Higher-Dimensional Timed Automata

Uli Fahrenberg

LIX, École polytechnique, Palaiseau, France

Abstract: We introduce a new formalism of higher-dimensional timed automata, based on van Glabbeek’s higher-dimensional automata and Alur’s timed automata. We prove that their reachability is PSPACE-complete and can be decided using zone-based algorithms. We also show how to use tensor products to combat state-space explosion and how to extend the setting to higher-dimensional hybrid automata.

Keywords: timed automata, higher-dimensional automata, real time, non-interleaving concurrency, hybrid automata, state-space explosion

1. INTRODUCTION

In approaches to non-interleaving concurrency, more than one event may happen concurrently. There is a plethora of formalisms for modeling and analyzing such concurrent systems, e.g., Petri nets (Petri, 1962), event structures (Nielsen et al., 1981), configuration structures (van Glabbeek and Plotkin, 1995, 2009), or more recent variations such as dynamic event structures (Arbach et al., 2015) and Unravel nets (Casu and Pinna, 2017). They all share the convention of differentiating between concurrent and interleaving executions; using CCS notation (Milner, 1989), $a \parallel b \neq a.b + b.a$.

For modeling and analyzing embedded or cyber-physical systems, formalisms which use real time are available. These include timed automata (Alur and Dill, 1994), time Petri nets (Merlin and Farber, 1976), timed-arc Petri nets (Hanisch, 1993), or various classes of hybrid automata (Alur et al., 1995). Common for them all is that they identify concurrent and interleaving executions; here, $a \parallel b = a.b + b.a$.

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 (Goltz and Reisig, 1983) which is also used in practice (Esparza, 2010; Esparza and Heljanko, 2008), but almost all work on real-time extensions of Petri nets (Hanisch, 1993; Merlin and Farber, 1976; Sifakis, 1977; Srba, 2008), including the popular tool TAPAAL1, use an interleaving semantics.

Also Uppaal2, 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, we believe, can be avoided with a non-interleaving semantics such as we propose here.

---

1 This research is supported by the Chaire ISC : Engineering Complex Systems – École polytechnique – Thales – FX – DGA – Dassault Aviation – DCNS Research – ENSTA ParisTech – Télécom ParisTech

2 http://www.tapaal.net/

3 The author wishes to thank Kim G. Larsen for pointing him towards this paper.
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 “if the 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” (Aceto et al., 2007), our extension to higher dimensions and non-interleaving concurrency is completely free of such trouble.

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.

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 denote independence of three events, etc. To introduce them properly, we need to start with precubical sets.

A precubical set \( X \) 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 \( \delta_k^n : X_n \rightarrow X_{n-1} \), \( k = 1, \ldots, n \), \( \nu = 0, 1 \), satisfying the precubical identity
\[
\delta_k^{n-1} \delta_{\nu, n} = \delta^{n-1}_k \delta_{\nu, n-1} \delta_{\nu, n} \quad (k < \ell).
\]

Elements of \( X_n \) are called \( n \)-cubes, and for \( x \in X_n \), \( n = \dim x \) is its dimension. The mappings \( \delta_k^n \) are called face maps, and we will usually omit the extra subscript \( n \) and write \( \delta_k^n \) instead of \( \delta_k^{n,n} \). Intuitively, each \( n \)-cube \( x \in X_n \) has \( n \) lower faces \( \delta_0^n x, \ldots, \delta_{n-1}^n x \) and \( n \) upper faces \( \delta_1^n x, \ldots, \delta_n^n x \), and the precubical identity expresses the fact that \( (n-1) \)-faces of an \( n \)-cube meet in common \((n-2)\)-faces; see Fig. 1 for an example.

![Fig. 1. A 2-cube \( x \) with its four faces \( \delta_0^1 x, \delta_1^1 x, \delta_2^1 x, \delta_3^1 x \) and four corners](image)

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 \( \Sigma \) be a finite set of actions and recall that a multiset over \( \Sigma \) is a mapping \( \Sigma \rightarrow \mathbb{N} \). We denote multisets using double braces \( \{\cdot\} \) and the set of multisets over \( \Sigma \) by \( \Sigma^\ast \).

A higher-dimensional automaton (HDA) is a structure \( (X, x^0, X^f, \lambda) \), where \( X \) is a finite precubical set with initial state \( x^0 \in X_0 \) and accepting states \( X^f \subseteq X_0 \), and \( \lambda : X \rightarrow \Sigma^\ast \) is a labeling function such that for every \( x \in X \),
\[
\begin{align*}
&\cdot |\lambda(x)| = \dim x, \\
&\lambda(\delta^n_k x) = \lambda(\delta^n_0 x) \quad \text{for all } k \leq n, \text{ and} \\
&\lambda(x) \setminus \lambda(\delta^n_k x) \text{ is a singleton for all } k \leq \dim x.
\end{align*}
\]

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 \( \delta_k^n x \) of \( x \in X \) to \( x \) itself, the event in \( \lambda(x) \setminus \lambda(\delta_k^n x) \) is started, and when passing from \( x \) to an upper face \( \delta_k^n x \), the event in \( \lambda(x) \setminus \lambda(\delta_k^n x) \) is terminated.

HDA can indeed model higher-order concurrency of actions. As an example, the hollow cube on the left of Fig. 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 Fig. 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.

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
\[
\Phi(C) \ni \phi_1, \phi_2 ::= c \in \mathbb{R} \quad \text{or} \quad k \cdot \phi_1 \land \phi_2 \\
\quad (c \in C, k \in \mathbb{Z}, \varnothing \in \{<, \leq, >\}).
\]

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

A clock valuation is a mapping \( v : C \rightarrow \mathbb{R}_{\geq 0} \), where \( \mathbb{R}_{\geq 0} \) denotes the set of non-negative real numbers. The initial clock valuation is \( v^0 : C \rightarrow \mathbb{R}_{\geq 0} \) given by \( v^0(c) = 0 \) for
all \( c \in C \). For \( v \in \mathbb{R}_{\geq 0}^c, d \in \mathbb{R}_{\geq 0}^l, \) 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 \( \phi \) and \( [[\phi]] = \{ v : C \to \mathbb{R}_{\geq 0} \mid v \models \phi \} \).

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 infinite) transition system \([A] = (S, s^0, S^f, \sim)\), with \( \sim \subseteq S \times S \), given as follows:
\[
S = \{(q, v) \subseteq Q \times \mathbb{R}_{\geq 0}^C \mid v \models I(q)\}
\]
\[
s^0 = (q^0, v^0), \quad S^f = S \cap Q^f \times \mathbb{R}_{\geq 0}^C
\]
\[
\sim = \{(q, v), (q', v+d) \mid \forall 0 \leq d \leq d : v + d' \models I(q)\}
\]
\[
\cup \{(q, v), (q', v') \mid \exists (q, \phi, a, C', q') \in E : v \models \phi, v' = v[C' \leftarrow 0]\}
\]

Note that we are ignoring the labels here, as we will be concerned with reachability only. As usual, we say that \( A \) is **reachable** if there exists a finite path \( s^0 \sim \cdots \sim s \) in \([A]\) for which \( s \in S^f \).

The purpose of reachability is to ensure that actions are immediate: whenever \((q, v), (q', v') \in E\), then \( A \) passes from \((q, v)\) to \((q', v')\) without any delay. Time progresses only during delays \((q, v) \sim (q, v + d)\) in locations.

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

The **semantics** of a HDTA \( A = (L, l^0, L^f, \lambda, \text{inv, exit}) \) is a (usually infinite) transition system \([A] = (S, s^0, S^f, \sim)\), with \( \sim \subseteq S \times S \), given as follows:
\[
S = \{(l, v) \subseteq L \times \mathbb{R}_{\geq 0}^C \mid v \models \text{inv}(l)\}
\]
\[
s^0 = (l^0, v^0), \quad S^f = S \cap L^f \times \mathbb{R}_{\geq 0}^C
\]
\[
\sim = \{(l, v), (l', v+d) \mid \forall 0 \leq d \leq d : v + d' \models \text{inv}(l)\}
\]
\[
\cup \{(l, v), (l', v') \mid k \in \{1, \ldots, \dim l\}, v' = v[\text{inv}(\delta^k_l) \leftarrow 0] \models \text{inv}(l)\}
\]
\[
\cup \{(l, v), (l', v') \mid k \in \{1, \ldots, \dim l\}, v' = v[\text{exit}(l) \leftarrow 0] \models \text{inv}(\delta^k_l)\}
\]

We omit labels from the semantics, as we will be concerned only with reachability. Given a HDTA \( A \), does there exist a finite path \( s^0 \sim \cdots \sim s \) in \([A]\) such that \( s \in S^f \)?
We work out the relation between one-dimensional HDTA (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 time. In contrast, in our 1DTA, resets can occur in states and transitions may take time.

Proposition 5. 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 precequivalence set, with $L_0 = Q$, $L_1 = E$, $\delta_1(q, \phi, a, C', q') = q$, and $\delta_0(q, \phi, a, C', q') = q$. Let $l_0 = q_0$ and $L_1 = Q_f$. In order to make transitions immediate, we introduce a fresh clock $c \notin C$. For $q \in Q$, let $\lambda(q) = \emptyset$, $\text{inv}(q) = I(q)$, and $\text{exit}(q) = \{c\}$. For $c = (q, \phi, a, C', q') \in E$, put $\lambda(c) = \{a\}$, $\text{inv}(c) = \phi \land (c \leq 0)$, and $\text{exit}(c) = C'$. We have defined a 1DTA $A' = (L, l_0, L_1, \lambda, \text{inv}, \text{exit})$ (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 6. 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 $A = (L, l_0, L_1, \lambda, \text{inv}, \text{exit})$ 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, $Q = L$, $I = \text{inv}$, and $E = \{(\delta_0^\ell(x, \phi, \tau, \text{exit}(x), x), (x, \tau, \lambda(x), \text{exit}(x), \delta_0^\ell(x)) | x \in L_1\}$, where $\tau \notin \Sigma$ is a fresh (silent) action. See Fig. 6.

Note that even though silent transitions in timed automata are a delicate matter (Bérard et al., 1998), 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 now implies the following: Corollary 7. Reachability for HDTA is PSPACE-hard.

5. REACHABILITY FOR HDTA IS IN PSPACE

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

Definition 8. Let $(L, l_0, L_1, \lambda, \text{inv}, \text{exit})$ be a HDTA and $R \subseteq L \times \mathbb{R}_0^\geq \times L \times \mathbb{R}_0^\geq$. Then $R$ is an un timed bisimulation if $((l_0, v_0), (l_0, v_0)) \in R$ and, for all $((l_1, v_1), (l_2, v_2)) \in R$, the following hold:

1. $l_1 \in L_1$ iff $l_2 \in L_1$;
2. whenever $(l_1, v_1) \sim (l_1', v_1')$, then also $(l_2, v_2) \sim (l_2', v_2')$ for some $(l_2', v_2') \in R$;
3. whenever $(l_2, v_2) \sim (l_2', v_2')$, then also $(l_1, v_1) \sim (l_1', v_1')$ for some $(l_1', v_1') \in R$.

For a HDTA $A$, let $M_A$ denote the maximal constant appearing in any $\text{inv}(l)$ for $l \in L$, and let $\equiv_{M_A}$ denote standard region equivalence (Alur and Dill, 1994). Extend $\equiv_{M_A}$ to $[A]$ by defining $(l, v) \equiv_{M_A} (l', v')$ iff $l = l'$ and $v \equiv_{M_A} v'$.

Lemma 9. $\equiv_{M_A}$ is an un timed bisimulation.

Proof. This follows from standard properties of region equivalence (Alur and Dill, 1994).

For any HDTA $A$, the quotient of $[A]$ is $(S, s_0, S_f, \sim)$ under an untimed bisimulation $R$ is defined, as usual, as $[A]/R = (S/R, [s_0]_R, S_f /R, \sim)$, where $S/R$ is the set of equivalence classes, $[s_0]_R$ is the equivalence class in which $s_0$ belongs, and $\sim \subseteq S/R \times S/R$ is defined by $\tilde{s} \sim \tilde{s'}$ iff $\exists s \in \tilde{s}, \tilde{s'} \in \tilde{s} : s \sim s'$.

Lemma 10. 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 11. For any HDTA $A$, the quotient $[A]/\equiv_{M_A}$ is finite.

Proof. This follows immediately from the standard fact that the set of clock regions, i.e., $\mathbb{R}_0^\geq /\equiv_{M_A}$ is finite (Alur and Dill, 1994).
The size of \(|A|/\Sigma M_A\) is exponential in the size of \(A\), but reachability in \(|A|/\Sigma M_A\) can be decided in \(\text{PSPACE}\), see (Alur and Dill, 1994). Together with Corollary 7, we conclude:

**Theorem 12.** Reachability for HDTA is \(\text{PSPACE}\)-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 (Larsen et al., 1997).

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

\[
\Phi^+(C) \ni \phi_1, \phi_2 ::= \text{~}\in \text{~}, c \in k \mid c_1 - c_2 \in k \mid \phi_1 \land \phi_2 \quad (c, c_1, c_2 \in C),
\]

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

For a zone \(Z \in Z(C)\) and \(C' \subseteq C\), the delay and reset of \(Z\) are given by \(Z' = \{v + d \mid v \in Z\}\) 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 (Bengtsson and Yi, 2003). Also zone inclusion \(Z' \subseteq Z\) can be efficiently decided.

The *zone graph* of a HDTA \(A = (L, l_0, L', l, \lambda, \text{inv, exit})\) is a (usually infinite) transition system \(Z(A) = (S, s^0, S^f, \leadsto)\), with \(\leadsto \subseteq S \times S\), given as follows:

\[
S = \{(l, Z) \subseteq L \times Z(C) \mid Z \subseteq \text{inv}(l)\},
\]

\[
s^0 = (l_0, [l_0]) \cap \text{inv}(l_0),
\]

\[
S^f = S \cap L^f \times Z(C)
\]

\[
\leadsto = \{(\text{delay}(l, Z), (l, Z')) \mid k \in \{1, \ldots, \dim l\},
\]

\[
\text{exit}(l, Z') = \text{exit}(l) \cup \text{exit}(Z')
\]

**Lemma 13.** For any HDTA \(A\), an accepting location is reachable in \(A\) if an accepting state is reachable in \(Z(A)\).

**Proof.** This follows from standard arguments as to the soundness and completeness of the zone abstraction (Alur and Dill, 1994). \(\square\)

Any standard normalization technique (Bengtsson and Yi, 2003) may now be used to ensure that the zone graph \(Z(A)\) is finite, and then the standard zone algorithms can be employed to efficiently decide reachability in HDTA. As an example, Fig. 7 shows the zone graph of the HDTA in Fig. 3 (Example 2), 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.)

### 7. PARALLEL COMPOSITION OF HDTA

There is a tensor product on preclockable sets which extends to HDTA and can be used for parallel composition:

**Definition 14.** Let \(A_i = (L^i, l_0^i, L_i^f, \lambda^i, \text{inv}^i, \text{exit}^i)\), for \(i = 1, 2\), be HDTA. The tensor product of \(A^1\) and \(A^2\) is \(A^1 \otimes A^2 = (L, l_0, L^f, \lambda, \text{inv}, \text{exit})\) given as follows:

\[
L_n = \bigcup_{p+q=n} L_p^1 \times L_q^2, \quad l_0 = (l_1^0, l_2^0), \quad L^f = L_1^f \times L_2^f.
\]

\[
\delta^l_l = \{\delta^l_1, \delta^l_2\} \text{ if } i \leq \dim l,
\]

\[
\delta^l_l = \{l_1, \delta^l_{\dim l, l_2}\} \text{ if } i > \dim l.
\]

\[
\lambda(l_1, l_2) = \lambda(l_1) \cup \lambda(l_2), \quad \text{inv}(l_1, l_2) = \text{inv}(l_1) \land \text{inv}(l_2),
\]

\[
\text{exit}(l_1, l_2) = \text{exit}(l_1) \cup \text{exit}(l_2)
\]

Intuitively, tensor product is asynchronous parallel composition, or independent product. In combination with relabeling and restriction, any parallel composition operator can be obtained, see Winskel and Nielsen (1995) or Fahrenberg (2005) for the special case of HDA.

**Example 15.** Of the two 1DTA in Fig. 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 Cardelli (1982), these are \(a[2:4] b[0:3]\) and \(b[1:3] a[0:2]\).) Their tensor product is precisely the HDTA of Example 2.

Using tensor product for parallel composition, one can avoid introducing spurious interleavings and thus combat state-space explosion. Take the real-time version of Milner’s scheduler from David et al. (2015) as an example. This is essentially a real-time round-robin scheduler in which the nodes are simple timed automata, see Fig. 9 for the Uppaal model.

Notice that there are, essentially, two transitions from the initial to the topmost state, one which outputs \(\text{w}[i]\) ("work")

\[
x \leftarrow 0 \quad x \leq 4 \quad x \geq 2 \quad y \leftarrow 0 \quad y \leq 3 \quad y \geq 1
\]

Fig. 7. Zone graph of the HDTA in Fig. 3

![Fig. 7. Zone graph of the HDTA in Fig. 3](image)

Fig. 8. The two 1DTA of Example 15

![Fig. 8. The two 1DTA of Example 15](image)
Fig. 9. Uppaal model of a single node in Milner’s scheduler

and another which 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. Thus, when a number of such nodes (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.

David et al. (2015) show that especially when d is much smaller than D (say, d = 4 and D = 30), verification of the scheduler becomes impossible already for N = 6 nodes. The authors then show that this can be amenable by using compositional verification techniques, which are out of scope of our current work.

Another possibility is to use methods from partial order reduction (Godefroid, 1996) to detect spurious interleavings. Aside from the fact that this has proven to be largely impractical for timed automata, see for example Hansen et al. (2014), 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.

A 2DTA corresponding to the Uppaal model in Fig. 9 is essentially the tensor product of the w[i]! and rec[(i+1)%N]! transitions, together with 1-dimensional loops and a transition from the upper to lower corner for the rec[i]? input transitions.

8. HIGHER-DIMENSIONAL HYBRID AUTOMATA

For completeness, we show that our definition of HDTA easily extends to one for higher-dimensional hybrid automata. Let X be a finite set of variables, $\bar{X} = \{ \bar{x} | x \in X \}$, $X' = \{ x^t | x \in X \}$, and Pred(Y) the set of (arithmetic) predicates on free variables in Y.

Definition 16. A higher-dimensional hybrid automaton (HDHA) is a structure $(L, \lambda, \text{inv}, \text{flow}, \text{exit})$, where $(L, \lambda)$ is a finite higher-dimensional automaton and init, inv : $L \to \text{Pred}(X)$, flow : $L \to \text{Pred}(X \cup X)$, and exit : $L \to \text{Pred}(X \times X')$ 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.

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

Fig. 10. Two independently bouncing balls

\[ S = \{(l, v) \leq L \times \mathbb{R}^X_{\geq 0} \mid v \models \text{inv}(l)\} \]

\[ S^0 = \{(l, v) \in S \mid v \models \text{init}(l)\} \]

\[ \sim = \{(\langle l, v \rangle, \langle l', v' \rangle) \mid \exists d \geq 0, f \in D([0, d], \mathbb{R}^X) : f(0) = v, (d) = v', \forall t \in [0, d) : f(t) \models \text{inv}(q), (f(t), f(t)) \models \text{flow}(q)\} \]

\[ \cup \{(\langle \delta^l_0(t), v \rangle, \langle l', v' \rangle) \mid k \in \{1, \ldots, \dim l\}, (v, v') \models \text{exit}(\delta^l_k(t))\} \]

\[ \cup \{(\langle l, v \rangle, \langle \delta^l_k(t), v' \rangle) \mid k \in \{1, \ldots, \dim l\}, (v, v') \models \text{exit}(l)\} \]

Here $D(D_1, D_2)$ denotes the set of differentiable functions $D_1 \to D_2$.

Example 17. As a non-trivial example, we show a 2DHA which models two independently bouncing balls, following the temporal regularization from Johansson et al. (1999), in Fig. 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 logically, 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 $\epsilon$-regularized transition ($\epsilon > 0$) from falling to raising ($v'_2 = -cv$, 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 $\epsilon$-transitioning.

Due to the identifications, there is only one 0-cube, which models the state in which both balls are $\epsilon$-transitioning; its inv, flow and exit conditions can be inferred from the ones given. With a notion of tensor product similar to the one for HDTA, this model can also be obtained as tensor product of the one-dimensional models for the individual balls.

9. CONCLUSION

We have seen that our new formalism of higher-dimensional timed automata 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 believe that our notion that in a non-interleaving real-time setting, events should have a time duration, is quite natural. Working on non-interleaving real-time semantics for Petri nets, Chatain and Jard (2013) remark that “[t]ime and causality [do] not necessarily blend well in [...] Petri
nets” and propose to 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 also seen 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 real-time systems.

REFERENCES

Aceto, L., Ingólfsdóttir, A., Larsen, K.G., and Srba, J. (2007). Reactive Systems. Cambridge Univ. Press.

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P., Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S. (1995). The algorithmic analysis of hybrid systems. Theor. Comput. Sci., 138(1), 3–34.

Alur, R. and Dill, D.L. (1990). Automata for modeling real-time systems. In ICALP, volume 443 of LNCS, 322–335. Springer.

Alur, R. and Dill, D.L. (1994). A theory of timed automata. Theor. Comput. Sci., 126(2), 183–235.

Arbab, Y., Karcher, D., Peters, K., and Nestmann, U. (2015). Dynamic causality in event structures. In FORTE, volume 9039 of LNCS, 83–97. Springer.

Bengtsson, J. and Yi, W. (2003). Timed automata: Semantics, algorithms and tools. In Lectures on Concurrency and Petri Nets, volume 3098 of LNCS, 87–124. Springer.

Bérard, B., Petit, A., Diekert, V., and Gastin, P. (1998). Characterization of the expressive power of silent transitions in timed automata. Fund. Inf., 36(2-3), 145–182.

Cardelli, L. (1982). Real time agents. In TLCA, volume 187–201. Springer.

Casu, G. and Pinna, G.M. (2017). Petri nets and dynamic causality for service-oriented computations. In ACM SAC. ACM.

Chattain, T. and Jard, C. (2013). Back in time Petri nets. In FORMATS, volume 8053 of LNCS, 91–105. Springer.

David, A., Larsen, K.G., Legay, A., Nyman, U., Traonouez, L., and Wasowski, A. (2015). Real-time specifications. Int. J. Softw. Tools Techn. Transfer, 17(1), 17–45.

Esparza, J. (2010). A false history of true concurrency: From Petri to tools (invited talk). In SPIN, volume 6349 of LNCS, 180–186. Springer.

Esparza, J. and Heljanko, K. (2008). Unfoldings – A Partial-Order Approach to Model Checking. Monographs Theor. Comput. Sci. Springer.

Fahrenberg, U. and Legay, A. (2015). Partial higher-dimensional automata. In CALCO, volume 35 of LIPIcs, 101–115. Springer.

Fahrenberg, U. (2005). A category of higher-dimensional automata. In FoSSaCS, volume 3441 of LNCS, 187–201. Springer.

Godefroid, P. (1996). Partial-Order Methods for the Verification of Concurrent Systems, volume 1032 of LNCS. Springer.

Goltz, U. and Reisig, W. (1983). The non-sequential behavior of Petri nets. Inf. Cont., 57(2/3), 125–147.

Hanisch, H.M. (1993). Analysis of place/transition nets with timed arcs and its application to batch process control. In ATPN, volume 691 of LNCS, 282–299. Springer.

Hansen, H., Lin, S., Liu, Y., Nguyen, T.K., and Sun, J. (2014). Partial order reduction for timed automata with abstractions. In CAV, volume 8559 of LNCS, 391–406. Springer.

Johansson, K.H., Egerstedt, M., Lygeros, J., and Sastry, S. (1999). On the regularization of zeno hybrid automata. Systems & Control Letters, 38(3), 141 – 150.

Larsen, K.G., Pettersson, P., and Yi, W. (1997). UPPAAL in a nutshell. Int. J. Softw. Tools Techn. Transfer, 1(1-2), 134–152.

Merlin, P.M. and Farber, D.J. (1976). Recoverability of communication protocols–implications of a theoretical study. IEEE Trans. Commun., 24(9), 1036–1043.

Milner, R. (1989). Communication and Concurrency. Prentice Hall.

Nielsen, M., Plotkin, G.D., and Winskel, G. (1981). Petri nets, event structures and domains, part I. Theor. Comput. Sci., 13, 85–108.

Petri, C.A. (1962). Kommunikation mit Automaten. Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2.

Pratt, V.R. (1991). Modeling concurrency with geometry. In POPL, 311–322. ACM Press.

Pratt, V.R. (2000). Higher dimensional automata revisited. Math. Struct. Comput. Sci., 10(4), 525–548.

Sifakis, J. (1977). Use of Petri nets for performance evaluation. In Measuring, Modelling and Evaluating Computer Systems, 75–93. North-Holland.

Sifakis, J. and Yovine, S. (1996). Compositional specification of timed systems. In STACS, volume 1046 of LNCS, 347–359. Springer.

Srba, J. (2008). Comparing the expressiveness of timed automata and timed extensions of Petri nets. In FORMATS, volume 5215 of LNCS, 15–32. Springer.

van Glabbeek, R.J. (1991). Bisimulations for higher dimensional automata. Email message. http://theory.stanford.edu/~rvg/htda.

van Glabbeek, R.J. (2006a). On the expressiveness of higher dimensional automata. Theor. Comput. Sci., 356(3), 265–290. See also (van Glabbeek, 2006b).

van Glabbeek, R.J. (2006b). Erratum to “On the expressiveness of higher dimensional automata”. Theor. Comput. Sci., 368(1-2), 168–194.

van Glabbeek, R.J. and Plotkin, G.D. (1995). Configuration structures. In LICS, 199–209. IEEE Computer Society.

van Glabbeek, R.J. and Plotkin, G.D. (2009). Configuration structures, event structures and Petri nets. Theor. Comput. Sci., 410(41), 4111–4159.

Winskel, G. and Nielsen, M. (1995). Models for concurrency. In Handbook of Logic in Computer Science, volume 4. Clarendon Press, Oxford.