Durations for Truly-Concurrent Transitions

Eric Goubault*

Abstract

In this article we take a rather different view on models for real-time systems. First of all, transitions are not instantaneous. They really bear time changes. Secondly, the model is of geometric inspiration (following the ideas of [24]). It is intuitively clearer than other models in that executions can really be pictured as curves (or "trajectories"). Finally it is based on a model of true concurrency which can express scheduling properties (see [12]). We present the model in a very progressive way, starting from ordinary transition systems, then going through some truly concurrent operational models, to end up with a fully formalized model for real-time systems (with an application to a subset of timed CCS). The model (timed higher-dimensional automata or timed HDA in short) is made into a category where morphisms are simulations. It is shown to have many interesting algebraic (complete, co-complete, cartesian closed, monoidal closed) and computer-scientific properties (the timing laws are given naturally by the categorical combinators). A discussion of important matters such as fairness and Zeno is also provided.

1 Introduction

In [15], real-time models were considered good enough if they were **refinable**, **digitizable**, and **operational**. This means in particular that we should be able to look at a real-time system at different levels of precision (this rules out formalisms depending on a base of time) and that its description should be based on systems of transitions. Timed automata [3], generalizing finite state machines over infinite strings by adding a finite set of real-valued clocks verify these requirements. The same holds for timed transition systems [16] which extend the formalism of transition systems by imposing timing constraints on transitions. In the first model, states are waiting periods for clock constraints to be satisfied and in the second one, transitions are **instantaneous** as well but are due to occur within precise time bounds. This is not a natural view on real-time systems. A transition should really **take time** in the sense that it corresponds to an abstraction of some computation. As a matter of fact, we asked for refinability so we cannot assume actions to be only "elementary" - almost instantaneous - ones. Unfortunately models for real-time concurrent systems having transitions bearing time changes can no longer be based on ordinary transition systems since interleaving of two actions a, b will result in having an execution time equal to the sum of the times a and b take. This obviously ruins all future reasoning and explains why this natural idea has never been formalized up to now (except in some restricted way in [7]). A solution is to follow a **truly concurrent** operational approach. These approaches are discussed in Section 2.1. As more generally scheduling policies of processes onto processors have a direct impact on the measure of time[1], it appears that we need more than that. We need to be able to describe the **level of parallelism**, i.e. the number of busy processors at a given time. Some work has been done on this [14] and is introduced in

*LIENS, École Normale Supérieure, 45 rue d'Ulm, 75230 Paris Cedex 05, FRANCE, email:goubault@dmi.ens.fr

[1] In most work on analysis of real-time languages, a "maximal parallelism" assumption is assumed. This clearly is too rigid when it comes to real machines, and leads to complicated discussions when one wants to change the scheduling policy in a semantics.

Section 2.2. The main idea is to conceive executions as **geometric shapes**. Ordinary transition systems can already be thought of as one-dimensional **trajectories**. Then the asynchronous execution of n actions is a trajectory (or transition) of dimension n. It is fully formalized in Section 2.2. We carry on by realizing these shapes in some euclidean space $\mathbb{R}^n$ (Section 2.4) as a basic step towards having execution time of transitions measured by their **length**. This situation is abstracted in Section 3.1 where the length depends on a norm associated with every transition. We construct a category of models (timed HDA) by defining morphisms to be "simulations" (as in recent work in concurrency, [31]). A correctness criterion with respect to untimed semantics is obtained by forgetting the geometry and the norms (Section 3.3). Fairness (Section 3.2) is also discussed. In Section 3.4, Zeno behaviours are shown to be of a topological nature. Similarly to fairness properties, we propose to give a choice between allowing or not these behaviours. Finally in Section 4, the model is shown to be **natural** in the sense that parallel composition, non-determistic choice, etc. with suitable timing laws are categorical combinators in the category of timed HDA. Moreover the model covers **different paradigms** since synchronized product and function spaces are again *natural constructions*. The category of timed HDA is actually a model for non-commutative intuitionistic **linear logic**. It has then enough categorical properties for being used for **denotational (or categorical) semantics**. A **SOS**-like metalanguage is defined for the operational semanticians. We give an application in Section 5 (timed CCS).

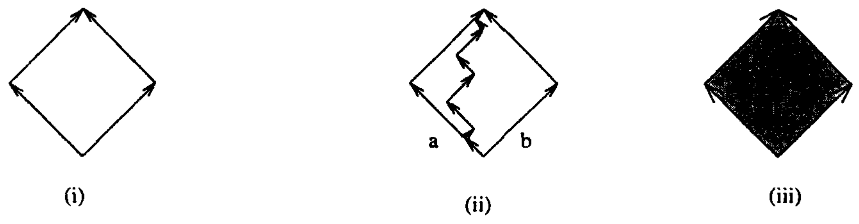
2 Untimed higher-dimensional automata

We begin by presenting a very simple geometric model for true concurrency, based on ideas by Vaughan Pratt and Rob van Glabbeek [24], [29] and formalized in different ways in [11], [14].

2.1 An introduction to HDA

Operational models for concurrency start with (ordinary) transition systems. A transition system is a structure $(S,i,L,Tran)$ where S is a set of states, i is the initial state, L is the set of labels and $Tran \subseteq S \times L \times S$ is the transition relation. This definition has already some geometry in it since we are all used to represent them as arrows (transitions) between states (points or small circles). This does not fulfill the aim we had at the beginning, i.e. it does not provide us a semantics stable by refinement [30] nor it distinguishes non-determinism from truly concurrent (or asynchronous) execution. This should be fixed (as said in the introduction) before using it as a basis for real-time modeling. A possible answer is to decorate the transition systems with some relation prescribing the independence of some actions (or transitions). This can be done in more than one manner; just to mention a few: asynchronous transition systems [6], [27], concurrent automata [28] and transition systems with independence [31]. We comment on the former only, since exhaustivity would be too space consuming. Asynchronous transition systems are equipped with an irreflexive symmetric binary independence relation I on actions verifying a few conditions. The most important is that independence of actions means confluence of the transition relation for the actions involved. This decoration on ordinary transition systems (the independence relation I) is enough to make the distinction between non-determinism and true concurrency. Suitable refinement operators can be defined as well on these structures. There is a slight problem though. The level of parallelism is not defined in a very precise manner. This is due to the fact that the independence relation is only a binary one. Of course, a straightforward generalization would be to replace the binary relation I by an n-ary relation. This could be done (though we do not have any pointers in the literature) but the generalization to real-time concurrent systems seems too heavy work (how to measure time for asynchronous executions?). This can be tackled if we get back to our geometric intuition. Things have been made overly unnatural by adding an object (the independence relation) which is not of the same nature as transitions and states. Just think of $a|b$ as an

Figure 1: Non-determinism (i) versus overlap in time (ii) abstracted by a transition of dimension 2 (iii).



abstraction of all possible asynchronous executions of a and b. As in [24], this can be pictured as the filled-in square of the right-hand side of Figure 1, distinguishing it in a striking manner with the interleaving at the left-hand side of the same figure. Notice that geometrically, the interior of the square consists of the union of all paths where executions of a and b overlap "in time" (middle picture of Figure 1). Time already makes its way into the model, though not quantified yet. As a direct generalization, asynchronous execution of n transitions give rise to hypercubes of dimension n, called n-transitions (ordinary transitions are 1-transitions, states are 0-transitions). Interestingly enough, all this has a very neat algebraic formulation.

2.2 Formalization

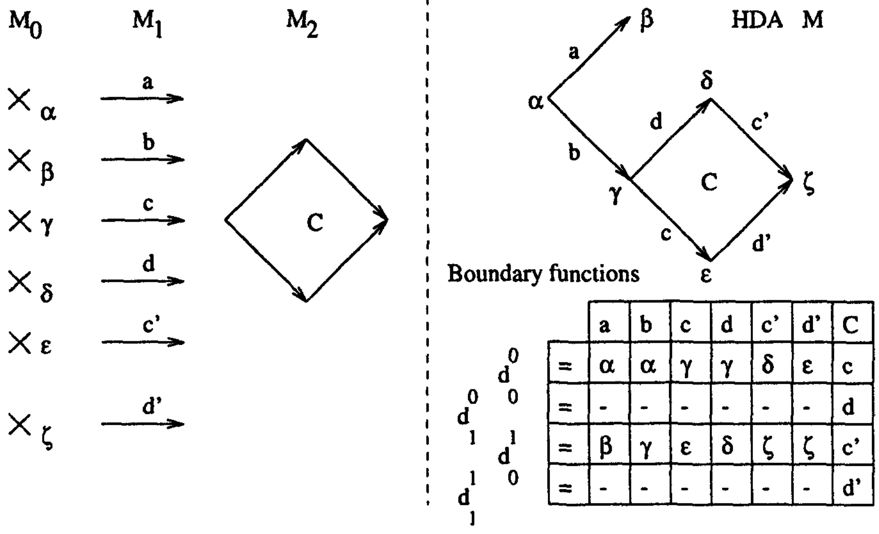
We present the geometric shapes we are interested in as unions of points, segments, squares, etc. hypercubes, i.e., as collections of n-transitions ($n \in \mathbb{N}$). We glue them together by means of boundary relations (see Figure 2), given by two boundary operators: d^0, the start boundary operator and d^1 the end boundary operator. They generalize the source and target functions for ordinary automata.

Consider the square, $$\begin{tikzcd}(0,0) \arrow[r, "a"] \arrow[d, "b"'] & (0,1) \arrow[d, "b'"] \\ (1,0) \arrow[r, "a'"] \arrow[ur, phantom, "A"] & (1,1)\end{tikzcd}$$. This corresponds to the asynchronous execution of actions a and b (a' and b' are copies of transitions of label a and b respectively). The object of dimension 2 "interior of the square" A should certainly have two source boundaries, up to the order on $\{a,b\}$, $d_0^0(A) = a$ and $d_1^0(A) = b$ since from state $(0,0)$ we can fire a and b. Similarly, it should have two target boundary operators $d_0^1(A) = a'$ and $d_1^1(A) = b'$ since from the parallel execution of a and b (represented by A) we can end first action a (giving "residue" b') or action b (giving "residue" a'). We will see that again when speaking about paths. Notice that with this ordering on vertices, we have, $d_0^0(d_1^0(A)) = (0,0) = d_0^0(d_0^0(A))$ and $d_0^1(d_1^0(A)) = (1,0) = d_0^0(d_0^1(A))$. We can show that for any hypercube of dimension n, we can choose an ordering on vertices, squares etc. such that the $2 * n$ boundary operators verify the commutation rules[2], $d_i^k \circ d_j^l = d_{j-1}^l \circ d_i^k$ for $k = 0, 1$, $l = 0, 1$ and $i < j$ ($\circ$ is the ordinary composition of functions). Now we can glue these elementary shapes in order to get HDA. This is exemplified in Figure 2. We verify on the example the commutation rule between the source and target boundary operators d^0 and d^1 respectively. We can then introduce these formally under the name of unlabeled semi-regular HDA.

Definition 1 *A semi-regular HDA is a collection of sets M_n ($n \in \mathbb{N}$) together with functions $M_n \overset{d_i^0}{\underset{d_j^1}{\rightrightarrows}} M_{n-1}$ for all $n \in \mathbb{N}$ and $0 \leq i, j \leq n-1$, such that $d_i^k \circ d_j^l = d_{j-1}^l \circ d_i^k$ ($i < j, k, l = 0, 1$) and $\forall n, m\ n \neq m$, $M_n \cap M_m = \emptyset$.*

[2] Very much alike the ones we have for simplicial complexes. Ideas of many constructions of the article actually come from combinatorial algebraic topology.

Figure 2: Glueing of elementary shapes to get a semi-regular HDA.



	a	b	c	d	c'	d'	C
d^0_0 =	α	α	γ	γ	δ	ε	c
d^0_1 =	-	-	-	-	-	-	d
d^1_0 =	β	γ	ε	δ	ζ	ζ	c'
d^1_1 =	-	-	-	-	-	-	d'

Elements x of M_n ($dim\ x = n$) are called n-transitions (or states if $n = 0$). In order to be able to study "natural" constructions on HDA, we define a notion of **morphism** between them. As customary in recent work in concurrency [31], morphisms look like **simulations**. In geometrical terms, morphisms preserve shapes (every n-transition is mapped onto a n-transition), time and orientation.

Definition 2 *Let M and N be two semi-regular HDA, and f a family $f_n : M_n \to N_n$ of functions. f is a morphism of semi-regular HDA if and only if $f_n \circ d^0_i = d^0_i \circ f_{n+1}$ and $f_n \circ d^1_i = d^1_i \circ f_{n+1}$ for all $n \in \mathbb{N}$, $0 \leq i \leq n$.*

This defines the **category Υ_{sr}** of semi-regular HDA.

Now, traces of execution are described as sequences of states and transitions satisfying certain properties. A **path** is to be understood as a sequence of *allocation* (case (*ii*) below) of one action at a time on a new processor or *deallocation* (case (*i*) below) of one action at a time (i.e. its execution has ended on a given processor). An example of a path in an automaton M is given in Figure 3 together with its inclusion morphism into M (M simulates all of his paths).

Definition 3 *A path in a semi-regular HDA M is $p = (p_0, \ldots, p_n)$ such that p_0 and p_n are states and $\forall k, 0 < k \leq n$, $\exists j$, $p_k = d^1_j(p_{k-1})$ (i) or, $p_k = d^0_j(p_{k+1})$ (ii)*

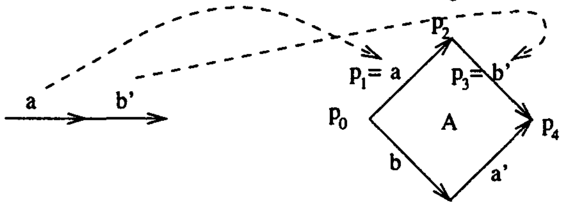
The definition of paths explains why the morphisms are (higher-dimensional) simulations. The commutation with the start boundary operator d^0 for example can be seen as asserting: "whenever M fires a new action, N fires a similar one".

Properties of the category of semi-regular HDA will be seen as a special case of those of timed HDA in Section 4.

2.3 From the untimed to the timed world

In the formalization, we have forgotten the geometry. Let us have it back. As a matter of fact, in order to introduce time into the model we already have, we are going to represent transitions

Figure 3: A path and its inclusion morphism in a semi-regular HDA.



as real continuous geometric objects. Continuous geometry is good for measuring time: the principle here is to have time measured by the **length** of transitions (or paths). Traces are then real **trajectories** as in mechanics. This is close to intuition, contrarily to most approaches (see [22] for an excellent overview of these approaches), **transitions take time**[3]. Being interested by program analysis, where transitions are in fact abstractions of some complex process, this approach is very natural. In particular refinement comes then for free (look at Figure 6 for an easy example).

Recovering the geometry will be done in the same style as the geometric realization functor between simplicial sets and CW-complexes (see for instance [20] or [10]). We associate with every n-transition x a unit cube of dimension n in $\mathbb{R}^n$, $\square_n = \{(t_0, \ldots, t_n)/\forall i, 0 \leq t_i \leq 1\}$. Then, similarly to the process seen in Figure 2, we glue these cubes together according to the values of the boundary functions. In order to do this, we need to define functions characterizing the boundaries of these unit cubes in $\mathbb{R}^n$. Let δ_i^k, $0 \leq i \leq n$, be the continuous functions ($n > 0$) from $\square_{n-1}$ to $\square_n$ with $\delta_i^k(t_0, \ldots, t_{n-1}) = (t_0, \ldots, t_{i-1}, k, t_i, \ldots, t_{n-1})$. They describe how the boundaries of a cube can be included into it. Then $\delta_i^k \circ \delta_j^l = \delta_{j+1}^l \circ \delta_i^k$, $(i \leq j)$. Consider now, for a semi-regular HDA M, the set $R(M) = \bigcup_{n, x \in M_n} (x, \square_n)$. Each $(x, \square_n)$ inherits a topology given by the standard one on $\mathbb{R}^{n+1}$, thus $R(M)$ is a topological space with the disjoint sum topology. Let $\equiv$ be the equivalence relation (the "glueing" relation) induced by the identities: $\forall k, i$, $x \in M_{n+1}$, $t \in \square_n$, $n \geq 0$, $(d_i^k(x), t) \equiv (x, \delta_i^k(t))$. Let $\mid M \mid = R(M)/\equiv$. It has a structure of topological space induced by $R(M)$. $\mid M \mid$ is called the **geometric realization** of M. It is easy to make this construction into a functor from Υ_{sr} to **Top**, the category of topological spaces with continuous maps. As observed in [10], we can actually work in *Ke* the full subcategory of *Kelley spaces* (i.e. compactly generated topological spaces[4], [1]) instead of the entire category *Top*. The geometric realization functor has then fairly nice properties. When taken in value in *Ke* it commutes with finite inverse limits and all colimits. All this gives us a hint about how to define timed higher-dimensional automata. A first step towards a general definition is given in next section.

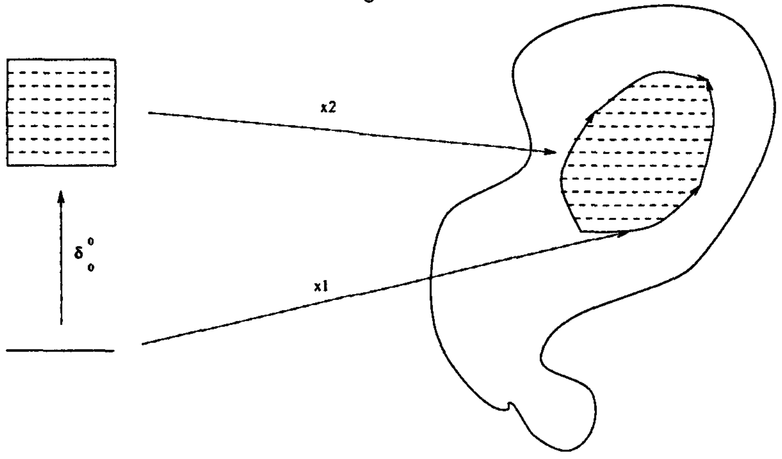
2.4 Timing a semi-regular HDA

Let M be a semi-regular HDA. The standard way in mathematics to measure the length (time) of transitions in $\mid M \mid$ is to have a norm $\|.\|_x$ on the tangent space at every $x \in M$ of the shapes we have. Then the length of a transition a is the integral of the speed $\|\frac{d\gamma(t)}{dt}\|_{\gamma(t)}$ for a parametrization γ of a (it does not depend on the parametrization chosen). $\mid M \mid$ has a well known differential structure. On every transition of dimension n, we put the norm $\|u_1, \ldots, u_n\|_{x_1, \ldots, x_n} = max\{\mid u_1 \mid, \ldots, \mid u_n \mid\}$. The norm chosen corresponds to giving all 1-transitions the unity duration and to have that when n processes run asynchronously, the time to complete them is the maximum of the times necessary to complete each of them,

[3]There have been some semantics in which transitions take time in some way [2, 19], but they relied on coding this explicitly in the term language. Our approach is language independent.

[4]A Kelley space is a Hausdorff topological space X such that a subset F of X is closed whenever its intersection with each compact subspace of X is closed. In particular, locally compact spaces are Kelley spaces.

Figure 4: Deformed cubes



as already observed in [24]. This corresponds to our view of independent processes running asynchronously. For instance, in Figure 3, the geometric realization of the path is of length 2. The fully synchronous execution in the automaton at the right-hand side (the diagonal of the square from the starting point to the end) is of length 1. This view to timed HDA, if encouraging, is not yet satisfactory. We have a very **rigid** notion of time in the sense that the norm has to be chosen uniformly for all transitions. In general, the interaction between actions makes the norm (or "local cost of computation") vary as one of the actions takes over the others (i.e. as we approach the boundaries, or interleavings). We only have to abstract away from a so concrete representation in order to get what we need. Look for instance at the classical billiard example (Figure 6): the representation of transitions has been chosen in order to fit to the trajectory of the ball; states are then the coordinates of the point representing them. The picture will become more general in next section.

3 Timed higher-dimensional automata

3.1 Basic definitions

First of all, we need a geometric shape X to define a timed HDA, i.e. we need a topological space. There are many kinds of topological spaces. We have seen that timing semi-regular HDA only requires Kelley spaces[5]. A good point is that they have very good algebraic properties: they form a complete and cocomplete cartesian closed subcategory of **Top** [1].

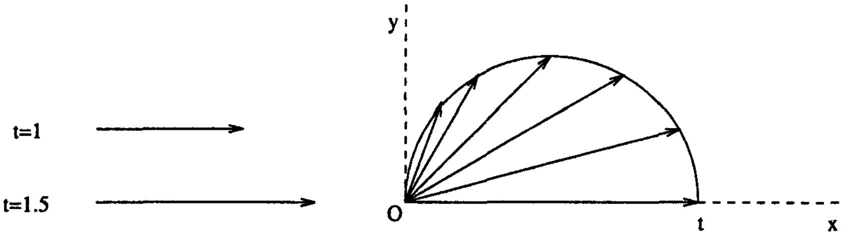
Then we have to give a differential structure on X to be able to measure time. This is difficult to do so in full generality. In particular, when it comes to algebraic properties, differential manifolds are difficult to handle[6]. We therefore choose to present here a very particular mathematical object, in which the differential structure is given by the transitions. Thus we have to look at transitions now. Intuitively they should be sort of **deformed cubes** (see Figure 4). This leads us to define them as almost inclusion functions, i.e. as continuous functions $x : \square_n \rightarrow X$ (called singular cubes[7]). They are required to be continuously deformed cubes

[5] This is purely a technical point for having good categorical constructions. It does not affect in any manner the geometric intuition underlying the model we are presenting.

[6] Quotients, function spaces are hard work (they need submersion theorems and infinite dimensional differential geometry respectively).

[7] By analogy with singular simplices, [20].

Figure 5: Delay transitions (left) and timeout HDA (right).



only in their interior since we may want to identify some of their boundaries to get cyclic shapes. This is formalized by saying that all singular cubes $x : \Box_n \rightarrow X$ induce homeomorphisms from[8] $\mathring{\Box}_n$ to their images[9]. Moreover, we want X to be covered by all its transitions, i.e. we impose $\{x(\mathring{\Box}_n)/n \in \mathbb{N}, x \in X_n\}$ to partition X, i.e. X is the disjoint union $\bigcup_{n\in\mathbb{N},x\in X_n} x(\mathring{\Box}_n)$. We should be able to take boundaries, i.e. the collection of singular cubes should be stable by composition with δ_j^ϵ (by Section 2.3). Finally, on every tangent space $T_xX =_{def} T_xu(\mathring{\Box}_n)$ (where $x \in u(\mathring{\Box}_n)$, $u \in X_n$) of X at $x \in X$ we have a norm $||.||_x$ such that $F(x,\dot{x}) = ||\dot{x}||_x$ is a continuous function[10]. The norm can be seen as an **infinitesimal cost** for the computation at some point. To sum up things,

Definition 4 *A (unlabeled) timed HDA is a Kelley space X together with a presentation of X by singular cubes. This means that we have sets X_n containing singular cubes $x : \Box_n \rightarrow X$ stable by composition with δ_j^ϵ ($\epsilon = 0,1$, $0 \leq j \leq n-1$). Moreover we impose the following conditions on X*[11]*,*

- $\{x(\mathring{\Box}_n)/n \in \mathbb{N}, x \in X_n\}$ *partition X,*
- *all singular cubes $x : \Box_n \rightarrow X$ induce homeomorphisms from $\mathring{\Box}_n$ to its image.*
- *X is given a family of norms $||.||_x$ on every tangent space $T_xX = \{(x,\dot{x})\}$ (where $x \in u(\mathring{\Box}_n)$, $u \in X_n$) of X such that $F(x,\dot{x}) = ||\dot{x}||_x$ is a continuous function*

Example 1 *(see Figure 5)*

- *Let X_t ($t \in \mathbb{R}$) be the timed HDA generated by the unique 1-transition $\lambda x.tx : \Box_1 \rightarrow t\Box_1 = \{0 \leq x_1 \leq t\}$. $t\Box_1$ is equipped with the norm $||\dot{x}||_x = |\dot{x}|$. We will see that it is a delay transition of duration t (similar to the δ_t operator of timed CCS [21]).*
- *Define T_t to be the upper half circle of diameter t centered at coordinates $(\frac{t}{2},0)$ in the plane $\mathbb{R}^2$ (with its standard basis). It is given the structure of a timed HDA with the norm induced by the euclidean one in $\mathbb{R}^2$, and with the covering of 1-transitions (for $\theta \in [0,\pi/2]$) $x_\theta : \Box_1 \rightarrow T_t$, $x_\theta(u) = (tucos^2(\theta), tusin(\theta)cos(\theta))$. We will see that it allows us to represent a timeout operator (t is the maximum waiting time).*

When X is a timed HDA, it is easy to see that the collection of sets X_n defines a semi-regular HDA. We define in a similar manner morphisms of timed HDA,

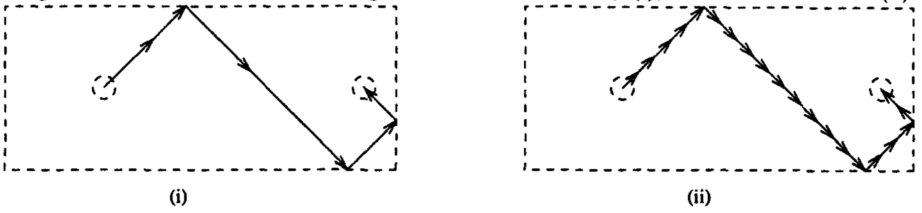
[8] $\mathring{\Box}_n$ denotes the topological interior of $\Box_n$ i.e. $\mathring{\Box}_n = \{0 < t_i < 1\}$, $n \geq 1$ and $\mathring{\Box}_0 = \{0\}$.

[9] Therefore the singular cubes give a (trivial !) structure of manifold to all the $x(\mathring{\Box}_n)$.

[10] If F is at least C^3 then this defines a Finsler space ([26]). Recall that a norm F verifies the properties, $\forall k \in \mathbb{R}$, $F(x,k\dot{x}) = |k| F(x,\dot{x})$, $F(x,\dot{x}) \geq 0$ and $F(x,\dot{x}) = 0$ if and only if $\dot{x} = 0$, and $\forall x$, $\dot{x}$ and $\dot{x}'$, $F(x,\dot{x}+\dot{x}') \leq F(x,\dot{x}) + F(x,\dot{x}')$.

[11] Which make it into a combinatorial cell complex in the terminology of [18].

Figure 6: A timed HDA representing a billiard ball trajectory (i), and a refined version (ii).



Definition 5 *Let X and Y be timed HDA. A continuous function $f : X \to Y$ is a morphism of timed HDA if and only if,*

(i) *for all n-transition $x \in X_n$, $y = f \circ x$ is a n-transition (i.e. $y \in Y_n$).*

(ii) *f commutes with all the boundary operators.*

Actually, since we are in a very special case, (i) implies that f is differentiable on every manifold $x(\mathring{\square}_n)$ since f is then the identity function in the local coordinates, thus a C^∞ diffeomorphism. (ii) can be seen to be partly redundant as well. We write $\mathcal{TT}$ for the category of timed HDA. Notice that no requirement has been made on the way morphims behave with respect to time. Choices are not so easy for "computer-scientific" reasons as well as for "technical reasons"[12]. Nevertheless, we will consider two subcategories of $\mathcal{TT}$, $\mathcal{TT}_=$ whose objects are timed HDA and whose morphisms $f : X \to Y$ preserve time (are *isometries*), i.e. $||df(u).\dot{u}||^Y_{f(u)} = ||\dot{u}||^X_u$ (where df is the differential of f), and $\mathcal{TT}_\leq$ whose objects are timed HDA and whose morphisms f contract time, i.e. $||df(u).\dot{u}||^Y_{f(u)} \leq ||\dot{u}||^X_u$[13].

Now, labeled timed HDA are morphisms $l : P \to L$, where L is the "labelling" automaton. l is nothing but the canonical projection associated with the equivalence "having the same label as" on transitions. If we are in $\mathcal{TT}_=$ or $\mathcal{TT}_\leq$ then the labelling prescribed the allowed durations of transitions.

Timed HDA are in particular semi-regular HDA with boundary operators $d^k_l(x) = x \circ \delta^k_l$ (where x is a n-transition $x : \square_n \to X$). As such we know what a path in it is (we may add in particular initial and final states to timed HDA). But it is not clear though how to decide how much time a transition may take. To answer this question we define "virtual paths" in a timed HDA X as being particular curves on X which paths are in some way abstractions of. Basically, they are continuous functions $\gamma : [0, \infty[\to X$ such that there exist open intervals $I_k =]\alpha_k, \alpha_{k+1}[$, n_k-transitions x^k, $k = 0, \ldots, m-1$(or ∞) with $\bigcup\limits_k I_k = [0,1]\backslash\{\alpha_i\}$ (disjoint union) and $\gamma_{|I_k} : I_k \to x^k(\mathring{\square}_{n_k})$. Moreover, $\gamma_{|I_k}$ must be a differentiable function (I_k has the standard differentiable structure of $\mathbb{R}$) and $(x^k)^{-1}_i \circ \gamma_{|I_k}$ $(0 \leq i \leq n_k)$ should be increasing maps. The set of virtual paths from a point u to a point v is denoted by $\mathcal{V}(u,v)$. To determine the time that a path takes from its initial to its final point we use the metric generated by the norm on TX[14].

Definition 6 *(see [26]) The distance (or time) inf between two points u and v in X is defined to be[15] (with value in $\mathbb{R} \cup \infty$), $T^X_i(u,v) = inf_{\gamma \in \mathcal{V}(u,v)} \int_0^\infty ||\frac{d\gamma}{dt}(t)||_{\gamma(t)} dt$. We have also the distance (or time) sup between two points (with value in $\mathbb{R} \cup \infty$), $T^X_s(u,v) = sup_{\gamma \in \mathcal{V}(u,v)} \int_0^\infty ||\frac{d\gamma}{dt}(t)||_{\gamma(t)} dt$.*

[12] Categories of metric spaces, hence categories with norms, do not have very good algebraic properties in general. One must be careful when defining morphisms !

[13] Called conservative or non-expansive functions in categories of generalized metric spaces.

[14] This is very close to the intuition behind the metric spaces models for real-time of [25].

[15] Where the integrals are in fact the sum of the integrals on the open intervals I_k

T_i^X defines actually a distance function thus a metric on X (generalized in the sense that it may take infinite values).

In $T\Upsilon_=$, automata are simulated exactly in the same time (i.e. all virtual paths and their images have the same length). In $T\Upsilon_\leq$, we allow to simulate by **faster automata**. This is a sensible notion of simulation since programs can only be safely implemented on faster machines than needed[16].

Example 2 *A simple computation shows now that X_t (example 1) has length t, i.e. has execution time t. For T_t, the 1-transitions x_θ have execution time from 0 to t. The transition x_0 leads to the escape sequence, all the other ones lead to the normal ending of the program. Finally, a hypercube of dimension n timed as in Section 2.4 has maximal execution time n (all interleavings) and minimal execution time 1 (synchronous execution of the n 1-transitions, i.e. the diagonal of the hypercube).*

Similarly to the untimed case, we can defined **labeled** timed HDA to be unlabeled timed HDA plus a labeling morphism in $T\Upsilon$. **Timed higher-dimensional transition systems** are labeled timed HDA together with an initial state.

3.2 Fairness

Notice that we can easily define a time **local to a processor**. We can take for granted that in $\mid M \mid$ the length of the projection of a path γ on the ith coordinate is the cpu time of the ith processor on γ. More generally, we suppose that $\dot{x}_i(\frac{d\gamma}{dt})$[17] is the infinitesimal cost of computation on processor i. **Quantitative weak fairness** is expressed as a property of the norm: all processors must be used for some time on every (fair) paths, i.e. $\|(0,\ldots,\dot{x}_i(\frac{d\gamma}{dt}),0,\ldots,0)\|$ should be strictly positive function of time. **Quantitative strong fairness** is a weaker property on the norm: whenever the global time diverges, the local times of every processor must diverge as well.

3.3 Correctness Timed/Untimed

Similarly to work in program analysis, we can define a way to go from the timed to the untimed world and then back to the timed one which has special properties. It is done in general [8] by means of Galois connections which ensure that an analysis (or a non-standard semantics) is **correct** with respect to a semantics. Being in a completely categorical framework, the right mathematical tool is then pairs of adjoint functors. We actually have here a right-adjoint to the functor $\mid \cdot \mid$, $F : T\Upsilon \rightarrow \Upsilon_{sr}$ defined by $F(X) = (X_n)_n$ (F forgets time). Moreover, the units and counits of the adjunction are isometries. This entails that this adjunction restricts to adjunctions between $T\Upsilon_=$ and Υ_{sr}, and $T\Upsilon_\leq$ and Υ_{sr} respectively. Having simulations as morphisms in these categories, this shows that **simulation properties** (and bisimulation ones in particular) in the timed world are **correct** with respect to the corresponding ones in the untimed world.

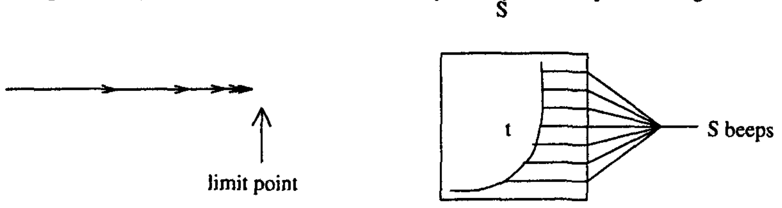
3.4 Zeno behaviours

Let γ be an infinite virtual path. If $\gamma([0,\infty[)$ is compact, then there is a limit point a in the sequence $(\gamma(\alpha_k))_k$. Therefore, even if time always increases by strictly positive steps , there may be a (sub)path in which time "**slows down**" up to some point. This is exemplified by the Zeno paradox (which can be explicitly given a timed HDA representation, Figure 7) in which a door is seen to be closed through observations of the type "it is closed half way from the

[16]There exist properties that are not preserved when going on a faster machine (see [7]), but this goes beyond the scope of this article.

[17]Where $\dot{x}_i$ denotes the ith coordinate in the tangent space.

Figure 7: Typical Zeno behaviour and a hybrid system implementing it.



A timed HDA that could represent this behaviour is X with,

- $X = [0, 1]$,
- $X_0 = \{x_i / i \in \mathbb{N}, x_i = 1 - \frac{1}{2^i}\}$,
- $X_1 = \{[x_i, x_{i+1}] / i \in \mathbb{N}\}$ and the obvious boundary operators,
- the norm is induced by the euclidean norm on $[0, 1]$.

end". The time it needs to be closed is finite, the number of allowed obervations (transitions) is infinite. No lower bound whatsoever is imposed on the time of transitions. This precisely creates the paradox.

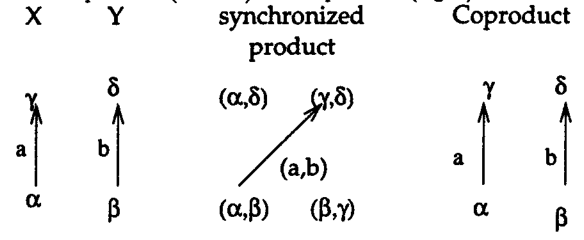
There are easy ways to prevent Zeno paradoxes to occur in a timed HDA X. As they happen when there exist some limit points, it suffices to prevent them to crop up. A sufficient condition is to have a **lower bound** on the time transitions take (even locally), very much alike the "finite-variability" condition of [5]. Why not put this condition in the model from the very beginning? We argue that for hybrid systems (or even just ordinary real-time systems like in [21]), it may be interesting to consider Zeno paradoxes as well. Suppose we have a system S in which the temperature t diverges in finite time (grows at an exponential rate in practise). Suppose also that S is equipped with a measuring apparatus which beeps every time the temperature grows of one degree Farenheit. We model S by a timed HDA in which the states represent the number of times S has beeped (i.e. the temperature of S minus its initial value) and the 1-transitions are the delay transitions from one state to the next. Then it implements a Zeno behaviour: no strictly positive lower bound can be given to the time of execution of any transition. As we do not know the **precision** at which time can be measured, we cannot eliminate this Zeno behaviour when studying the system S.

4 Timed higher-dimensional automata as denotational and operational models

In this section, we show that $T\Upsilon$ is a complete and co-complete cartesian closed, monoidal closed category similarly to Υ_{sr}. Some constructions will be exemplified in both categories. $T\Upsilon_{\leq}$ is shown to be a complete and co-complete monoidal category. $T\Upsilon_{=}$ has only filtered limits and colimits and a tensor product. As customary since [31], categorical combinators will be recognized to be **timed-process-algebra sort of combinators** (as those of [21]). In order to see this, we introduce a **SOS-like metalanguage** which gives an operational view to the constructions.

The idea is to write n-transitions a of some timed HDA X as arrows $s \xrightarrow[{[t_1 t_2]}]{a} s'$ where s and s' are the beginning state (i.e. the beginning state of a beginning 1-transition of ... a beginning $(n-1)$-transition of a) and end state of a respectively, and t_1 is the minimal execution time, t_2 the maximum execution time of a (t_2 may be ∞ as we are working in $\mathbb{R} \cup \{\infty\}$. More

Figure 8: Synchronized product (middle) and coproduct (right) of two transitions (left).



formally, we define an entailment relation $\models$ to relate X to its transitions, and we write, $X \models s \xrightarrow[{[t_1,t_2]}]{a} s' \Leftrightarrow \begin{cases} d_0^0 d_1^0 \dots d_{n-1}^0 a &= s, \\ d_0^1 d_1^1 \dots d_{n-1}^1 a &= s', \\ T_i^{x(\square_n)}(s,s') &= t_1, \\ T_s^{x(\square_n)}(s,s') &= t_2 \end{cases}$. Sometimes, we specify the dimension n of the n-transition a by adding $dim\ a = n$.

Let X and Y be two timed HDA. Then their **cartesian product** is the timed HDA Z described operationally by the rule, $\dfrac{X \models u \xrightarrow[{[\alpha_1,\alpha_2]}]{t} v \quad X' \models u' \xrightarrow[{[\alpha'_1,\alpha'_2]}]{t'} v'}{X \times X' \models (u,u') \xrightarrow[{[max(\alpha_1,\alpha'_1), max(\alpha_2,\alpha'_2)]}]{(t,t')} (v,v')}$ and $dim\ t = dim\ t' = dim\ (t,t')$

This shows that this is really a **synchronized product** (see Figure 8) of the two automata X and Y. More formally, it is defined as

- $Z_n = \{z : \square_n \xrightarrow{\Delta} \square_n \times \square_n \xrightarrow{x \times y} X \times Y / x \in X_n, y \in Y_n\}$ where Δ is the diagonal $\Delta(x) = (x,x)$,
- $Z = \bigcup_{n \in \mathbb{N}, z \in Z_n} z(\square_n) \subseteq X \times Y$,
- $\|\dot{x}, \dot{y}\|_{(x,y)} = max(\|\dot{x}\|_x, \|\dot{y}\|_y)$.

The projections here are not isometries in general, but they are contracting maps, i.e. $T_i^X(p_1(u), p_1(v)) \leq T_i^Z(u,v)$. Notice that the norm (and then the timing laws above) is given by the categorical construction and is by no means arbitrary.

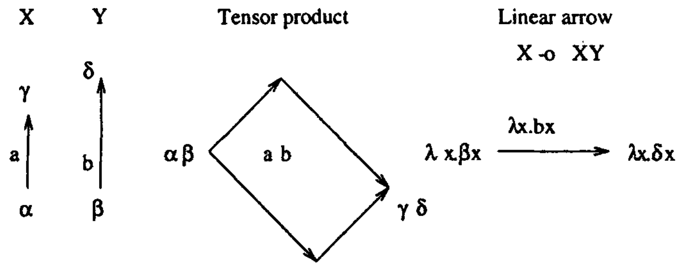
The union (or coproduct) of two timed HDA X and Y is described operationally by the two rules,

$$\frac{X \models u \xrightarrow[{[\alpha_1,\alpha_2]}]{t} v}{X \cup X' \models u \xrightarrow[{[\alpha_1,\alpha_2]}]{t} v} \qquad \frac{X' \models u' \xrightarrow[{[\alpha'_1,\alpha'_2]}]{t'} v'}{X \cup X' \models u' \xrightarrow[{[\alpha'_1,\alpha'_2]}]{t'} v'}$$

We recognize a rule for **non-deterministic choice** (see Figure 8) as in timed CCS with the operator $+$. Again, the intuitively clear timing laws are given directly by the structure of the model.

The **parallel composition with no interference** can be defined operationally by the rule

Figure 9: Parallel composition (middle) of two transitions (left) and linear function space (right).



(see Figure 9)

$$\frac{X \models u \xrightarrow[{[\alpha_1,\alpha_2]}]{t} v \quad X' \models u' \xrightarrow[{[\alpha'_1,\alpha'_2]}]{t'} v'}{X \otimes X' \models u \otimes u' \xrightarrow[{[max(\alpha_1,\alpha'_1),\alpha_2+\alpha'_2]}]{t\otimes t'} v \otimes v'}$$

and $dim\ t \otimes t' = dim\ t + dim\ t'$ More formally, it is a **tensor product** $Z = X \otimes Y$ defined by,

- $Z = X \times Y$,
- $Z_n = \{z : \Box_n \cong \Box_k \times \Box_{n-k} \xrightarrow{x \times y} Z / x \in X_k, y \in Y_{n-k}\}$
- $\|\dot{x}, \dot{y}\|_{(x,y)} = max(\|\dot{x}\|_x, \|\dot{y}\|_y)$.

Now $Z = X \multimap Y$, such that $X \multimap .$ is the right-adjoint to $. \otimes X$, is,

- $Z_n = \{z : \Box_n \to (X \to Y) / z' : \Box_n \otimes X \to Y, z'(u,x) = z(u)(x)$ is a morphism$\}$, where $\Box_n$ is considered as the timed HDA with the unique n-transition $Id : \Box_n \to \Box_n$,
- $Z = \bigcup_{n\in\mathbb{N}, z\in Z_n} z(\Box_n) \subseteq (X \to Y)$ endowed with the compact-open topology [1],
- for $f \in Z$, $\|\dot{f}\|_f = sup_{x\in X, \|\dot{x}\|_x^X = 1} \|\dot{f}(x)\dot{x}\|^Y_{f(x)}$.

We conjecture that operationally[18], $\dfrac{X \models u \xrightarrow[{[\alpha_1,\alpha_2]}]{t} v \quad X \multimap X' \models u' \xrightarrow[{[\alpha'_1,\alpha'_2]}]{t'} v'}{X' \models u'(u) \xrightarrow[{[max(\alpha_1,\alpha'_1),\alpha_2+\alpha'_2]}]{t'(t)} v'(v)}$. In $X \multimap X'$ we have functions which **fork** new actions (dynamically) as $\lambda x.b \otimes x$ in Figure 9. The argument of these functions may be computed **in parallel** with the body of the function.

5 Semantics of a toy language

We consider the following language (a subset of RTCCS, [17]),

$$\begin{array}{rcllll} P & ::= & a^j;P & \mid \mathbf{nil} & \mid P+P \\ & \mid & P\|P & \mid (t)P & \mid \mathbf{rec}\ x.P[x] \end{array}$$

The atomic actions a^j are supposed to take unit time. $(t)P$ can behave like P after time t. For the sake of simplicity, we do not consider synchronization here. We refer the reader to [13] for details.

[18]The untimed part is easy to verify though.

5.1 Semantic domains

As in [11] we want to give to terms of the language denotations which are higher-dimensional transition systems. To this end, we want to define a huge timed HDA D (called *domain*) which will contain all possible operational behaviours of terms of the language. Elements of the domain, and thus denotations, will be sub-timed-HDA of D (i.e. inclusion morphisms into D) as in the untimed case.

All this is most conveniently done by recursive domain definitions (see [23]). As a matter of fact, we generally want a domain to contain a few specified actions and to be closed under such constructs as the parallel composition (the tensor product). $\coprod$ (and then amalgamated sums – i.e. pushouts – $+$), $\times$ and $\otimes$ are covariant functors commuting with colimits, therefore standard results [4] guarantee the existence of solutions to recursive equations like $D \cong U + D \otimes D$ (U is a given timed HDA) which is precisely a timed HDA closed under parallel composition. More complex constructions can be done (similar to the homotopical constructions of [11]), for instance to give domains for imperative languages where states are mappings from variables to actual values but we will not need them here.

5.1.1 Denotational semantics

We first define semi-regular HDA (α_i^j) and (a^j), to be the following HDA (geometrically),

$$\begin{tikzcd}(\alpha_i^j): \; 1 \arrow[r, "\alpha_i^j"] & \alpha_i^j\end{tikzcd} \qquad , \; \begin{tikzcd}(a^j): 1 \arrow[loop above, "a^j"]\end{tikzcd}$$

Let P and K be the domains given by the recursive equations,

$$P \cong (|\, \alpha_i^j \,|)_{i,j} + \sum_{t \in \mathbb{R}^+} X_t + P \otimes P$$

$$K \cong (|\, a^j \,|) + K \otimes K$$

where $|\cdot|$ is the geometric realization functor of Section 2.3, and the norm is induced by the Euclidean norm in $\mathbb{R}^n$.

Let $l : D \rightarrow L$ the morphism of HDA defined by,

- $\forall i \in \mathbb{N}$, $l(\alpha_i^j) = a^j$
- $\forall x, y \in P$, $l(x \otimes y) = l(x) \otimes l(y)$

It lifts easily to $k = |\, l \,| : P \rightarrow K$, making P into a labeled timed HDA.

The domain of HDA in which we give the semantics of the language is $k : P \rightarrow K$. We can actually give it in $D = P$, and recover the full definition by applying the labelling l. Then,

- $[\![\mathbf{nil}]\!] = (1)$
- $[\![p + q]\!] = [\![p]\!] \coprod [\![q]\!]$ ($\coprod$ is the coproduct in $T\Upsilon/K$, it corresponds to an amalgamated sum in $T\Upsilon$)
- $[\![a^j; q]\!] = (|\, a_i \,|) \coprod \left(\alpha_i^j \otimes [\![q]\!]\right)$ for some fresh i
- $[\![(t).p]\!] = X_t \coprod (t \otimes [\![p]\!])$
- $[\![p || q]\!] = [\![p]\!] \otimes [\![q]\!]$
- $[\![\mathbf{rec}\ x.p[x]]\!] = \underrightarrow{lim}\ [\![p^i[\mathbf{nil}]]\!]$ where the direct limit is taken on the full subcategory of $T\Upsilon/K$ whose objects are the $[\![p^i[\mathbf{nil}]]\!]$

5.2 Operational semantics

Its operational semantics is then (by results of Section 4),

$$\frac{}{\mathbf{nil} \models 1 \xrightarrow[{[0,0]}]{1} 1} \qquad \frac{P \models s \xrightarrow[{[\alpha_1,\alpha_2]}]{u} s'}{(t).P \models t \otimes s \xrightarrow[{[t+\alpha_1, t+\alpha_2]}]{u} t \otimes s'}$$

$$\frac{}{a^j; P \models 1 \xrightarrow[{[1,1]}]{a} \alpha^j_i} \qquad \frac{P \models s \xrightarrow[{[\alpha_1,\alpha_2]}]{u} s'}{a^j; P \models \alpha^j_i \otimes s \xrightarrow[{[\alpha_1,\alpha_2]}]{u} \alpha^j_i \otimes s'}$$

$$\frac{Q \models s \xrightarrow[{[\alpha_1,\alpha_2]}]{a} t}{Q + Q' \models s \xrightarrow[{[\alpha_1,\alpha_2]}]{a} t} \qquad \frac{Q' \models s' \xrightarrow[{[\alpha'_1,\alpha'_2]}]{a'} t'}{Q + Q' \models s' \xrightarrow[{[\alpha'_1,\alpha'_2]}]{a'} t'}$$

$$\frac{Q \models s \xrightarrow[{[\alpha_1,\alpha_2]}]{a} t \qquad Q' \models s' \xrightarrow[{[\alpha'_1,\alpha'_2]}]{a'} t'}{Q \| Q' \models s \otimes s' \xrightarrow[{[max(\alpha_1,\alpha'_1), \alpha_2 + \alpha'_2]}]{a \otimes a'} t \otimes t'}$$

$$\frac{Q[\mathbf{rec}\ x.Q[x]] \models s \xrightarrow[{[\alpha_1,\alpha_2]}]{u} t}{\mathbf{rec}\ x.Q[x] \models s \xrightarrow[{[\alpha_1,\alpha_2]}]{u} t}$$

The last rule expresses that $[\![\mathbf{rec}\ x.Q[x]]\!]$ forms a co-cone with the diagram $([\![Q^i[\mathbf{nil}]]\!])_i$.

Example: By the rules above, or a calculation on the denotational semantics, we see that the term $(2.5).(a^1;\mathbf{nil} \| (1).a^1;\mathbf{nil})$ has minimum execution time 4.5 (with a synchronous schedule on two processors) and maximum execution time 5.5 (with a schedule on one processor only, hence by interleaving).

6 Future directions and Conclusion

In this article, we have presented an **operational model for real-time truly concurrent systems**. The model has **neat algebraic properties**. In particular, the category of models (timed HDA) is complete, co-complete, cartesian closed and monoidal closed. Categorical combinators are **timed process algebra** operators. Timed HDA can be used for **denotational** as well as **operational** semantics.

In the future, we would like to use the model for different purposes. The first one is to develop a theory of **complexity** for constrained concurrency, i.e. for machines which can use at most n processors. Mathematically, it would rely on a careful study of the effect of truncation (forget all transitions of dimension more than n) on geodesics. A second one would be program analysis in the style of [9].

References

[1] S. Abramsky and T. Maibaum. *Handbook of Logic in Computer Science*, volume 1. Oxford Press, 1993.

[2] L. Aceto and D. Murphy. On the ill-timed, but well-caused. In *Proc. of CONCUR'93*, number 715 in LNCS, pages 97 - 111. Springer-Verlag.

[3] R. Alur and D. Dill. The theory of timed automata. In *Proceedings of the REX Workshop, Real-Time: Theory in Practice*, LNCS. Springer-Verlag, 1991.

[4] A. Asperti and G. Longo. *Categories, types and structures.* The MIT Press, second edition, 1991.

[5] H. Barringer, R. Kuiper, and A. Pnueli. A really abstract concurrent model and its temporal logic. In *Proc. of the 13th POPL*, pages 173–183. ACM Press, 1986.

[6] M. A. Bednarczyk. *Categories of asynchronous systems.* PhD thesis, University of Sussex, 1988.

[7] R. Cleaveland and A. Zwarico. A theory of testing for real-time. In *Proceedings of LICS*. IEEE Press, 1991.

[8] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction of approximations of fixed points. *Principles of Programming Languages 4*, pages 238–252, 1977.

[9] R. Cridlig and E. Goubault. Semantics and analyses of Linda-based languages. In *Proc. of WSA'93*, number 724 in LNCS. Springer-Verlag, 1993.

[10] P. Gabriel and M. Zisman. Calculus of fractions and homotopy theory. In *Ergebnisse der Mathematik und ihrer Grenzgebiete*, volume 35. Springer Verlag, 1967.

[11] E. Goubault. Domains of higher-dimensional automata. In *Proc. of CONCUR'93*, Hildesheim, August 1993. Springer-Verlag.

[12] E. Goubault. Schedulers as abstract interpretations of HDA. In *Proc. of PEPM'95*, La Jolla, June 1995. ACM Press.

[13] E. Goubault. *The Geometry of Concurrency.* PhD thesis, Ecole Normale Supérieure, to appear, 1995.

[14] E. Goubault and T. P. Jensen. Homology of higher-dimensional automata. In *Proc. of CONCUR'92*, Stonybrook, New York, August 1992. Springer-Verlag.

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

[16] T.A. Henzinger, Z. Manna, and A. Pnueli. Timed transition systems. In *Proc. of the REX Workshop, Real-Time: Theory in Practice*, LNCS. Springer-Verlag, 1993.

[17] P. Krishnan. A model for real-time systems. In *Proc. of Mathematical Foundations of Computer Science*, number 520 in LNCS. Springer-Verlag, 1991.

[18] A.T. Lundell and S. Weingram. *The Topology of CW-Complexes.* Van Nostrand Reinhold Company, 1969.

[19] A. Maggiolo-Schettini and J. Winkowski. Towards an algebra for timed behaviours. *Theoretical Computer Science*, 103:335 - 363.

[20] J. P. May. *Simplicial objects in algebraic topology.* D. van Nostrand Company, inc, 1967.

[21] F. Moller and C. Tofts. A temporal calculus of communicating systems. In *Proceedings of CONCUR'90*, number 458 in LNCS. Springer-Verlag, 1990.

[22] X. Nicollin and J. Sifakis. An overview and synthesis of timed process algebras. In *Proc. of CAV'92*, number 575 in LNCS, pages 376–398. Springer-Verlag, 1992.

[23] G. Plotkin. Domains. Technical report, Computer Science Department, Edinbourgh, 1984.

[24] V. Pratt. Modeling concurrency with geometry. In *Proc. of the 18th ACM Symposium on Principles of Programming Languages*. ACM Press, 1991.

[25] G. M. Reed and A. W. Roscoe. Metric spaces as models for real-time concurrency. In *Proc. of Mathematical Foundations of Programming Languages and Semantics*, number 298 in LNCS, pages 331–343. Springer-Verlag, 1987.

[26] H. Rund. *The Differential Geometry of Finsler Spaces.* Springer-Verlag, 1959.

[27] M.W. Shields. Concurrent machines. *Computer Journal*, 28, 1985.

[28] A. Stark. Concurrent transition systems. *Theoretical Computer Science*, 64:221–269, 1989.

[29] R. van Glabbeek. Bisimulation semantics for higher dimensional automata. Technical report, Stanford University, 1991.

[30] R. van Glabbeek and U. Goltz. Partial order semantics for refinement of actions. *Bulletin of the EATCS*, (34), 1989.

[31] G. Winskel and M. Nielsen. *Models for concurrency*, volume 3 of Handbook of Logic in Computer Science, pages 100–200. Oxford University Press, 1994.