Higher-Dimensional Timed and Hybrid Automata

Uli Fahrenberg
EPITA Research and Development Laboratory (LRDE), France

Abstract

We introduce a new formalism of higher-dimensional timed automata, based on Pratt and van Glabbeek's higher-dimensional automata and Alur and Dill's timed automata. We prove that their reachability is PSPACE-complete and can be decided using zone-based algorithms. We also extend the setting to higher-dimensional hybrid automata.

The interest of our formalism is in modeling systems which exhibit both real-time behavior and concurrency. Other existing formalisms for real-time modeling identify concurrency and interleaving, which, as we shall argue, is problematic.

2012 ACM Subject Classification Theory of computation → Concurrency; Theory of computation → Timed and hybrid models

Keywords and Phrases concurrency, real time, higher-dimensional automaton, timed automaton

Digital Object Identifier 10.4230/LITES.8.2.3

Acknowledgements The author acknowledges the support of the *Chaire ISC : Engineering Complex Systems* and École polytechnique where most of this work was carried out. He is most grateful to Kim G. Larsen and Eric Goubault for numerous interesting discussions on the subject of this paper.

Received 2020-08-04 **Accepted** 2022-01-28 **Published** 2022-12-07

Editor Alessandro Abate, Uli Fahrenberg, and Martin Fränzle

Special Issue Special Issue on Distributed Hybrid Systems

1 Introduction

In approaches to non-interleaving concurrency, more than one event may happen at the same time. There is a multitude of formalisms for modeling and analyzing such concurrent systems, e.g., Petri nets [47], event structures [46], configuration structures [58, 57], asynchronous transition systems [8, 50], or more recent variations such as dynamic event structures [6] and Unravel nets [16]. They all share the convention of differentiating between concurrent and interleaving executions; using CCS notation [44], $a|b \neq a.b + b.a$.

For modeling and analyzing embedded or cyber-physical systems, formalisms which use real time are available. These include timed automata [5], time Petri nets [43], timed-arc Petri nets [38], or various classes of hybrid automata [3]. Common for them all is that they identify concurrent and interleaving executions; here, $a|b = a.b + b.a$.

We are interested in formalisms for real-time non-interleaving concurrency. Hence we would like to differentiate between concurrent and interleaving executions *and* be able to model and analyze real-time properties. Few such formalisms seem to be available in the literature. The situation is perhaps best epitomized by the fact that there is a natural non-interleaving semantics for Petri nets [34] which is also used in practice [22, 23], but almost all work on real-time extensions of Petri nets [43, 38, 51, 53], including the popular tool TAPAAL,[1] use an interleaving semantics. (A notable exception here are the time Petri nets of [43] which do have a non-interleaving real-time semantics [18, 17, 7, 32] which has also been used for networks of timed automata [15].)

Also Uppaal,[2] the successful tool for modeling and analyzing networks of timed automata, uses an interleaving semantics for such networks. This leads to great trouble with state-space explosion (see also Sect. 7 of this paper) which can be avoided with a non-interleaving semantics such as we

[1] `http://www.tapaal.net/`
[2] `http://www.uppaal.org/`

propose here. Intuitively, interleaving composition of networks adds $n!$ different interleavings for every concurrent composition of n independent events, whereas in the non-interleaving semantics, only one (n-dimensional) object is added to the system.

We introduce higher-dimensional timed automata (HDTA), a formalism based on the (non-interleaving) higher-dimensional automata of Pratt and van Glabbeek [48, 54] (see also [56]) and the timed automata of Alur and Dill [5, 4]. We show that HDTA can model interesting phenomena which cannot be captured by neither of the formalisms on which they are based, but that their analysis remains just as accessible as the one of timed automata. That is, reachability for HDTA is PSPACE-complete and can be decided using zone-based algorithms.

In the above-mentioned interleaving real-time formalisms, continuous flows and discrete actions are orthogonal in the sense that executions alternate between real-time delays and discrete actions which are immediate, i.e., take no time. (In the hybrid setting, these are usually called flows and mode changes, respectively.) Already Sifakis and Yovine [52] notice that this significantly simplifies the semantics of such systems and hints that this is a main reason for the success of these formalisms (see the more recent [53] for a similar statement).

In the (untimed) non-interleaving setting, on the other hand, events have a (logical; unspecified) duration. This can be seen, for example, in the ST-traces of [55] where actions have a start (a^+) and a termination (a^-) and are (implicitly) running between their start and termination, or in the representation of concurrent systems as Chu spaces over $\mathbf{3} = \{0, \frac{1}{2}, 1\}$, where 0 is interpreted as “before”, $\frac{1}{2}$ as “during”, and 1 as “after”, see [49]. Intuitively, only if events have duration can one make statements such as “while a is running, b starts, and then while b is running, a terminates”.

In our non-interleaving real-time setting, we hence abandon the assumption that actions are immediate. Instead, we take the view that actions start and then run during some *specific* time before terminating. While this runs counter to the standard assumption in most of real-time and hybrid modeling, a similar view can be found, for example, in Cardelli’s [14].

Given that we abandon the orthogonality between continuous flows and discrete actions, we find it remarkable to see that the standard techniques used for timed automata transfer to our non-interleaving setting. Equally remarkable is, perhaps, the fact that even though *“[t]he timed-automata model is at the very border of decidability, in the sense that even small additions to the formalism [...] will soon lead to the undecidability of reachability questions”* [1], our extension to higher dimensions and non-interleaving concurrency is completely free of such trouble.

We also show that our HDTA model naturally extends to a formalism of higher-dimensional *hybrid* automata (HDHA), which can be used to model cyber-physical systems which exhibit concurrency. We introduce tensor products both for HDTA and HDHA which can be used for a concurrent composition of systems which avoids state-space explosion.

The contributions of this paper are, thus, (1) the introduction of a new formalism of HDTA, a natural extension of higher-dimensional automata and timed automata, in Sect. 3; (2) the proof that reachability for HDTA is PSPACE-complete and decidable using zone-based algorithms, in Sects. 5 and 6; (3) the introduction of a tensor product for HDTA which can be used for parallel composition, in Sect. 7; and (4) the extension of the definition to higher-dimensional hybrid automata together with a non-trivial example of two independently bouncing balls, in Sect. 8.

This paper is based on the conference contribution [27], which has been presented at the 6th IFAC Conference on Analysis and Design of Hybrid Systems in Oxford, UK. Compared to this previous paper, we have included proofs of all statements, improved the presentation and examples, and added a precise definition of tensor product of higher-dimensional hybrid automata.

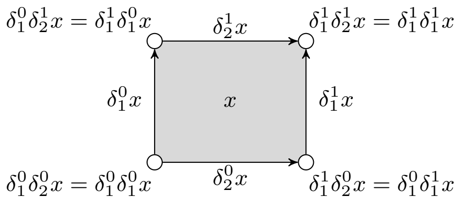

$\delta_1^0\delta_2^1 x = \delta_1^1\delta_1^0 x$
$\delta_2^1 x$
$\delta_1^1\delta_2^1 x = \delta_1^1\delta_1^1 x$
$\delta_1^0 x$
x
$\delta_1^1 x$
$\delta_1^0\delta_2^0 x = \delta_1^0\delta_1^0 x$
$\delta_2^0 x$
$\delta_1^1\delta_2^0 x = \delta_1^0\delta_1^1 x$

Figure 1 A 2-cube x with its four faces $\delta_1^0 x$, $\delta_1^1 x$, $\delta_2^0 x$, $\delta_2^1 x$ and four corners.

2 Preliminaries

We recall a few facts about higher-dimensional automata and timed automata.

2.1 Higher-Dimensional Automata

Higher-dimensional automata are a generalization of finite automata which permit the specification of independence of actions through higher-dimensional elements. That is, they consist of states and transitions, but also squares which signify that two events are independent, cubes which denote independence of three events, etc. To introduce them properly, we need to start with precubical sets.

A *precubical set* is a graded set $X = \bigcup_{n\in\mathbb{N}} X_n$, with $X_n \cap X_m = \emptyset$ for $n \neq m$, together with mappings $\delta_{k,n}^\nu : X_n \to X_{n-1}$, $k = 1, \dots, n$, $\nu = 0, 1$, satisfying the *precubical identity*

$$\delta_{k,n-1}^\nu \delta_{\ell,n}^\mu = \delta_{\ell-1,n-1}^\mu \delta_{k,n}^\nu \qquad (k < \ell)\,.$$

Elements of X_n are called *n-cubes*, and for $x \in X_n$, $n = \dim x$ is its *dimension*. The mappings $\delta_{k,n}^\nu$ are called *face maps*, and we will usually omit the extra subscript n and write δ_k^ν instead of $\delta_{k,n}^\nu$. Intuitively, each n-cube $x \in X_n$ has n *lower faces* $\delta_1^0 x, \dots, \delta_n^0 x$ and n *upper faces* $\delta_1^1 x, \dots, \delta_n^1 x$, and the precubical identity expresses the fact that non-parallel $(n-1)$-faces of an n-cube meet in common $(n-2)$-faces; see Figure 1 for an example.

A precubical set X is *finite* if X is finite as a set. This means that X_n is finite for each $n \in \mathbb{N}$ and that X is *finite-dimensional*: there exists $N \in \mathbb{N}$ such that $X_n = \emptyset$ for all $n \geq N$.

Let Σ be a finite set of *actions* and recall that a *multiset* over Σ is a mapping $\Sigma \to \mathbb{N}$; we denote the set of such by $\mathbb{N}^\Sigma$. The *cardinality* of $S \in \mathbb{N}^\Sigma$ is $|S| = \sum_{a\in\Sigma} S(a)$.

▶ **Definition 1.** A *higher-dimensional automaton* (HDA) is a structure (X, x^0, X^f, λ), where X is a finite precubical set with initial state $x^0 \in X_0$ and accepting states $X^f \subseteq X_0$, and $\lambda : X \to \mathbb{N}^\Sigma$ is a labeling function such that for every $x \in X$,
- $|\lambda(x)| = \dim x$,
- $\lambda(\delta_k^0 x) = \lambda(\delta_k^1 x)$ for all $k \leq \dim x$, and
- $\lambda(x) \setminus \lambda(\delta_k^0 x)$ is a singleton for all $k \leq \dim x$.

The conditions on the labeling ensure that the label of an n-cube is an extension, by one event, of the label of any of its faces. The computational intuition is that when passing from a lower face $\delta_k^0 x$ of $x \in X$ to x itself, the (unique) event in $\lambda(x) \setminus \lambda(\delta_k^0 x)$ is started, and when passing from x to an upper face $\delta_\ell^1 x$, the event in $\lambda(x) \setminus \lambda(\delta_\ell^1 x)$ is terminated.

HDA can indeed model higher-order concurrency of actions. As an example, the hollow cube on the left of Figure 2, consisting of all six faces of a cube but not of its interior, models the situation where the actions a, b and c are mutually independent, but cannot be executed all three concurrently. The full cube on the right of Figure 2, on the other hand, has a, b and c independent as a set. The left HDA might model a system of three users connected to two printers, so that every two of the users can print concurrently but not all three, whereas the right HDA models a system of three users connected to (at least) three printers.

▶ Remark. Instead of using multisets as we do here, labeling of precubical sets is commonly introduced by defining a precubical set $!\Sigma$ induced by Σ and then letting the labeling be a precubical morphism, see for example [24, 36]. This has the advantage that HDA can be posed as a slice category, but we will not need this here.

There is a rich literature on the geometric and topological analysis of HDA, starting with their *geometric realization* as directed topological spaces. The interested reader is referred to [37, 30, 31, 28, 24, 25].

2.2 Timed Automata

Timed automata extend finite automata with clock variables and invariants which permit the modeling of real-time properties. Let C be a finite set of *clocks*. $\Phi(C)$ denotes the set of *clock constraints* defined as

$$\Phi(C) \ni \phi_1, \phi_2 ::= c \bowtie k \mid \phi_1 \wedge \phi_2 \qquad (c \in C, k \in \mathbb{N}, \bowtie \in \{<, \leq, \geq, >\}) .$$

Hence a clock constraint is a conjunction of comparisons of clocks to integers.

A *clock valuation* is a mapping $v : C \to \mathbb{R}_{\geq 0}$, where $\mathbb{R}_{\geq 0}$ denotes the set of non-negative real numbers. The *initial* clock valuation is $v^0 : C \to \mathbb{R}_{\geq 0}$ given by $v^0(c) = 0$ for all $c \in C$. For $v \in \mathbb{R}_{\geq 0}^C$, $d \in \mathbb{R}_{\geq 0}$, and $C' \subseteq C$, the clock valuations $v + d$ and $v[C' \leftarrow 0]$ are defined by

$$(v + d)(c) = v(c) + d\,; \quad v[C' \leftarrow 0](c) = \begin{cases} 0 & \text{if } c \in C' , \\ v(c) & \text{if } c \notin C' . \end{cases}$$

For $v \in \mathbb{R}_{\geq 0}^C$ and $\phi \in \Phi(C)$, we write $v \models \phi$ if v satisfies ϕ and $[\![\phi]\!] = \{v : C \to \mathbb{R}_{\geq 0} \mid v \models \phi\}$.

▶ **Definition 2.** A *timed automaton* is a structure (Q, q^0, Q^f, I, E), where Q is a finite set of locations with initial location $q^0 \in Q$ and accepting locations $Q^f \subseteq Q$, $I : Q \to \Phi(C)$ assigns invariants to states, and $E \subseteq Q \times \Phi(C) \times \Sigma \times 2^C \times Q$ is a set of guarded transitions.

The *semantics* of a timed automaton $A = (Q, q^0, Q^f, I, E)$ is a (usually uncountably infinite) transition system $[\![A]\!] = (S, s^0, S^f, \rightsquigarrow)$, with $\rightsquigarrow \subseteq S \times S$, given as follows:

$$\begin{aligned} S &= \{(q, v) \in Q \times \mathbb{R}_{\geq 0}^C \mid v \models I(q)\} \\ s^0 &= (q^0, v^0) \quad S^f = S \cap Q^f \times \mathbb{R}_{\geq 0}^C \\ \rightsquigarrow &= \{((q, v), (q, v + d)) \mid \forall 0 \leq d' \leq d : v + d' \models I(q)\} \\ &\quad \cup \{((q, v), (q', v')) \mid \exists (q, \phi, a, C', q') \in E : v \models \phi, v' = v[C' \leftarrow 0]\} \end{aligned}$$

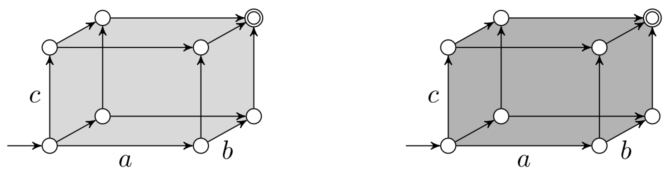


Figure 2 Two example HDA. Left, the hollow cube; right, the full cube.

Note that we are ignoring the labels of transitions here, as we will be concerned with reachability for now. As usual, we say that A is *reachable* iff there exists a finite path $s^0 \rightsquigarrow \cdots \rightsquigarrow s$ in $[\![A]\!]$ for which $s \in S^f$.

The definition of $\rightsquigarrow$ ensures that actions are immediate: whenever $(q, \phi, a, C', q') \in E$, then A passes from (q, v) to (q', v') without any delay. Time progresses only during delays $(q, v) \rightsquigarrow (q, v + d)$ in locations.

▶ Remark. Timed automata have a long and successful history in the modeling and verification of real-time computing systems. Several tools exist which are routinely applied in industry, such as Uppaal [42, 9], RED [59] or Kronos [13]. The interested reader is referred to [12, 41, 1].

3 Higher-Dimensional Timed Automata

Unlike timed automata, higher-dimensional automata make no formal distinction between states (0-cubes), transitions (1-cubes), and higher-dimensional cubes. We transfer this intuition to higher-dimensional timed automata, so that each n-cube has an invariant which specifies when it is enabled and an exit condition giving the clocks to be reset when leaving:

▶ **Definition 3.** A *higher-dimensional timed automaton* (HDTA) is a structure $(L, l^0, L^f, \lambda, \mathsf{inv}, \mathsf{exit})$, where (L, l^0, L^f, λ) is a finite higher-dimensional automaton and $\mathsf{inv} : L \to \Phi(C)$, $\mathsf{exit} : L \to 2^C$ assign *invariant* and *exit* conditions to each n-cube.

The *semantics* of a HDTA $A = (L, l^0, L^f, \lambda, \mathsf{inv}, \mathsf{exit})$ is a (usually uncountably infinite) transition system $[\![A]\!] = (S, s^0, S^f, \rightsquigarrow)$, with $\rightsquigarrow \subseteq S \times S$, given as follows:

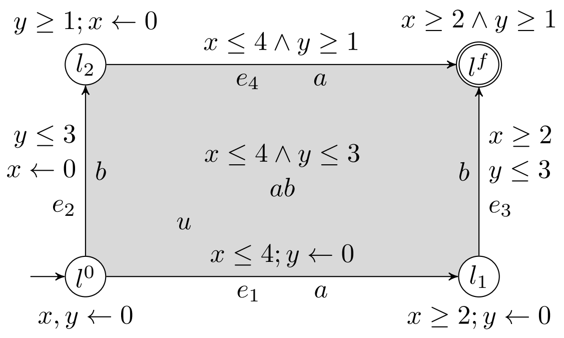
$$\begin{aligned}
&S = \{(l, v) \in L \times \mathbb{R}_{\geq 0}^C \mid v \models \mathsf{inv}(l)\}\\
&s^0 = (l^0, v^0) \quad S^f = S \cap L^f \times \mathbb{R}_{\geq 0}^C\\
&\rightsquigarrow = \{((l, v), (l, v + d)) \mid \forall 0 \leq d' \leq d : v + d' \models \mathsf{inv}(l)\}\\
&\qquad \cup \{((\delta_k^0 l, v), (l, v')) \mid k \in \{1, \dots, \dim l\}, v' = v[\mathsf{exit}(\delta_k^0 l) \leftarrow 0] \models \mathsf{inv}(l)\}\\
&\qquad \cup \{((l, v), (\delta_k^1 l, v')) \mid k \in \{1, \dots, \dim l\}, v' = v[\mathsf{exit}(l) \leftarrow 0] \models \mathsf{inv}(\delta_k^1 l)\}
\end{aligned}$$

We omit labels from the semantics, as we will be concerned only with *reachability* for now: Given a HDTA A, does there exist a finite path $s^0 \rightsquigarrow \cdots \rightsquigarrow s$ in $[\![A]\!]$ such that $s \in S^f$?

Note that in the definition of $\rightsquigarrow$ above, we allow time to evolve in any n-cube in L. Hence transitions (i.e., 1-cubes) are not immediate. The second line in the definition of $\rightsquigarrow$ defines the passing from an $(n-1)$-cube to an n-cube, i.e., the start of a new concurrent event, and the third line describes what happens when finishing a concurrent event. Exit conditions specify which clocks to reset when leaving a cube.

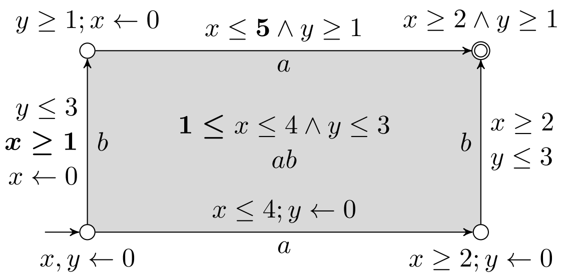
▶ **Example 4.** We give a few examples of two-dimensional timed automata. The first, in Figure 3, models two actions, a and b, which can be performed concurrently. It consists of four states (0-cubes) l^0, l_1, l_2, l^f, four transitions (1-cubes) e_1 through e_4, and one ab-labeled square (2-cube) u. This HDTA models that performing a takes between two and four time units, whereas performing b takes between one and three time units. To this end, we use two clocks x and y which are reset when the respective actions are started and then keep track of how long they are running.

The clocks are reset by the condition $\mathsf{exit}(l^0) = \{x, y\}$, and the invariants $x \leq 4$ at the a-labeled transitions e_1, e_4 and at the square u ensure that a takes at most four time units. The invariants $x \geq 2$ at l_1, e_3 and l^f take care that a cannot finish before two time units have passed. Note that x is also reset when exiting e_2 and l_2, ensuring that regardless when a is started, whether before b, while b is running, or after b is terminated, it must take between two and four time units.



$y \geq 1; x \leftarrow 0$
$x \leq 4 \wedge y \geq 1$
$x \geq 2 \wedge y \geq 1$
l_2
l^f
e_4
a
$y \leq 3$
$x \leftarrow 0$
b
e_2
$x \leq 4 \wedge y \leq 3$
ab
u
b
$x \geq 2$
$y \leq 3$
e_3
$x \leq 4; y \leftarrow 0$
l^0
l_1
e_1
a
$x, y \leftarrow 0$
$x \geq 2; y \leftarrow 0$

Figure 3 The HDTA of Example 4.

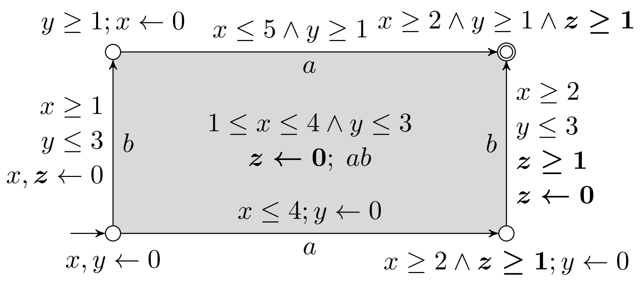


$y \geq 1; x \leftarrow 0$
$x \leq \mathbf{5} \wedge y \geq 1$
$x \geq 2 \wedge y \geq 1$
a
$y \leq 3$
$\boldsymbol{x \geq 1}$
$x \leftarrow 0$
b
$\mathbf{1 \leq} x \leq 4 \wedge y \leq 3$
ab
b
$x \geq 2$
$y \leq 3$
$x \leq 4; y \leftarrow 0$
$x, y \leftarrow 0$
a
$x \geq 2; y \leftarrow 0$

Figure 4 The HDTA of Example 5.

▶ **Example 5.** In the HDTA shown in Figure 4 (where we have omitted the names of states etc. for clarity and show changes to Figure 3 in bold), invariants have been modified so that b can only start after a has been running for one time unit, and if b finishes before a, then a may run one time unit longer. Hence an invariant $x \geq 1$ is added to the two b-labeled transitions and to the ab-square (at the right-most b-transition $x \geq 1$ is already implied), and the condition on x at the top a-transition is changed to $x \leq 5$. Note that the left edge e_2 is now permanently disabled: before entering it, x is reset to zero, but its edge invariant is $x \geq 1$. This is as expected, as b should not be able to start before a.

▶ **Example 6.** The HDTA in Figure 5, where we again show changes to Figure 4 in bold, models the additional constraint that b also *finish* one time unit before a. To this end, an extra clock z is introduced which is reset when b terminates and must be at least 1 when a is terminating. After these changes, the right b-labeled edge is deadlocked: when leaving it, z is reset to zero but needs to be at least one when entering the accepting state. Again, this is expected, as a should not terminate before b.



$y \geq 1; x \leftarrow 0$
$x \leq 5 \wedge y \geq 1$
$x \geq 2 \wedge y \geq 1 \wedge \boldsymbol{z \geq 1}$
a
$x \geq 1$
$y \leq 3$
$x, \boldsymbol{z} \leftarrow 0$
b
$1 \leq x \leq 4 \wedge y \leq 3$
$\boldsymbol{z \leftarrow 0}; ab$
b
$x \geq 2$
$y \leq 3$
$\boldsymbol{z \geq 1}$
$\boldsymbol{z \leftarrow 0}$
$x \leq 4; y \leftarrow 0$
$x, y \leftarrow 0$
a
$x \geq 2 \wedge \boldsymbol{z \geq 1}; y \leftarrow 0$

Figure 5 The HDTA of Example 6.

As both vertical edges are now permanently disabled, the accepting state can only be reached through the square. This shows that reachability for HDTA cannot be reduced to one-dimensional reachability along transitions and relates them to the partial HDA of [29, 20].

▶ Remark. In our model of HDTA, the exit conditions of a cube are the same regardless of *how* the cube is exited. One could imagine an extension of the model where exit conditions may be different depending on whether an action is terminated or a new one is started, so that $\mathsf{exit} : L \times L \rightharpoonup 2^C$ would be a partial function from *pairs* of cubes, one a face of the other. Our results still hold for this extension of the model, but we have not seen use for it in examples, and for the sake of simplicity, we do not pursue it here.

4 One-Dimensional Timed Automata

We work out the relation between one-dimensional HDTA (i.e., 1DTA) and standard timed automata. Note that this is not trivial, as in timed automata, clocks can only be reset at transitions, and, semantically, transitions take no time. In contrast, in our 1DTA, resets can occur in states and transitions may take time.

▶ **Proposition 7.** *There is a linear-time algorithm which, given any timed automaton A, constructs a 1DTA A', with one extra clock, so that A is reachable iff A' is.*

Proof. Let $A = (Q, q^0, Q^f, I, E)$ be a timed automaton. It is clear that $L = Q \cup E$ forms a one-dimensional precubical set, with $L_0 = Q$, $L_1 = E$, $\delta_1^0(q, \phi, a, C', q') = q$, and $\delta_1^1(q, \phi, a, C', q') = q'$. Let $l^0 = q^0$ and $L^f = Q^f$. In order to make transitions immediate, we introduce a fresh clock $c \notin C$. For $q \in Q$, let $\lambda(q) = \emptyset$, $\mathsf{inv}(q) = I(q)$, and $\mathsf{exit}(q) = \{c\}$. For $e = (q, \phi, a, C', q') \in E$, put $\lambda(e) = \{a\}$, $\mathsf{inv}(e) = \phi \wedge (c \leq 0)$, and $\mathsf{exit}(e) = C'$. We have defined a 1DTA $A' = (L, l^0, L^f, \lambda, \mathsf{inv}, \mathsf{exit})$ (over clocks $C \cup \{c\}$). As c is reset whenever exiting a state, and every transition has $c \leq 0$ as part of its invariant, it is clear that transitions in A' take no time, and the claim follows. ◀

▶ **Proposition 8.** *There is a linear-time algorithm which, given any 1DTA A, constructs a timed automaton A' over the same clocks such that A is reachable iff A' is.*

Proof. Let $A = (L, l^0, L^f, \lambda, \mathsf{inv}, \mathsf{exit})$ be a 1DTA, we construct a timed automaton $A' = (Q, q^0, Q^f, I, E)$. Because transitions in A may take time, we cannot simply let $Q = L_0$, but need to add extra states corresponding to the edges in L_1. Let, thus, $Q = L$, $I = \mathsf{inv}$, and

$$E = \{(\delta_1^0 x, \mathbf{tt}, \tau, \mathsf{exit}(\delta_1^0 x), x), (x, \mathbf{tt}, \lambda(x), \mathsf{exit}(x), \delta_1^1 x) \mid x \in L_1\},$$

where $\tau \notin \Sigma$ is a fresh (silent) action; see Figure 6 for an illustration.

Semantically, this converts the 0-cube $\delta_1^0 x$ to the location $\delta_1^0 x$; but its exit condition C_1 is moved to the new outgoing τ-edge. Thus, the location $\delta_1^0 x$ is enabled precisely when the 0-cube $\delta_1^0 x$ is enabled, and when it is exited by immediate execution of the τ-edge, its exit condition is applied. Similar considerations apply to the conversion of the 1-cube x to the location x; thus, A' is reachable iff A is. ◀

Note that even though silent transitions in timed automata are a delicate matter [11], the fact that we add them in the last proof is unimportant as we are only concerned with reachability. PSPACE-completeness of reachability for timed automata [2] and Proposition 7 now imply the following:

▶ **Corollary 9.** *Reachability for HDTA is PSPACE-hard.*

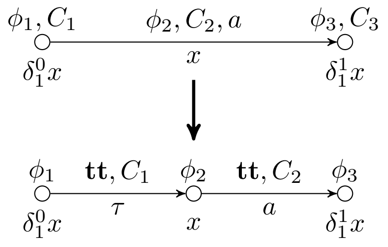


Figure 6 Conversion of 1DTA edge to timed automaton.

5 Reachability for HDTA is PSPACE-Complete

We now turn to extend the notion of *regions* to HDTA, in order to show that reachability for HDTA is decidable in PSPACE.

▶ **Definition 10.** Let $(L, l^0, L^f, \lambda, \mathsf{inv}, \mathsf{exit})$ be a HDTA and $R \subseteq L \times \mathbb{R}_{\geq 0}^C \times L \times \mathbb{R}_{\geq 0}^C$. Then R is an *untimed bisimulation* if $((l^0, v^0), (l^0, v^0)) \in R$ and, for all $((l_1, v_1), (l_2, v_2)) \in R$,
- $l_1 \in L^f$ iff $l_2 \in L^f$;
- for all $(l_1, v_1) \rightsquigarrow (l_1', v_1')$, also $(l_2, v_2) \rightsquigarrow (l_2', v_2')$ for some $((l_1', v_1'), (l_2', v_2')) \in R$;
- for all $(l_2, v_2) \rightsquigarrow (l_2', v_2')$, also $(l_1, v_1) \rightsquigarrow (l_1', v_1')$ for some $((l_1', v_1'), (l_2', v_2')) \in R$.

For a HDTA A, let M_A denote the maximal constant appearing in any $\mathsf{inv}(l)$ for $l \in L$, and let $\cong_{M_A}$ denote the standard region equivalence [5] on $\mathbb{R}_{\geq 0}^C$ defined as follows. For $d \in \mathbb{R}_{\geq 0}$, write $\lfloor d \rfloor$ and $\langle d \rangle$ for the integral, respectively fractional, parts of d, and then for $v, v' : C \to \mathbb{R}_{\geq 0}$, $v \cong_{M_A} v'$ iff
- $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ or $v(x), v'(x) > M_A$, for all $x \in C$,
- $\langle v(x) \rangle = 0$ iff $\langle v'(x) \rangle = 0$, for all $x \in C$ with $v(x) \leq M_A$, and
- $\langle v(x) \rangle \leq \langle v(y) \rangle$ iff $\langle v'(x) \rangle \leq \langle v'(y) \rangle$ for all $x, y \in C$ with $v(x) \leq M_A$ and $v(y) \leq M_A$.

Extend $\cong_{M_A}$ to $[\![A]\!]$ by defining $(l, v) \cong_{M_A} (l', v')$ iff $l = l'$ and $v \cong_{M_A} v'$.

▶ **Lemma 11.** *$\cong_{M_A}$ is an untimed bisimulation.*

Proof. This follows from standard properties of region equivalence [5]. First, $(l^0, v^0) \cong_{M_A} (l^0, v^0)$, and for all $(l_1, v_1) \cong_{M_A} (l_2, v_2)$, $l_1 \in L^f \Leftrightarrow l_2 \in L^f$ because $l_1 = l_2$.

Let $(l, v_1) \cong_{M_A} (l, v_2)$ and $(l, v_1) \rightsquigarrow (l', v_1')$; we show that there is v_2' such that $(l, v_2) \rightsquigarrow (l', v_2')$ and $(l', v_1') \cong_{M_A} (l', v_2')$. The symmetric case is analogous.

Assume $v_1' = v_1 + d$ and $l' = l$, then we have d' such that $v_2' := v_2 + d' \cong_{M_A} v_1 + d$, but then also $(l, v_2) \rightsquigarrow (l', v_2')$ and $(l, v_1') \cong_{M_A} (l, v_2')$.

Assume $l = \delta_k^0 l'$ for some k and $v_1' = v_1[\mathsf{exit}(l) \leftarrow 0] \models \mathsf{inv}(l')$. Let $v_2' = v_2[\mathsf{exit}(l) \leftarrow 0]$, then $v_2' \cong_{M_A} v_1'$ and $v_2' \models \mathsf{inv}(l')$, hence $(l, v_2) \rightsquigarrow (l', v_2')$ and $(l, v_1') \cong_{M_A} (l, v_2')$.

Assume $l' = \delta_k^1 l$ for some k and $v_1' = v_1[\mathsf{exit}(l) \leftarrow 0] \models \mathsf{inv}(l')$. Let $v_2' = v_2[\mathsf{exit}(l) \leftarrow 0]$, then $v_2' \cong_{M_A} v_1'$ and $v_2' \models \mathsf{inv}(l')$, hence $(l, v_2) \rightsquigarrow (l', v_2')$ and $(l, v_1') \cong_{M_A} (l, v_2')$. ◀

For any HDTA A, the *quotient* of $[\![A]\!] = (S, s^0, S^f, \rightsquigarrow)$ under an untimed bisimulation R is defined, as usual, as $[\![A]\!]/R = (S/R, [s^0]_R, S^f/R, \tilde{\rightsquigarrow})$, where S/R is the set of equivalence classes, $[s^0]_R$ is the equivalence class which contains s^0, and $\tilde{\rightsquigarrow} \subseteq S/R \times S/R$ is defined by $\tilde{s} \tilde{\rightsquigarrow} \tilde{s}'$ iff $\exists s \in \tilde{s}, s' \in \tilde{s}' : s \rightsquigarrow s'$.

▶ **Lemma 12.** *Let A be a HDTA and R an untimed bisimulation on A. Then A is reachable iff $[\![A]\!]/R$ is.*

Proof. By definition, an accepting location is reachable in A iff an accepting state is reachable in $[\![A]\!]$. On $[\![A]\!]$, R is a standard bisimulation, hence the claim follows. ◀

▶ **Lemma 13.** *For any HDTA A, the quotient $[\![A]\!]/{\cong_{M_A}}$ is finite.*

Proof. This follows immediately from the standard fact that the set of clock regions, i.e., $\mathbb{R}_{\ge 0}^C/{\cong_{M_A}}$, is finite [5]. ◀

The size of $[\![A]\!]/{\cong_{M_A}}$ is exponential in the size of A, but reachability in $[\![A]\!]/{\cong_{M_A}}$ can be decided in PSPACE, see [5]. Together with Corollary 9, we conclude:

▶ **Theorem 14.** *Reachability for HDTA is PSPACE-complete.*

6 Zone-Based Reachability

We show that the standard zone-based algorithm for checking reachability in timed automata also applies in our HDTA setting. This is important, as zone-based reachability checking is at the basis of the success of tools such as Uppaal, see [42].

Recall that the set $\Phi^+(C)$ of *extended clock constraints* over C is defined by the grammar

$$\Phi^+(C) \ni \phi_1, \phi_2 ::= c \bowtie k \mid c_1 - c_2 \bowtie k \mid \phi_1 \land \phi_2 \qquad (c, c_1, c_2 \in C, k \in \mathbb{Z}, \bowtie \in \{<, \le, \ge, >\}),$$

and that a *zone* over C is a subset $Z \subseteq \mathbb{R}_{\ge 0}^C$ which can be represented by an extended clock constraint ϕ, i.e., such that $Z = [\![\phi]\!]$. Let $\mathcal{Z}(C)$ denote the set of zones over C.

For a zone $Z \in \mathcal{Z}(C)$ and $C' \subseteq C$, the *delay* and *reset* of Z are given by $Z^\uparrow = \{v + d \mid v \in Z\}$ and $Z[C' \leftarrow 0] = \{v[C' \leftarrow 0] \mid v \in Z\}$; these are again zones, and their representation by an extended clock constraint can be efficiently computed [10]. Also zone inclusion $Z' \subseteq Z$ can be efficiently decided.

The *zone graph* of a HDTA $A = (L, l^0, L^f, \lambda, \mathsf{inv}, \mathsf{exit})$ is a (usually infinite) transition system $Z(A) = (S, s^0, S^f, \rightsquigarrow)$, with $\rightsquigarrow \subseteq S \times S$, given as follows:

$$\begin{aligned}
&S = \{(l, Z) \in L \times \mathcal{Z}(C) \mid Z \subseteq [\![\mathsf{inv}(l)]\!]\}\\
&s^0 = (l^0, [\![v^0]\!]^\uparrow \cap [\![\mathsf{inv}(l^0)]\!]) \qquad S^f = S \cap L^f \times \mathcal{Z}(C)\\
&\rightsquigarrow = \{((\delta_k^0 l, Z), (l, Z')) \mid k \in \{1, \dots, \dim l\}, Z' = Z[\mathsf{exit}(\delta_k^0 l) \leftarrow 0]^\uparrow \cap [\![\mathsf{inv}(l)]\!]\}\\
&\qquad \cup \{((l, Z), (\delta_k^1 l, Z')) \mid k \in \{1, \dots, \dim l\}, Z' = Z[\mathsf{exit}(l) \leftarrow 0]^\uparrow \cap [\![\mathsf{inv}(\delta_k^1 l)]\!]\}
\end{aligned}$$

As an example, Figure 7 shows the zone graph of the HDTA in Figure 3 (Example 4), with zones displayed graphically using x as the horizontal axis and y as the vertical. (We have taken the liberty to simplify by computing unions of zones at the locations u, e_3 and e_4 before proceeding.)

▶ **Lemma 15.** *For any HDTA A, an accepting location is reachable in A iff an accepting state is reachable in $Z(A)$.*

Proof. This follows from standard arguments as to the soundness and completeness of the zone abstraction, see [5]. ◀

Any standard *normalization* technique [10] may now be used to ensure that only a finite portion of the zone graph $Z(A)$ is visited, and then the standard zone algorithms can be employed to efficiently decide reachability in HDTA.

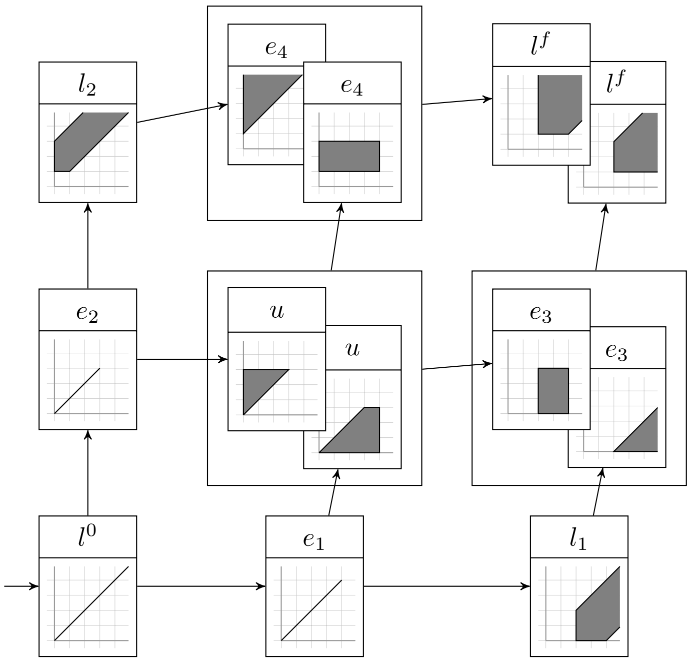


Figure 7 Zone graph of the HDTA in Figure 3.

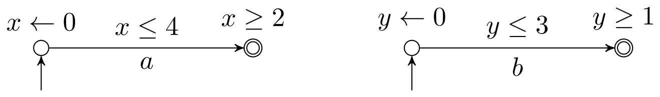


Figure 8 The two 1DTA of Example 17.

7 Parallel Composition of HDTA

There is a *tensor product* on precubical sets which extends to HDTA and can be used for parallel composition (below we use $\sqcup$ for disjoint unions):

▶ **Definition 16.** Let $A_i = (L^i, l^{i,0}, L^{i,f}, \lambda^i, \mathsf{inv}^i, \mathsf{exit}^i)$, for $i = 1, 2$, be HDTA. The *tensor product* of A^1 and A^2 is $A^1 \otimes A^2 = (L, l^0, L^f, \lambda, \mathsf{inv}, \mathsf{exit})$ given as follows:

$$L_n = \bigsqcup_{p+q=n} L^1_p \times L^2_q \qquad l^0 = (l^{1,0}, l^{2,0}) \qquad L^f = L^{1,f} \times L^{2,f}$$

$$\delta^\nu_i(l^1, l^2) = \begin{cases} (\delta^\nu_i l^1, l^2) & \text{if } i \leq \dim l^1 \\ (l^1, \delta^\nu_{i-\dim l^1} l^2) & \text{if } i > \dim l^1 \end{cases}$$

$$\lambda(l^1, l^2) = \lambda(l^1) \sqcup \lambda(l^2) \qquad \mathsf{inv}(l^1, l^2) = \mathsf{inv}(l^1) \wedge \mathsf{inv}(l^2)$$

$$\mathsf{exit}(l^1, l^2) = \mathsf{exit}(l^1) \sqcup \mathsf{exit}(l^2)$$

Intuitively, tensor product is asynchronous parallel composition, or independent product. In combination with relabeling and restriction, any parallel composition operator can be obtained, see [60] or [24] for the special case of HDA.

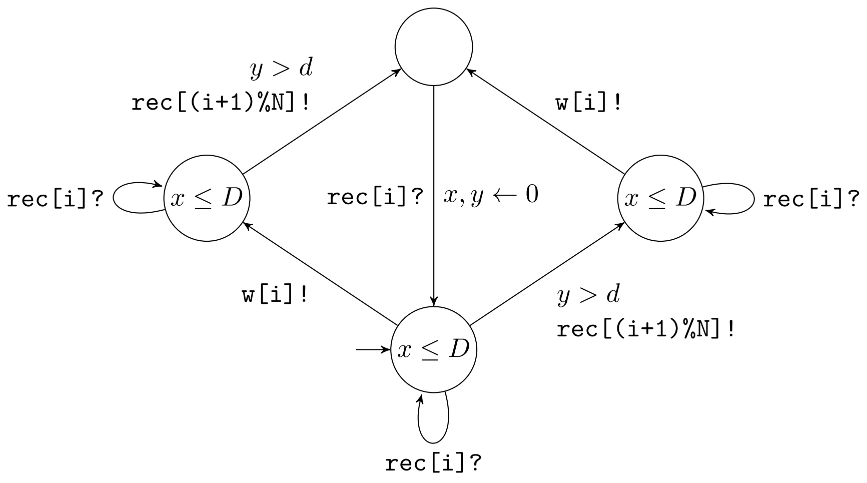


Figure 9 A single node in Milner's scheduler from [19].

▶ **Example 17.** Of the two 1DTA in Figure 8, the first models the constraint that performing the action a takes between two and four time units, and the second, that performing b takes between one and three time units. (In the notation of [14], these are the processes $a[2]$:$a(2)$:0 and $b[1]$:$b(2)$:0.) Their tensor product is precisely the HDTA of Example 4.

▶ **Example 18.** Using tensor product for parallel composition, one can avoid introducing spurious interleavings and thus combat state-space explosion. As an example, we recall the real-time version of Milner's scheduler from [19], a real-time round-robin scheduler in which the nodes are simple timed automata. Figure 9 shows one node in the scheduler, with two transitions from the initial to the topmost state, one that outputs w[i] ("work") and another that passes on the token (rec[(i+1)%N]!). These transitions are independent, but because of the limitations of the timed-automata formalism, they have to be modeled as an interleaving diamond. Apart from the number N of nodes the model has two other parameters, real numbers $d < D$ that specify the time interval within which the tokens have to be passed on.

When a larger number of nodes ($\texttt{N} = 30$, say) are composed into a scheduler, a high amount of interleaving is generated: but most of it is spurious, owing to constraints of the modeling language rather than properties of the system at hand. That is, most of the interleaving in the composed model is an artefact of the modeling formalism and denotes, so to say, higher-dimensional concurrent structures which have been forgotten. The authors of [19] show that especially when d is much smaller than D (for example, $d = 4$ and $D = 30$), verification of the scheduler becomes impossible already for $\texttt{N} = 6$ nodes.

One can use methods from partial order reduction [33] to *detect* spurious interleavings. Aside from the fact that this has proven to be largely impractical for timed automata [39], we also argue that by using HDTA as a modeling language, partial order reduction is, so to speak, *built into* the model. Spurious interleavings are taken care of during the modeling phase, instead of having to be detected during the verification phase.

8 Higher-Dimensional Hybrid Automata

We show that our definition of HDTA extends to one for higher-dimensional hybrid automata. Let X be a finite set of variables, $\dot{X} = \{\dot{x} \mid x \in X\}$, $X' = \{x' \mid x \in X\}$, and $\mathrm{Pred}(Y)$ the set of (arithmetic) predicates on free variables in Y.

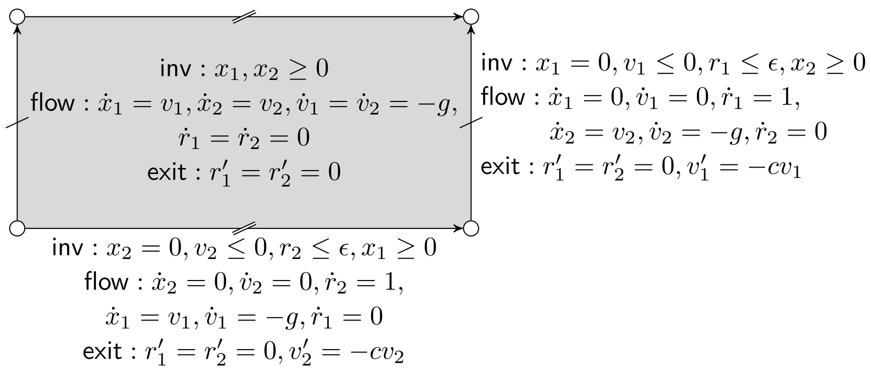

inv : $x_1, x_2 \geq 0$
flow : $\dot{x}_1 = v_1, \dot{x}_2 = v_2, \dot{v}_1 = \dot{v}_2 = -g,$
$\dot{r}_1 = \dot{r}_2 = 0$
exit : $r'_1 = r'_2 = 0$
inv : $x_1 = 0, v_1 \leq 0, r_1 \leq \epsilon, x_2 \geq 0$
flow : $\dot{x}_1 = 0, \dot{v}_1 = 0, \dot{r}_1 = 1,$
$\dot{x}_2 = v_2, \dot{v}_2 = -g, \dot{r}_2 = 0$
exit : $r'_1 = r'_2 = 0, v'_1 = -cv_1$
inv : $x_2 = 0, v_2 \leq 0, r_2 \leq \epsilon, x_1 \geq 0$
flow : $\dot{x}_2 = 0, \dot{v}_2 = 0, \dot{r}_2 = 1,$
$\dot{x}_1 = v_1, \dot{v}_1 = -g, \dot{r}_1 = 0$
exit : $r'_1 = r'_2 = 0, v'_2 = -cv_2$

Figure 10 Two independently bouncing balls (opposite edges identified).

▶ **Definition 19.** A *higher-dimensional hybrid automaton* (HDHA) is a structure $(L, \lambda, \mathsf{init}, \mathsf{inv}, \mathsf{flow}, \mathsf{exit})$, where (L, λ) is a finite higher-dimensional automaton and $\mathsf{init}, \mathsf{inv} : L \to \text{Pred}(X)$, $\mathsf{flow} : L \to \text{Pred}(X \cup \dot{X})$, and $\mathsf{exit} : L \to \text{Pred}(X \cup X')$ assign *initial*, *invariant*, *flow*, and *exit* conditions to each n-cube.

Note that we have removed initial and final locations from the definition; this is standard for hybrid automata. Also, remark how this continues our mantra that there is no conceptual difference between states and transitions; everything is an n-cube, and what are transition *guards* in hybrid automata are now invariants.

The *semantics* of a HDHA $A = (L, \lambda, \mathsf{init}, \mathsf{inv}, \mathsf{flow}, \mathsf{exit})$ is a (usually uncountably infinite) transition system $[\![A]\!] = (S, S^0, \rightsquigarrow)$, with $\rightsquigarrow \subseteq S \times S$, given as follows:

$$\begin{aligned}
S &= \{(l, v) \in L \times \mathbb{R}^X_{\geq 0} \mid v \models \mathsf{inv}(l)\} \\
S^0 &= \{(l, v) \in S \mid v \models \mathsf{init}(l)\} \\
\rightsquigarrow &= \{((l, v), (l, v')) \mid \exists d \geq 0, f \in \mathcal{D}([0, d], \mathbb{R}^X) : f(0) = v, f(d) = v', \\
&\qquad\qquad \forall t \in]0, d[: f(t) \models \mathsf{inv}(q), (f(t), \dot{f}(t)) \models \mathsf{flow}(q)\} \\
&\quad \cup \{((\delta^0_k l, v), (l, v')) \mid k \in \{1, \dots, \dim l\}, (v, v') \models \mathsf{exit}(\delta^0_k l)\} \\
&\quad \cup \{((l, v), (\delta^1_k l, v')) \mid k \in \{1, \dots, \dim l\}, (v, v') \models \mathsf{exit}(l)\}
\end{aligned}$$

Here $\mathcal{D}(D_1, D_2)$ denotes the set of differentiable functions $D_1 \to D_2$, and we write $(v, \dot{v}) \models \mathsf{flow}(q)$ to mean that the predicate $\mathsf{flow}(q)$ on $X \cup \dot{X}$ evaluates to true when the variables are replaced by their values in v and $\dot{v}$; similarly for $(v, v') \models \mathsf{exit}(l)$.

▶ **Example 20.** As a non-trivial example, we show a 2DHA which models two independently bouncing balls, following the *temporal regularization* from [40], in Figure 10. Here, the 2-cube models the state in which both balls are in the air. Its left and right edges are identified, as are its lower and upper edges, so that topologically, this model is a torus.

Its left / right edge is the state in which the second ball is in the air, whereas the first ball is in its ϵ-regularized transition ($\epsilon > 0$) from falling to raising ($v'_1 = -cv$, for some $c \in]0, 1[$). Similarly, its lower / upper edge is the state in which the first ball is in the air, while the second ball is ϵ-transitioning.

Due to the identifications, there is only one 0-cube, which models the state in which both balls are ϵ-transitioning; its inv, flow and exit conditions can be inferred from the ones given.

We can extend the tensor product of HDTA to HDHA:

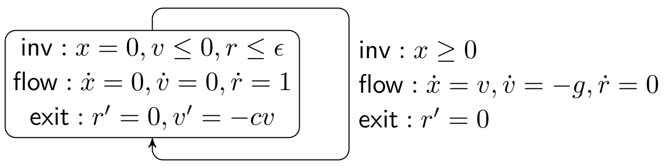

inv : $x = 0, v \leq 0, r \leq \epsilon$
flow : $\dot{x} = 0, \dot{v} = 0, \dot{r} = 1$
exit : $r' = 0, v' = -cv$
inv : $x \geq 0$
flow : $\dot{x} = v, \dot{v} = -g, \dot{r} = 0$
exit : $r' = 0$

Figure 11 HDHA for a bouncing ball.

▶ **Definition 21.** Let $A_i = (L^i, \lambda^i, \mathsf{init}^i, \mathsf{inv}^i, \mathsf{flow}^i, \mathsf{exit}^i)$, for $i = 1, 2$, be HDHA. The *tensor product* of A^1 and A^2 is $A^1 \otimes A^2 = (L, \lambda, \mathsf{inv}, \mathsf{flow}, \mathsf{exit})$ given as follows:

$$L_n = \bigsqcup_{p+q=n} L_p^1 \times L_q^2 \qquad \delta_i^\nu(l^1, l^2) = \begin{cases} (\delta_i^\nu l^1, l^2) & \text{if } i \leq \dim l^1 \\ (l^1, \delta_{i-\dim l^1}^\nu l^2) & \text{if } i > \dim l^1 \end{cases}$$

$$\lambda(l^1, l^2) = \lambda(l^1) \sqcup \lambda(l^2)$$

$$\mathsf{init}(l^1, l^2) = \mathsf{init}(l^1) \land \mathsf{init}(l^2) \qquad \mathsf{inv}(l^1, l^2) = \mathsf{inv}(l^1) \land \mathsf{inv}(l^2)$$

$$\mathsf{flow}(l^1, l^2) = \mathsf{flow}(l^1) \land \mathsf{flow}(l^2) \qquad \mathsf{exit}(l^1, l^2) = \mathsf{exit}(l^1) \land \mathsf{exit}(l^2)$$

▶ **Example 22.** Figure 11 shows the HDHA for a bouncing ball, with one 1-cube in which the ball is in the air and one 0-cube for its ϵ-regularized transition from falling to raising. Denoting the model as $A(x, v, r)$, we can now construct models for arbitrarily many independently bouncing balls as

$$\bigotimes_{i=1}^{k} A(x_i, v_i, r_i)\,;$$

note that for $k = 2$ we obtain the HDHA from Figure 10. We emphasize that such a simple construction for systems of bouncing balls is not available in the standard interleaving formalisms for hybrid automata [3].

9 Conclusion and Further Work

We have seen that our new formalism of higher-dimensional timed automata (HDTA) is useful for modeling interesting properties of non-interleaving real-time systems, and that reachability for HDTA is PSPACE-complete, but can be decided using zone-based algorithms.

We have also shown how tensor product of HDTA can be used for parallel composition, and that HDTA can easily be generalized to higher-dimensional hybrid automata. We believe that altogether, this defines a powerful modeling formalism for non-interleaving cyber-physical systems.

We have argued that in a non-interleaving real-time setting, events should have a time duration. Note that this differs from [15, 18, 17], where, going back to [7], processes are partial orders of events which are fired punctually. Chatain and Jard in [18] notice that *"[t]ime and causality [do] not necessarily blend well in [...] Petri nets"* and propose to (locally) let time run backwards to get nicer semantics. We should like to argue that our proposal of letting events have duration appears more natural.

We have not paid any attention to categorical notions or results here. Higher-dimensional automata have a natural categorical semantics [24], and also for timed automata, works on categorical semantics are available [26, 45, 21], so this should be a natural extension. We would have liked the semantics of HDTA to be precubical in some sense, but this does not seem easy. A coalgebraic formulation of higher-dimensional automata would help here, but also this is not available.

We have mentioned that techniques from geometry and (directed) topology are used to analyze higher-dimensional automata. We have not used any such techniques here, but it appears only natural to try to extend them to work for HDTA. A starting point could be the *timed higher-dimensional automata* of Goubault's [35], which are essentially connected complexes of singular cubes in a locally compact Hausdorff space. We believe that HDTA can be given semantics as sets of such Goubault automata.

Finally, we should mention that this paper is part of a long-term effort to develop useful theory and tools for the analysis of *distributed hybrid systems*. Such systems consist of cyber-physical components which are distributed in the sense that no central clock synchronization mechanism is available, and the current state-of-the-art in distributed and hybrid systems analysis does not allow for the modeling and analysis of such systems. We believe that through convergence and interaction of methods and tools from concurrency theory, hybrid systems, control theory, and distributed systems, significant advances can be obtained in this area.

References

1 Luca Aceto, Anna Ingólfsdóttir, Kim G. Larsen, and Jiří Srba. *Reactive Systems*. Cambridge University Press, 2007.

2 Luca Aceto and François Laroussinie. Is your model checker on time? On the complexity of model checking for timed modal logics. *Journal of Logic and Algebraic Methods in Programming*, 52-53:7–51, 2002.

3 Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

4 Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In Mike Paterson, editor, *ICALP*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, 1990.

5 Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

6 Youssef Arbach, David Karcher, Kirstin Peters, and Uwe Nestmann. Dynamic causality in event structures. In Susanne Graf and Mahesh Viswanathan, editors, *FORTE*, volume 9039 of *Lecture Notes in Computer Science*, pages 83–97. Springer, 2015.

7 Tuomas Aura and Johan Lilius. Time processes for time Petri-nets. In Pierre Azéma and Gianfranco Balbo, editors, *ICATPN*, volume 1248 of *Lecture Notes in Computer Science*, pages 136–155. Springer, 1997.

8 Marek A. Bednarczyk. *Categories of asynchronous systems*. PhD thesis, University of Sussex, UK, 1987.

9 Gerd Behrmann, Alexandre David, and Kim Guldstrand Larsen. A tutorial on uppaal. In Marco Bernardo and Flavio Corradini, editors, *SFM-RT*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer, 2004.

10 Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 87–124. Springer, 2003.

11 Béatrice Bérard, Antoine Petit, Volker Diekert, and Paul Gastin. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, 36(2-3):145–182, 1998.

12 Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, Nicolas Markey, Joël Ouaknine, and James Worrell. Model checking real-time systems. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking.*, pages 1001–1046. Springer, 2018.

13 Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. Kronos: A model-checking tool for real-time systems. In Alan J. Hu and Moshe Y. Vardi, editors, *CAV*, volume 1427 of *Lecture Notes in Computer Science*, pages 546–550. Springer, 1998.

14 Luca Cardelli. Real time agents. In Mogens Nielsen and Erik Meineche Schmidt, editors, *ICALP*, volume 140 of *Lecture Notes in Computer Science*, pages 94–106. Springer, 1982.

15 Franck Cassez, Thomas Chatain, and Claude Jard. Symbolic unfoldings for networks of timed automata. In Susanne Graf and Wenhui Zhang, editors, *ATVA*, volume 4218 of *Lecture Notes in Computer Science*, pages 307–321. Springer, 2006.

16 Giovanni Casu and G. Michele Pinna. Petri nets and dynamic causality for service-oriented computations. In Ahmed Seffah, Birgit Penzenstadler, Carina Alves, and Xin Peng, editors, *SAC*, pages 1326–1333. ACM, 2017.

17 Thomas Chatain and Claude Jard. Complete finite prefixes of symbolic unfoldings of safe time Petri nets. In Susanna Donatelli and P. S. Thiagarajan, editors, *ICATPN*, volume 4024 of *Lecture Notes in Computer Science*, pages 125–145. Springer, 2006.

18 Thomas Chatain and Claude Jard. Back in time Petri nets. In Víctor A. Braberman and Laurent Fribourg, editors, *FORMATS*, volume 8053 of *Lecture Notes in Computer Science*, pages 91–105. Springer, 2013.

19 Alexandre David, Kim G. Larsen, Axel Legay, Ulrik Nyman, Louis-Marie Traonouez, and Andrzej Wasowski. Real-time specifications.

Int. J. Software Tools for Technology Transfer, 17(1):17–45, 2015.

20 Jérémy Dubut. Trees in partial higher dimensional automata. In Mikołaj Bojańczyk and Alex Simpson, editors, *FOSSACS*, volume 11425 of *Lecture Notes in Computer Science*, pages 224–241. Springer, 2019.

21 Jérémy Dubut, Ichiro Hasuo, Shin-ya Katsumata, and David Sprunger. Quantitative bisimulations using coreflections and open morphisms. *CoRR*, abs/1809.09278, 2018.

22 Javier Esparza. A false history of true concurrency: From Petri to tools (invited talk). In Jaco van de Pol and Michael Weber, editors, *SPIN*, volume 6349 of *Lecture Notes in Computer Science*, pages 180–186. Springer, 2010.

23 Javier Esparza and Keijo Heljanko. *Unfoldings – A Partial-Order Approach to Model Checking*. Monographs Theor. Comput. Sci. Springer, 2008.

24 Uli Fahrenberg. A category of higher-dimensional automata. In Vladimiro Sassone, editor, *FoSSaCS*, volume 3441 of *Lecture Notes in Computer Science*, pages 187–201. Springer, 2005.

25 Uli Fahrenberg. *Higher-Dimensional Automata from a Topological Viewpoint*. PhD thesis, Aalborg University, Denmark, 2005.

26 Uli Fahrenberg. How to pull back open maps along semantics functors. In Jochen Pfalzgraf, editor, *ACCAT*, 2008.

27 Uli Fahrenberg. Higher-dimensional timed automata. In Alessandro Abate, Antoine Girard, and Maurice Heemels, editors, *ADHS*, volume 51 of *IFAC-PapersOnLine*, pages 109–114. Elsevier, 2018.

28 Uli Fahrenberg, Christian Johansen, Georg Struth, and Krzysztof Ziemiański. Languages of higher-dimensional automata. *Mathematical Structures in Computer Science*, pages 1–39, 2021.

29 Uli Fahrenberg and Axel Legay. Partial higher-dimensional automata. In Lawrence S. Moss and Pawel Sobocinski, editors, *CALCO*, volume 35 of *LIPIcs*, pages 101–115, 2015.

30 Lisbeth Fajstrup, Eric Goubault, Emmanuel Haucourt, Samuel Mimram, and Martin Raussen. *Directed Algebraic Topology and Concurrency*. Springer, 2016.

31 Lisbeth Fajstrup, Martin Raussen, and Éric Goubault. Algebraic topology and concurrency. *Theoretical Computer Science*, 357(1-3):241–278, 2006.

32 Hans Fleischhack and Christian Stehno. Computing a finite prefix of a time Petri net. In Javier Esparza and Charles Lakos, editors, *ICATPN*, volume 2360 of *Lecture Notes in Computer Science*, pages 163–181. Springer, 2002.

33 Patrice Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems*, volume 1032 of *Lecture Notes in Computer Science*. Springer, 1996.

34 Ursula Goltz and Wolfgang Reisig. The non-sequential behavior of Petri nets. *Information and Control*, 57(2/3):125–147, 1983.

35 Eric Goubault. Durations for truly-concurrent transitions. In Hanne Riis Nielson, editor, *ESOP*, volume 1058 of *Lecture Notes in Computer Science*, pages 173–187. Springer, 1996.

36 Eric Goubault. Labelled cubical sets and asynchronous transition systems: an adjunction. In *Preliminary Proceedings CMCIM'02*, 2002.

37 Marco Grandis. *Directed algebraic topology: models of non-reversible worlds.* New mathematical monographs. Cambridge University Press, 2009.

38 Hans-Michael Hanisch. Analysis of place/transition nets with timed arcs and its application to batch process control. In Marco Ajmone Marsan, editor, *ATPN*, volume 691 of *Lecture Notes in Computer Science*, pages 282–299. Springer, 1993.

39 Henri Hansen, Shang-Wei Lin, Yang Liu, Truong Khanh Nguyen, and Jun Sun. Partial order reduction for timed automata with abstractions. In Armin Biere and Roderick Bloem, editors, *CAV*, volume 8559 of *Lecture Notes in Computer Science*, pages 391–406. Springer, 2014.

40 Karl Henrik Johansson, Magnus Egerstedt, John Lygeros, and Shankar Sastry. On the regularization of Zeno hybrid automata. *Systems & Control Letters*, 38(3):141–150, 1999.

41 Kim G. Larsen, Uli Fahrenberg, and Axel Legay. From timed automata to stochastic hybrid games. In *Dependable Software Systems Engineering*, pages 60–103. IOS Press, 2017.

42 Kim G. Larsen, Paul Pettersson, and Wang Yi. Uppaal in a nutshell. *Int. J. Software Tools for Technology Transfer*, 1(1-2):134–152, 1997.

43 Philip M. Merlin and David J. Farber. Recoverability of communication protocols–implications of a theoretical study. *IEEE Transactions on Communications*, 24(9):1036–1043, 1976.

44 Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.

45 Mogens Nielsen and Thomas Hune. Bisimulation and open maps for timed transition systems. *Fundamenta Informaticae*, 38(1-2):61–77, 1999.

46 Mogens Nielsen, Gordon D. Plotkin, and Glynn Winskel. Petri nets, event structures and domains, part I. *Theoretical Computer Science*, 13:85–108, 1981.

47 Carl A. Petri. *Kommunikation mit Automaten*. Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.

48 Vaughan R. Pratt. Modeling concurrency with geometry. In David S. Wise, editor, *POPL*, pages 311–322. ACM Press, 1991.

49 Vaughan R. Pratt. Higher dimensional automata revisited. *Mathematical Structures in Computer Science*, 10(4):525–548, 2000.

50 Mike W. Shields. Concurrent machines. *The Computer Journal*, 28(5):449–465, 1985.

51 Joseph Sifakis. Use of Petri nets for performance evaluation. In *Measuring, Modelling and Evaluating Computer Systems*, pages 75–93. North-Holland, 1977.

52 Joseph Sifakis and Sergio Yovine. Compositional specification of timed systems. In Claude Puech and Rüdiger Reischuk, editors, *STACS*, volume 1046 of *Lecture Notes in Computer Science*, pages 347–359. Springer, 1996.

53 Jiří Srba. Comparing the expressiveness of timed automata and timed extensions of Petri nets. In Franck Cassez and Claude Jard, editors, *FORMATS*, volume 5215 of *Lecture Notes in Computer Science*, pages 15–32. Springer, 2008.

54 Rob J. van Glabbeek. Bisimulations for higher dimensional automata. Email message, June 1991.

55 Rob J. van Glabbeek. On the expressiveness of higher dimensional automata. *Theoretical Computer Science*, 356(3):265–290, 2006.

56 Rob J. van Glabbeek. Erratum to "On the expressiveness of higher dimensional automata". *Theoretical Computer Science*, 368(1-2):168–194, 2006.

57 Rob J. van Glabbeek and Gordon D. Plotkin. Configuration structures. In *LICS*, pages 199–209. IEEE Computer Society, 1995.

58 Rob J. van Glabbeek and Gordon D. Plotkin. Configuration structures, event structures and Petri nets. *Theoretical Computer Science*, 410(41):4111–4159, 2009.

59 Farn Wang, Aloysius K. Mok, and E. Allen Emerson. Symbolic model checking for distributed real-time systems. In Jim Woodcock and Peter Gorm Larsen, editors, *FME*, volume 670 of *Lecture Notes in Computer Science*, pages 632–651. Springer, 1993.

60 Glynn Winskel and Mogens Nielsen. Models for concurrency. In *Handbook of Logic in Computer Science*, volume 4. Clarendon Press, Oxford, 1995.