mathcal{C}$, because $\mathbf{z}$ and $\mathbf{z}'$ are language equivalent with respect to the class of one-sided timed automata. This equivalence does not depend on the relative order of lower-bounded clocks, nor upon the relative order of upper-bounded clocks, but only on the relative order of the lower-bounded clocks with respect to the upper-bounded clocks.

Simulation Equivalence of Rectangular Hybrid Automata. A structural explanation for the finiteness of the language equivalence quotient of RHA was sought in [HHK95]. It was there shown that every 2D RHA has a finite synchronous simulation equivalence quotient (and therefore a finite asynchronous simulation equivalence quotient). It was conjectured that this result generalized to RHA of arbitrary dimension. We disprove this conjecture, showing that, in fact, the only synchronous simulation on three-dimensional RHA is equality. Asynchronous simulation equivalence relation is equality in four or more dimensions (we do not know if the asynchronous simulation equivalence quotient is finite or infinite in three dimensions).

The remainder of the paper is organized as follows. Section 2 provides definitions and previous results. Section 3 presents an analysis of one-sided timed automata. In Section 4 the results of Section 3 are applied to the language equivalence problem for rectangular hybrid automata, and then the synchronous and asynchronous simulation equivalence relations for RHA are characterized. Section 5 summarizes our results; a tabulation is given in Figure 7.

2 Definitions and Previous Results

Fix a positive integer n. A *rectangle* B is a closed subset of $\mathbb{R}^n$ that is the cartesian product of (possibly unbounded) intervals on $\mathbb{R}$, all of whose finite endpoints are integers. We write B_i for the projection of B on the ith coordinate, so that $B = \prod_{i=1}^{n} B_i$.

For a closed interval $I \subseteq \mathbb{R}$ and a number $x \in \mathbb{R}$, define $\mathit{closest}(x, I)$ to be the closest point to x lying in I. So if $x \in I$, then $\mathit{closest}(x, I) = x$; if $x \leq \min I$ then $\mathit{closest}(x, I) = \min I$; and if $x \geq \max I$ then $\mathit{closest}(x, I) = \max I$. A *guarded command* is a triple $g = (B, U, F)$, where the *guard* B and the *reset* F are rectangles, and $U \subseteq \{1, \ldots, n\}$ is the *update set*. The guarded command g defines a binary relation jump_g on $\mathbb{R}^n$ by $(\mathbf{x}, \mathbf{x}') \in \mathit{jump}_g$ iff (1) $\mathbf{x} \in B$, (2) for each $i \in U$, $x'_i \in F_i$, and (3) for each $i \notin U$, $x'_i = \mathit{closest}(x_i, F_i)$. Thus the guarded command g may act on $\mathbf{x}$ iff $\mathbf{x}$ satisfies the guard. The result of the action is to reassign each coordinate $i \in U$ nondeterministically to a value in F_i, and to reassign each coordinate $i \notin U$ to the closest value

in F_i. In particular, if $F = B$, then the coordinates $i \notin U$ are left unchanged. The set of all guarded commands is denoted $\mathcal{G}_n$.

Rectangular Hybrid Automata.

An n-dimensional *rectangular hybrid automaton* A [HKPV95] over the alphabet Σ consists of a directed multigraph (V_A, E_A), an n-dimensional compact (closed and bounded) rectangle act_A with positive endpoints called the *activity rectangle*, and two edge labeling functions $guard_A : E \to \mathcal{G}_n$ and $event_A : E_A \to \Sigma$.[3] When only one automaton is under consideration, we omit the subscript on the components of A.

A *state* of A is a pair $(v, \mathbf{x})$ consisting of a discrete part $v \in V_A$ and a continuous part $\mathbf{x} \in \mathbb{R}^n$; the *state space* of A is $V_A \times \mathbb{R}^n$. The automaton A defines a labeled transition system on its state space as follows. The *time action* $\stackrel{time}{\to}$ is defined by $(v, \mathbf{x}) \stackrel{time}{\to} (v', \mathbf{x}')$ iff (1) $v' = v$ and (2) either $\mathbf{x}' = \mathbf{x}$ or $\frac{\mathbf{x}'-\mathbf{x}}{t} \in act_A$ for some $t \in \mathbb{R}_{>0}$. Therefore as time passes, the continuous state of A follows a trajectory satisfying the nondeterministic differential equation $\frac{d\mathbf{x}}{dt} \in act_A$. For $\sigma \in \Sigma$, the *discrete action* $\stackrel{\sigma}{\to}$ is defined by $(v, \mathbf{x}) \stackrel{\sigma}{\to} (v', \mathbf{x}')$ iff there exists an edge $e = (v, v') \in E_A$ such that $event(e) = \sigma$ and $(\mathbf{x}, \mathbf{x}') \in jump_{guard(e)}$. A discrete action $\stackrel{\sigma}{\to}$ is thus only enabled in vertex v when the continuous state satisfies the guard of an edge e with source v and event σ, and the resulting continuous state is obtained by applying the guarded command of e.

A *run of A beginning at* $(v, \mathbf{x})$ is a finite sequence

$$(v^0, \mathbf{x}^0) \stackrel{time}{\to} (v^1, \mathbf{x}^1) \stackrel{\sigma_1}{\to} \cdots \stackrel{time}{\to} (v^{2k-1}, \mathbf{x}^{2k-1}) \stackrel{\sigma_k}{\to} (v^{2k}, \mathbf{x}^{2k})$$

of alternating time actions and discrete actions, where $(v^0, \mathbf{x}^0) = (v, \mathbf{x})$. This run *generates* the string $\sigma_1\sigma_2 \cdots \sigma_k \in \Sigma^*$. The language $L_A(v, \mathbf{x})$ of state $(v, \mathbf{x})$ is the set of strings $\overline{\sigma} \in \Sigma^*$ that are generated by runs of A beginning at $(v, \mathbf{x})$.

One-sided Timed Automata.

A *timed automaton* is a rectangular hybrid automaton T such that act_T is the singleton $\{(1, 1, \ldots, 1)\}$ consisting of the vector with all components equal to 1. Thus each coordinate of the continuous state of a timed automaton moves deterministically and uniformly at slope 1 as time passes. We say that coordinate i is an *upper-bounded clock* if for each edge $e \in E_T$, if $guard(e) = (B, U, F)$, then B_i (which is an interval) is unbounded from below. This means that the ith coordinate of the continuous state of T is constrained by guards only from above. We say that coordinate j is a *lower-bounded clock* if for each edge $e \in E_T$, if $guard(e) = (B, U, F)$, then B_j is unbounded from above. This means that the jth coordinate of the continuous state of T is constrained by guards only from below. The timed automaton T is *upper-bounded* (resp. *lower-bounded*) if each coordinate is an upper-bounded (resp. lower-bounded) clock. The timed automaton T is *one-sided* if every coordinate is either an upper-bounded clock or a lower-bounded clock.

Every n-dimensional rectangular hybrid automaton can be translated into a $2n$-dimensional one-sided timed automaton with the same language. The mapping from $\mathbb{R}^n$ to $\mathbb{R}^{2n}$ is uniform among all RHA with the same activity rectangle, and therefore allows characterizations of language

[3] There is one important difference between the model of RHA we present and the one used in [HKPV95]. Our automata have a constant activity rectangle, and therefore correspond to the *initialized* RHA of [HKPV95]. Three other differences appear in order to simplify the presentation; they do not materially affect our results. First, we have restricted attention to closed rectangles. Second, we have not included vertex invariants in the definition. Third, we have slightly generalized the reset mechanism.

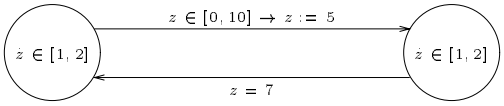

Figure 2: A rectangular automaton A

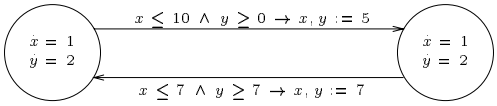

Figure 3: The one-sided, constant-slope automaton M_A

equivalence for one-sided timed automata to yield sufficient conditions for language equivalence for the class of rectangular hybrid automata with a fixed activity rectangle.

Theorem 1 [HKPV95] *Let $R \subseteq \mathbb{R}^n$ be a compact rectangle. For every vector $\mathbf{z} \in \mathbb{R}^n$, there exist vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ such that for every rectangular hybrid automaton A with $act_A = R$, there exists a $2n$-dimensional one-sided timed automaton T_A such that for every vertex v, $L_A(v, \mathbf{z}) = L_{T_A}(v, (\mathbf{x}, \mathbf{y}))$.*

Proof Outline. Given A, we define an RHA M_A in which each continuous variable has constant slope. The automaton M_A essentially acts in the same way as the subset construction for finite automata, giving bounds on all possible behaviors of A. Each continuous variable z with $\dot{z} \in [a, b]$ is replaced by two variables x and y, where $\dot{x} = a$ and $\dot{y} = b$, which give exact lower and upper bounds on the possible values for z. Testing if $z \in [c, d]$ is equivalent to testing if $x \leq d \wedge y \geq c$, and then resetting x and y to the closest points in $[c, d]$.[4] Thus in M_A, x is only bounded from above, and y is only bounded from below. The constant-slope automaton M_A is transformed into a one-sided timed automaton by first dividing each guard and reset for x by a, and for y by b, and then multiplying all guards and resets by the least common multiple of all of the slopes. An example is given in Figures 2, 3, and 4. ■

Equivalences for Rectangular Hybrid Automata.

Language Equivalence. Let $\mathcal{C}$ be a class of n-dimensional rectangular hybrid automata. We that say two vectors $\mathbf{x}, \mathbf{u} \in \mathbb{R}^n$ are *language equivalent with respect to* $\mathcal{C}$, and write $\mathbf{x} \approx^{\mathcal{C}}_{\text{lang}} \mathbf{u}$, iff for every automaton $A \in \mathcal{C}$, and every vertex $v \in V_A$, $L_A(v, \mathbf{x}) = L_A(v, \mathbf{u})$. Define $\mathbf{x} \preceq^{\mathcal{C}}_{\text{lang}} \mathbf{u}$ iff for every automaton $A \in \mathcal{C}$, and every vertex $v \in V_A$, $L_A(v, \mathbf{x}) \subseteq L_A(v, \mathbf{u})$.

Simulation Equivalence. Let $\mathcal{C}$ be a class of n-dimensional rectangular hybrid automata, each having the same activity rectangle R. Let G be the set of guarded commands appearing in some automaton in $\mathcal{C}$. A *synchronous simulation* on $\mathbb{R}^n$ with respect to $\mathcal{C}$ is a binary relation $\preceq$ such that for every $\mathbf{x}, \mathbf{u} \in \mathbb{R}^n$, if $\mathbf{x} \preceq \mathbf{u}$ then (1) for every $\mathbf{x}' \in \mathbb{R}^n$ and every $t \in \mathbb{R}_{>0}$, if $\frac{\mathbf{x}' - \mathbf{x}}{t} \in R$ then either $\mathbf{x}' \preceq \mathbf{u}$ or there exist a $t' \in \mathbb{R}_{>0}$ and a $\mathbf{u}' \in \mathbb{R}^n$ such that $\mathbf{x}' \preceq \mathbf{u}'$ and $\frac{\mathbf{u}' - \mathbf{u}}{t'} \in R$, and (2) for every $\mathbf{x}' \in \mathbb{R}^n$ and every guarded command $g \in G$, if $(\mathbf{x}, \mathbf{x}') \in jump_g$ then there exists a $\mathbf{u}' \in \mathbb{R}^n$ such that $\mathbf{x}' \preceq \mathbf{u}'$ and $(\mathbf{u}, \mathbf{u}') \in jump_g$. The largest synchronous simulation on $\mathbb{R}^n$ with respect to $\mathcal{C}$

[4] It is for this reason that the generalized reset mechanism is introduced: it simplifies the translation from RHA into timed automata.

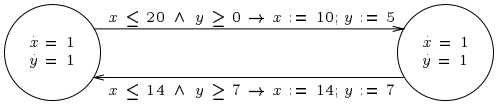

Figure 4: The one-sided timed automaton T_A

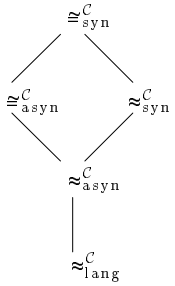

Figure 5: Lattice of equivalence relations

is denoted by $\preceq^{\mathcal{C}}_{\text{syn}}$. If $\mathbf{x} \preceq^{\mathcal{C}}_{\text{syn}} \mathbf{u}$, we say $\mathbf{u}$ *synchronously simulates* $\mathbf{x}$ with respect to $\mathcal{C}$. If $\mathbf{x} \preceq^{\mathcal{C}}_{\text{syn}} \mathbf{u}$, then every time action with source $\mathbf{x}$ can be matched by a time action with source $\mathbf{u}$, and every immediate application of a guarded command to $\mathbf{x}$ can be matched by the immediate application of the same guarded command to $\mathbf{u}$. It follows that if $\mathbf{x} \preceq^{\mathcal{C}}_{\text{syn}} \mathbf{u}$, then $\mathbf{x} \preceq^{\mathcal{C}}_{\text{lang}} \mathbf{u}$. We write $\mathbf{x} \approx^{\mathcal{C}}_{\text{syn}} \mathbf{u}$ if $\mathbf{x} \preceq^{\mathcal{C}}_{\text{syn}} \mathbf{u}$ and $\mathbf{u} \preceq^{\mathcal{C}}_{\text{syn}} \mathbf{x}$. This equivalence relation is known as *synchronous simulation equivalence* with respect to $\mathcal{C}$.

For asynchronous simulations, time is folded into the application of guarded commands. For a guarded command g, define the binary relation $\mathit{delayed}^R_g$ on $\mathbb{R}^n$ by $(\mathbf{x}, \mathbf{x}') \in \mathit{delayed}^R_g$ iff there exists an $\mathbf{x}'' \in \mathbb{R}^n$ such that (1) $(\mathbf{x}'', \mathbf{x}') \in \mathit{jump}_g$ and (2) either $\mathbf{x}'' = \mathbf{x}$ or there exists a time $t > 0$ such that $\frac{\mathbf{x}''-\mathbf{x}}{t} \in R$. An *asynchronous simulation* on $\mathbb{R}^n$ with respect to $\mathcal{C}$ is a binary relation $\preceq$ such that for every $\mathbf{x}, \mathbf{u} \in \mathbb{R}^n$, if $\mathbf{x} \preceq \mathbf{u}$ then for every $\mathbf{x}' \in \mathbb{R}^n$ and every guarded command $g \in G$ with $(\mathbf{x}, \mathbf{x}') \in \mathit{delayed}^R_g$, there exists a $\mathbf{u}' \in \mathbb{R}^n$ such that $\mathbf{x}' \preceq \mathbf{u}'$ and $(\mathbf{u}, \mathbf{u}') \in \mathit{delayed}^R_g$. The largest asynchronous simulation on $\mathbb{R}^n$ with respect to $\mathcal{C}$ is denoted by $\preceq^{\mathcal{C}}_{\text{asyn}}$. If $\mathbf{x} \preceq^{\mathcal{C}}_{\text{asyn}} \mathbf{u}$, we say $\mathbf{u}$ *asynchronously simulates* $\mathbf{x}$ with respect to $\mathcal{C}$. If $\mathbf{x} \preceq^{\mathcal{C}}_{\text{asyn}} \mathbf{u}$, then every guarded command executable by $\mathbf{x}$ after an unspecified waiting period can be matched by $\mathbf{u}$. It follows that if $\mathbf{x} \preceq^{\mathcal{C}}_{\text{asyn}} \mathbf{u}$, then $\mathbf{x} \preceq^{\mathcal{C}}_{\text{lang}} \mathbf{u}$. We write $\mathbf{x} \approx^{\mathcal{C}}_{\text{asyn}} \mathbf{u}$ if $\mathbf{x} \preceq^{\mathcal{C}}_{\text{asyn}} \mathbf{u}$ and $\mathbf{u} \preceq^{\mathcal{C}}_{\text{asyn}} \mathbf{x}$. This equivalence relation is known as *asynchronous simulation equivalence* with respect to $\mathcal{C}$. Notice that every synchronous simulation is an asynchronous simulation. Therefore if $\mathbf{x} \preceq^{\mathcal{C}}_{\text{syn}} \mathbf{u}$, then $\mathbf{x} \preceq^{\mathcal{C}}_{\text{asyn}} \mathbf{u}$. Hence asynchronous simulation equivalence is at least as coarse as synchronous simulation equivalence.

Bisimulation Equivalence. A *synchronous* (resp. *asynchronous*) *bisimulation* with respect to $\mathcal{C}$ is a symmetric synchronous (resp. asynchronous) simulation with respect to $\mathcal{C}$. The largest synchronous (resp. asynchronous) bisimulation on $\mathbb{R}^n$ with respect to $\mathcal{C}$ is denoted by $\cong^{\mathcal{C}}_{\text{syn}}$ (resp. $\cong^{\mathcal{C}}_{\text{asyn}}$), and called *synchronous bisimulation equivalence* (resp. *asynchronous bisimulation equivalence*). Bisimulation equivalence is at least as fine as simulation equivalence, and it is finer if simulation equivalence is not a simulation.

Figure 5 depicts the relative coarseness of each of the five equivalence relations with respect to $\mathcal{C}$. Language equivalence is the coarsest; synchronous bisimulation equivalence is the finest.

Equivalences with Respect to the Class of Timed Automata.

The ceiling or floor of a vector $\mathbf{x}$ is taken coordinatewise, so $\lceil \mathbf{x} \rceil = (\lceil x_1 \rceil, \lceil x_2 \rceil, \ldots, \lceil x_n \rceil)$. Recall that the *region equivalence relation* on $\mathbb{R}^n$ is defined by $\mathbf{x} \cong_{\text{reg}} \mathbf{x}'$ iff $\lceil \mathbf{x} \rceil = \lceil \mathbf{x}' \rceil$, $\lfloor \mathbf{x} \rfloor = \lfloor \mathbf{x}' \rfloor$, and for each $k, \ell \in \{1, \ldots, n\}$, $\lfloor x_k - x_\ell \rfloor = \lfloor x'_k - x'_\ell \rfloor$. That is, (1) each x_k and x'_k must have the same integer part, and both or neither must be an integer, and (2) the fractional parts of x_k and x_ℓ must be ordered in the same way as the fractional parts of x'_k and x'_ℓ.

Theorem 2 [AD94] *With respect to the class of timed automata, all five equivalence relations (language equivalence, and synchronous and asynchronous simulation and bisimulation equivalence) coincide with region equivalence.*

3 One-sided Timed Automata

We begin the analysis of one-sided timed automata by first considering automata whose clocks are all of the same type. For a vector $\mathbf{x} \in \mathbb{R}^n$ and a time $t \geq 0$, define $\mathbf{x} + t$ to be the vector $(x_1 + t, \ldots, x_n + t)$.

3.1 Upper-bounded Timed Automata

For the class of upper-bounded timed automata, language equivalence and both types of simulation equivalence coincide, while both types of bisimulation equivalence are much finer. The latter two coincide with region equivalence.

Proposition 1 *For all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$, the following four conditions are equivalent:*

- *clock vectors $\mathbf{x}$ and $\mathbf{x}'$ are language equivalent with respect to the class of upper-bounded timed automata,*
- *clock vectors $\mathbf{x}$ and $\mathbf{x}'$ are synchronously simulation equivalent with respect to the class of upper-bounded timed automata,*
- *clock vectors $\mathbf{x}$ and $\mathbf{x}'$ are asynchronously simulation equivalent with respect to the class of upper-bounded timed automata,*
- $\lceil \mathbf{x} \rceil = \lceil \mathbf{x}' \rceil$.

In two or more dimensions, the synchronous and asynchronous bisimulation equivalence relations with respect to the class of upper-bounded timed automata coincide with region equivalence.

Proof. Let $\mathcal{C}$ be the class of all upper-bounded timed automata. Call a guarded command *upper-bounded* if it appears in some upper-bounded timed automaton. Every guard B of an upper-bounded guarded command is of the form $\prod_{i=1}^{n} (-\infty, q_i]$, where each $q_i \in \mathbb{Z} \cup \{\infty\}$. It follows that $\mathbf{x} \in B$ iff $\mathbf{x} \leq \mathbf{q}$ coordinatewise. Thus for any upper-bounded guarded command $g = (B, U, F)$, the domain of $jump_g$ is $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \leq \mathbf{q}\}$. Put $\mathbf{x} \preceq \mathbf{x}'$ iff $\lceil \mathbf{x}' \rceil \leq \lceil \mathbf{x} \rceil$. Then $\mathbf{x} \preceq \mathbf{x}'$ iff for every upper-bounded guarded command g, $\mathbf{x} \in \text{dom } delayed_g$ implies $\mathbf{x}' \in \text{dom } delayed_g$. Thus $\mathbf{x} \preceq^{\mathcal{C}}_{\text{lang}} \mathbf{x}'$ implies $\mathbf{x} \preceq \mathbf{x}'$. On the other hand, $\preceq$ is a synchronous simulation. For suppose $\mathbf{x} \preceq \mathbf{u}$. If $(\mathbf{x}, \mathbf{x}') \in jump_g$, let $\mathbf{u}'$ be the unique vector such that $(\mathbf{u}, \mathbf{u}') \in jump_g$ and for each $i \in U$, $u'_i = x'_i$. Then $\lceil \mathbf{u}' \rceil \leq \lceil \mathbf{x}' \rceil$. Time actions of duration 0 preserve the simulation: for every $t \geq 0$,

$\lceil \mathbf{u} + 0 \rceil \leq \lceil \mathbf{x} + t \rceil$. This proves the equivalence of the four conditions in the statement of the proposition.

Since region equivalence is a synchronous bisimulation, it suffices to show that asynchronous bisimulation equivalence coincides with region equivalence. Suppose $\mathbf{x} \cong^{\mathcal{C}}_{\text{asyn}} \mathbf{u}$. Since $\mathbf{x} \approx^{\mathcal{C}}_{\text{asyn}} \mathbf{u}$, it follows from the above that $\lceil \mathbf{x} \rceil = \lceil \mathbf{u} \rceil$. Suppose that $\lfloor \mathbf{x} \rfloor \neq \lfloor \mathbf{u} \rfloor$. Without loss of generality, suppose $x_i < u_i$ and $u_i \in \mathbb{Z}$. Consider the upper-bounded guarded command g traditionally denoted by the expression $z_i \leq u_i \rightarrow z_j := 0$, where z_i refers to the ith clock variable, and $j \neq i$. There is a unique $\mathbf{u}'$ such that $(\mathbf{u}, \mathbf{u}') \in \mathit{delayed}_g$. Let $\mathbf{x}' = \mathbf{x}[x_j := 0]$. Then $(\mathbf{x}, \mathbf{x}') \in \mathit{delayed}_g$. We claim $\mathbf{x}' \ncong^{\mathcal{C}}_{\text{asyn}} \mathbf{u}'$. This is because for small enough $t > 0$, $\lceil x'_i + t \rceil = \lceil x_i + t \rceil = u_i$ and $\lceil x'_j + t \rceil = 1$; but for every $t' > 0$, $\lceil u'_i + t' \rceil > u_i$, and for $t' = 0$, $\lceil u'_j + t' \rceil = 0$. Thus if g' is the trivial guarded command $(\mathbb{R}^n, \emptyset, \mathbb{R}^n)$, then $(\mathbf{x}', \mathbf{x}' + t) \in \mathit{delayed}_{g'}$, but no $\mathit{delayed}_{g'}$-successor of $\mathbf{u}'$ is asynchronously bisimulation equivalent to $\mathbf{x}' + t$. We have proven by contradiction that $\lfloor \mathbf{x} \rfloor = \lfloor \mathbf{u} \rfloor$. To complete the proof, we show that the fractional parts of the coordinates of $\mathbf{x}$ and $\mathbf{y}$ must have the same relative order. Suppose $\lfloor x_j - x_i \rfloor > \lfloor u_j - u_i \rfloor$. Then because $\lceil \mathbf{x} \rceil = \lceil \mathbf{u} \rceil$ and $\lfloor \mathbf{x} \rfloor = \lfloor \mathbf{u} \rfloor$, the fractional part of x_j is greater than the fractional part of x_i, while the reverse holds for u_j and u_i. Thus there exists a time $t \geq 0$ (namely, $\lceil x_j \rceil - x_j + \epsilon$ for small enough ϵ) such that for every time $t' \geq 0$, $\lceil \mathbf{x} + t \rceil \neq \lceil \mathbf{u} + t' \rceil$. Thus again there is a $\mathit{delayed}_{g'}$-successor of $\mathbf{x}$ that is not asynchronously bisimulation equivalent to any $\mathit{delayed}_{g'}$-successor of $\mathbf{u}$. ■

3.2 Lower-bounded Timed Automata

For the class of lower-bounded timed automata, the asynchronous notions of equivalence collapse to the universal relation, while the synchronous equivalences are much finer.

Proposition 2 *1. Language equivalence, asynchronous simulation equivalence, and asynchronous bisimulation equivalence with respect to the class of lower-bounded timed automata coincide with the universal relation $\mathbb{R}^n \times \mathbb{R}^n$.*

2. Clock vectors $\mathbf{y}, \mathbf{y}' \in \mathbb{R}^n$ are synchronously simulation equivalent with respect to the class of lower-bounded timed automata iff $\lfloor \mathbf{y} \rfloor = \lfloor \mathbf{y}' \rfloor$.

3. In two or more dimensions, synchronous bisimulation equivalence with respect to the class of lower-bounded timed automata coincides with region equivalence.

Proof. Call a guarded command *lower-bounded* if it appears in a lower-bounded timed automaton. Every guard B of a lower-bounded guarded command is of the form $\prod_{j=1}^{n} [p_j, \infty)$, where each $p_j \in \mathbb{Z} \cup \{-\infty\}$. It follows that $\mathbf{y} \in B$ iff $\mathbf{y} \geq \mathbf{p}$ coordinatewise. Thus for any lower-bounded guarded command $g = (B, U, F)$, the domain of jump_g is $\{\mathbf{y} \in \mathbb{R}^n \mid \mathbf{y} \geq \mathbf{p}\}$. But since for any $\mathbf{y}$ there is a $t \geq 0$ such that $\mathbf{y} + t \geq \mathbf{p}$, the domain of $\mathit{delayed}_g$ is $\mathbb{R}^n$. This proves (1).

Let $\mathcal{C}$ be the class of all lower bounded timed automata. To prove (2), put $\mathbf{y} \preceq \mathbf{y}'$ iff $\lfloor \mathbf{y} \rfloor \leq \lfloor \mathbf{y}' \rfloor$. Then $\mathbf{y} \preceq \mathbf{y}'$ iff for every lower bound guarded command g, $\mathbf{y} \in \text{dom } \mathit{jump}_g$ implies $\mathbf{y}' \in \text{dom } \mathit{jump}_g$. Thus $\mathbf{y} \preceq^{\mathcal{C}}_{\text{syn}} \mathbf{y}'$ implies $\mathbf{y} \preceq \mathbf{y}'$. On the other hand, $\preceq$ is a synchronous simulation. For if $\mathbf{y} \preceq \mathbf{v}$, then (a) if $(\mathbf{y}, \mathbf{y}') \in \mathit{jump}_g$, then $\lfloor \mathbf{y}' \rfloor \leq \lfloor \mathbf{v}' \rfloor$, where $\mathbf{v}'$ is the unique vector such that $(\mathbf{v}, \mathbf{v}') \in \mathit{jump}_g$ and $v'_i = y'_i$ for $i \in U$, and (b) for every time $t \geq 0$, there exists a time $t' \geq 0$ such that $\lfloor \mathbf{y} + t \rfloor \leq \lfloor \mathbf{v} + t' \rfloor$. This proves (2). The proof of (3) is almost identical to the proof of the last assertion of Proposition 1. ■

3.3 One-sided Timed Automata

Let T be an $(n+m)$-dimensional one-sided timed automaton with n upper-bounded clocks and m lower-bounded clocks. After suitable rearrangement, we can assume that coordinates 1 through n are upper-bounded clocks and that coordinates $n+1$ through $n+m$ are lower-bounded clocks. Then for each guarded command $guard_T(e) = (B, U, F)$, it follows that the guard B is of the form

$$\prod_{i=1}^{n}(-\infty, q_i] \times \prod_{j=1}^{m}[p_j, \infty),$$

where each $q_i \in \mathbb{Z} \cup \{\infty\}$ and each $p_j \in \mathbb{Z} \cup \{-\infty\}$. So we represent guards B of one-sided timed automata by pairs of vectors $(\mathbf{q}, \mathbf{p}) \in (\mathbb{Z} \cup \{\infty\})^n \times (\mathbb{Z} \cup \{-\infty\})^m$. We call such guards and the guarded commands they comprise *one-sided*. We write a vector $\mathbf{u} \in \mathbb{R}^{n+m}$ as the pair $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} = (u_1, \ldots, u_n) \in \mathbb{R}^n$ and $\mathbf{y} = (u_{n+1}, \ldots, u_{n+m}) \in \mathbb{R}^m$. We use the notational convention that i ranges over $\{1, \ldots, n\}$ and j ranges over $\{1, \ldots, m\}$.

Definition 1 The binary relation $\preceq_1$ on $\mathbb{R}^{n+m}$ is defined by $(\mathbf{x}, \mathbf{y}) \preceq_1 (\mathbf{x}', \mathbf{y}')$ iff $\lceil \mathbf{x}' \rceil \leq \lceil \mathbf{x} \rceil$ and $\forall i. \forall j. \lfloor y'_j - x'_i \rfloor \geq \lfloor y_j - x_i \rfloor$. ■

Put $(\mathbf{x}, \mathbf{y}) \approx_1 (\mathbf{x}', \mathbf{y}')$ iff $(\mathbf{x}, \mathbf{y}) \preceq_1 (\mathbf{x}', \mathbf{y}')$ and $(\mathbf{x}', \mathbf{y}') \preceq_1 (\mathbf{x}, \mathbf{y})$. The difference between region equivalence and $\approx_1$-equivalence is that the latter only considers the relative order of fractional parts of upper-bounded clocks with respect to the fractional parts of lower-bounded clocks. The relative order of the fractional parts of two different lower-bounded clocks (or two different upper-bounded clocks) is irrelevant to $\approx_1$-equivalence.

Lemma 1 *For every one-sided guarded command* $g = ((\mathbf{q}, \mathbf{p}), U, F)$,

$$\text{dom } delayed_g = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n+m} \mid \mathbf{x} \leq \mathbf{q} \text{ and } (\forall i)(\forall j)[p_j - q_i \leq y_j - x_i]\}.$$

Proof.

$$\begin{aligned}
\text{dom } delayed_g &= \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n+m} \mid (\exists t \geq 0)[(\mathbf{x}, \mathbf{y}) + t \in \text{dom } jump_g]\} \\
&= \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n+m} \mid (\exists t \geq 0)[(\mathbf{x} + t) \leq \mathbf{q} \text{ and } (\mathbf{y} + t) \geq \mathbf{p}]\} \\
&= \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n+m} \mid \mathbf{x} \leq \mathbf{q} \text{ and } \max_j p_j - y_j \leq \min_i q_i - x_i\} \\
&= \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n+m} \mid \mathbf{x} \leq \mathbf{q} \text{ and } (\forall i)(\forall j)[p_j - y_j \leq q_i - x_i]\} \\
&= \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n+m} \mid \mathbf{x} \leq \mathbf{q} \text{ and } (\forall i)(\forall j)[p_j - q_i \leq y_j - x_i]\}. \blacksquare
\end{aligned}$$

Lemma 2 *For all* $(\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}') \in \mathbb{R}^{n+m}$, $(\mathbf{x}, \mathbf{y}) \preceq_1 (\mathbf{x}', \mathbf{y}')$ *iff for every one-sided guarded command* $g = (B, U, F)$, $(\mathbf{x}, \mathbf{y}) \in \text{dom } delayed_g$ *implies* $(\mathbf{x}', \mathbf{y}') \in \text{dom } delayed_g$.

Proof. Suppose $(\mathbf{x}, \mathbf{y}) \preceq_1 (\mathbf{x}', \mathbf{y}')$. Let $g = ((\mathbf{q}, \mathbf{p}), \rho)$. Since $(\mathbf{x}, \mathbf{y}) \preceq_1 (\mathbf{x}', \mathbf{y}')$, $\lceil \mathbf{x}' \rceil \leq \lceil \mathbf{x} \rceil \leq \mathbf{q}$, and for each i and j, $y'_j - x'_i \geq \lfloor y_j - x_i \rfloor \geq p_j - q_i$. The *only if* now follows from Lemma 1.

We now prove the *if*. Case 1: $\lceil \mathbf{x}' \rceil \not\leq \lceil \mathbf{x} \rceil$. Let B be the one-sided guard $(\lceil \mathbf{x} \rceil, (-\infty, \ldots, -\infty))$, and let $g = (B, \emptyset, B)$. Then $(\mathbf{x}, \mathbf{y}) \in \text{dom } delayed_g$ and $(\mathbf{x}', \mathbf{y}') \notin \text{dom } delayed_g$. Case 2: $\lfloor y'_j - x'_i \rfloor < \lfloor y_j - x_i \rfloor$. Let $q_i, p_j \in \mathbb{Z}$ be such that $p_j - q_i = \lfloor y_j - x_i \rfloor$ and $x_i \leq q_i$. Let B be the one-sided guard $((\infty, \ldots, \infty, q_i, \infty, \ldots, \infty)(-\infty, \ldots, -\infty, p_j, -\infty, \ldots, -\infty))$, and let $g = (B, \emptyset, B)$. By Lemma 1, $(\mathbf{x}, \mathbf{y}) \in \text{dom } delayed_g$ and $(\mathbf{x}', \mathbf{y}') \notin \text{dom } delayed_g$. ■

3.3.1 Synchronous Analysis.

It is not the case that $(\mathbf{x},\mathbf{y}) \preceq_1 (\mathbf{x}',\mathbf{y}')$ implies that for every one-sided guarded command g, $(\mathbf{x},\mathbf{y}) \in \text{dom } \mathit{jump}_g$ implies $(\mathbf{x}',\mathbf{y}') \in \text{dom } \mathit{jump}_g$. For this we need the additional condition that $\lfloor \mathbf{y}' \rfloor \geq \lfloor \mathbf{y} \rfloor$.

Definition 2 The binary relation $\preceq_2$ on $\mathbb{R}^{n+m}$ is defined by $(\mathbf{x},\mathbf{y}) \preceq_2 (\mathbf{x}',\mathbf{y}')$ iff $(\mathbf{x},\mathbf{y}) \preceq_1 (\mathbf{x}',\mathbf{y}')$ and $\lfloor \mathbf{y}' \rfloor \geq \lfloor \mathbf{y} \rfloor$. ■

In order to prove that $\preceq_2$ is a synchronous simulation, we need a few basic lemmas about differences of floors and ceilings.

Lemma 3 *For all $a, b \in \mathbb{R}$, $\lfloor a \rfloor - \lfloor b \rfloor \geq \lfloor a - b \rfloor$.* ■

Lemma 4 *For all $a, b \in \mathbb{R}$, $\lceil a \rceil - \lceil b \rceil \geq \lfloor a - b \rfloor$.* ■

Lemma 5 *For all $a, b \in \mathbb{R}$, $\lfloor a - b \rfloor \geq \lfloor a \rfloor - \lceil b \rceil$.* ■

Proposition 3 *The relation $\preceq_2$ is the largest synchronous simulation on $\mathbb{R}^{n+m}$ with respect to the class of one-sided timed automata.*

Proof. Let $\mathcal{C}$ be the class of one-sided timed automata. From the proofs of Propositions 1 and 2 we know that $(\mathbf{x},\mathbf{y}) \preceq^{\mathcal{C}}_{\text{syn}} (\mathbf{u},\mathbf{v})$ implies both $\lceil \mathbf{u} \rceil \leq \lceil \mathbf{x} \rceil$ and $\lfloor \mathbf{v} \rfloor \geq \lfloor \mathbf{y} \rfloor$. That $(\mathbf{x},\mathbf{y}) \preceq^{\mathcal{C}}_{\text{syn}} (\mathbf{u},\mathbf{v})$ implies $\forall i.\forall j.\, \lfloor v_j - u_i \rfloor \geq \lfloor y_j - u_i \rfloor$ follows from Lemma 2. For if $\lfloor v_j - u_i \rfloor < \lfloor y_j - u_i \rfloor$, then there exists a guarded command g such that $(\mathbf{x},\mathbf{y}) \in \text{dom } \mathit{delayed}_g$ and $(\mathbf{u},\mathbf{v}) \notin \text{dom } \mathit{delayed}_g$. Thus there exists a time $t \geq 0$ such that for all $t' \geq 0$, $(\mathbf{x}+t,\mathbf{y}+t) \not\preceq^{\mathcal{C}}_{\text{syn}} (\mathbf{u}+t',\mathbf{v}+t')$. We have proven that $\preceq^{\mathcal{C}}_{\text{syn}} \subseteq \preceq_2$, so if $\preceq_2$ is a synchronous simulation, then it is the largest.

We now show that $\preceq_2$ is a synchronous simulation. Suppose $(\mathbf{x},\mathbf{y}) \preceq_2 (\mathbf{u},\mathbf{v})$. Analysis of time transitions is simple. For $t \in \mathbb{R}_{\geq 0}$, let B be the one-sided guard $(\lceil \mathbf{x}+t \rceil, \lfloor \mathbf{y}+t \rfloor)$, and let $g = (B,\emptyset,B)$. Since $(\mathbf{x},\mathbf{y}) \in \text{dom } \mathit{delayed}_g$, it follows from Lemma 2 that $(\mathbf{u},\mathbf{v}) \in \text{dom } \mathit{delayed}_g$ as well. Thus there exists a $t' \in \mathbb{R}_{\geq 0}$ such that $\mathbf{u}+t' \leq \lceil \mathbf{x}+t \rceil$ and $\mathbf{v}+t' \geq \lfloor \mathbf{y}+t \rfloor$. For the final requirement, $\lfloor (v_j+t') - (u_i+t') \rfloor = \lfloor v_j - u_i \rfloor \geq \lfloor y_j - x_i \rfloor = \lfloor (y_j+t) - (x_i+t) \rfloor$. Therefore $(\mathbf{x}+t,\mathbf{y}+t) \preceq_2 (\mathbf{u}+t',\mathbf{v}+t')$.

Analysis of guarded command transitions is more difficult. Let $g = (B,U,F)$ be a one-sided guarded command, and suppose $((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')) \in \mathit{jump}_g$. Let $(\mathbf{u}',\mathbf{v}')$ be the unique vector such that (1) $((\mathbf{u},\mathbf{v}),(\mathbf{u}',\mathbf{v}')) \in \mathit{jump}_g$, (2) for each $i \in U$, $u'_i = \lfloor x'_i \rfloor$, and (3) for each $j \in U$, $v'_j = \lceil y'_j \rceil$. We will show that $(\mathbf{x}',\mathbf{y}') \preceq_2 (\mathbf{u}',\mathbf{v}')$.

First we show that $\lceil \mathbf{u}' \rceil \leq \lceil \mathbf{x}' \rceil$. For $i \in U$, $\lceil u'_i \rceil \leq \lceil x'_i \rceil$ because $u'_i = \lfloor x'_i \rfloor$. For $i \notin U$, $u'_i = \mathit{closest}(u_i, F_i)$ and $x'_i = \mathit{closest}(x_i, F_i)$. If $u_i \leq x_i$ it follows that $u'_i \leq x'_i$, and therefore that $\lceil u'_i \rceil \leq \lceil x'_i \rceil$. So suppose $u_i > x_i$. Since $\mathbf{u} \leq \lceil \mathbf{x} \rceil$, we have $\lceil x_i \rceil \geq u_i > x_i$. Case 1: $x_i \in F_i$. Since $x_i \notin \mathbb{Z}$ it follows that $\lceil x_i \rceil \in F_i$, and therefore that $u_i \in F_i$. Therefore $x'_i = x_i$ and $u'_i = u_i$, and so $\lceil u'_i \rceil \leq \lceil x'_i \rceil$. Case 2: $x_i \notin F_i$, $u_i \in F_i$. It follows that $u'_i = u_i = \lceil x_i \rceil = \min F_i = x'_i$, so $\lceil u'_i \rceil = \lceil x'_i \rceil$. Case 3: $x_i, u_i \notin F_i$. In this case either $\min F_i \geq u_i > x_i$, or $\max F_i < x_i < u_i$. In either case, $x'_i = u'_i$, and so $\lceil u'_i \rceil = \lceil x'_i \rceil$.

It follows by symmetry that $\lceil \mathbf{v}' \rceil \geq \lceil \mathbf{y}' \rceil$.

Finally we show $\forall i.\forall j.\, \lfloor v'_j - u'_i \rfloor \geq \lfloor y'_j - x'_i \rfloor$. Case 1: $i,j \in U$. Here $v'_j \geq y'_j$ and $u'_i \leq x'_i$, so $\lfloor v'_j - u'_i \rfloor \geq \lfloor y'_j - x'_i \rfloor$ follows immediately. Case 2: $i \in U$, $j \notin U$. Here $u'_i \in \mathbb{Z}$, and $u'_i = \lfloor x'_i \rfloor$. Therefore

$$\lfloor v'_j - u'_i \rfloor = \lfloor v'_j \rfloor - u'_i \geq \lfloor y'_j \rfloor - u'_i = \lfloor y'_j \rfloor - \lfloor x'_i \rfloor \geq \lfloor y'_j - x'_i \rfloor,$$

where the first inequality follows from $\lfloor \mathbf{v}' \rfloor \geq \lfloor \mathbf{y}' \rfloor$, and the last inequality follows from Lemma 3. Case 3: $i \notin U$, $j \in U$. Here $v'_j = \lceil y'_j \rceil$. Therefore

$$\lfloor v'_j - u'_i \rfloor = v'_j - \lceil u'_i \rceil = \lceil y'_j \rceil - \lceil u'_i \rceil \geq \lceil y'_j \rceil - \lceil x'_i \rceil \geq \lfloor y'_j - x'_i \rfloor,$$

where the second inequality follows from $\lceil \mathbf{u}' \rceil \leq \lceil \mathbf{x}' \rceil$, and the last inequality follows from Lemma 4. Case 4: $i, j \notin U$. If $x'_i \in \mathbb{Z}$, then

$$\lfloor y'_j - x'_i \rfloor = \lfloor y'_j \rfloor - x'_i \leq \lfloor y'_j \rfloor - \lceil u'_i \rceil \leq \lfloor v'_j \rfloor - \lceil u'_i \rceil \leq \lfloor v'_j - u'_i \rfloor,$$

where the first inequality follows from $\lceil \mathbf{u}' \rceil \leq \lceil \mathbf{x}' \rceil$, the second follows from $\lfloor \mathbf{v}' \rfloor \geq \lfloor \mathbf{y}' \rfloor$, and the third follows from Lemma 5. If $y'_j \in \mathbb{Z}$, then

$$\lfloor y'_j - x'_i \rfloor = y'_j - \lceil x'_i \rceil \leq y'_j - \lceil u'_i \rceil \leq \lfloor v'_j \rfloor - \lceil u'_i \rceil \leq \lfloor v'_j - u'_i \rfloor,$$

with the same attributions as above. So suppose $x'_i, y'_j \notin \mathbb{Z}$. Since $i, j \notin U$, it follows that $x'_i \in F_i$ and $y'_j \in F_j$, and so $x'_i = x_i$ and $y'_j = y_j$. Subcase A: $u_i \in F_i$ and $v_j \in F_j$. Here $u'_i = u_i$ and $v'_j = v_j$, so $\lfloor v'_j - u'_i \rfloor = \lfloor v_j - u_i \rfloor \geq \lfloor y_j - x_i \rfloor = \lfloor y'_j - x'_i \rfloor$. Subcase B: $u_i \in F_i$ and $v_j \notin F_j$. Since $[\lfloor y_j \rfloor, \lceil y_j \rceil] \subseteq F_j$ and $\lfloor v_j \rfloor \geq \lfloor y_j \rfloor$, it follows that $v_j > \lceil y_j \rceil = \lceil y'_j \rceil$, and therefore that $v'_j \geq \lceil y'_j \rceil$. Thus

$$\lfloor v'_j - u'_i \rfloor \geq \lfloor \lceil y'_j \rceil - u'_i \rfloor \geq \lfloor \lceil y'_j \rceil - \lceil x'_i \rceil \rfloor = \lceil y'_j \rceil - \lceil x'_i \rceil \geq \lfloor y'_j - x'_i \rfloor,$$

where the last inequality follows from Lemma 4. Subcase C: $u_i \notin F_i$ and $v_j \in F_j$. Since $[\lfloor x_i \rfloor, \lceil x_i \rceil] \subseteq F_i$ and $\lceil u_i \rceil \leq \lceil x_i \rceil$, it follows that $u_i < \lfloor x_i \rfloor = \lfloor x'_i \rfloor$, and therefore that $u'_i \leq \lfloor x'_i \rfloor$. Thus

$$\lfloor v'_j - u'_i \rfloor \geq \lfloor v'_j - \lfloor x'_i \rfloor \rfloor = \lfloor v'_j \rfloor - \lfloor x'_i \rfloor \geq \lfloor y'_j \rfloor - \lfloor x'_i \rfloor \geq \lfloor y'_j - x'_i \rfloor,$$

where the last inequality follows from Lemma 3. ■

3.3.2 Asynchronous Analysis.

It turns out that synchronous simulation equivalence is more discriminating than asynchronous simulation equivalence. The latter coincides with language equivalence.

Proposition 4 *The relation $\preceq_1$ is the largest asynchronous simulation on $\mathbb{R}^{n+m}$ with respect to the class of one-sided timed automata.*

Proof. We show that $\preceq_1$ is an asynchronous simulation. That it is the largest follows from Lemma 2. Suppose $(\mathbf{x}, \mathbf{y}) \preceq_1 (\mathbf{u}, \mathbf{v})$ and $((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) \in \mathit{delayed}_g$. We must show that there exists a $(\mathbf{u}', \mathbf{v}') \in \mathbb{R}^{n+m}$ such that $((\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}')) \in \mathit{delayed}_g$ and $(\mathbf{x}', \mathbf{y}') \preceq_1 (\mathbf{u}', \mathbf{v}')$.

There exists a $t \geq 0$ such that $((\mathbf{x}+t, \mathbf{y}+t), (\mathbf{x}', \mathbf{y}')) \in \mathit{jump}_g$. Let h be any one-sided guarded command with guard $(\lceil \mathbf{x} + t \rceil, \lfloor \mathbf{y} + t \rfloor)$. Then $(\mathbf{x}, \mathbf{y}) \in \mathrm{dom}\ \mathit{delayed}_h$. By Lemma 2, $(\mathbf{u}, \mathbf{v}) \in \mathrm{dom}\ \mathit{delayed}_h$ as well. Therefore there exists a time $t' \geq 0$ such that $(\mathbf{u} + t', \mathbf{v} + t') \in \mathrm{dom}\ \mathit{jump}_h$. Then $\lceil \mathbf{u} + t' \rceil \leq \lceil \mathbf{x} + t \rceil$, $\lfloor \mathbf{v} + t' \rfloor \geq \lfloor \mathbf{y} + t \rfloor$, and for all i and j, $\lfloor (v_j + t') - (u_i + t') \rfloor = \lfloor v_j - u_i \rfloor \geq \lfloor y_j - x_i \rfloor = \lfloor (y_j + t) - (x_i + t) \rfloor$. So $(\mathbf{x} + t, \mathbf{y} + t) \preceq_2 (\mathbf{u} + t', \mathbf{v} + t')$. It follows that there exists a $(\mathbf{u}', \mathbf{v}')$ such that $((\mathbf{u} + t', \mathbf{v} + t'), (\mathbf{u}', \mathbf{v}')) \in \mathit{jump}_g$ and $(\mathbf{x}', \mathbf{y}') \preceq_2 (\mathbf{u}', \mathbf{v}')$. Immediately from the definitions follow $((\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}')) \in \mathit{delayed}_g$ and $(\mathbf{x}', \mathbf{y}') \preceq_1 (\mathbf{u}', \mathbf{v}')$. ■

Theorem 3 *For all $(\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}') \in \mathbb{R}^{n+m}$, the following are equivalent:*

- *clock vectors $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$ are language equivalent with respect to the class of one-sided timed automata,*

- *clock vectors* $(\mathbf{x}, \mathbf{y})$ *and* $(\mathbf{x}', \mathbf{y}')$ *are asynchronously simulation equivalent with respect to the class of one-sided timed automata,*
- 1. $\lceil \mathbf{x} \rceil = \lceil \mathbf{x}' \rceil$, *and*
  2. $\forall i \in \{1, \ldots, n\}. \forall j \in \{1, \ldots, m\}. \lfloor y_j - x_i \rfloor = \lfloor y'_j - x'_i \rfloor$.

Clock vectors $(\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}') \in \mathbb{R}^{n+m}$ *are synchronously simulation equivalent with respect to the class of one-sided timed automata iff they are asynchronously simulation equivalent and* $\lceil \mathbf{y} \rceil = \lceil \mathbf{y}' \rceil$. *Synchronous and asynchronous bisimulation equivalence with respect to the class of one-sided timed automata coincide with region equivalence.*

Proof. The equivalence of the first three statements follows immediately from Proposition 4 and Lemma 2. The characterization of synchronous simulation equivalence is a restatement of Proposition 3. The characterizations of synchronous and asynchronous bisimulation equivalence follow from Proposition 1 and the fact that region equivalence is a synchronous bisimulation. ■

3.3.3 Size of the Language Equivalence Quotient.

The coarseness of an equivalence relation on $\mathbb{R}^n$ is measured by the number of equivalence classes per unit volume. Let $Regions(n)$ be the number of region equivalence classes on n dimensions where all clocks are constrained to lie in the interval $(0, 1)$, and let $OneSidedRegions(n)$ be the number of language equivalence classes for one-sided timed automata with $n/2$ upper-bounded clocks and $n/2$ lower-bounded clocks, where all clocks are constrained to lie in the interval $(0, 1)$, as defined by Theorem 1. These classes are called *regions* and *one-sided regions* respectively. While the region equivalence of $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$ requires that the relative ordering of the fractional parts of each pair of coordinates be identical for the two vectors, one-sided region equivalence only requires the same relative ordering of the fractional parts of pairs of coordinates for which one is a lower-bounded clock and one is an upper-bounded clock. It follows that one-sided region equivalence is considerably coarser than region equivalence.

Figure 6 illustrates the difference between regions and one-sided regions. Any one of the following four conditions bars the vectors $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$ from being region equivalent: (1) $x_1 > x_2$ but $x'_1 = x'_2$, (2) $y_1 > y_2$ but $y'_1 < y'_2$, (3) $x_3 > x_4$ but $x'_3 < x'_4$, and (4) $y_1 > x_4$ but $y'_1 = x'_4$. However, the relative order of the fractional parts of two upper-bounded clocks (or two lower-bounded clocks) is irrelevant to one-sided region equivalence. Therefore (1), (2), and (3) do not prevent the one-sided region equivalence of the two vectors. Nor does condition (4) prevent it, because $\lfloor y_1 - x_4 \rfloor = \lfloor y'_1 - x'_4 \rfloor$. The two vectors are in fact one-sided region equivalent.

Using characterizations of $Regions(n)$ and $OneSidedRegions(n)$ in terms of Stirling numbers of the second kind (see, e.g., [GKP89]), we show that, while the number of one-sided regions is still exponential, it is less than the number of regions by a multiplicative exponential factor.

Theorem 4 $\frac{Regions(2n)}{OneSidedRegions(2n)} = \Omega(2^n)$.

Proof. Let $\left\{ \begin{array}{c} n \\ k \end{array} \right\}$ be the number of ways to partition a set of n elements into k subsets. Each region in $(0, 1)^{2n}$ defines a partition of $\{1, \ldots, 2n\}$ into k subsets, where each subset defines a set of coordinates with the same value, and a permutation of the partition classes, giving the relative ordering of these values. Therefore

$$Regions(2n) = \sum_{k=1}^{2n} \left\{ \begin{array}{c} 2n \\ k \end{array} \right\} k!.$$

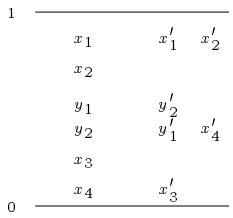

Figure 6: Vectors $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$ are one-sided region equivalent but not region equivalent

Each one-sided region with an upper-bounded clock having the highest value determines a partition of the upper-bounded clocks into k subsets, and a permutation thereof. The upper-bounded clock partition is interleaved with a partition of the lower-bounded clocks into k or $k-1$ subsets. The same analysis applies if a lower-bounded clock has the highest value. It follows that

$$\begin{aligned} \mathit{OneSidedRegions}(2n) &= 2\textstyle\sum_{k=1}^{n}(\left\{ \begin{array}{c} n \\ k \end{array} \right\})^2 k!^2 + 2\sum_{k=2}^{n} \left\{ \begin{array}{c} n \\ k \end{array} \right\} \left\{ \begin{array}{c} n \\ k-1 \end{array} \right\} k!(k-1)! \\ &\leq 4(\textstyle\sum_{k=1}^{n} \left\{ \begin{array}{c} n \\ k \end{array} \right\} k!)^2 \\ &= 4\mathit{Regions}(n)^2. \end{aligned}$$

Therefore $\mathit{OneSidedRegions}(2n) = O(\mathit{Regions}(n)^2)$. Since every pair of n-dimensional regions can be used to form a distinct $2n$-dimensional region by placing one "on top" of the other, $\frac{\mathit{Regions}(2n)}{\mathit{Regions}(n)^2} \geq \left(\begin{array}{c} 2n \\ n \end{array} \right)$. Now $\left(\begin{array}{c} 2n \\ n \end{array} \right) = \frac{(2n)!}{n!^2}$, $(2n)! = \Omega((\frac{2n}{e})^{2n})$, and $n! = O(n(\frac{n}{e})^n)$. So

$$\frac{\mathit{Regions}(2n)}{\mathit{OneSidedRegions}(2n)} = \Omega(\frac{(2n)!}{n!^2}) = \Omega(\frac{(\frac{2n}{e})^{2n}}{n^2(\frac{n}{e})^{2n}}) = \Omega(\frac{2^{2n}}{n^2}) = \Omega(2^n). \blacksquare$$

4 Rectangular Hybrid Automata

4.1 Language Equivalence

Combining Theorems 1 and 3, we obtain the following sufficient condition for the language equivalence of two vectors with respect to the class of rectangular hybrid automata with fixed activity rectangle R.

Corollary 1 *Let $R = \prod_{k=1}^{n}[a_k, b_k]$, and let $\lambda = lcm(a_1, b_1, \ldots, a_n, b_n)$. For all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$, if*

- $\forall i \in \{1, \ldots, n\}. \lceil x_i \frac{\lambda}{a_i} \rceil = \lceil x_i' \frac{\lambda}{a_i} \rceil$, *and*
- $\forall i, j \in \{1, \ldots, n\}. \lfloor x_j \frac{\lambda}{b_j} - x_i \frac{\lambda}{a_i} \rfloor = \lfloor x_j' \frac{\lambda}{b_j} - x_i' \frac{\lambda}{a_i} \rfloor$,

then $\mathbf{x}$ and $\mathbf{x}'$ are language equivalent with respect to the class of rectangular hybrid automata with activity rectangle R.

4.2 Simulation Equivalence

4.2.1 Synchronous analysis.

Theorem 5 *With respect to the set of 3D rectangular hybrid automata with activity rectangle* $\{1\} \times \{1\} \times [1,2]$*, the only synchronous simulation is equality.*

Proof. We refer to the three coordinates as x, y, and z. Let $\mathcal{C}$ be the set of rectangular hybrid automata with activity rectangle $\{1\} \times \{1\} \times [1,2]$. It suffices to show that for all vectors $(x,y,z),(x',y',z')$ in the unit cube $[0,1]^3$, $(x,y,z) \preceq^{\mathcal{C}}_{\text{syn}} (x',y',z')$ implies $(x,y,z) = (x',y',z')$. We do so in several steps. First we show that for every n, if $x \leq \frac{1}{2^n}$ and $x' > \frac{1}{2^n}$ then $(x,y,z) \not\preceq^{\mathcal{C}}_{\text{syn}} (x',y',z')$ and $(x',y',z') \not\preceq^{\mathcal{C}}_{\text{syn}} (x,y,z)$. We prove the analogous results for y and z simultaneously by induction. Second, we extend these results from numbers of the form $\frac{1}{2^n}$ to every dyadic rational $\frac{k}{2^n} \in [0,1]$. The theorem then follows from the density of the dyadic rationals.

We use a traditional notation to denote guarded commands. For example, $x = 1 \to z := 0$ denotes the guarded command $(\{1\} \times \mathbb{R} \times \mathbb{R}, \{3\}, \mathbb{R} \times \mathbb{R} \times \{0\})$. We define X to be the guarded command $true \to x := 0$, and similarly define Y and Z. For a guarded command g, Define $\xrightarrow{g}$ by $(x,y,z) \xrightarrow{g} (x',y',z')$ iff $((x,y,z),(x',y',z')) \in jump_g$. Define $\xrightarrow{T}$ by $\mathbf{u} \xrightarrow{T} \mathbf{u}'$ iff $\mathbf{u} = \mathbf{u}'$ or there is a time $t \in \mathbb{R}_{>0}$ with $\frac{\mathbf{u}'-\mathbf{u}}{t} \in \{1\} \times \{1\} \times [1,2]$. Given a predicate ϕ on $[0,1]^3$, we write $[\![\phi]\!]$ for $\{(x,y,z) \in [0,1]^3 \mid \phi(x,y,z)\}$. If ψ is another predicate on $[0,1]^3$, and $g \in \mathcal{G}_3 \cup \{T\}$, define $[\![\phi]\!] \xrightarrow{g} [\![\psi]\!]$ iff for every $(x,y,z) \in [\![\phi]\!]$, there exists a $(x',y',z') \in [\![\psi]\!]$ such that $(x,y,z) \xrightarrow{g} (x',y',z')$. Extend the definition of $\xrightarrow{g}$ to strings $\bar{g} \in (\mathcal{G}_3 \cup \{T\})^*$ in the natural way. Concatenation is denoted by a semicolon, so $g;g'$ is a string of length two. Define $Post_{\bar{g}}([\![\phi]\!])$ to be the set $\{(x',y',z') \in [0,1]^3 \mid \exists (x,y,z) \in [\![\phi]\!].\, (x,y,z) \xrightarrow{\bar{g}} (x',y',z')\}$ of $\bar{g}$-successors, inside the unit cube, of elements of $[\![\phi]\!]$. If no element of $[\![\phi]\!]$ is synchronously simulated by any element of $[\![\psi]\!]$, we write $[\![\phi]\!] \not\preceq_{\text{syn}} [\![\psi]\!]$. We say that three predicates ϕ, ψ, and θ on $[0,1]^3$ are *synchronously distinguishable* if $[\![\alpha]\!] \not\preceq_{\text{syn}} [\![\beta]\!]$ for all $\alpha \neq \beta \in \{\phi, \psi, \theta\}$.

We prove by induction that for all $n \geq 1$,

- $1(n)$: $x < \frac{1}{2^n}$, $x = \frac{1}{2^n}$, and $x > \frac{1}{2^n}$ are synchronously distinguishable,
- $2(n)$: $y < \frac{1}{2^n}$, $y = \frac{1}{2^n}$, and $y > \frac{1}{2^n}$ are synchronously distinguishable,
- $3(n)$: $z < \frac{1}{2^n}$, $z = \frac{1}{2^n}$, and $z > \frac{1}{2^n}$ are synchronously distinguishable.

In fact we prove that $1(n+1) \wedge 2(n+1) \wedge 3(n)$ implies $1(n+2) \wedge 2(n+2) \wedge 3(n+1)$, which makes the base case somewhat lengthy. Since x and y occupy symmetrical positions, we prove results for x only, and refer to the x result when a y citation is needed.

A). $[\![x \leq \frac{1}{2}]\!] \not\preceq_{\text{syn}} [\![x > \frac{1}{2}]\!]$: $[\![x \leq \frac{1}{2}]\!] \xrightarrow{Y;Z;T;x=1 \wedge z=1} [\![true]\!]$, but $Post_{Y;Z;T;x=1 \wedge z=1}([\![x > \frac{1}{2}]\!]) = \emptyset$.

B). $[\![x \geq \frac{1}{2}]\!] \not\preceq_{\text{syn}} [\![0 \leq x < \frac{1}{2}]\!]$: $[\![x \geq \frac{1}{2}]\!] \xrightarrow{Y;Z;T;x=1} [\![y \leq \frac{1}{2}]\!]$, but $Post_{Y;Z;T;x=1}([\![x < \frac{1}{2}]\!]) \subseteq [\![y > \frac{1}{2}]\!]$. Thus B) follows from A).

C). $1(1)$ and $2(1)$: immediate from A) and B).

D). $[\![z \leq \frac{1}{2}]\!] \not\preceq_{\text{syn}} [\![z > \frac{1}{2}]\!]$: $[\![z \leq \frac{1}{2}]\!] \xrightarrow{X;Y;T;z=1} [\![x = \frac{1}{2}]\!]$, but $Post_{X;Y;T;z=1}([\![z > \frac{1}{2}]\!]) \subseteq [\![x < \frac{1}{2}]\!]$. Thus D) follows from $1(1)$.

E). $[\![z \geq 2y]\!] \not\preceq_{\text{syn}} [\![z < 2y]\!]$: $[\![z \geq 2y]\!] \xrightarrow{X;T;z=1} [\![y \leq \frac{1}{2}]\!]$, but $Post_{X;T;z=1}([\![z < 2y]\!]) \subseteq [\![y > \frac{1}{2}]\!]$. Thus E) follows from $2(1)$.

F). $[\![\frac{1}{2} \geq x \geq \frac{1}{4}]\!] \not\preceq_{\text{syn}} [\![x < \frac{1}{4}]\!]$: $[\![\frac{1}{2} \geq x \geq \frac{1}{4}]\!] \xrightarrow{Y;Z;T} [\![x = \frac{1}{2} \wedge z \leq \frac{1}{2} \wedge z \geq 2y]\!]$, but

$$Post_{Y;Z;T}([\![x < \frac{1}{4}]\!]) \cap [\![x = \frac{1}{2} \wedge z \geq 2y]\!] \subseteq [\![z > \frac{1}{2}]\!].$$

Thus F) follows from 1(1), D), and E).

G). $[\![x \le \frac{1}{4}]\!] \not\preceq_{\text{syn}} [\![x > \frac{1}{4}]\!]$: $[\![x \le \frac{1}{4}]\!] \overset{Y;Z;T}{\rightarrow} [\![x = \frac{1}{2} \wedge y \ge \frac{1}{4}]\!]$, but

$$Post_{Y;Z;T}([\![x > \frac{1}{4}]\!]) \cap [\![x = \frac{1}{2}]\!] \subseteq [\![y < \frac{1}{4}]\!].$$

Thus G) follows from 1(1) and F).

H). 1(2) and 2(2): immediate from F), G), 1(1), and 2(1).

I). $[\![z \ge \frac{1}{2}]\!] \not\preceq_{\text{syn}} [\![z < \frac{1}{2}]\!]$: $[\![z \ge \frac{1}{2}]\!] \overset{X;Y;T;z=1}{\rightarrow} [\![x \le \frac{1}{4}]\!]$, but $Post_{XYT;z=1}([\![z < \frac{1}{2}]\!]) \subseteq [\![x > \frac{1}{4}]\!]$. Thus I) follows from G).

J) 3(1): immediate from D) and I).

The proof of the base case is complete. Recall that the induction step consists of proving that $1(n+1) \wedge 2(n+1) \wedge 3(n)$ implies $1(n+2) \wedge 2(n+2) \wedge 3(n+1)$. The proof of this step is a slight variant of the proof of the base case. Assume $1(n+1) \wedge 2(n+1) \wedge 3(n)$.

K). $[\![z \le \frac{1}{2^{n+1}}]\!] \not\preceq_{\text{syn}} [\![z > \frac{1}{2^{n+1}}]\!]$: $[\![z \le \frac{1}{2^{n+1}}]\!] \overset{X;Y;T}{\rightarrow} [\![x = \frac{1}{2^{n+1}} \wedge z \le \frac{1}{2^n}]\!]$, but

$$Post_{X;Y;T}([\![z > \frac{1}{2^{n+1}}]\!]) \cap [\![x = \frac{1}{2^{n+1}}]\!] \subseteq [\![z > \frac{1}{2^n}]\!].$$

Thus K) follows from $1(n+1)$ and $3(n)$.

L). $[\![\frac{1}{2^{n+1}} \ge x \ge \frac{1}{2^{n+2}}]\!] \not\preceq_{\text{syn}} [\![x < \frac{1}{2^{n+2}}]\!]$: $[\![\frac{1}{2^{n+1}} \ge x \ge \frac{1}{2^{n+2}}]\!] \overset{Y;Z;T}{\rightarrow} [\![x = \frac{1}{2^{n+1}} \wedge z \le \frac{1}{2^{n+1}} \wedge z \ge 2y]\!]$, but

$$Post_{Y;Z;T}([\![x < \frac{1}{2^{n+2}}]\!]) \cap [\![x = \frac{1}{2^{n+1}} \wedge z \ge 2y]\!] \subseteq [\![z > \frac{1}{2^{n+1}}]\!].$$

Thus L) follows from $1(n+1)$, E), and K).

M). $[\![x \le \frac{1}{2^{n+2}}]\!] \not\preceq_{\text{syn}} [\![x > \frac{1}{2^{n+2}}]\!]$: $[\![x \le \frac{1}{2^{n+2}}]\!] \overset{Y;Z;T}{\rightarrow} [\![x = \frac{1}{2^{n+1}} \wedge \frac{1}{2^{n+1}} \ge y \ge \frac{1}{2^{n+2}}]\!]$, but

$$Post_{Y;Z;T}([\![x > \frac{1}{2^{n+2}}]\!]) \cap [\![x = \frac{1}{2^{n+1}}]\!] \subseteq [\![y < \frac{1}{2^{n+2}}]\!].$$

Thus M) follows from $1(n+1)$ and L).

N). $1(n+2)$ and $2(n+2)$: immediate from L), M), $1(n+1)$, and $2(n+1)$.

O). $[\![\frac{1}{2^n} \ge z \ge \frac{1}{2^{n+1}}]\!] \not\preceq_{\text{syn}} [\![z < \frac{1}{2^{n+1}}]\!]$: $[\![\frac{1}{2^n} \ge z \ge \frac{1}{2^{n+1}}]\!] \overset{X;Y;T}{\rightarrow} [\![x \le \frac{1}{2^{n+2}} \wedge z = \frac{1}{2^n}]\!]$, but

$$Post_{X;Y;T}([\![z < \frac{1}{2^{n+1}}]\!]) \cap [\![z = \frac{1}{2^n}]\!] \subseteq [\![x > \frac{1}{2^{n+2}}]\!].$$

Thus O) follows from $3(n)$ and N).

P). $3(n+1)$: immediate from K), O), and $3(n)$.

The induction is complete; $1(n)$, $2(n)$, and $3(n)$ hold for all $n \ge 1$.

We prove that the following three conditions hold for every dyadic rational $\frac{k}{2^n} \in [0,1]$:

$I(k,n)$: $x < \frac{k}{2^n}$, $x = \frac{k}{2^n}$, and $x > \frac{k}{2^n}$ are synchronously distinguishable,

$II(k,n)$: $y < \frac{k}{2^n}$, $y = \frac{k}{2^n}$, and $y > \frac{k}{2^n}$ are synchronously distinguishable,

$III(k,n)$: $z < \frac{k}{2^n}$, $z = \frac{k}{2^n}$, and $z > \frac{k}{2^n}$ are synchronously distinguishable.

The argument for general dyadic rationals is now straightforward by induction on the denominator. The base case is supplied by 1(1), 2(1), and 3(1), the synchronous distinguishability of $x = 0$, $0 < x < 1$, and $x = 1$, and the synchronous distinguishability of $z = 0$, $0 < z < 1$, and $z = 1$. Let $I(n) = \bigwedge_{k=1}^{n-1} I(k,n)$, $II(n) = \bigwedge_{k=1}^{n-1} II(k,n)$, and $III(n) = \bigwedge_{k=1}^{n-1} III(k,n)$. Assume $I(n)$, $II(n)$,

and $III(n)$. If k is even, then $I(k,n+1)$, $II(k,n+1)$, and $III(k,n+1)$ follow immediately from the induction hypothesis, because in this case $\alpha(k,n+1)$ is equivalent to $\alpha(k/2,n)$ for $\alpha \in \{I, II, III\}$. So suppose k is odd.

Q). $[\![\frac{\frac{k+1}{2}}{2^n} \geq x \geq \frac{k}{2^{n+1}}]\!] \not\trianglelefteq_{\text{syn}} [\![x < \frac{k}{2^{n+1}}]\!]$: $[\![\frac{\frac{k+1}{2}}{2^n} \geq x \geq \frac{k}{2^{n+1}}]\!] \stackrel{Y;Z;T}{\rightarrow} [\![x = \frac{\frac{k+1}{2}}{2^n} \wedge y \leq \frac{1}{2^{n+1}}]\!]$, but

$$Post_{Y;Z;T}([\![x < \frac{k}{2^{n+1}}]\!]) \cap [\![x = \frac{\frac{k+1}{2}}{2^n}]\!] \subseteq [\![y > \frac{1}{2^{n+1}}]\!].$$

Thus Q) follows from $I(n)$ and $2(n+1)$.

R). $[\![x \leq \frac{k}{2^{n+1}}]\!] \not\trianglelefteq_{\text{syn}} [\![x > \frac{k}{2^{n+1}}]\!]$: $[\![x \leq \frac{k}{2^{n+1}}]\!] \stackrel{Y;Z;T}{\rightarrow} [\![x = \frac{\frac{k+1}{2}}{2^n} \wedge y \geq \frac{1}{2^{n+1}}]\!]$, but

$$Post_{Y;Z;T}([\![x > \frac{k}{2^{n+1}}]\!]) \cap [\![x = \frac{\frac{k+1}{2}}{2^n}]\!] \subseteq [\![y < \frac{1}{2^{n+1}}]\!].$$

Thus R) follows from $I(n)$ and $2(n+1)$.

S). $I(n+1)$: immediate from Q), R), and $I(n)$.

T). $[\![\frac{\frac{k+1}{2}}{2^n} \geq z \geq \frac{k}{2^{n+1}}]\!] \not\trianglelefteq_{\text{syn}} [\![z < \frac{k}{2^{n+1}}]\!]$: $[\![\frac{\frac{k+1}{2}}{2^n} \geq z \geq \frac{k}{2^{n+1}}]\!] \stackrel{X;Y;T}{\rightarrow} [\![z = \frac{\frac{k+1}{2}}{2^n} \wedge x \leq \frac{1}{2^{n+2}}]\!]$, but

$$Post_{X;Y;T}([\![z < \frac{k}{2^{n+1}}]\!]) \cap [\![z = \frac{\frac{k+1}{2}}{2^n}]\!] \subseteq [\![x > \frac{1}{2^{n+2}}]\!].$$

Thus T) follows from $III(n)$ and $1(n+2)$.

U). $[\![z \leq \frac{k}{2^{n+1}}]\!] \not\trianglelefteq_{\text{syn}} [\![z > \frac{k}{2^{n+1}}]\!]$: $[\![z \leq \frac{k}{2^{n+1}}]\!] \stackrel{X;Y;T}{\rightarrow} [\![z = \frac{\frac{k+1}{2}}{2^n} \wedge x \geq \frac{1}{2^{n+1}}]\!]$, but

$$Post_{X;Y;T}([\![z > \frac{k}{2^{n+1}}]\!]) \cap [\![z = \frac{\frac{k+1}{2}}{2^n}]\!] \subseteq [\![x < \frac{1}{2^{n+1}}]\!].$$

Thus U) follows from $III(n)$ and $1(n+1)$.

V). $III(n+1)$: immediate from T), U), and $III(n)$.

The induction is complete.

Finally suppose $(x,y,z) \neq (x',y',z')$, where $(x,y,z),(x',y',z') \in [0,1]^3$. If $x < x'$ (resp. $x > x'$), then there is a dyadic rational $\frac{k}{2^n}$ with $x < \frac{k}{2^n} < x'$ (resp. $x > \frac{k}{2^n} > x'$). By $I(k,n)$, $(x,y,z) \not\trianglelefteq_{\text{syn}} (x',y',z')$. Similar remarks apply to $y \neq y'$ or $z \neq z'$. ■

4.2.2 Asynchronous analysis.

Since every time predecessor of a point simulates that point, equality is not the only asynchronous simulation with respect to the class of RHA with a given activity rectangle. Nevertheless, asynchronous simulation equivalence coincides with equality. The proof of the next theorem is similar to the proof of Theorem 5, but it is slightly more complicated due to the lack of synchrony.

Theorem 6 *With respect to the class of 4D rectangular hybrid automata with activity rectangle* $\{1\}^3 \times [1,2]$*, asynchronous simulation equivalence coincides with equality.*

Proof. Put $R = \{1\}^3 \times [1,2]$. We refer to the four coordinates as w, x, y, and z. For a guarded command g, Define $\stackrel{g}{\rightarrow}$ by $(w,x,y,z) \stackrel{g}{\rightarrow} (w',x',y',z')$ iff $((w,x,y,z),(w',x',y',z')) \in delayed_X^R$. We use a traditional notation for guarded commands. For example, $x = 1 \rightarrow z := 0$ denotes the guarded command $(\mathbb{R} \times \{1\} \times \mathbb{R} \times \mathbb{R}, \{4\}, \mathbb{R}^3 \times \{0\})$. All of the action takes place on the unit cube $[0,1]^4$. Given a predicate ϕ on $[0,1]^4$, we write $[\![\phi]\!]$ for $\{(w,x,y,z) \in [0,1]^4 \mid \phi(w,x,y,z)\}$. If ψ is another

predicate on $[0,1]^4$, and g is a guarded command, define $[\![\phi]\!] \xrightarrow{g} [\![\psi]\!]$ iff for every $(w,x,y,z) \in [\![\phi]\!]$, there exists a $(w',x',y',z') \in [\![\psi]\!]$ such that $(w,x,y,z) \xrightarrow{g} (w,x',y',z')$. Extend the definition of $\xrightarrow{g}$ to strings $\bar{g} \in \mathcal{G}_4^*$ in the natural way. Concatenation is indicated by a semicolon, so $g;g'$ is a string of length two. Define $Post_{\bar{g}}([\![\phi]\!])$ to be the set

$$\{(w',x',y',z') \in [0,1]^4 \mid \exists (w,x,y,z) \in [\![\phi]\!].\, (w,x,y,z) \xrightarrow{\bar{g}} (w',x',y',z')\}$$

of $delayed^R_{\bar{g}}$-successors, inside the unit cube, of elements of $[\![\phi]\!]$. If no element of $[\![\phi]\!]$ is asynchronously simulated by any element of $[\![\psi]\!]$, we write $[\![\phi]\!] \not\preceq_{\text{asyn}} [\![\psi]\!]$. We say that three predicates ϕ, ψ, and θ on $[0,1]^4$ are *asynchronously distinguishable* if $[\![\alpha]\!] \not\preceq_{\text{asyn}} [\![\beta]\!]$ for all $\alpha \neq \beta \in \{\phi,\psi,\theta\}$.

We first prove by induction that for all $n \geq 1$,

- $1(n)$: For all $a \neq b \in \{w,x,y\}$, $a < \frac{1}{2^n} \wedge b = 0$, $a = \frac{1}{2^n} \wedge b = 0$, and $a > \frac{1}{2^n} \wedge b = 0$ are asynchronously distinguishable,
- $2(n)$: For all $b \in \{w,x,y\}$, $z < \frac{1}{2^n} \wedge b = 0$, $z = \frac{1}{2^n} \wedge b = 0$, and $z > \frac{1}{2^n} \wedge b = 0$ are asynchronously distinguishable.

In fact we prove that $1(n+1) \wedge 2(n)$ implies $1(n+2) \wedge 2(n+1)$. It then follows quickly that for every dyadic rational $\frac{k}{2^n} \in [0,1]$,

- $3(k,n)$: For all $a \neq b \in \{w,x,y\}$, $a < \frac{k}{2^n} \wedge b = 0$, $a = \frac{k}{2^n} \wedge b = 0$, and $a > \frac{k}{2^n} \wedge b = 0$ are asynchronously distinguishable,
- $4(k,n)$: For all $b \in \{w,x,y\}$, $z < \frac{k}{2^n} \wedge b = 0$, $z = \frac{k}{2^n} \wedge b = 0$, and $z > \frac{k}{2^n} \wedge b = 0$ are asynchronously distinguishable.

Finally, we prove that for every dyadic rational $\frac{k}{2^n} \in [0,1]$,

- $5(k,n)$: For all $a \in \{w,x,y\}$, $[\![a < \frac{k}{2^n}]\!] \not\preceq_{\text{asyn}} [\![a \geq \frac{k}{2^n}]\!]$,
- $6(k,n)$: $[\![z < \frac{2^k}{n}]\!] \not\preceq_{\text{asyn}} [\![z \geq \frac{k}{2^n}]\!]$.

It follows from the density of the dyadic rationals that asynchronous simulation equivalence coincides with equality.

Since w, x, and y occupy symmetrical positions, we only prove results for one (usually x), and refer to this proof when a symmetrical result is needed. We write W for the guarded command $true \to w := 0$, and similarly for x, y, and z.

A). $[\![x \leq \frac{1}{2} \wedge z = 0]\!] \not\preceq_{\text{asyn}} [\![x > \frac{1}{2}]\!]$: $[\![x \leq \frac{1}{2} \wedge z = 0]\!] \xrightarrow{W;Y;x=1 \wedge z=1} [\![true]\!]$ but

$$Post_{W;Y;x=1 \wedge z=1}([\![x > \frac{1}{2}]\!]) = \emptyset.$$

B). $[\![x \geq \frac{1}{2} \wedge y = 0]\!] \not\preceq_{\text{asyn}} [\![x < \frac{1}{2} \wedge y = 0]\!]$: $[\![x \geq \frac{1}{2} \wedge y = 0]\!] \xrightarrow{W;Z;x=1 \to z:=0} [\![y \leq \frac{1}{2} \wedge z = 0]\!]$, but $Post_{W;Z;x=1 \to z:=0}([\![x < \frac{1}{2} \wedge y = 0]\!]) \subseteq [\![y > \frac{1}{2}]\!]$. Thus B) follows from A).

C). $[\![x \leq \frac{1}{2} \wedge y = 0]\!] \not\preceq_{\text{asyn}} [\![x > \frac{1}{2} \wedge y = 0]\!]$: $[\![x \leq \frac{1}{2} \wedge y = 0]\!] \xrightarrow{W;Z;x=1 \to x:=0} [\![y \geq \frac{1}{2} \wedge x = 0]\!]$, but $Post_{W;Z;x=1 \to x:=0}([\![x > \frac{1}{2} \wedge y = 0]\!]) \subseteq [\![y < \frac{1}{2} \wedge x = 0]\!]$. Thus C) follows from B).

D). $1(1)$: immediate from B) and C).

E). $[\![1 \geq z \geq 2y \geq 0]\!] \not\preceq_{\text{asyn}} [\![0 \leq z < 2y \leq 1]\!]$: $[\![1 \geq z \geq 2y \geq 0]\!] \xrightarrow{W;X;z=1 \to x:=0} [\![y \leq \frac{1}{2} \wedge x = 0]\!]$, but $Post_{W;X;z=1 \to x:=0}([\![0 \leq z < 2y \leq 1]\!]) \subseteq [\![y > \frac{1}{2} \wedge x = 0]\!]$. Thus E) follows from C).

F). $[\![z \leq \frac{1}{2} \wedge x = 0]\!] \not\preceq_{\text{asyn}} [\![z > \frac{1}{2}]\!]$: $[\![z \leq \frac{1}{2} \wedge x = 0]\!] \overset{W;Y;z=1\rightarrow y:=0}{\rightarrow} [\![x \geq \frac{1}{2} \wedge y = 0]\!]$ but

$$Post_{W;Y;z=1\rightarrow y:=0}([\![z > \frac{1}{2}]\!]) \subseteq [\![x < \frac{1}{2} \wedge y = 0]\!].$$

Thus F) follows from B).

G). $[\![\frac{1}{2} \geq x \geq \frac{1}{4} \wedge y = 0]\!] \not\preceq_{\text{asyn}} [\![x < \frac{1}{4} \wedge y = 0]\!]$:

$$[\![\frac{1}{2} \geq x \geq \frac{1}{4} \wedge y = 0]\!] \overset{W;Z;W}{\rightarrow} [\![x = \frac{1}{2} \wedge w = 0 \wedge z \leq \frac{1}{2} \wedge 1 \geq z \geq 2y \geq 0]\!],$$

but $Post_{W;Z;W}([\![x < \frac{1}{4} \wedge y = 0]\!]) \cap [\![x = \frac{1}{2} \wedge w = 0 \wedge 1 \geq z \geq 2y \geq 0]\!] \subseteq [\![z > \frac{1}{2}]\!]$. Thus G) follows from 1(1), E), and F).

H). $[\![x \leq \frac{1}{4} \wedge y = 0]\!] \not\preceq_{\text{asyn}} [\![x > \frac{1}{4} \wedge y = 0]\!]$: $[\![x \leq \frac{1}{4} \wedge y = 0]\!] \overset{W;Z;W}{\rightarrow} [\![x = \frac{1}{2} \wedge w = 0 \wedge y \geq \frac{1}{4}]\!]$, but $Post_{W;Z;W}([\![x > \frac{1}{4} \wedge y = 0]\!]) \cap [\![x = \frac{1}{2} \wedge w = 0]\!] \subseteq [\![y < \frac{1}{4} \wedge w = 0]\!]$. Hence H) follows from 1(1) and G).

I). 1(2): immediate from G), H), and 1(1).

J). $[\![z \geq \frac{1}{2} \wedge x = 0]\!] \not\preceq_{\text{asyn}} [\![z < \frac{1}{2} \wedge x = 0]\!]$: $[\![z \geq \frac{1}{2} \wedge x = 0]\!] \overset{W;Y;z=1\rightarrow y:=0}{\rightarrow} [\![x \leq \frac{1}{4} \wedge y = 0]\!]$, but $Post_{W;Y;z=1\rightarrow y:=0}([\![z < \frac{1}{2} \wedge x = 0]\!]) \subseteq [\![x > \frac{1}{4} \wedge y = 0]\!]$. Thus J) follows from 1(2).

K). 2(1): immediate from F) and J).

This completes the base case. We now turn to the induction step. Assume $1(n+1)$ and $2(n)$.

L). $[\![z \leq \frac{1}{2^{n+1}} \wedge y = 0]\!] \not\preceq_{\text{asyn}} [\![z > \frac{1}{2^{n+1}} \wedge y = 0]\!]$:

$$[\![z \leq \frac{1}{2^{n+1}} \wedge y = 0]\!] \overset{W;X;Y}{\rightarrow} [\![z = \frac{1}{2^n} \wedge x \geq \frac{1}{2^{n+1}} \wedge y = 0]\!],$$

but $Post_{W;X;Y}([\![z > \frac{1}{2^{n+1}} \wedge y = 0]\!]) \cap [\![z = \frac{1}{2^n} \wedge y = 0]\!] \subseteq [\![x < \frac{1}{2^{n+1}} \wedge y = 0]\!]$. Thus L) follows from $1(n+1)$ and $2(n)$.

M). $[\![\frac{1}{2^{n+1}} \geq x \geq \frac{1}{2^{n+2}} \wedge y = 0]\!] \not\preceq_{\text{asyn}} [\![x < \frac{1}{2^{n+2}} \wedge y = 0]\!]$:

$$[\![\frac{1}{2^{n+1}} \geq x \geq \frac{1}{2^{n+2}} \wedge y = 0]\!] \overset{W;Z;W}{\rightarrow} [\![x = \frac{1}{2^{n+1}} \wedge 1 \geq z \geq 2y \geq 0 \wedge z \leq \frac{1}{2^{n+1}} \wedge w = 0]\!],$$

but

$$Post_{W;Z;W}([\![x < \frac{1}{2^{n+2}} \wedge y = 0]\!]) \cap [\![x = \frac{1}{2^{n+1}} \wedge w = 0 \wedge 1 \geq z \geq 2y \geq 0]\!] \subseteq [\![z > \frac{1}{2^{n+1}} \wedge w = 0]\!].$$

Thus M) follows from $1(n+1)$, E), and L).

N). $[\![x \leq \frac{1}{2^{n+2}} \wedge y = 0]\!] \not\preceq_{\text{asyn}} [\![x > \frac{1}{2^{n+2}} \wedge y = 0]\!]$:

$$[\![x \leq \frac{1}{2^{n+2}} \wedge y = 0]\!] \overset{W;Z;W}{\rightarrow} [\![x = \frac{1}{2^{n+1}} \wedge \frac{1}{2^{n+1}} \geq y \geq \frac{1}{2^{n+2}} \wedge w = 0]\!],$$

but $Post_{W;Z;W}([\![x > \frac{1}{2^{n+2}} \wedge y = 0]\!]) \cap [\![x = \frac{1}{2^{n+1}} \wedge w = 0]\!] \subseteq [\![y < \frac{1}{2^{n+2}} \wedge w = 0]\!]$. Thus N) follows from $1(n+1)$ and M).

O). $1(n+2)$: immediate from M), N), and $1(n+1)$.

P). $[\![\frac{1}{2^n} \geq z \geq \frac{1}{2^{n+1}} \wedge y = 0]\!] \not\preceq_{\text{asyn}} [\![z < \frac{1}{2^{n+1}} \wedge y = 0]\!]$:

$$[\![\frac{1}{2^n} \geq z \geq \frac{1}{2^{n+1}} \wedge y = 0]\!] \overset{W;X;W}{\rightarrow} [\![z = \frac{1}{2^n} \wedge y \leq \frac{1}{2^{n+2}} \wedge w = 0]\!],$$

but $Post_{W;X;W}([\![z < \frac{1}{2^{n+1}} \wedge y = 0]\!]) \cap [\![z = \frac{1}{2^n} \wedge w = 0]\!] \subseteq [\![y > \frac{1}{2^{n+2}} \wedge w = 0]\!]$. Thus P) follows from $2(n)$ and $1(n+2)$.

Q). $2(n+1)$: immediate from L), P), and $2(n)$.

The proof of $1(n)$ and $2(n)$ for all n is complete. The truth of $3(k,n)$ and $4(k,n)$ follows by the same argument used to prove $I(k,n)$, $II(k,n)$, and $III(k,n)$ in the proof of Theorem 5, except that the case of $\frac{k}{2^n} = 1$ must be handled differently.

R). $[\![a = 1 \wedge b = 0]\!] \not\preceq_{\text{asyn}} [\![a < 1 \wedge b = 0]\!]$: $[\![a = 1 \wedge b = 0]\!] \stackrel{a=1 \wedge b=0}{\rightarrow} [\![true]\!]$, but

$$Post_{a=1 \wedge b=0}([\![a < 1 \wedge b = 0]\!]) = \emptyset.$$

S). $[\![a < 1 \wedge b = 0]\!] \not\preceq_{\text{asyn}} [\![a = 1 \wedge b = 0]\!]$: Given $\mathbf{x} \in [\![a < 1 \wedge b = 0]\!]$, let n be large enough so that $a < 1 - \frac{1}{2^n}$. Then $\mathbf{x} \stackrel{c:=0;a=1 \rightarrow b=0}{\rightarrow} [\![c \geq \frac{1}{2^n} \wedge b = 0]\!]$ but

$$Post_{c:=0;a=1 \rightarrow b=0}(a = 1 \wedge b = 0) \cap [\![c \geq \frac{1}{2^n} \wedge b = 0]\!] = \emptyset.$$

Thus $3(2^n, n)$ and $4(2^n, n)$ follow from $1(n)$ and $2(n)$ respectively.

We omit the inductive step of the proof of $3(k,n)$ and $4(k,n)$, and turn to $5(k,n)$ and $6(k,n)$.

T). $[\![x < \frac{k}{2^n}]\!] \not\preceq_{\text{asyn}} [\![x \geq \frac{k}{2^n}]\!]$: $[\![x < \frac{k}{2^n}]\!] \stackrel{Y}{\rightarrow} [\![x < \frac{k}{2^n} \wedge y = 0]\!]$, but

$$Post_Y(x \geq \frac{k}{2^n}) \subseteq [\![x \geq \frac{k}{2^n} \wedge y = 0]\!].$$

Thus $5(k,n)$ follows from $3(k,n)$.

U). $[\![z < \frac{k}{2^n}]\!] \not\preceq_{\text{asyn}} [\![z \geq \frac{k}{2^n}]\!]$: $[\![z < \frac{k}{2^n}]\!] \stackrel{Y}{\rightarrow} [\![z < \frac{k}{2^n} \wedge y = 0]\!]$, but

$$Post_Y(z \geq \frac{k}{2^n}) \subseteq [\![z \geq \frac{k}{2^n} \wedge y = 0]\!].$$

Thus $6(k,n)$ follows from $4(k,n)$. ■

5 Summary

A summary of our results is given in Figure 7. All of the results are new, except those in the column regarding the class of timed automata [AD94]. In the four columns regarding subclasses of timed automata, exact characterizations of each of the five equivalence relations are given. Our main results relate to rectangular hybrid automata. First, language equivalence is finite, and a sufficient condition for language equivalence, superior by a multiplicative exponential factor to previously-known conditions, is derivable from the language equivalence relation that is displayed for one-sided timed automata. Second, the only synchronous simulation in three dimensions is equality, and asynchronous simulation equivalence in four dimensions is equality. Finally, our results suggest that simulation equivalence can be a more useful approximation to language equivalence than bisimulation equivalence. Indeed, for all of the subclasses of timed automata that we consider, synchronous bisimulation equivalence is finer than language equivalence. Moreover, except in the case of lower-bounded timed automata, the same can be said for asynchronous bisimulation equivalence.

References

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

	Upper-bounded Timed Automata	Lower-bounded Timed Automata	One-Sided Timed Automata	Timed Aut.	RHA
Language Equivalence	$\lceil \mathbf{x} \rceil = \lceil \mathbf{x}' \rceil$	universal relation	1. $\lceil \mathbf{x} \rceil = \lceil \mathbf{x}' \rceil$ 2. $\forall i. \forall j. \lfloor y_j - x_i \rfloor = \lfloor y'_j - x'_i \rfloor$	region equiv.	finite quotient (Cor. 1)
Asynchronous Simulation Equivalence	$\lceil \mathbf{x} \rceil = \lceil \mathbf{x}' \rceil$	universal relation	1. $\lceil \mathbf{x} \rceil = \lceil \mathbf{x}' \rceil$ 2. $\forall i. \forall j. \lfloor y_j - x_i \rfloor = \lfloor y'_j - x'_i \rfloor$	region equiv.	equality
Synchronous Simulation Equivalence	$\lceil \mathbf{x} \rceil = \lceil \mathbf{x}' \rceil$	$\lfloor \mathbf{y} \rfloor = \lfloor \mathbf{y}' \rfloor$	1. $\lceil \mathbf{x} \rceil = \lceil \mathbf{x}' \rceil$ 2. $\forall i. \forall j. \lfloor y_j - x_i \rfloor = \lfloor y'_j - x'_i \rfloor$ 3. $\lfloor \mathbf{y} \rfloor = \lfloor \mathbf{y}' \rfloor$	region equiv.	equality
Asynchronous Bisimulation Equivalence	region equivalence	universal relation	region equivalence	region equiv.	equality
Synchronous Bisimulation Equivalence	region equivalence	region equivalence	region equivalence	region equiv.	equality

Figure 7: Summary of results

[ACHH93] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems I*, Lecture Notes in Computer Science 736, pages 209–229. Springer-Verlag, 1993.

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

[AHH96] R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.

[BBLS92] S. Bensalem, A. Bouajjani, C. Loiseaux, and J. Sifakis. Property-preserving simulations. In G. von Bochmann and D.K. Probst, editors, *CAV 92: Computer-aided Verification*, Lecture Notes in Computer Science 663, pages 260–273. Springer-Verlag, 1992.

[BCG88] M.C. Browne, E.M. Clarke, and O. Grümberg. Characterizing finite kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59:115–131, 1988.

[DY95] C. Daws and S. Yovine. Two examples of verification of multirate timed automata with Kronos. In *Proceedings of the 16th Annual Real-time Systems Symposium*, pages 66–75. IEEE Computer Society Press, 1995.

[GKP89] R. Graham, D. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley Publishing Company, 1989.

[Hen95] T.A. Henzinger. Hybrid automata with finite bisimulations. In Z. Fülöp and F. Gécseg, editors, *ICALP 95: Automata, Languages, and Programming*, Lecture Notes in Computer Science 944, pages 324–335. Springer-Verlag, 1995.

[Hen96] T.A. Henzinger. The theory of hybrid automata. In *Proceedings of the Eleventh Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1996.

[HH95] T.A. Henzinger and P.-H. Ho. Algorithmic analysis of nonlinear hybrid systems. In P. Wolper, editor, *CAV 95: Computer-aided Verification*, Lecture Notes in Computer Science 939, pages 225–238. Springer-Verlag, 1995.

[HHK95] M.R. Henzinger, T.A. Henzinger, and P.W. Kopke. Computing simulations on finite and infinite graphs. In *Proceedings of the 36rd Annual Symposium on Foundations of Computer Science*, pages 453–462. IEEE Computer Society Press, 1995.

[HHWT95] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: the next generation. In *Proceedings of the 16th Annual Real-time Systems Symposium*, pages 56–65. IEEE Computer Society Press, 1995.

[HKPV95] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proceedings of the 27th Annual Symposium on Theory of Computing*, pages 373–382. ACM Press, 1995.

[HW95] P.-H. Ho and H. Wong-Toi. Automated analysis of an audio control protocol. In P. Wolper, editor, *CAV 95: Computer-aided Verification*, Lecture Notes in Computer Science 939, pages 381–394. Springer-Verlag, 1995.

[LPY95] K.G. Larsen, P. Pettersson, and W. Yi. Compositional and symbolic model checking of real-time systems. In *Proceedings of the 16th Annual Real-time Systems Symposium*, pages 76–87. IEEE Computer Society Press, 1995.

[NOSY93] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems I*, Lecture Notes in Computer Science 736, pages 149–178. Springer-Verlag, 1993.

[PV94] A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular differential inclusions. In D.L. Dill, editor, *CAV 94: Computer-aided Verification*, Lecture Notes in Computer Science 818, pages 95–104. Springer-Verlag, 1994.

[TY96] S. Tripakis and S. Yovine. Analysis of timed systems based on time-abstracting bisimulations. In *CAV 96: Computer-aided Verification*, Lecture Notes in Computer Science. Springer-Verlag, 1996.