On the Expressiveness of Higher Dimensional Automata

R.J. van Glabbeek [1,2]

National ICT Australia
and School of Computer Science and Engineering
The University of New South Wales
Locked Bag 6016, Sydney, NSW 1466, Australia

Abstract

In this paper I compare the expressive power of several models of concurrency based on their ability to represent causal dependence. To this end, I translate these models, in behaviour preserving ways, into the model of higher dimensional automata, which is the most expressive model under investigation. In particular, I propose four different translations of Petri nets, corresponding to the four different computational interpretations of nets found in the literature.

I also extend various equivalence relations for concurrent systems to higher dimensional automata. These include the history preserving bisimulation, which is the coarsest equivalence that fully respects branching time, causality and their interplay, as well as the ST-bisimulation, a branching time respecting equivalence that takes causality into account to the extent that it is expressible by actions overlapping in time. Through their embeddings in higher dimensional automata, it is now well-defined whether members of different models of concurrency are equivalent.

Keywords: Concurrency, expressiveness, causality, higher dimensional automata, Petri nets, event structures, history preserving bisimulation, ST-bisimulation.

1 A hierarchy of concurrency models

Figure 1 lists the main models of concurrency proposed in the literature, ordered by expressive power. Of all models, the labelled variant is understood. Here I only treat models of processes that perform actions a, b, c, ... whose internal structure is not further examined, and real-time and stochastic aspects

[1] This work was supported by EPSRC under grant number GR/S22097
[2] Email: rvg@cs.stanford.edu

An extended abstract of this paper appeared in *Electronic Notes in Theoretical Computer Science* 128(2), April 2005: Proceedings of the 11th International Workshop on *Expressiveness in Concurrency*, EXPRESS 2004, pp. 5–34.

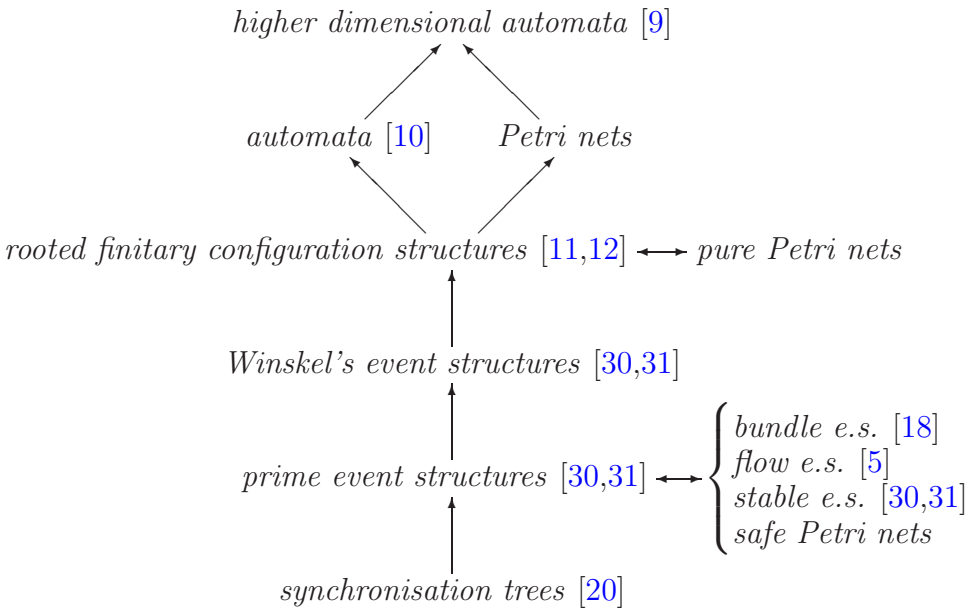

Fig. 1. A hierarchy of concurrency models, ordered by expressive power up to history preserving bisimulation

of processes are completely ignored. Furthermore, I only study the representation of *processes*, not the representation of *operators* on processes. I restrict myself to models that take branching time fully into account, and hence skip models that represent processes by the sets of their executions. Thus, the expressive power of the models differs only to the extent that certain forms of causal dependence are expressible. I limit myself to models that take a fully *asynchronous* view on parallelism: whenever a number of actions can happen simultaneously, this must be because they are causally independent, and hence they can also happen in any order. Because of this, I do not include Petri nets with inhibitor arcs, or Chu spaces [25].

There is an arrow from model A to model B in Figure 1 iff there exists a translation from processes representable in model A to processes representable in model B that fully respects branching time, causality, and their interplay. Thus a process and its translation ought to be *history preserving bisimulation equivalent* [26,11].

Part of the contribution of this paper is a definition of what this means. Forms of history preserving bisimulation were defined on behaviour structures in [26], on stable event structures in [11], and on Petri nets under the individual token interpretation in [4]; however, it has never been formally defined what it means for an event structure, for instance, to be history preserving bisimulation equivalent to a Petri net.

In Section 2 I introduce the model of higher dimensional automata. Then, in Sections 4, 5 and 6, I define behaviour preserving translations from the other models of Figure 1 into this model; for the model of Petri nets I do this in four different ways, corresponding to the four different computational

interpretations of nets found in the literature. In Section 8 I define history preserving bisimulation equivalence on higher dimensional automata. The embeddings of the other models of concurrency into higher dimensional automata make this definition apply to processes representable in any of these models: two processes are equivalent iff their representations as higher dimensional automata are. Naturally, I have to ensure that the new definition agrees with the existing ones on the models where it has already been defined.

With this tool in hand, the hierarchy of Figure 1 is justified in Section 9. In particular, counterexamples will be presented to illustrate the strictness of the expressiveness ordering.

Besides history preserving bisimulation equivalence, I also define *interleaving bisimulation equivalence* [20,11] on higher dimensional automata, and thus on the other models, as well as *ST-bisimulation equivalence* [14], a branching time respecting semantic equivalence that takes causality into account to the extent that it can be expressed by the possibility of durational actions to overlap in time. If I compare the models merely up to interleaving bisimulation equivalence they turn out to be all equally expressive. If I compare them up to ST-bisimulation equivalence, I conjecture that just two equivalence classes of models remain: the models that do not take causality into account—*in Figure 1 just the model of synchronisation trees*—and the models that do—*in Figure 1 all other models*. It follows that the more interesting hierarchy of Figure 1 is due to causal subtleties that evaporate when considering processes up to ST-bisimulation equivalence.

2 Higher dimensional automata

One of the most commonly used models of concurrency is that of *automata*, also known as *process graphs*, *state transition diagrams* or *labelled transition systems*. In ordinary automata, the parallel composition P of two actions a and b is displayed in the same way as a system M that executes a and b in either order, ending in the same state each way, such that a and b are mutually exclusive (see Figure 2). Nevertheless, it is often important to tell these systems apart. This happens for instance when a and b take time: the total running time of M is at least the sum of the running times of a and b, whereas P may be as quick as the maximum of the running times of a and b. Another occasion where it is essential to distinguish between P and M is when designing systems using *action refinement*, as described in [11]. In many other models of concurrency P and M are represented distinctly. For the model of Petri nets, this is illustrated in Figure 2.

Throughout the years, people have wondered whether the elegance of automata could be combined with the expressiveness of models like Petri nets or event structures, that are able to capture causal relationships between action occurrences. These relationships include the causal independence of a and b in P, the dependence of b on a in the left branch of the automaton representing M,

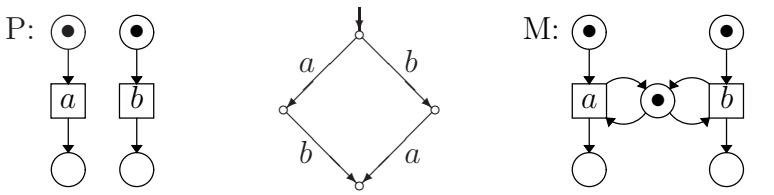

Fig. 2. The automaton in the middle represents both *parallelism*, as does the Petri net P, and *mutual exclusion*, as does the Petri net M.

and the dependence of a on b in the right branch. As a result, several models of concurrency have been proposed that are essentially automata, upgraded with some extra structure to express causal independence [28,2,26,29,32].

In [7] it was pointed out that ordinary automata, without such extra structure, are already sufficiently expressive to capture these causal relationships. All that is needed is a reading of automata that assumes squares to represent concurrency, and nonconfluent branching to represent a choice or conflict. Under this *concurrent interpretation* the automaton of Figure 2 represents parallelism only, whereas mutual exclusion is represented by the automaton of Figure 3. The square of Figure 2 is now seen as the *product* of the transi-

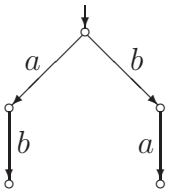

Fig. 3. Representation of mutual exclusion in ordinary automata under the concurrent interpretation

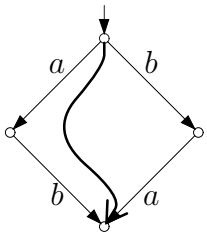

Fig. 4. Travelling through the inside of a square

tions a and b that form its sides. As each of these two transitions has 3 cells, a *start node*, an *end node*, and the *edge* between them, their product has 9 cells, namely the 4 nodes and 4 edges displayed in Figure 2, and the *inside of the square*. The latter represents the concurrent execution of a and b.

Modelling concurrency by means of squares (or cubes, hypercubes etc. in case of three or more concurrent actions) is particularly useful when actions are thought to have a duration or structure. A concurrent execution of a and b in the process $a\|b$ can then be thought of as a continuous path through the surface of the square, starting at the top and terminating at the bottom node, while being nondecreasing when projected on any edge. The execution displayed in Figure 4 for instance passes through a point in which 50% of a and about 17% of b has happened. As long as we want to identify all such paths (when abstracting from timing information and structural knowledge of a and b) we simply represent their equivalence class as the square. In [15] this representation of concurrent systems turned out to be essential in finding and explaining an essential counterexample in the study of semantic equivalences.

The price to pay for this simple solution is that it is no longer possible to represent the system M of Figure 2 in such a way that both executions *ab* and *ba* end up in the same state. In order to overcome this deficiency, and to distinguish *choice*, as displayed in Figure 3, from *mutual exclusion*, as in M in Figure 2, Vaughan Pratt [24] preferred for every square, cube, hypercube etc. to indicate explicitly whether it represents concurrency or not. An n-dimensional hypercube that represents the concurrent execution of the n transitions that span its sides is thought of as being "filled in", which is represented by an additional n-dimensional cell in the automaton. Those that represent mutual exclusion stay empty. This gives rise to what he baptised *higher dimensional automata*.

A totally sequential implementation of a system, in which no two actions happen in parallel, can be represented by an ordinary automaton, which is a higher dimensional automaton in which there are no higher dimensional cells, so no squares are filled in. At the other side of the spectrum, the very same ordinary automata under the concurrent interpretation of [7] form the special case of higher dimensional automata in which *every* (nondegenerate) square, cube or hypercube is filled in. In between, Pratt's higher dimensional automata can model systems that feature both parallelism and mutual exclusion, without having to resort to untying the join of two branches that surround an area of mutual exclusion.

The idea of higher dimensional automata above had to some extent been contemplated before in [28] and applied in [23]. However, [24] offered the first formalisation of the idea. Pratt's formalisation, based on n-categories, takes a *globular* or *hemispherical* approach, in which an n-cell has only two boundaries of dimension $n-1$. Desperate attempts by me to fully grok the globular approach led to an exchange with Pratt in December 1990 that gave rise to the joint conclusion that a *cubical approach*, in which an n-cell has $2n$ boundaries of dimension $n-1$, would be preferable (and easier to understand). A *higher dimensional automaton* (HDA) was henceforth defined as a presheaf over $\square$, the category of *cubical complexes*.[3] At the occasion of proposing notions of *bisimulation equivalence*, *homotopy* and *unfolding* for higher dimensional

[3] The objects of $\square$ are the symbols 0,1,2,... denoting the hypercubes of each dimension, and its morphisms from k to n are the embeddings of the k-dimensional hypercube as a k-dimensional face of the n-dimensional hypercube. All dimensions are directed and the morphisms preserve this direction. Thus, there are for example 6 morphisms from 2 to 3, corresponding to the fact that a cube has 6 sides. A *presheaf* over a category $\mathbf{C}$ is a functor $F : \mathbf{C}^{\mathrm{op}} \to \mathbf{Set}$. Thus a presheaf F over $\square$ associates a set $F(n)$ to every object n in $\square$. $F(n)$ is thought of as the set of n-dimensional hypercubes in the HDA. Also, for every morphism $m : k \to n$, recognising the k-dimensional hypercube as a face of the n-dimensional one, there exists a function $F(m) : F(n) \to F(k)$, giving for every n-dimensional hypercube its k-dimensional face on the side indicated by m. These functions must satisfy exactly those composition laws that hold for the morphisms in $\square$. The advantage of the categorical approach is that the concept of an HDA is thus completely defined without the need for figuring out these laws.

automata in [9], I reworded this definition in set-theoretical terms as follows.

Definition 1 A *cubical set* consists of a family of sets $(Q_n)_{n\geq 0}$ and for every $n \in \mathbb{N}$ a family of maps $s_i, t_i : Q_n \to Q_{n-1}$ for $1 \leq i \leq n$, such that

$$\alpha_i \circ \beta_j = \beta_{j-1} \circ \alpha_i \qquad \text{for all } 1 \leq i < j \leq n \text{ and } \alpha, \beta \in \{s, t\}. \qquad (*)$$

A *higher dimensional automaton*, labelled over an alphabet A, is a tuple (Q, s, t, I, F, l) [4] with (Q, s, t) a cubical set, $I \in Q_0$, $F \subseteq Q_0$ and $l : Q_1 \to A$, such that $l(s_i(q)) = l(t_i(q))$ for all $q \in Q_2$ and $i = 1, 2$.

The elements of Q_0 are called *nodes* and those of Q_1, Q_2 and Q_3 are *edges*, *squares* and *cubes*, respectively. In general, the elements of Q_n are called *n-dimensional hypercubes*, or *n-cells*. An n-dimensional hypercube represents a state of a concurrent system in which n transitions are firing concurrently. Because the dimensions of the hypercube are numbered $1, \ldots, n$, these n transitions are *de facto* stored as a list.

An edge q connects 2 nodes: its *source* $s_1(q)$ and its *target* $t_1(q)$. Likewise, an n-dimensional hypercube q has n sources $s_i(q)$ and n targets $t_i(q)$, one in each dimension $i = 1, ..., n$, and each being an $(n-1)$-dimensional hypercube. The source $s_i(q)$ represents the possible state prior to q in which $n-1$ out of n transitions are already firing, but the i^{th} one has not yet started. Likewise, $t_i(q)$ represents the possible state past q in which $n-1$ out of n transitions are still firing, but the i^{th} one has terminated.

When removing the i^{th} transition out of a list of n transitions, an implicit renumbering takes place, and what was formerly the j^{th} transition, for $j > i$, is now called the $(j-1)^{th}$ transition. Hence, first leaving out the j^{th} and then the i^{th}, with $i < j$, leaves us with the same list as first removing the i^{th} and then the $(j-1)^{th}$ transition. This is the content of the *cubical laws* (*).

I is the initial state, and F the set of final states of the represented system. These states are required to have dimension 0, meaning that no transition is currently firing. An edge $q \in Q_1$ represents a state where exactly 1 transition is firing, and l indicates the label of that transition. It is required that opposite sides of a square have the same label. This is because they represent the same transition, scheduled before and after the firing of another one. The labelling function can trivially be extended to a labelling of arbitrary n-dimensional hypercubes by lists of n actions (cf. Section 7.2).

Based on this definition, and the computational motivation of Pratt [24], numerous papers on higher dimensional automata have emerged [33–72].

A semantics of CCS in terms of higher dimensional automata is provided in Goubault & Jensen [62], and also studied by Lanzmann [65]. In Goubault [54], higher dimensional automata are used as a semantic domain for richer lan-

[4] I write Q for $\bigcup_{k=0}^{\infty} Q_k$ and the families of all maps s_i and t_i are denoted s and t. The components of an HDA A are called Q^A, s^A, t^A, I^A, F^A and l^A, a convention that also applies to other structures given as tuples. When clear from context, the index A is omitted.

guages, and to compute local invariants which can decide a few computational properties of interest. Cridlig & Goubault [39] provide a semantics of Linda in terms of higher dimensional automata. Applications of higher dimensional automata to scheduling problems and wait-free protocols for distributed systems are studied by Goubault [55,56,58], and model checking applications by Cridlig [37,38]. Algorithms for deadlock detection in terms of higher dimensional automata appear in Fajstrup et al. [41,40]. Takayama [70,71,72] studies parallelisation algorithms by means of higher dimensional automata. Extensions of higher dimensional automata with time are investigated by Goubault [55,57].

In Goubault [60] the relations between 1-dimensional automata, asynchronous transition systems [28,2], and higher dimensional automata are cast in a categorical framework, following the work of Winskel & Nielsen [32]. To this end it appears to be fruitful to equip the cubical sets of Definition 1 with degeneracy mappings, so that they become exactly the cubical sets studied before in algebraic topology [27].

Homotopy for higher dimensional automata is defined in Pratt [24] and Van Glabbeek [9], and further investigated by Goubault [55,61] and Raussen [68]. Gunawardena [64] uses homotopy theory to show that 2-phase locking is safe. Goubault & Raussen [63] show, using homotopy theory, how geometric models of concurrency like higher dimensional automata are suitable for attacking the state-explosion problem.

Homology theories for higher dimensional automata are proposed by Goubault and Gaucher [62,55,42–52]. Gaucher [42–47], like Pratt [24], takes a globular approach to higher dimensional automata, and relates it to the cubical approach. Variations and generalisations of this approach are studied by Gaucher & Goubault [52,48–51].

Buckland et al. [34,35] study a simplified version of globular higher dimensional automata, and equip it with process algebraic operations. A generalisation of higher dimensional automata suitable for modelling continuous rather than discrete processes has been proposed by Sokolowsky [69].

In Cattani & Sassone [36] a simplified form of higher dimensional automata is proposed, called *higher dimensional transition systems*. The simplification is payed for in expressive power though: not all higher dimensional automata that arise as the image of a Petri net under the self-concurrent collective token interpretation (see Section 6) exist as higher dimensional transition systems.

An event oriented form of higher dimensional automata, called *labelled step transition systems*, appears in Van Glabbeek [53] and Badouel [33], where it is used to represent the behaviour of Petri nets.

In [66,67], Pratt argues that acyclic higher dimensional automata can be seen as subsets of a single infinite dimensional cube, which in turn can be modelled as Chu spaces over 3.

An overview of work related to higher dimensional automata and the use of other geometric methods in computer science is presented in Goubault [59].

3 Isomorphism of reachable parts

When used for representing concurrent systems, HDA need to be considered modulo a suitable semantic equivalence. One of the finest possible candidates is the following notion of isomorphism of reachable parts.

Definition 2 Two higher dimensional automata A and B are *isomorphic* if they differ only in the names of their cells, i.e. if there exists a dimension preserving bijection $\mathcal{I} : \mathcal{Q}^{\mathrm{A}} \to \mathcal{Q}^{\mathrm{B}}$ between their cells, such that
- $s_i^{\mathrm{B}}(\mathcal{I}(q)) = \mathcal{I}(s_i^{\mathrm{A}}(q))$,
- $t_i^{\mathrm{B}}(\mathcal{I}(q)) = \mathcal{I}(t_i^{\mathrm{A}}(q))$,
- $\mathcal{I}(I^{\mathrm{A}}) = I^{\mathrm{B}}$,
- $\mathcal{I}(q) \in F^{\mathrm{B}} \Leftrightarrow q \in F^{\mathrm{A}}$,
- $l^{\mathrm{B}}(\mathcal{I}(q)) = l^{\mathrm{A}}(q)$ for $q \in Q_1^{\mathrm{A}}$.

Definition 3 A cell $q \in Q^{\mathrm{A}} = \bigcup_{n=0}^{\infty} Q_n^{\mathrm{A}}$ in a higher dimensional automaton A is *reachable* if it belongs to the smallest set $R(\mathrm{A}) \subseteq Q^{\mathrm{A}}$ such that
- $I^{\mathrm{A}} \in R(\mathrm{A})$,
- if $s_i(q) \in R(\mathrm{A})$ for some $1 \leq i \leq d(q)$ then $q \in R(\mathrm{A})$, and
- if $q \in R(\mathrm{A})$ and $1 \leq i \leq d(q)$ then $t_i(q) \in R(\mathrm{A})$.

Here $d(q)$, the *dimension* of q, is n when $q \in Q_n$.

Thus, a cell $q \in Q$ in an HDA A is reachable iff it occurs in a path of A as will be defined in Section 7.3. By definition $R(\mathrm{A})$ is closed under the maps t_i. The following lemma shows that $R(\mathrm{A})$ is also closed under the maps s_i. Its proof applies the cubical laws (*) of Definition 1.

Lemma 1 *Let* A *be an HDA. If* $q \in R(\mathrm{A})$ *and* $1 \leq i \leq d(q)$ *then* $s_i(q) \in R(\mathrm{A})$.

Proof. Define $R_k(\mathrm{A})$ for $k \in \mathbb{N}$ as the smallest sets of cells of A satisfying
(1) $I^{\mathrm{A}} \in R_0(\mathrm{A})$,
(2) if $s_i(q) \in R_k(\mathrm{A})$ for $1 \leq i \leq d(q)$ then $q \in R_{k+1}(\mathrm{A})$, and
(3) if $q \in R_k(\mathrm{A})$ and $1 \leq i \leq d(q)$ then $t_i(q) \in R_{k+1}(\mathrm{A})$.

Now $R(\mathrm{A}) = \bigcup_{k=0}^{\infty} R_k(\mathrm{A})$. With induction on k I show that if $q \in R_{k+1}(\mathrm{A})$ and $1 \leq i \leq d(q)$ then $s_i(q) \in R_k(\mathrm{A})$.

Induction base ($k = 0$): If $q \in R_1(\mathrm{A})$, then $d(q) = 1$ and $s_1(q) = I^{\mathrm{A}} \in R_0(\mathrm{A})$.

Induction step ($k > 0$): Let $q \in R_{k+1}(\mathrm{A})$ and $1 \leq i \leq d(q)$. By definition, either $s_j(q) \in R_k(\mathrm{A})$ for some $1 \leq j \leq d(q)$, or $q = t_j(q')$ with $q' \in R_k(\mathrm{A})$ and $1 \leq j \leq d(q')$.

Case 1: $s_j(q) \in R_k(\mathrm{A})$ and $i < j$. By induction, $s_i(s_j(q)) \in R_{k-1}(\mathrm{A})$. By a cubical law, $s_{j-1}(s_i(q)) = s_i(s_j(q)) \in R_{k-1}(\mathrm{A})$, so $s_i(q) \in R_k(\mathrm{A})$ by (2).

Case 2: The case $s_j(q) \in R_k(\mathrm{A})$ with $i = j$ is trivial.

Case 3: $s_j(q) \in R_k(\mathrm{A})$ and $i > j$. By induction, $s_{i-1}(s_j(q)) \in R_{k-1}(\mathrm{A})$. By a cubical law, $s_j(s_i(q)) = s_{i-1}(s_j(q)) \in R_{k-1}(\mathrm{A})$, so $s_i(q) \in R_k(\mathrm{A})$ by (2).

Case 4: $q = t_j(q')$ with $q' \in R_k(\mathrm{A})$ and $i < j \leq d(q')$. By induction, $s_i(q') \in R_{k-1}(\mathrm{A})$, so $t_{j-1}(s_i(q')) \in R_k(\mathrm{A})$ by (3). By one of the cubical laws, $s_i(q) = s_i(t_j(q')) = t_{j-1}(s_i(q')) \in R_k(\mathrm{A})$.

Case 5: $q = t_j(q')$ with $q' \in R_k(\mathrm{A})$ and $j \leq i \leq d(q)$, i.e. $j < i+1 \leq d(q')$. By induction, $s_{i+1}(q') \in R_{k-1}(\mathrm{A})$, so $t_j(s_{i+1}(q')) \in R_k(\mathrm{A})$ by (3). By a cubical law, $s_i(q) = s_i(t_j(q')) = t_j(s_{i+1}(q')) \in R_k(\mathrm{A})$. □

By Lemma 1, the following concept of *isomorphism of reachable parts*, $\cong_{\mathcal{R}}$, is well-defined.

Definition 4 The *reachable part* $\mathcal{R}(\mathrm{A})$ of an HDA A is the HDA consisting of its reachable cells: $(R(\mathrm{A}), s^{\mathrm{A}} \restriction R(\mathrm{A}), t^{\mathrm{A}} \restriction R(\mathrm{A}), I^{\mathrm{A}}, F^{\mathrm{A}} \restriction R(\mathrm{A}), l^{\mathrm{A}} \restriction R(\mathrm{A})$.

Write $\mathrm{A} \cong_{\mathcal{R}} \mathrm{B}$ if $\mathcal{R}(\mathrm{A})$ and $\mathcal{R}(\mathrm{B})$ are isomorphic.

When two HDA are isomorphic, then also their reachable parts are isomorphic. Thus, to check $\mathrm{A} \cong_{\mathcal{R}} \mathrm{B}$ it suffices to restrict A and B to subsets of Q^{A} and Q^{B} that contain all reachable states, and construct an isomorphism between the resulting HDA. This method will be applied in the proof of Theorem 2.

4 Embedding ordinary automata in HDA

A (1-dimensional) automaton is a tuple $(Q_0, Q_1, s_1, t_1, I, F, l)$, the special case of a higher dimensional automaton in which $Q_n = \emptyset$ for $n > 1$. Often automata are required to be *extensional*, meaning that a transition is completely determined by its source, target and label:

$$s_1(q) = s_1(q') \wedge t_1(q) = t_1(q') \wedge l(q) = l(q') \Rightarrow q = q' \qquad \text{for } q, q' \in Q_1.$$

In that case, a transition $q \in Q_1$ can be named after the triple $(s_1(q), l(q), t_1(q))$, and, writing S for the set of states Q_0, the quadruple (Q_1, s_1, t_1, l) can be conveniently represented by a relation $T \subseteq S \times A \times S$, thereby contracting the definition of an automaton to a quadruple (S, T, I, F).

A straightforward embedding of ordinary automata in higher dimensional automata is given by recognising them as HDA in which $Q_n = \emptyset$ for $n > 1$. However, the concurrent interpretation of automata from [7], elaborated in [10], yields a more expressive model of concurrency. Here an extensional 1-dimensional automaton $\mathrm{G} = (S, T, I, F)$ is, in essence, seen as an abbreviation of a higher dimensional automaton $\mathcal{A}(\mathrm{G})$ by assuming that any nondegenerate n-dimensional hypercube that can be recognised in G is "filled in", i.e. constitutes an n-dimensional cell in $\mathcal{A}(\mathrm{G})$. In [7,10] this was merely a computational interpretation of automata; here I use it to formally define $\mathcal{A}(\mathrm{G})$. An n-dimensional hypercube in G consists of a string $\ell = a_1 \cdots a_n \in A^n$ of n action labels and 2^n corners $p_\xi \in S$, with $\xi \in \{0,1\}^n$ ranging over the strings of n 0s and 1s, such that $(p_\xi, a_i, p_\chi) \in T$ whenever ξ and χ differ only on their i^{th} bit and that bit is 0 in ξ and 1 in χ. The source $s_i(q)$ (resp. target $t_i(q)$) in dimension i of such an n-dimensional hypercube q in G is the $(n-1)$-dimensional hypercube in G obtained by deleting a_i from ℓ and restricting the set of corners p_ξ of q to those in which the i^{th} bit of ξ is 0 (resp. 1).

A hypercube $q = (\ell, p_\xi\ (\xi \in \{0,1\}^n))$ in G is *degenerate* iff there are indices $1 \leq i < j \leq n$ such that $a_i = a_j$ in ℓ and, for certain bits $b_k \in \{0,1\}$ $(k \neq i, j)$,

$$p_{b_1 \cdots b_{i-1} 0 b_{i+1} \cdots b_{j-1} 1 b_{j+1} \cdots b_n} = p_{b_1 \cdots b_{i-1} 1 b_{i+1} \cdots b_{j-1} 0 b_{j+1} \cdots b_n}.$$

In particular, a degenerate square consists of two transitions (p, a, q) and (q, a, r). It is a square by taking $\ell = aa$ and the corners $p_{00} = p$, $p_{01} = p_{10} = q$ and $p_{11} = r$. The reason for not assuming this square to be filled in, is that if I do, I loose the expressive power to specify a sequence of two identical actions that have to occur in sequential order. Hence the property that at least all synchronisation trees are representable as automata under the concurrent interpretation would be lost. In general, a hypercube in G is nondegenerate iff all its 2-dimensional faces are nondegenerate. So the definition of degeneracy above is the most stringent one I could get away with.

5 Embedding event oriented models in HDA

Winskel [30,31] introduced six kinds of event structures: the *prime*, *stable* and [general] *event structures*, each optionally with the restriction of *binary conflict*. The behaviour of these event structures is fully specified by the *families of configurations* that can be associated to them; moreover, the family of all configurations of an event structure is fully determined by the finite configurations in the family. Hence, event structures embed faithfully in the model of *(finitary) configuration structures* of [12], which generalises the families of (finite) configurations of event structures.

In Figure 5, taken from [12], the six models of event structures from [30,31] are ordered with respect to their expressive power as measured by the class of (finitary) configuration structures they can denote. In addition, the figure includes the *flow event structures* of Boudol [5], and the *bundle event structures* of Langerak [18]. The *synchronisation trees* of Milner [20], which are just tree shaped (1-dimensional) automata, can be seen as special kinds of prime event structures with binary conflict, and, naturally, the maximal expressive power is obtained by the model of all (finitary) configuration structures.

In [11] a configuration structure was defined as follows.

Definition 5 A *(labelled) configuration structure (over an alphabet A)* is a triple $\mathrm{C} = (C, F, l)$ where C is a family of finite sets (the *configurations*), $F \subseteq C$ a *termination predicate*, satisfying $X \in F \wedge X \subseteq Y \in C \Rightarrow X = Y$ (i.e. terminating configurations must be maximal), and $l : \bigcup_{X \in C} X \to A$ is a *labelling function*. The set E of *events* of C is defined by $E := \bigcup_{X \in C} X$.

This definition differs from the one in [12] by not listing the set of events E in the tuple (E, C, F, l), thereby employing a coarser notion of isomorphism, by employing finite configurations only (this is called *finitariness* in [12]), and by the addition of the termination predicate, thereby distinguishing *deadlock*—a maximal but non-terminating configuration—from successful termination.

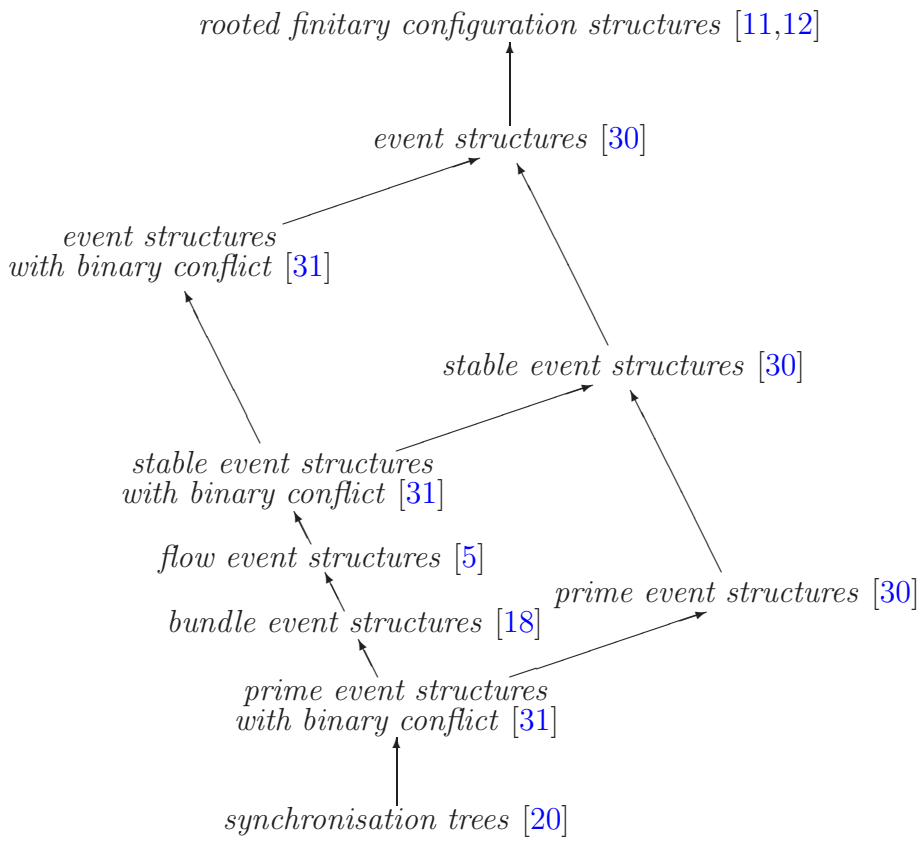

Fig. 5. A hierarchy of event oriented models of concurrency

Definition 6 [11] A configuration structure C is

- *rooted* iff $\emptyset \in C^{\mathrm{C}}$,
- *connected* iff $\emptyset \neq X \in C^{\mathrm{C}} \Rightarrow \exists e \in X : X - \{e\} \in C^{\mathrm{C}}$,
- *closed under bounded unions* iff $X, Y, Z \in C^{\mathrm{C}},\ X \cup Y \subseteq Z \Rightarrow X \cup Y \in C^{\mathrm{C}}$,
- *closed under bounded intersections* iff $X, Y, Z \in C^{\mathrm{C}},\ X \cup Y \subseteq Z \Rightarrow X \cap Y \in C^{\mathrm{C}}$,
- and *stable* iff it has all four properties above.

As observed in [30,12,11], a configuration structure arises as the family of finite configurations of an event structure iff $F = \emptyset$ and it is rooted, connected and closed under bounded intersections; it arises as the family of finite configurations of a stable event structure iff $F = \emptyset$ and it is stable.

In [10], the rooted finitary configuration structures are faithfully embedded in the model of ordinary automata under the concurrent interpretation of [7]. To this end, the concurrent interpretation of [7] was extended to cover *all* (extensional) automata—in [7] it applied merely to automata of a certain shape: the ones arising as the images of (labelled) prime event structures.

Definition 7 The extensional 1-dimensional automaton associated to a rooted finitary configuration structure $\mathrm{C} = (C, F, l)$ is $\mathcal{G}(\mathrm{C}) = (C, T, \emptyset, F)$ with

$$(X, a, Y) \in T \text{ iff } Y = X \dot{\cup} \{e\} \text{ for an event } e \text{ with } l(e) = a.$$

By composing the mappings of [12], embedding the models of Figure 5 in rooted finitary configuration structures, of [10], embedding the latter in automata under the concurrent interpretation, and of Section 4 above, embedding automata in HDA, all models of Figure 5 embed faithfully in HDA.

6 Embedding Petri nets in HDA

In this section I show that the model of higher dimensional automata is at least as expressive as the model of Petri nets, by giving translations from Petri nets to higher dimensional automata that capture exactly the dynamic behaviour of nets, as expressed by the *firing rule*, informally described below.

The essence of the material in this section appeared originally in [53], but using a different model of higher dimensional automata, called *labelled step transition systems*. Although extremely appealing as a model of concurrency, labelled step transition systems lack the expressiveness to crown the spectrum of Figure 1: they do not capture all ordinary automata under the concurrent interpretation.

Definition 8 A (labelled, marked) *Petri net* is a tuple (S, T, F, I, l) with
- S and T two disjoint sets of *places* (*Stellen* in German) and *transitions*,
- $F : (S \times T \cup T \times S) \to \mathbb{N}$, the *flow relation*,
- $I : S \to \mathbb{N}$, the *initial marking*,
- and $l : T \to A$, for A a set of *actions*, the *labelling function*.

Petri nets are pictured by drawing the places as circles and the transitions as boxes, containing their label. For $x, y \in S \cup T$ there are $F(s,t)$ *arcs* from x to y.

When a Petri net represents a concurrent system, a global state of this system is given as a *marking*, a function $M : S \to \mathbb{N}$. Such a state is depicted by placing $M(s)$ dots (*tokens*) in each place s. The initial state is given by the marking I. A transition t can *fire* (occur) by taking $F(s,t)$ tokens out of each place s, one for each arc from s to t. If this is possible in the state given by the marking $M : S \to \mathbb{N}$, i.e. if $F(s,t) < M(s)$ for all $s \in S$, one says that t is *enabled* under M. After t has fired, $F(t,s)$ tokens are added to each place s, one of each arc from t to s. For t a transition in a Petri net N, let ${}^\bullet t : S^{\mathrm{N}} \to \mathbb{N}$ be given by ${}^\bullet t(s) = F(s,t)$ and $t^\bullet : S^{\mathrm{N}} \to \mathbb{N}$ by $t^\bullet(s) = F(t.s)$. The multiset ${}^\bullet t$ describes which tokens are consumed by firing t, and $t^\bullet$ which tokens are produced.

In Van Glabbeek & Vaandrager [14], Petri nets were studied from the point of view that transitions may take time, and we introduced global states in which any number of transitions may be currently firing. The states represented by markings as defined above are those in which no transition is currently firing. In general, a state is given by a multiset of places *and transitions*. In order to precisely keep track of causal relationships between transition firings, we found it convenient to represent the (multi)set of transitions in a state as a list. This made it possible to distinguish different occurrences of

currently firing transitions (i.e. the third and the fifth). Hence an *ST-marking* is defined as a pair $(M, \sigma) \in \mathcal{M}(S) \times T^*$ of a multiset of places and a list of transitions. This is exactly the notion of state that I need below in my translations of Petri nets to higher dimensional automata.

As pointed out in [12], there are $2 \times 2 = 4$ computational interpretations of Petri nets, called the *individual token* and the *collective token* interpretation, and, orthogonally, the *self-sequential* and the *self-concurrent* interpretation. In the individual token interpretation one distinguishes different tokens residing in the same place, keeping track of where they come from. If a transition fires by using a token that has been produced by another transition, there is a causal link between the two. Consequently, the causal relations between the transitions in a run of a net can always be described by means of a partial order. In the collective token interpretation, on the other hand, tokens cannot be distinguished: if there are two tokens in a place, all that is present there is the number 2. This gives rise to more subtle causal relationships between transitions in a run of a net, which cannot be expressed by partial orders. In the self-concurrent interpretation, a transition may fire concurrently with itself. This is not allowed in the self-sequential interpretation.

The below can be understood as a way of formally pinpointing the differences between these computational interpretations, by giving four translations from Petri nets into higher dimensional automata, one for each interpretation. In some sense this amounts to giving four different semantics of Petri nets.

6.1 The self-concurrent collective token interpretation

First I define the higher dimensional automaton $\mathcal{A}_{\mathrm{CT}}(\mathrm{N})$ associated to a Petri net N according to the self-concurrent collective token interpretation of nets. As cells of $\mathcal{A}_{\mathrm{CT}}(\mathrm{N})$ I take the ST-markings of N, each being a pair (M, σ) of a multiset M of places and a list σ of transitions in N. The number of transitions in the list is the dimension of the cell. The source $s_i(q)$ (resp. the target $t_i(q)$) in dimension i of a cell $q = (M, \sigma)$ is obtained by omitting the i^{th} transition t from σ, and adding the multiset of places ${}^\bullet t$ (resp. $t^\bullet$) to M. Below, $+$ denotes the union of multisets, given by $(M+X)(s) = M(s)+X(s)$. The empty list is denoted ε. As there is no standard definition of successful termination in Petri nets, I map them to HDA without final states; however, for any definition of successful termination of Petri nets found in the literature, most likely a corresponding definition of the final states of the associated HDA can be obtained.

Definition 9 Let N be a Petri net. The higher dimensional automaton $\mathcal{A}_{\mathrm{CT}}(\mathrm{N}) = (Q, s, t, I, F, l)$ is given by
- $Q_n = \mathcal{M}(S^{\mathrm{N}}) \times (T^{\mathrm{N}})^n$ for $n \in \mathbb{N}$,
- $s_i(M, t_1 \cdots t_n) = (M + {}^\bullet t_i,\ t_1 \cdots t_{i-1} t_{i+1} \cdots t_n)$ for $1 \leq i \leq n$,
- $t_i(M, t_1 \cdots t_n) = (M + t_i^\bullet,\ t_1 \cdots t_{i-1} t_{i+1} \cdots t_n)$ for $1 \leq i \leq n$,
- $I = (I^{\mathrm{N}}, \varepsilon)$ and $F = \emptyset$,
- $l(M, t) = l^{\mathrm{N}}(t)$ for $(M, t) \in Q_1$.

The source and target functions defined above correspond exactly to the start and termination of a transition firing as defined in clauses 2 and 4 of Definition 7.1.1 in [14]. They are also consistent with the informal description of the firing rule given above.

6.2 The self-concurrent individual token interpretation

Below I will define the notion of a token as it could occur in a Petri net, in such a way that all possible token occurrences have a different name. A token will be a triple (t', k, s), with s the place where the token occurs, and t' the transition firing that brought it there. For tokens that are in s initially, I take $t' = *$. When the number of tokens that t' deposits in s is n, I distinguish these tokens by giving them ordinal numbers $k = 0, 1, 2, ..., n-1$. In order to define tokens as announced above I need to define transition firings simultaneously. These will be pairs (X, t) with t the transition that fires, and X the set of tokens that is consumed in the firing. Transitions t that can fire without consuming tokens can fire multiple times on the same (empty) input; these firings will be called (k, t) with $k \in \mathbb{N}$ instead of $(\emptyset, t)$. I define the functions β from tokens to the places where they occur by $\beta(x, k, s) = s$, and η from transition firings to the transition that fires by $\eta(x, t) = t$. The function β extends to a function from sets of tokens X to multisets of places $\beta(X) : S \to \mathbb{N}$, by $\beta(X)(s) = |\{s' \in X \mid \beta(s') = s\}|$.

Definition 10 Given a Petri net $\mathrm{N} = (S, T, F, I, l)$, the sets of *tokens* $S_\bullet$ and *transition firings* $T_\bullet$ of N are recursively defined by
- $(*, k, s) \in S_\bullet$ for $s \in S$ and $k < I(s)$;
- $(t', k, s) \in S_\bullet$ for $s \in S$, $t' \in T_\bullet$ and $k < F(\eta(t'), s)$;
- $(X, t) \in T_\bullet$ for $t \in T$ and $X \subseteq S_\bullet$ such that $\beta(X) = {}^\bullet t \neq \emptyset$;
- $(k, t) \in T_\bullet$ for $k \in \mathbb{N}$ and $t \in T$ such that ${}^\bullet t = \emptyset$.

Now I define the higher dimensional automaton $\mathcal{A}_{\mathrm{IT}}(\mathrm{N})$ associated to a Petri net N according to the self-concurrent individual token interpretation of nets. First I will treat the case that N is a *standard net*, a net in which each transition has at least one incoming arc: $\forall t \in T.\ {}^\bullet t > 0$. A net is standard iff its set of *spontaneous transition firings* $T_\circ = \{(k, t) \in T_\bullet \mid k \in \mathbb{N}\}$ is empty.

As cells of $\mathcal{A}_{\mathrm{IT}}(\mathrm{N})$ I take the *ST-markings with individual tokens* of N, each being a pair (M, σ) of a multiset M of tokens of N (each token allocated to a place in N) and a list σ of transition firings of N. The number of transition firings in the list is the dimension of the cell. The source $s_i(q)$ in dimension i of a cell $q = (M, \sigma)$ is obtained by omitting the i^{th} transition firing (X, t) from σ, and adding the set of tokens X to M. Likewise, the target $t_i(q)$ of q in dimension i is obtained by omitting the i^{th} transition firing (X, t) from σ and upgrading M by adding $F(t, s)$ tokens to each place s. Below, in applying the multiset union $+$, sets X are identified with multisets by taking $X(s) = 1$ if $s \in X$ and $X(s) = 0$ otherwise.

Definition 11 Let N be a standard net. The higher dimensional automaton $\mathcal{A}_{\mathrm{IT}}(\mathrm{N}) = (Q, s, t, I, F, l)$ is given by

- $Q_n = \mathcal{M}(S^{\mathrm{N}}_\bullet) \times (T^{\mathrm{N}}_\bullet)^n$ for $n \in \mathbb{N}$,
- $s_i(M, t'_1 \cdots t'_n) = (M + X, t'_1 \cdots t'_{i-1} t'_{i+1} \cdots t'_n)$ for $1 \leq i \leq n$ and $t'_i = (X, t)$,
- $t_i(M, t'_1 \cdots t'_n) = (M + \{(t'_i, k, s) \mid k < F^{\mathrm{N}}(\eta(t'_i), s)\}, t'_1 \cdots t'_{i-1} t'_{i+1} \cdots t'_n)$,
- $I = (\{(*, k, s) \mid k < I^{\mathrm{N}}(s)\}, \varepsilon)$ and $F = \emptyset$,
- $l(M, t') = l^{\mathrm{N}}(\eta(t'))$ for $(M, t') \in Q_1$.

It may be helpful to observe that $\mathcal{A}_{\mathrm{CT}}(\mathrm{N})$ can be obtained from $\mathcal{A}_{\mathrm{IT}}(\mathrm{N})$ by applying β and η to the tokens and transition firings that make up a cell in $\mathcal{A}_{\mathrm{IT}}(\mathrm{N})$; in particular one has $s_i((\beta, \eta)(q)) = (\beta, \eta)(s_i(q))$, and likewise for t_i.

The following proposition says that I succeeded in giving all possible token occurrences a different name: each reachable cell in $\mathcal{A}_{\mathrm{IT}}(\mathrm{N})$ is an ST-marking with individual tokens (M, σ) of N in which M is a plain set. The reason to involve multisets of tokens in the definition above is to avoid the problems related to unions $M \cup X$ not being disjoint.

Proposition 1 *For* N *a standard net and* $(M, \sigma) \in R(\mathcal{A}_{\mathrm{IT}}(\mathrm{N}))$ *a reachable ST-marking with individual tokens, no token occurs twice in* M*, i.e.* M *is a plain set.*

Proof. Define the *direct causality relation* $\prec$ between the tokens and transition firings of N by (1) $s' \prec (X, t)$ for $s' \in X$ and (2) $t' \prec (t', k, s)$. Let $<$, the *causality relation*, be the transitive closure of $\prec$.

I claim that in any reachable ST-marking with individual tokens (M, σ) of N, no token occurs twice in M, no transition firing occurs twice in σ, and there are no tokens or transition firings x and y in (M, σ) such that $x < y$.

This follows with a straightforward induction on reachability, using for the $s_i(q)$-clause that $y < (X, t) \Leftrightarrow \exists x \in X : y \leq x$, and for the $t_i(q)$-clause that $y < (t'_i, k, s) \Leftrightarrow y \leq t'_i$. □

For non-standard nets the question arises how to generalise the second clause in Definition 11 to transition firings of the form (k, t) with $k \in \mathbb{N}$. The simplest solution would be to treat such firings as $(\emptyset, t)$. However, this would lead to a failure of Proposition 1 for non-standard nets, as a spontaneous transition firing (k, t) could occur multiple times in a firing sequence, leaving multiple copies of its output tokens in the resulting reachable ST-marking.

Instead, I want to ensure that each spontaneous transition firing may only occur once in a run of a Petri net. This requirement is motivated by the idea that every time a transition t with ${}^\bullet t = 0$ fires, its firing gets a different identifier. Thus, my notion of state should incorporate, besides the information contained in an ST-marking with individual tokens, some bookkeeping to remember which spontaneous transition firings are still possible. This I do by adding to the place-component of an ST-marking the set of names ${}^t k$ of all spontaneous transition firings $(k, t) \in T_0$ that may still occur.

Definition 12 Let N be net. The HDA $\mathcal{A}_{\mathrm{IT}}(\mathrm{N}) = (Q, s, t, I, F, l)$ is given by
- $Q_n = \mathcal{M}(S^{\mathrm{N}}_\bullet \cup \{\, {}^t k \mid (k,t) \in T_0\}) \times (T^{\mathrm{N}}_\bullet)^n$ for $n \in \mathbb{N}$,
- $s_i(M, t'_1 \cdots t'_n) = (M + X, t'_1 \cdots t'_{i-1} t'_{i+1} \cdots t'_n)$ for $1 \leq i \leq n$ and $t'_i = (X,t)$,
  $s_i(M, t'_1 \cdots t'_n) = (M + \{{}^t k\}, t'_1 \cdots t'_{i-1} t'_{i+1} \cdots t'_n)$ for $1 \leq i \leq n$ and $t'_i = (k,t)$,
- $t_i(M, t'_1 \cdots t'_n) = (M + \{(t'_i, k, s) \mid k < F^{\mathrm{N}}(\eta(t'_i), s)\}, t'_1 \cdots t'_{i-1} t'_{i+1} \cdots t'_n)$,
- $I = (\{(*, k, s) \mid k < I^{\mathrm{N}}(s)\} \cup \{\, {}^t k \mid (k,t) \in T_0\}, \varepsilon)$ and $F = \emptyset$,
- $l(M, t') = l^{\mathrm{N}}(\eta(t'))$ for $(M, t') \in Q_1$.

With this definition, Proposition 1 generalises straightforwardly.

Corollary 1 *For* N *a net and* $(M, \sigma) \in R(\mathcal{A}_{\mathrm{IT}}(\mathrm{N}))$, *$M$ is a plain set.* □

6.3 The self-sequential interpretations

A self-sequential version $\mathcal{A}^{ss}_{\mathrm{CT}}$ of the collective token interpretation of Petri nets into higher dimensional automata is obtained by only allowing, in the first clause of Definition 9, cells (M, σ) in which no transition occurs twice in σ. Likewise, on standard nets, a self-sequential version of the individual token interpretation above is obtained by only allowing cells $(M, t'_1 \cdots t'_n)$ such that $\eta(t'_i) = \eta(t'_j) \Rightarrow i = j$.

On non-standard nets, before employing the same definition, I take the opportunity to rectify an unfortunate aspect of the map $\mathcal{A}_{\mathrm{IT}}$ that was unavoidable under the self-concurrent interpretation. Namely, if a net contains a transition t without input places, Definition 12 yields an infinitely branching HDA: the initial transition firing can be any (k,t) for $k \in \mathbb{N}$. The reason this is unavoidable under the self-concurrent interpretation is that any number of transition firings (k,t) can happen simultaneously (can occur in one ST-marking with individual tokens), and I want the interpretation to preserve the fundamental property of Petri nets that whenever a number of transition firings can happen in one step, they can happen in any order; so any of the firings (k,t) can happen first. Under the self-sequential interpretation, on the other hand, it is much more natural so take the point of view that although the transition t allows arbitrary many firings to occur sequentially, there is no point in distinguishing different kinds of first firings. Thus, I will use k not merely as a label taken from an arbitrary countable set, but as an actual number, (k,t) denoting the $k{+}1^{th}$ firing of transition t. Thus, the presence of t_k in a state signifies that the $k{+}1^{th}$ firing of t is enabled.

Definition 13 Let N be net. The HDA $\mathcal{A}^{ss}_{\mathrm{IT}}(\mathrm{N}) = (Q, s, t, I, F, l)$ is given by
- $Q_n = \{(M, t'_1 \cdots t'_n) \in \mathcal{M}(S^{\mathrm{N}}_\bullet \cup \{{}^t k \mid (k,t) \in T_0\}) \times (T^{\mathrm{N}}_\bullet)^n \mid \eta(t'_i) = \eta(t'_j) \Rightarrow i = j\}$ for $n \in \mathbb{N}$,
- $s_i(M, t'_1 \cdots t'_n) = (M + X, t'_1 \cdots t'_{i-1} t'_{i+1} \cdots t'_n)$ for $1 \leq i \leq n$ and $t'_i = (X,t)$,
  $s_i(M, t'_1 \cdots t'_n) = (M + \{{}^t k\}, t'_1 \cdots t'_{i-1} t'_{i+1} \cdots t'_n)$ for $1 \leq i \leq n$ and $t'_i = (k,t)$,
- $t_i(M, t'_1 \cdots t'_n) = (M + \{(t'_i, k, s) \mid k < F^{\mathrm{N}}(\eta(t'_i), s)\} \cup \{{}^t(k{+}1) \mid t'_i = (k,t)\}, t'_1 \cdots t'_{i-1} t'_{i+1} \cdots t'_n)$,

- $I = (\{(*,k,s) \mid k < I^{\mathrm{N}}(s)\} \cup \{\,{}^t0 \mid t \in T_0 \wedge {}^\bullet t = 0\}, \varepsilon)$ and $F = \emptyset$,
- $l(M,t') = l^{\mathrm{N}}(\eta(t'))$ for $(M,t') \in Q_1$.

Corollary 2 *For* N *a net and* $(M,\sigma) \in R(\mathcal{A}^{ss}_{\mathrm{IT}}(\mathrm{N}))$, *$M$ is a plain set.* □

6.4 The relative expressiveness of the four interpretations

Each of the four computational interpretations above makes a different model of concurrency out of Petri nets. These models can now be compared with respect to their expressive power in denoting higher dimensional automata. A first result is easily obtained: Petri nets under the self-concurrent collective token interpretation are at least as expressive as Petri nets under the self-concurrent individual token interpretation, in the sense that any higher dimensional automaton that can be denoted by a net under the self-concurrent individual token interpretation can also be a denoted by a net under the self-concurrent collective token interpretation.

Theorem 1 *For every net* N *there is a net* $\mathrm{N}_\bullet$ *such that* $\mathcal{A}_{\mathrm{CT}}(\mathrm{N}_\bullet) = \mathcal{A}_{\mathrm{IT}}(\mathrm{N})$.

Proof. $\mathrm{N}_\bullet = (S_\bullet \cup \{\,{}^tk \mid (k,t) \in T_0\}, T_\bullet, F_\bullet, I_\bullet, l_\bullet)$ with
- $S_\bullet$ and $T_\bullet$ as in Definition 10.
- $F_\bullet(s',t') = 1$ if $t' = (X,t)$ with $s' \in X$, or $t' = (k,t) \in T_\circ$ and $s' = {}^tk$; $F_\bullet(s',t') = 0$ otherwise.
- $F_\bullet(t',s') = 1$ if s' has the form (t',k,s); $F_\bullet(t',s') = 0$ otherwise.
- $I_\bullet(*,k,s) = I_\bullet({}^tk) = 1$ and $I_\bullet(t',k,s) = 0$ for $t' \in T_\bullet$.
- $l_\bullet(t') = l^{\mathrm{N}}(\eta(t'))$ for $t' \in T_\bullet$.

That $\mathcal{A}_{\mathrm{CT}}(\mathrm{N}_\bullet) = \mathcal{A}_{\mathrm{IT}}(\mathrm{N})$ is straightforward. □

The net $\mathrm{N}_\bullet$ constructed above is a close relative of the *unfolding* of a Petri net into an *occurrence net*, as defined in [30,6,19]. The difference is that I have not bothered to eliminate unreachable places and transitions. In [53] I define the unfolding of an arbitrary Petri net N as the reachable part of $\mathrm{N}_\bullet$. This construction extends the unfolding construction of [30,6,19] by applying also to non-standard nets.

In general, results as strong as Theorem 1 can not be obtained: in order to compare expressiveness in a meaningful way, processes represented by higher dimensional automata, Petri nets, or other models of concurrency should be regarded modulo some semantic equivalence relation. A particularly fine equivalence relation that allows one to totally order the computational interpretations of Petri nets is isomorphism of reachable parts of HDA.

The following theorem shows that, up to isomorphism of reachable parts of associated HDA, the behaviour of nets under the self-sequential interpretations can easily be encoded into the behaviour of nets under the corresponding self-concurrent interpretation.

Theorem 2 *For any net* N *there is a net* N^{ss} *such that* $\mathcal{A}_{\mathrm{CT}}(\mathrm{N}^{ss}) \cong_{\mathcal{R}} \mathcal{A}^{ss}_{\mathrm{CT}}(\mathrm{N})$ *and* $\mathcal{A}_{\mathrm{IT}}(\mathrm{N}^{ss}) \cong_{\mathcal{R}} \mathcal{A}^{ss}_{\mathrm{IT}}(\mathrm{N})$.

Proof. Following [17], N^{ss} is obtained from N by adding for every transition t a *self-loop*, consisting of a place s_t with $I(s_t) = F(s_t, t) = F(t, s_t) = 1$ and $F(s_t, u) = F(u, s_t) = 0$ for all $u \neq t$. Write S_{new} for the set of new places s_t.

To check that $\mathcal{A}_{\mathrm{CT}}(\mathrm{N}^{ss}) \cong_{\mathcal{R}} \mathcal{A}^{ss}_{\mathrm{CT}}(\mathrm{N})$, restrict the cells of $\mathcal{A}_{\mathrm{CT}}(\mathrm{N}^{ss})$, i.e. the ST-markings (M, σ) of N^{ss}, to the ones in which each transition t occurs at most once in σ, $M(s_t) = 1$ when t does not occur in σ, and $M(s_t) = 0$ when t does occur in σ. This set of cells surely contains all reachable ones. Let $\mathcal{I}(M, \sigma) \in \mathcal{M}(S^{\mathrm{N}}) \times (T^{\mathrm{N}})^*$ be obtained by restricting the domain of $M : S \cup S_{new} \to \mathbb{N}$ to S, and leaving σ invariant. The bijection $\mathcal{I}$ constitutes an isomorphism of reachable parts between $\mathcal{A}_{\mathrm{CT}}(\mathrm{N}^{ss})$ and $\mathcal{A}^{ss}_{\mathrm{CT}}(\mathrm{N})$.

To check that $\mathcal{A}_{\mathrm{IT}}(\mathrm{N}^{ss}) \cong_{\mathcal{R}} \mathcal{A}^{ss}_{\mathrm{IT}}(\mathrm{N})$, restrict the cells of $\mathcal{A}_{\mathrm{IT}}(\mathrm{N}^{ss})$, i.e. the ST-markings with individual tokens $(M, t'_1 \cdots t'_n)$ of N^{ss}, to the ones satisfying $\eta(t'_i) = \eta(t'_j) \Rightarrow i = j$, such that there is at most one token of the form $(x, 0, s_t)$ in M for each $s_t \in S_{new}$, and none when $\eta(t_i) = t$ for some $1 \leq i \leq n$. This set of states surely contains all reachable ones. Also, in view of Corollaries 1 and 2, the cells of $\mathcal{A}_{\mathrm{IT}}(\mathrm{N}^{ss})$ and $\mathcal{A}^{ss}_{\mathrm{IT}}(\mathrm{N})$ may be restricted to pairs (M, σ) with M a *set* of tokens and names of spontaneous transition firings, rather than a multiset. Let $S_\circ = \{s_t \in S_{new} \mid {}^\bullet t = 0 \ \text{(in N)}\}$. For $s_t \in S_\circ$, let $s^0_t = (*, 0, s_t)$ and $s^{k+1}_t = ((\{s^k_t\}, t), 0, s_t)$. Then all tokens (x, k, s_t) of N^{ss} are of the form s^k_t for $k \in \mathbb{N}$. Now the mappings $\overline{\eta}$ from the transition firings in N^{ss} to the transition firings in N, for convenience extended with $\overline{\eta}(*) = *$, and $\overline{\beta}$ from sets of tokens in N^{ss} to sets of tokens and names of spontaneous transition firings in N, are defined with recursion on the structure of transition firings and sets of tokens by $\overline{\eta}(X, t) = \begin{cases} (\overline{\beta}(X), t) & \text{if } {}^\bullet t \neq 0 \\ (k, t) & \text{if } {}^\bullet t = 0 \wedge X = \{s^k_t\} \end{cases}$ and $\overline{\beta}(X) = \{(\overline{\eta}(x), k, s) \mid (x, k, s) \in X \wedge s \notin S_{new}\} \cup \{ {}^t k \mid s^k_t \in X \wedge s_t \in S_\circ\}$. The bijection $\mathcal{I}$ given by $\mathcal{I}(M, t'_1 \cdots t'_n) = (\overline{\beta}(M), \overline{\eta}(t_1) \cdots \overline{\eta}(t_n))$ constitutes an isomorphism of reachable parts between $\mathcal{A}_{\mathrm{IT}}(\mathrm{N}^{ss})$ and $\mathcal{A}^{ss}_{\mathrm{IT}}(\mathrm{N})$. □

By Theorem 2, any HDA that can be denoted by a Petri net under the self-sequential collective token interpretation, can also be denoted by a net under the self-concurrent collective token interpretation, and likewise for nets under the individual token interpretations. The construction of N^{ss} above, reducing the self-sequential to the self-concurrent interpretation of nets, is well known [17]. The point of the proof above is to some extent just a sanity check on the definitions of $\mathcal{A}_{\mathrm{CT}}$, $\mathcal{A}^{ss}_{\mathrm{CT}}$, $\mathcal{A}_{\mathrm{IT}}$ and $\mathcal{A}^{ss}_{\mathrm{IT}}$.

The following strengthening of Theorem 2 says that the self-sequential collective token interpretation is *subsumed* by the self-concurrent collective token interpretation, in the sense that, up to isomorphism of reachable parts of associated HDA, the class of all Petri nets under the self-sequential collective token interpretation is equally expressive as a subclass of nets on which the two interpretations coincide. Likewise, the self-sequential individual token interpretation is subsumed by the self-concurrent individual token one.

Definition 14 A Petri net is *self-sequential* if in no reachable ST-marking a transition occurs twice.

Theorem 3 *Let* N *be a Petri net.*

(1) *Then* N^{ss} *is self-sequential,* $\mathcal{A}^{ss}_{\mathrm{CT}}(\mathrm{N}^{ss}) \cong_{\mathcal{R}} \mathcal{A}^{ss}_{\mathrm{CT}}(\mathrm{N})$ *and* $\mathcal{A}^{ss}_{\mathrm{IT}}(\mathrm{N}^{ss}) \cong_{\mathcal{R}} \mathcal{A}^{ss}_{\mathrm{IT}}(\mathrm{N})$.

(2) *If* N *is self-sequential, then* $\mathcal{A}^{ss}_{\mathrm{CT}}(\mathrm{N}) \cong_{\mathcal{R}} \mathcal{A}_{\mathrm{CT}}(\mathrm{N})$ *and* $\mathcal{A}^{ss}_{\mathrm{IT}}(\mathrm{N}) \cong_{\mathcal{R}} \mathcal{A}_{\mathrm{IT}}(\mathrm{N})$.

Proof. If N is self-sequential, trivially $\mathcal{R}(\mathcal{A}^{ss}_{\mathrm{CT}}(\mathrm{N})) = \mathcal{R}(\mathcal{A}_{\mathrm{CT}}(\mathrm{N}))$, and therefore $\mathcal{A}^{ss}_{\mathrm{CT}}(\mathrm{N}) \cong_{\mathcal{R}} \mathcal{A}_{\mathrm{CT}}(\mathrm{N})$. Likewise, $\mathcal{R}(\mathcal{A}^{ss}_{\mathrm{IT}}(\mathrm{N})) = \mathcal{R}(\mathcal{A}_{\mathrm{IT}}(\mathrm{N}))$, considering that self-sequential nets can have no transitions t with ${}^\bullet t = 0$. This yields (2).

That N^{ss} is self-sequential is trivial, and, using this, the remaining statements in (1) follow from (2) and Theorem 2. □

Theorem 4 below says that, up to isomorphism of reachable parts of associated HDA, the two individual token interpretations are equally expressive, and subsumed by each of the collective token interpretations: the class of all Petri nets under either of the individual token interpretations is equally expressive as a subclass of nets on which all four interpretations coincide.

Definition 15 A Petri net is a *unique-occurrence net* if $\forall t \in T.\ {}^\bullet t > 0$ (i.e. it is a standard net), $\forall s \in S.\ I(s) + \Sigma_{t \in T} F(t,s) = 1$ and the flow relation F is well-founded, i.e. there is no infinite alternating sequence $x_0, x_1, \ldots$ of places and transitions such that $F(x_{i+1}, x_i) > 0$ for $i \in \mathbb{N}$.

This class of nets is a close relative of the class of *occurrence nets* of Winskel [30]; it just lacks the requirements that cause the elimination of unreachable places and transitions.

Definition 16 Two Petri nets P and Q are *isomorphic*, written $P \cong Q$, if they differ only in the names of their places and transitions, i.e. if there are bijections $\beta : S^{\mathrm{P}} \to S^{\mathrm{Q}}$ and $\eta : T^{\mathrm{P}} \to T^{\mathrm{Q}}$ such that, for $s \in S^{\mathrm{P}}$ and $t \in T^{\mathrm{P}}$: $I^{\mathrm{Q}}(\beta(s)) = I^{\mathrm{P}}(s)$, $F^{\mathrm{Q}}(\beta(s), \eta(t)) = F^{\mathrm{P}}(s,t)$, $F^{\mathrm{Q}}(\eta(t), \beta(s)) = F^{\mathrm{P}}(t,s)$ and $l^{\mathrm{Q}}(\eta(t)) = l^{\mathrm{P}}(t)$.

Lemma 2 *For every Petri net* N*, the net* $\mathrm{N}_\bullet$ *is a unique-occurrence net. Moreover, if* N *is a unique-occurrence net, then* $\mathrm{N}_\bullet \cong \mathrm{N}$.

Proof. The first statement follows immediately from the construction of $N_\bullet$, the well-foundedness of F (i.e. the well-foundedness of $\prec$ in the proof of Proposition 1) being a consequence of the recursive nature of Definition 10.

The second statement follows with induction on the well-founded order F, using the mappings β and η of Section 6.2. □

Lemma 2 tells that in a unique-occurrence net there is a bijective correspondence between places and token occurrences, and between transitions and transition firings. In particular, in a run of a net each place will be visited at most once, and each transition will fire at most once. Hence the name "unique-occurrence nets". It follows that unique-occurrence nets are self-sequential.

Theorem 4 *Let* N *be a Petri net.*

(1) *Then* $\mathrm{N}_\bullet$ *is a unique-occurrence net satisfying* $\mathcal{A}_{\mathrm{IT}}(\mathrm{N}_\bullet) \cong_{\mathcal{R}} \mathcal{A}_{\mathrm{IT}}(\mathrm{N})$.

(2) $(\mathrm{N}^{ss})_\bullet$ *is a unique-occurrence net satisfying* $\mathcal{A}^{ss}_{\mathrm{IT}}((\mathrm{N}^{ss})_\bullet) \cong_{\mathcal{R}} \mathcal{A}^{ss}_{\mathrm{IT}}(\mathrm{N})$.

(3) *If* N *is a unique-occurrence net, then*

$$\mathcal{A}^{ss}_{\mathrm{IT}}(\mathrm{N}) \cong_{\mathcal{R}} \mathcal{A}_{\mathrm{IT}}(\mathrm{N}) \cong_{\mathcal{R}} \mathcal{A}_{\mathrm{CT}}(\mathrm{N}) \cong_{\mathcal{R}} \mathcal{A}^{ss}_{\mathrm{CT}}(\mathrm{N}).$$

Proof. The left and right statement of (3) follow from Theorem 3, given that unique-occurrence nets are self-sequential. As it is easy to check that $\mathrm{P} \cong \mathrm{Q} \Rightarrow \mathcal{A}_{\mathrm{CT}}(\mathrm{P}) \cong \mathcal{A}_{\mathrm{CT}}(\mathrm{Q})$ for Petri nets P and Q, Lemma 2 implies that $\mathcal{A}_{\mathrm{CT}}(\mathrm{N}_\bullet) \cong \mathcal{A}_{\mathrm{CT}}(\mathrm{N})$ for unique-occurrence nets N. Together with Theorem 1, this yields the middle statement of (3). By Lemma 2, $\mathrm{N}_\bullet$ and $(\mathrm{N}^{ss})_\bullet$ are unique-occurrence nets. Finally, $\mathcal{A}_{\mathrm{IT}}(\mathrm{N}_\bullet) \cong_{\mathcal{R}} \mathcal{A}_{\mathrm{CT}}(\mathrm{N}_\bullet) \cong_{\mathcal{R}} \mathcal{A}_{\mathrm{IT}}(\mathrm{N})$ by (3) and Theorem 1, and $\mathcal{A}^{ss}_{\mathrm{IT}}((\mathrm{N}^{ss})_\bullet) \cong_{\mathcal{R}} \mathcal{A}_{\mathrm{CT}}((\mathrm{N}^{ss})_\bullet) \cong_{\mathcal{R}} \mathcal{A}_{\mathrm{IT}}(\mathrm{N}^{ss}) \cong_{\mathcal{R}} \mathcal{A}^{ss}_{\mathrm{IT}}(\mathrm{N})$ by (3), Theorem 1 and Theorem 2. □

Together, Theorems 1 and 2 yield the expressiveness hierarchy of Figure 6.

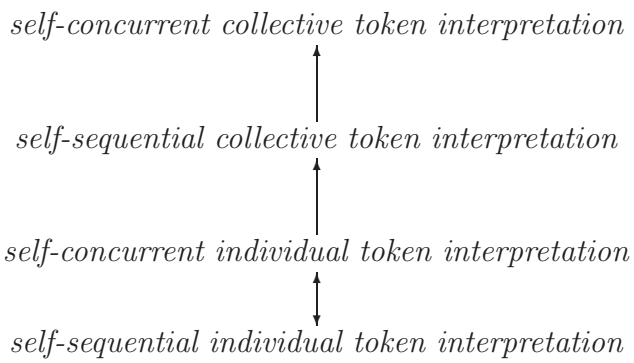

Fig. 6. Relative expressiveness of four computational interpretations of Petri nets

Motivated by this hierarchy, the self-concurrent collective token interpretation of Petri nets will be my default; this is the interpretation that comes with the Petri net entries in Figure 1. Instead of comparing the expressiveness of Petri nets under any of the other interpretations with that of other models of concurrency one can just as well talk about the expressiveness of the relevant subclass of Petri nets.

In order to integrate the hierarchies of Figures 5 and 6, it pays to consider higher dimensional automata up to a semantic equivalence coarser than isomorphism of reachable parts. I will define such equivalences in Section 8, using the material of Section 7. It turns out that up to history preserving bisimulation equivalence the self-sequential and the self-concurrent collective token interpretations of Petri nets coincide.

7 Homotopy for higher dimensional automata

7.1 Naming the faces of n-cells

In order to name the faces of a hypercube, following [67], I use the bits 0, ⌟ and 1 to indicate whether a transition has not yet started, is active, or has terminated, respectively. In other work, the bit ⌟ is sometimes called $\frac{1}{2}$, A, or T for *active* or *in transition*, and yet others use the bits 0, 1, 2, or $-$, 0, $+$.

Since an n-cell represents a list of n transitions being active concurrently, each of its 3^n faces can be represented by a list $b_1 \cdots b_n$ of n such bits, where b_i indicates the status of the i^{th} transition in the list. Thus, the dimension of the face $b_1 \cdots b_n$ is given by the number of ⌟s in the list, ⌟ $\cdots$ ⌟ is the identity map, and ⌟ $\cdots$ ⌟0⌟ $\cdots$ ⌟ (resp. ⌟ $\cdots$ ⌟1⌟ $\cdots$ ⌟), with only $b_i \neq$ ⌟, denotes the map s_i (resp. t_i). The face $b_1 \cdots b_n$ can also be expressed as $\alpha_1 \circ \ldots \circ \alpha_n$ with $\alpha_i = s_i$ if $b_i = 0$, $\alpha_i = t_i$ if $b_i = 1$, and $\alpha_i = Id$, the identity map, if $b_i =$ ⌟. Using this convention, the 2^n 0-dimensional corners of an n-cell q are named by lists $b_1 \cdots b_n$ with $b_i \in \{0, 1\}$, and the $n \cdot 2^{n-1}$ 1-dimensional edges of q by lists containing exactly one occurrence of ⌟.

7.2 Labelling n-cells

As an HDA is required to satisfy $l(s_i(q)) = l(t_i(q))$ for all $q \in Q_2$ and $i = 1, 2$, it follows that, for each $q \in Q_n$ and each $1 \leq i \leq n$, the 2^{n-1} edges $b_0 \cdots b_{i-1}$⌟$b_{i+1} \cdots b_n$ of q, with $b_j \in \{0, 1\}$ for $j \neq i$, all have the same label. Calling this label $l_i(q)$, the labelling function $l : Q_1 \to A$ can be extended to $Q := \bigcup_{k=0}^{\infty} Q_k$ by $l(q) = l_1(q) \cdots l_n(q)$ for $q \in Q_n$. Thus, the label of an n-dimensional hypercube q is the list of the labels of the n transitions whose concurrent execution is represented by q.

7.3 Paths and their observable content

Definition 17 A *path* in a higher dimensional automaton (Q, s, t, I, F, l) is a sequence of pairs $(\partial_1, q_1)(\partial_2, q_2) \cdots (\partial_m, q_m)$, denoted $I \underline{\partial_1} q_1 \underline{\partial_2} q_2 \underline{\partial_3} \cdots \underline{\partial_m} q_m$, with $q_k \in Q$ and $\partial_k \in \{s_i, t_i \mid i \geq 1\}$ for $1 \leq k \leq m$, such that

$$q_{k-1} = s_i(q_k) \text{ if } \partial_k = s_i \qquad \text{and} \qquad q_k = t_i(q_{k-1}) \text{ if } \partial_k = t_i.$$

Here $q_0 := I$, i.e. I consider only paths starting from the initial state. One writes $end(\pi)$ for q_m.

A path π in an HDA A represents a partial run of the system represented by A in which between every two consecutive states an action starts or terminates. It represents a total run iff $end(\pi) \in F$. If $\partial_k = s_i$, the transition from q_{k-1} to q_k represents the start of the action $l_i(q_k)$, and if $\partial_k = t_i$, it represents the termination of the action $l_i(q_{k-1})$.

Definition 18 Write *split-trace*(π) for the sequence $\sigma_1 \cdots \sigma_m$, where $\sigma_k = l_i(q_k)^+$ if $\partial_k = s_i$, and $\sigma_k = l_i(q_{k-1})^-$ if $\partial_k = t_i$.

Split-trace(π) approximates the observable content of a path π. It consists of the sequence of starts a^+ and terminations a^- of actions a occurring during the run represented by π. The HDA of Figure 2 in which the square is not filled in, for instance, has only two maximal paths, whose split-traces are $a^+a^-b^+b^-$ and $b^+b^-a^+a^-$. (The prefixes of these sequences are also split-traces of paths.) The HDA of Figure 2 in which the square *is* filled in moreover has paths with split-traces like $a^+b^+b^-a^-$.

If it is possible to keep track of parallel occurrences of the same action, a split-trace falls short of representing the full observable content of a path. The observable behaviour of a real-time execution would consist of a set of action occurrences, with for each action occurrence an interval given by a start time and a termination time, indicating the period during with the action takes place. Abstracting from real-time information, what remains is a split-trace σ—a sequence $\sigma = \sigma_1 \cdots \sigma_m$ of *action phases*, each being a start a^+ or a termination a^- of an action a—together with an injective function *start* from the (indices of) termination phases in σ to the (indices of) start phases in σ. This function tells for every termination of an action occurrence in σ where that action occurrence starts. Naturally, for every termination phase $\sigma_\ell = a^-$ in σ, one has $start(\ell) < \ell$ and $\sigma_{start(\ell)} = a^+$. Such an annotated split-trace is called an *ST-trace*. It can be compactly represented by writing $a^{start(\ell)}$ for σ_ℓ whenever $\sigma_\ell = a^-$. A formal definition of the ST-trace of a path in a higher dimensional automaton will follow in Section 7.5.

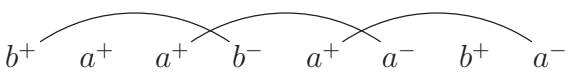

Fig. 7. An ST-trace

A typical ST-trace is depicted as Figure 7. It lists the starts and terminations of actions occurring in a path, and additionally links the start and termination of the same action. Its compact representation is $b^+a^+a^+b^1a^+a^3b^+a^5$. Note that it contains more information than the underlying split-trace. ST-traces were introduced in [8] in the context of event structures. Arguably, they constitute the best formalisation of the observable content of execution paths.

Paths and their ST-traces lack the possibility to express that action phases happen simultaneously. However, in higher dimensional automata two action phases can occur simultaneously iff they can occur in either order. Therefore, considering only paths in which all action phases are totally ordered does not lead to a decrease in expressive power.

7.4 Homotopy

Two paths in a higher dimensional automaton can be considered equivalent if they differ merely in the timing of causally independent actions. This applies for instance to the path ab and the path ba in the automaton of Figure 4, given that the actions a and b are causally independent. However, it would

not apply when the square is not filled in, as this signifies mutual exclusion, and the relative order of a and b would matter. As observed by Pratt [24], this notion of equivalence can be formalised beautifully by means of what he calls "monoidal homotopy". When seeing a higher dimensional automaton as a structure composed of higher dimensional cubes embedded in a higher dimensional Euclidean space, two paths are *homotopic* if one can be obtained out of the other by a continuous transformation, keeping the begin and endpoints the same, and allowing as intermediate stages of the transformation arbitrary paths going through the insides of higher dimensional cells, as long as they are monotonically increasing when projected on the axes of the cells they are going through. The path drawn in Figure 4 for instance could be one of the uncountably many stages in the continuous transformation of ab into ba.

This form of homotopy differs from the standard homotopy used in topology in that the directed nature of the underlying space needs to be preserved during transformations. Therefore it is called *monoidal homotopy* [24], or *directed homotopy* (*dihomotopy*) [63], as opposed to *group homotopy*, although in the context of higher dimensional automata it is simply called *homotopy*.

The following discrete analog of continuous deformation defines the same concept of homotopy without involving the notion of Euclidean space.

Definition 19 Two paths π and π' are *adjacent*—denoted $\pi \leftrightarrow \pi'$—if one can be obtained from the other by replacing, for $q, q' \in Q$ and $i < j$,

- a segment $\overset{s_i}{\underline{\ }} q \overset{s_j}{\underline{\ }}$ by $\overset{s_{j-1}}{\underline{\ }} q' \overset{s_i}{\underline{\ }}$,
- a segment $\overset{t_j}{\underline{\ }} q \overset{t_i}{\underline{\ }}$ by $\overset{t_i}{\underline{\ }} q' \overset{t_{j-1}}{\underline{\ }}$,
- a segment $\overset{s_i}{\underline{\ }} q \overset{t_j}{\underline{\ }}$ by $\overset{t_{j-1}}{\underline{\ }} q' \overset{s_i}{\underline{\ }}$,
- or a segment $\overset{s_j}{\underline{\ }} q \overset{t_i}{\underline{\ }}$ by $\overset{t_i}{\underline{\ }} q' \overset{s_{j-1}}{\underline{\ }}$.

Homotopy is the reflexive and transitive closure of adjacency.

The third adjacency replacement above can be motivated as follows: suppose we have a list of n actions, numbered 1 to n, and we first insert an action a at position i (thereby incrementing the slot-numbers $\geq i$ by one) and subsequently delete the j^{th} action ($j > i$) from the list (thereby decrementing the slot-numbers $> j$ by one), then we get the same result as when we first delete the $(j-1)^{th}$ action (thereby decrementing the slot-numbers $\geq j$ by one) and subsequently insert the action a at position i (incrementing the slot-numbers $\geq i$). The other replacements are motivated in a similar way.

The paths with split-traces $a^+a^-b^+b^-$ and $b^+b^-a^+a^-$ in Figure 4 for instance are homotopic, because the first can be transformed into the second through four adjacency replacements, namely (on the level of split-traces)

$$a^+a^-b^+b^- \leftrightarrow a^+b^+a^-b^- \leftrightarrow b^+a^+a^-b^- \leftrightarrow b^+a^+b^-a^- \leftrightarrow b^+b^-a^+a^-.$$

Homotopic paths share their endpoints. A homotopy class of paths in a higher dimensional automaton (with endpoint q) is called a *history* (of q). Histories form the analog of paths, after abstraction from the order of causally independent action occurrences.

7.5 Matching starts and terminations of action occurrences in paths

The following proposition illustrates the agreement between Definition 19 and the cubical laws of Definition 1.

Proposition 2 *For every segment $p \overset{s_i}{\underline{\ }} q \overset{t_j}{\underline{\ }} r$ with $i \neq j$, and for every segment $p \overset{s_i}{\underline{\ }} q \overset{s_j}{\underline{\ }} r$ or $p \overset{t_i}{\underline{\ }} q \overset{t_j}{\underline{\ }} r$ in a path π in an HDA* A*, there exists a unique path π' in* A*, adjacent to π, that can be obtained from π by replacing the indicated segment in the manner described in Definition 19 (going either right or left).*

Proof. Suppose π contains $p \overset{s_i}{\underline{\ }} q \overset{t_j}{\underline{\ }} r$, with $i < j$. Then $p = s_i(q)$ and $r = t_j(q)$. By the cubical laws in Definition 1, $t_{j-1}(p) = t_{j-1}((s_i(q)) = s_i(t_j(q)) = s_i(r)$. Hence the unique replacement is $p \overset{t_{j-1}}{\underline{\ }} q' \overset{s_i}{\underline{\ }} r$ with $q' = t_{j-1}(p) = s_i(r)$.

Likewise, suppose π contains $p \overset{s_i}{\underline{\ }} q \overset{s_j}{\underline{\ }} r$, with $i < j$. Then $p = s_i(q)$ and $q = s_j(r)$. By the cubical laws (*) in Definition 1 one has $p = s_{j-1}(s_i(r))$. So the unique replacement is $p \overset{s_{j-1}}{\underline{\ }} q' \overset{s_i}{\underline{\ }} r$ with $q' = s_i(r)$.

The other four cases go similarly. □

The intuition is that when two actions happen concurrently, they can start in either order as well as terminate in either order; moreover, if it is possible for action a to start before b terminates, b could just as well terminate before a starts, provided a and b are distinct action occurrences. Note, however, that Proposition 2 does not apply to segments $p \overset{t_i}{\underline{\ }} q \overset{s_j}{\underline{\ }} r$. If the termination of one action precedes the start of another, it may be that there is a causal link between the two that prevents this order from being interchanged.

Definition 20 Two paths π and π' are *ℓ-adjacent*—denoted $\pi \overset{\ell}{\leftrightarrow} \pi'$—if π' can be obtained from $\pi = I \overset{\partial_1}{\underline{\ }} q_1 \overset{\partial_2}{\underline{\ }} q_2 \overset{\partial_3}{\underline{\ }} \cdots \overset{\partial_m}{\underline{\ }} q_m$ $(m > \ell)$ by an adjacency replacement of the segment $\overset{\partial_\ell}{\underline{\ }} q_\ell \overset{\partial_{\ell+1}}{\underline{\ }}$ of π, inducing a swap of the action phases σ_ℓ and $\sigma_{\ell+1}$ in the split-trace $\sigma = \sigma_1 \cdots \sigma_m$ of π.

Assume $\partial_{\ell+1} = t_i$, i.e. $\sigma_{\ell+1}$ is the termination $l_i(q_\ell)^-$ of an action $l_i(q_\ell)$. In case $\partial_\ell = s_i$ we have that σ_ℓ is the start $l_i(q_\ell)^+$ of the very same occurrence of the action $l_i(q_\ell)$ in π. In this case σ_ℓ and $\sigma_{\ell+1}$ cannot be swapped: there is no path π' such that $\pi \overset{\ell}{\leftrightarrow} \pi'$. Proposition 2 tells that in all other cases (i.e. when $\partial_\ell \neq s_i$) σ_ℓ and $\sigma_{\ell+1}$ can be swapped: there exists a unique path π' in A with $\pi \overset{\ell}{\leftrightarrow} \pi'$. This makes it possible to tell which action phase in *split-trace*(π) is the start of the action occurrence whose termination happens as phase $\sigma_{\ell+1}$: it is the unique phase σ_k such that $\pi \overset{\ell}{\longleftrightarrow} \pi_\ell \overset{\ell-1}{\longleftrightarrow} \pi_{\ell-1} \overset{\ell-2}{\longleftrightarrow} \cdots \overset{k+1}{\longleftrightarrow} \pi_{k+1} \overset{k}{\not\longleftrightarrow} \pi_k$.

Definition 21 Let $\pi = I \overset{\partial_1}{\underline{\ }} q_1 \overset{\partial_2}{\underline{\ }} q_2 \overset{\partial_3}{\underline{\ }} \cdots \overset{\partial_m}{\underline{\ }} q_m$ be a path in a higher dimensional automaton. For $1 \leq \ell \leq m$ such that ∂_ℓ denotes a termination phase, let *start*(ℓ) denote the unique number k such that

$$\pi \overset{\ell-1}{\longleftrightarrow} \pi_{\ell-1} \overset{\ell-2}{\longleftrightarrow} \cdots \overset{k+1}{\longleftrightarrow} \pi_{k+1} \overset{k}{\not\longleftrightarrow} \pi_k.$$

Now *ST-trace*(π) is the sequence obtained from *split-trace*$(\pi) = \sigma_1 \cdots \sigma_k$, by replacing σ_ℓ by $a^{start(\ell)}$ whenever $\sigma_\ell = a^-$.

8 Bisimulation semantics for HDA

Using the material of Section 7, I now extend the main forms of bisimulation equivalence found in the literature that do not involve a special treatment of *hidden* or *internal* actions, to higher dimensional automata. For *interleaving bisimulation equivalence* this is trivial: it is just the standard notion of bisimulation equivalence on ordinary automata found in the literature [20,1], applied to higher dimensional automata by ignoring their higher dimensional cells. *ST-bisimulation equivalence* [14] is a branching time respecting equivalence that takes causality into account to the extent that it is expressible by durational actions overlapping in time. *History preserving bisimulation equivalence* [26,11] is the coarsest equivalence that fully respects branching time, causality and their interplay. *Hereditary history preserving bisimulation equivalence* [3] is a variant of the latter that strongly respects the internal structure of processes, while still collapsing choices between indistinguishable courses of action (i.e. satisfying the CCS law $x + x = x$ [20]).

By Definition 17, the empty path in a higher dimensional automaton A starts and ends in the initial state of A and hence is denoted I^{A}. I write $\pi \to \pi'$ if π is a prefix of a path π', i.e., if π' is an extension of π.

Definition 22 Two higher dimensional automata A and B are *history preserving bisimulation equivalent* if there exists a binary relation R between their paths—a *history preserving bisimulation*—such that

(1) the empty paths in A and B are related: $I^{\mathrm{A}} R I^{\mathrm{B}}$,
(2) if $\pi R \rho$ then then $ST\text{-}trace(\pi) = ST\text{-}trace(\rho)$,
(3) if $\pi R \rho$ and $\pi \overset{\ell}{\leftrightarrow} \pi'$ then $\exists \rho'$ with $\rho \overset{\ell}{\leftrightarrow} \rho'$ and $\pi' R \rho'$,
(4) if $\pi R \rho$ and $\rho \overset{\ell}{\leftrightarrow} \rho'$ then $\exists \pi'$ with $\pi \overset{\ell}{\leftrightarrow} \pi'$ and $\pi' R \rho'$,
(5) if $\pi R \rho$ and $\pi \to \pi'$ then $\exists \rho'$ with $\rho \to \rho'$ and $\pi' R \rho'$,
(6) if $\pi R \rho$ and $\rho \to \rho'$ then $\exists \pi'$ with $\pi \to \pi'$ and $\pi' R \rho'$,
(7) and if $\pi R \rho$ then $end(\pi) \in F^{\mathrm{A}} \Leftrightarrow end(\rho) \in F^{\mathrm{B}}$.

A and B are *hereditary history preserving bisimulation equivalent* if there exists a history preserving bisimulation R between their paths that moreover satisfies

(8) if $\pi R \rho$ and $\pi' \to \pi$ then $\exists \rho'$ with $\rho' \to \rho$ and $\pi' R \rho'$, and
(9) if $\pi R \rho$ and $\rho' \to \rho$ then $\exists \pi'$ with $\pi' \to \pi$ and $\pi' R \rho'$.

ST-bisimulation equivalence between HDA is defined exactly as history preserving bisimulation equivalence, but dropping clauses (3) and (4).

Note that in the presence of clause (2), related paths have the same length. Hence clauses (8) and (9) are equivalent. I listed them both solely to stress the symmetric nature of the definition. It is not hard to see that the notion of ST-bisimulation equivalence would not change upon adding clauses (8) and (9), but because of clauses (3) and (4), this does not apply to history preserving bisimulation equivalence. The clauses (3) and (4) express that the causal relations between action phases in the ST-traces of two related paths

are the same, for these relations are determined by the space of all allowed permutations of action phases. It follows from the material in Section 7.5 that in the presence of clauses (3) and (4), clause (2) can be simplified to

$$\text{if } \pi R \rho \text{ then } \textit{split-trace}(\pi) = \textit{split-trace}(\rho).$$

This clause can be omitted altogether when writing the names of action phases above the arrows in clauses (5), (6), (8) and (9), as is customary in many accounts of bisimulation relations found in the literature.

8.1 History preserving bisimulation semantics for 1-dimensional automata

The definitions of paths, their observable content, adjacency, homotopy and (hereditary) history preserving bisimulation equivalence can be drastically simplified when dealing with higher dimensional automata of the form $\mathcal{A}(\mathrm{G})$ with G an extensional 1-dimensional automaton; the reason being that all higher dimensional information is encoded in their 1-dimensional parts. As it is not necessary to consider paths that travel through the inside of squares, paths may be built from entire transitions rather than transition phases. Below I define the mentioned concepts directly on extensional 1-dimensional automata under the concurrent interpretation.

Definition 23 A *path* in an extensional automaton (S, T, I, F) is a sequence of pairs $(a_1, q_1)(a_2, q_2) \cdots (a_m, q_m)$, denoted $I \underline{a_1} q_1 \underline{a_2} q_2 \underline{a_3} \cdots \underline{a_m} q_m$, with $q_k \in S$ and $a_k \in A$ for $1 \leq k \leq m$, such that $(q_{k-1}, a, q_k) \in T$. Here $q_0 := I$, i.e. I consider only paths starting from the initial state. One writes $\textit{end}(\pi)$ for q_m.

Write $\textit{trace}(\pi)$ for the sequence $a_1 \cdots a_m$.

Two paths π and π' are *ℓ-adjacent*—denoted $\pi \overset{\ell}{\leftrightarrow} \pi'$—if $\pi = I \underline{a_1} \cdots \underline{a_m} q_m$ with $m > \ell$ and π' is obtained from π by replacing, for some $q \in Q$, the segment $\underline{a_\ell} q_\ell \underline{a_{\ell+1}}$ by $\underline{a_{\ell+1}} q \underline{a_\ell}$, provided that $a_\ell \neq a_{\ell+1}$ or $q \neq q_\ell$.

$\textit{Trace}(\pi)$ is the observable content of a path π. Again *homotopy* can be defined as the reflexive and transitive closure of *adjacency*, obtained by abstracting from ℓ. As before, I write $\pi \to \pi'$ if π is a prefix of a path π'.

Definition 24 Two extensional automata G and H are *history preserving bisimulation equivalent* if there exists a binary relation R between their paths —a *history preserving bisimulation*—such that

(1) the empty paths in G and H are related,
(2) if $\pi R \rho$ then then $\textit{trace}(\pi) = \textit{trace}(\rho)$,
(3) if $\pi R \rho$ and $\pi \overset{\ell}{\leftrightarrow} \pi'$ then $\exists \rho'$ with $\rho \overset{\ell}{\leftrightarrow} \rho'$ and $\pi' R \rho'$,
(4) if $\pi R \rho$ and $\rho \overset{\ell}{\leftrightarrow} \rho'$ then $\exists \pi'$ with $\pi \overset{\ell}{\leftrightarrow} \pi'$ and $\pi' R \rho'$,
(5) if $\pi R \rho$ and $\pi \to \pi'$ then $\exists \rho'$ with $\rho \to \rho'$ and $\pi' R \rho'$,
(6) if $\pi R \rho$ and $\rho \to \rho'$ then $\exists \pi'$ with $\pi \to \pi'$ and $\pi' R \rho'$,
(7) and if $\pi R \rho$ then $\textit{end}(\pi) \in F^{\mathrm{G}} \Leftrightarrow \textit{end}(\rho) \in F^{\mathrm{H}}$.

The equivalence is *hereditary* if R moreover satisfies

(8) if $\pi R \rho$ and $\pi' \to \pi$ then $\exists \rho'$ with $\rho' \to \rho$ and $\pi' R \rho'$.

The next proposition says that this simplified definition for 1-dimensional automata agrees with the general definition for higher dimensional automata.

Proposition 3 *Two extensional automata* G *and* H *are (hereditary) history preserving bisimulation equivalent in the sense of Definition 24 iff their higher dimensional representations* $\mathcal{A}(\mathrm{G})$ *and* $\mathcal{A}(\mathrm{H})$ *are (hereditary) history preserving bisimulation equivalent in the sense of Definition 22.*

Proof. Straightforward, albeit a bit tedious. □

Dropping clauses (3) and (4) in Definition 24 yields a definition of interleaving bisimulation; ST-bisimulation equivalence on extensional automata cannot be defined that way.

8.2 History preserving bisimulation semantics for configuration structures

Any semantic equivalence $\sim$ on 1- or higher dimensional automata is inherited by rooted finitary configuration structures by letting $\mathrm{C} \sim \mathrm{D}$ iff $\mathcal{G}(\mathrm{C}) \sim \mathcal{G}(\mathrm{D})$. In particular, Definition 24 extends (hereditary) history preserving bisimulation equivalence to arbitrary rooted finitary configuration structures. However, on stable configuration structures (Definition 6) these equivalences were already defined in [11]. Therefore I need to prove that for stable configuration structures the new definition (24) agrees with the one of [11]. As a first step, I reformulate the new definition for rooted finitary configuration structures by eliminating the reference to the embedding $\mathcal{G}$.

Definition 25 A *path* in a configuration structure C is a sequence of distinct events $e_1e_2\cdots e_m$ such that $\forall k \leq n.\ \{e_1,\ldots,e_k\} \in C^{\mathrm{C}}$. Write $end(\pi)$ for $\{e_1,\ldots,e_m\}$ and $trace(\pi)$ for the sequence $l^{\mathrm{C}}(e_1)\cdots l^{\mathrm{C}}(e_m)$. Two paths π and π' in C are *ℓ-adjacent*—denoted $\pi \overset{\ell}{\leftrightarrow} \pi'$—if $\pi = e_1\cdots e_m$ with $m > \ell$ and π' is obtained from π by swapping e_ℓ and $e_{\ell+1}$. Write $\pi \to \pi'$ if π is a prefix of π'.

Proposition 4 *Let* C *and* D *be rooted configuration structures. Then* $\mathcal{G}(\mathrm{C})$ *and* $\mathcal{G}(\mathrm{D})$ *are (hereditary) history preserving bisimulation equivalent iff there exists a relation between the paths of* C *and* D *(defined above) such that*

(1) the empty paths in C *and* D *are related,*
(2) if $\pi R\rho$ then then $trace(\pi) = trace(\rho)$,
(3) if $\pi R\rho$ and $\pi \overset{\ell}{\leftrightarrow} \pi'$ then $\exists\rho'$ with $\rho \overset{\ell}{\leftrightarrow} \rho'$ and $\pi' R\rho'$,
(4) if $\pi R\rho$ and $\rho \overset{\ell}{\leftrightarrow} \rho'$ then $\exists\pi'$ with $\pi \overset{\ell}{\leftrightarrow} \pi'$ and $\pi' R\rho'$,
(5) if $\pi R\rho$ and $\pi \to \pi'$ then $\exists\rho'$ with $\rho \to \rho'$ and $\pi' R\rho'$,
(6) if $\pi R\rho$ and $\rho \to \rho'$ then $\exists\pi'$ with $\pi \to \pi'$ and $\pi' R\rho'$,
(7) and if $\pi R\rho$ then $end(\pi) \in F^{\mathrm{C}} \Leftrightarrow end(\rho) \in F^{\mathrm{D}}$.

The equivalence is hereditary *if R moreover satisfies*

(8) if $\pi R\rho$ and $\pi' \to \pi$ then $\exists\rho'$ with $\rho' \to \rho$ and $\pi' R\rho'$.

Proof. Trivial. □

Let C be a stable configuration structure, and $X \in C^{\mathrm{C}}$. The *causality relation on* X is given by $d \leq_X e$ iff for all $Y \in C^{\mathrm{C}}$ with $Y \subseteq X$ one has $e \in Y \Rightarrow d \in Y$.

Proposition 5 *Let* C *be a stable configuration structure, and* $X \in C^{\mathrm{C}}$. *Then* $\leq_X$ *is a partial order, and the paths* π *in* C *with* $end(\pi) = X$ *are exactly the linearisations of* $\leq_X$.

Proof. The first statement is Proposition 5.2 in [11]. That paths ending in X are linearisations of $\leq_X$ is immediate from the definitions. That each linearisation of $\leq_X$ is a path in C is a restatement of Proposition 5.3 in [11].□

Corollary 3 *Any two paths* π *and* ρ *in a stable configuration structure that satisfy* $end(\pi) = end(\rho)$ *are homotopic.*

In [11] it is shown that a rooted finitary configuration structure is stable if and only if it satisfies the mentioned Propositions 5.2 and 5.3, hence if and only if it satisfies Proposition 5 above. Thus the stable ones are the largest class of rooted finitary configuration structures on which causality can faithfully be expressed in terms of partial orders.

Using this causality relation, (hereditary) history preserving bisimulation equivalence was defined in [11] on stable configuration structures as follows.

Definition 26 Two stable configuration structures C and D are *history preserving bisimulation equivalent* if there is a relation $R \subseteq C^{\mathrm{C}} \times C^{\mathrm{D}} \times \mathcal{P}(E^{\mathrm{C}} \times E^{\mathrm{D}})$ such that $(\emptyset, \emptyset, \emptyset) \in R$ and whenever $(X, Y, f) \in R$ then

- f is an isomorphism between $(X, \leq_X, l^{\mathrm{C}} \restriction X)$ and $(Y, \leq_Y, l^{\mathrm{D}} \restriction Y)$,
- $X \subseteq X' \in C^{\mathrm{C}} \Rightarrow \exists Y', f'$ with $Y \subseteq Y' \in C^{\mathrm{D}}$, $(X', Y', f') \in R$ and $f' \restriction X = f$,
- $Y \subseteq Y' \in C^{\mathrm{D}} \Rightarrow \exists X', f'$ with $X \subseteq X' \in C^{\mathrm{C}}$, $(X', Y', f') \in R$ and $f' \restriction X = f$,
- $X \in F^{\mathrm{C}} \Leftrightarrow Y \in F^{\mathrm{D}}$.

The equivalence is *hereditary* if R moreover satisfies

- $(X, Y, f) \in R \wedge X \supseteq X' \in C^{\mathrm{C}} \Rightarrow (X', f(X'), f \restriction X') \in R$.

This definition also applies to various kinds of event structures, through natural embeddings into stable configuration structures [11].

Theorem 5 *Two stable configuration structures* C *and* D *are (hereditary) history preserving bisimulation equivalent in the sense of Definition 26 iff their automata representations* $\mathcal{G}(\mathrm{C})$ *and* $\mathcal{G}(\mathrm{D})$ *are (hereditary) history preserving bisimulation equivalent in the sense of Definition 24.*

Proof. Let R be a relation between the paths of C and D satisfying the properties of Proposition 4. Define the relation $\widetilde{R} \subseteq C^{\mathrm{C}} \times C^{\mathrm{D}} \times \mathcal{P}(E^{\mathrm{C}} \times E^{\mathrm{D}})$ by $(X, Y, f) \in \widetilde{R}$ iff there are paths $\pi = e_1 \cdots e_m$ in C and $\rho = d_1 \cdots d_m$ in D with $\pi R \rho$, $X = end(\pi)$, $Y = end(\rho)$ and $f(e_k) = d_k$ for $k = 1, ..., m$.

I will show that $\widetilde{R}$ is a history preserving bisimulation between C and D. Using property (1), one obtains $(\emptyset, \emptyset, \emptyset) \in \widetilde{R}$. Now suppose $(X, Y, f) \in \widetilde{R}$.

- By construction f is a bijection between X and Y, and by (2) it satisfies $l^{\mathrm{D}}(f(e)) = l^{\mathrm{C}}(e)$ for $e \in X$. Suppose $d \not\leq_X e$. By Proposition 5 and Corollary 3 there is a path π' in C, homotopic with π, in which e precedes d. So by (3) there is a path ρ' in D, homotopic with ρ, in which $f(e)$ precedes $f(d)$. Hence $f(d) \not\leq_Y f(e)$. Likewise, using (4), one has $f(d) \not\leq_Y f(e) \Rightarrow d \not\leq_X e$. It follows that f is an isomorphism between $(X, \leq_X, l^{\mathrm{C}}{\restriction}X)$ and $(Y, \leq_Y, l^{\mathrm{D}}{\restriction}Y)$.
- Suppose $X \subseteq X' \in C^{\mathrm{C}}$. By Lemma 5.2.5 in [11], there is a path π' in C with $\pi \to \pi'$ and $end(\pi') = X'$. Thus, using (5), $\exists Y', f'$ with $Y \subseteq Y' \in C^{\mathrm{D}}$, $(X', Y', f') \in R$ and $f'{\restriction}X{=}f$.
- Likewise, using (6), one finds that $Y \subseteq Y' \in C^{\mathrm{D}}$ implies $\exists X', f'$ with $X \subseteq X' \in C^{\mathrm{C}}$, $(X', Y', f') \in R$ and $f'{\restriction}X{=}f$.
- $X \in F^{\mathrm{C}} \Leftrightarrow end(\pi) \in F^{\mathrm{C}} \Leftrightarrow end(\rho) \in F^{\mathrm{D}} \Leftrightarrow Y \in F^{\mathrm{D}}$, using (7).

In case R also satisfies property (8), the relation $\widetilde{R}$ is hereditary. Namely if $X \supseteq X' \in C^{\mathrm{C}}$ then by Lemma 5.2.5 in [11] there are paths π' and π'' in C with $\pi' \to \pi''$, $end(\pi') = X'$ and $end(\pi'') = X$. By Corollary 3 π'' and π are homotopic. So by (3) there is a path ρ'' in D, homotopic with ρ, with $\pi'' R \rho''$. Using (8) one obtains $(X', f(X'), f{\restriction}X') \in R$.

Conversely, let $\widetilde{R}$ be a history preserving bisimulation between C and D. Define the binary relation R between the paths of C and D by $\pi R \rho$ iff $|\pi| = |\rho|$, say $\pi = e_1 \cdots e_m$ and $\rho = d_1 \cdots d_m$, and $(end(\pi), end(\rho), f) \in \widetilde{R}$ for f the function given by $f(e_k) = d_k$ for $k = 1, ..., m$. I will show that R satisfies the properties of Proposition 4.

Property (1) follows because $(\emptyset, \emptyset, \emptyset) \in \widetilde{R}$, and (2) because f is an isomorphism. For (3), suppose $\pi \overset{\ell}{\leftrightarrow} \pi'$. As π' is a path in C, by Proposition 5 it must be a linearisation of $\leq_{end(\pi)}$. Since f is an isomorphism between $(end(\pi), \leq_{end(\pi)}, l^{\mathrm{C}}{\restriction}end(\pi))$ and $(end(\rho), \leq_{end(\rho)}, l^{\mathrm{D}}{\restriction}end(\rho))$, the sequence $f(\pi') := d_1 \cdots d_{\ell-1} d_{\ell+1} d_\ell d_{\ell+2} \cdots d_m$ must be a linearisation of $\leq_{end(\rho)}$, and hence a path in D. It follows that $\rho \overset{\ell}{\leftrightarrow} f(\pi')$ and $\pi' R f(\pi')$. Property (4) follows likewise, and (5), (6) and (7) are trivial. In case $\widetilde{R}$ is hereditary, also (8) holds trivially. □

9 A hierarchy of concurrency models

After having defined precisely what the arrows in Figure 1 mean, I will now proceed to argue for their soundness and completeness in describing the relative expressiveness of the models of concurrency under investigation.

9.1 Soundness of the inclusions of Figure 1

It is well known that stable event structures with binary conflict [31] are more expressive than flow event structures [5], which are more expressive than bundle event structures [18], which are in turn more expressive than prime event structures with binary conflict [31], cf. Figure 5. However, this holds

when comparing the families of configurations they can express; the difference disappears when working up to history preserving bisimulation. This follows immediately from the fact that prime and stable event structures with binary conflict specify the same *Scott domains* [30,31], and thus also the same [higher dimensional] automata. Alternatively, a direct proof can be found in [16]. In [16] and [22] it has furthermore been shown that, up to history preserving bisimulation, finitary conflict can be expressed in terms of binary conflict. Thus, by the criteria of this paper, the stable event structures of [30] do not rank higher in expressive power than the ones with binary conflict in [31]. Likewise, the general event structures of [30] do not rank higher in expressive power than the ones with binary conflict in [31]. This shows that up to history preserving bisimulation Figure 5 collapses into the bottom of Figure 1.

A Petri net is *safe* if no reachable marking will every have two tokens in the same place. In [21] the expressive equivalence has been established of the model of safe Petri nets with that of prime event structures with binary conflict. This was done by means of translations between these models that preserve more than history preserving bisimulation equivalence.

Up to history preserving bisimulation equivalence, on safe Petri nets there is no difference between any of the four computational interpretations of Petri nets discussed in Section 6. Moreover, the nets $N_\bullet$, constructed in the proof of Theorem 1, are safe. It follows that under the self-sequential individual token interpretation, as well as under the self-concurrent individual token interpretation, the class of all Petri nets is equally expressive as the class of safe Petri nets (and as the subclass of unique-occurrence nets).

In [12] two forms of *1-unfolding* are defined, one for Petri nets under the self-concurrent collective token interpretation, and one for Petri nets under the self-sequential collective token interpretation. Each of them converts any Petri net into a so-called *1-occurrence net*, a net in which each transition can fire only once. And each of these 1-unfoldings respects history preserving bisimulation equivalence w.r.t. to the computational interpretation of nets that comes with this 1-unfolding. Trivially, on 1-occurrence nets there is no difference between the self-concurrent and the self-sequential interpretation of nets. It follows that, up to history preserving bisimulation equivalence, Petri nets under the self-sequential collective token interpretation are equally expressive as Petri nets under the self-concurrent collective token interpretation.

A Petri net is *pure* if it has no *self-loops*, i.e. there are no places s and transitions t with $F(s,t) > 0$ and $F(t,s) > 0$. In [12] we showed that rooted finitary configuration structures are equally expressive as pure Petri nets under the collective token interpretation (or precisely, each of the two collective token interpretations). Taking into account that pureness is preserved by the two 1-unfoldings, this was done by means of translations between rooted finitary configuration structures and pure 1-occurrence nets that preserve more than history preserving bisimulation equivalence.

This concludes the justification of the arrows in Figure 1.

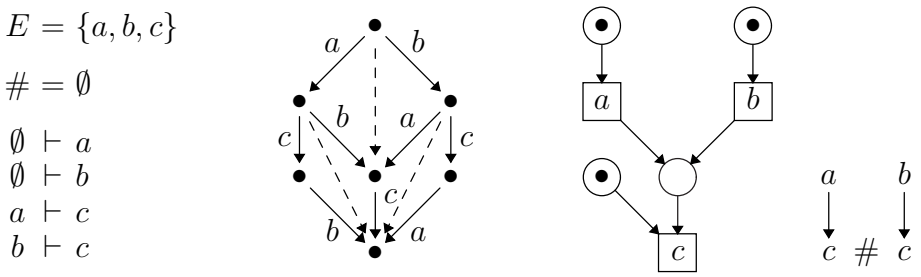

Fig. 8. A system with *disjunctive causality* represented as an event structure of [31], a [higher dimensional] automaton in which the dashed lines indicate that all three squares are filled in, and a Petri net. The last picture is the best representation of the same system as a prime event structure [21,31]. It requires the decomposition of the event c, which is causally dependent on the disjunction of a and b, into two events c_1 and c_2, only one of which may happen: c_1 being causally dependent only on a, and c_2 on b. This prime event structure is ST-bisimulation equivalent to the original one, but not history preserving bisimulation equivalent.

9.2 Completeness of the inclusions of Figure 1

That the model of synchronisation trees is less expressive than that of safe Petri nets is witnessed by the process $a\|b$, the parallel composition of two actions a and b. This process can be represented by the safe Petri net P in Figure 2. However, there is no synchronisation tree which is history preserving bisimulation equivalent, or even ST-bisimulation equivalent, to this process.

The models of prime and stable event structures are less expressive than that of (general) event structures of [31]. This is witnessed by the process of Figure 8, which is representable as an event structure of [31], but, up to history preserving bisimulation equivalence, not as a prime event structure.

In [13], a generalisation of Winskel's event structures called *rooted event structures with finite conflict* is proposed that (up to history preserving bisimulation equivalence) is equally expressive as Petri nets under the collective token interpretation. Also a subclass of *pure* rooted event structures with finite conflict is defined that matches the expressive power of rooted finitary configuration structures and pure nets. An example of a pure Petri net and a pure rooted event structure with finite conflict as in [13] that cannot be represented as an event structure of [30,31] appears in Figure 9.

Figure 10 shows a system represented as a Petri net, that cannot be represented as a pure Petri net, or as an automaton under the concurrent interpretation.

9.2.1 Beyond Petri nets

The final counterexample witnessing the completeness of the expressiveness hierarchy of Figure 1 concerns the process of Figure 11, that is representable as a higher dimensional automaton, and even as an ordinary automaton under the concurrent interpretation, but not as a Petri net.

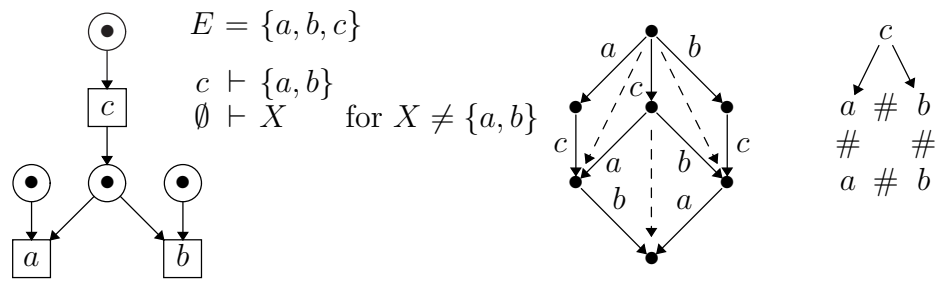

Fig. 9. A system with *resolvable conflict* represented as a pure Petri net, a pure rooted event structure with finite conflict as introduced in [13], and a [higher dimensional] automaton. The events a and b are initially in conflict (only one of them may happen), but as soon as c occurs this conflict is resolved. The last picture is the best representation of the same system as a prime event structure. It yields a system with two maximal runs, in one of which c causes just a, and in the other just b. Again it is ST-bisimulation equivalent to the original, but not history preserving bisimulation equivalent. There is no event structure as in [30,31] that is history preserving bisimulation equivalent to the system above.

Fig. 10. A *2-out-of-3 semaphore*, represented as a Petri net and as a higher dimensional automaton. In the latter, all six squares are supposed to be filled in, but the interior of the cube is not. Up to history preserving bisimulation equivalence this system cannot be represented as an automaton under the concurrent interpretation, because, due to the filled-in squares, the cube shape is unavoidable, and interior of the cube would by default be understood to be filled in. Hence the system can also not be represented as a pure Petri net.

The process displayed in Figure 11 was implemented during my presentation at EXPRESS 2004. Two computer scientists A and B were travelling from one end of the podium to the other. Their task to was perform the actions a resp. b of crossing a line on the podium. Due to strategic placing of obstacles, the only place where this was possible was at a narrow opening between the obstacles that had room for only one of the scientists A and B at a time. This made the actions a and b mutually exclusive, in the sense that they could not occur simultaneously. A third computer scientist, C, was assigned the task c of removing an obstacle that caused the bottleneck to exist. The action c was executed causally independent of a and b. The actions a and b were mutually exclusive only until c occurred, after which they became causally

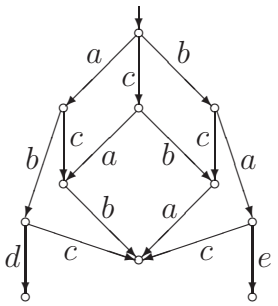

Fig. 11. A process, represented as an ordinary automaton under the concurrent interpretation, that, up to history preserving bisimulation equivalence, cannot be represented as a Petri net.

independent. Finally, a fourth participant was assigned the task of making a statement when a and b had both occurred before the action c started. This statement was going to be d in case A passed the bottleneck before B did, and e in case B passed the bottleneck before A did. Hearing this statement would prevent computer scientist C from carrying out the action c. The resulting process is described by the automaton above, in which all five squares are filled in.

In order to represent the process of Figure 11, up to history preserving bisimulation equivalence, as a Petri net, there must be a single transition representing the action a, regardless of whether it is scheduled before or after b or c. This is because of the concurrency inherent in the example. The same holds for b. However, in a Petri net, firing just these two transitions labelled a and b necessarily leads to a unique state, independent of the order in which a and b occur. This is in contradiction with the fact that the process under consideration has two states reachable by doing only a and b, in which different further actions are possible.

9.3 Comparisons modulo other notions of equivalence

If I compare the models of Figure 1 up to hereditary history preserving bisimulation equivalence the same hierarchy results. This is because all translations used in Section 9.1 even preserve hereditary history preserving bisimulation equivalence. If I compare them merely up to interleaving bisimulation equivalence, all models turn out to be equally expressive. This is because every higher dimensional automaton is trivially interleaving bisimulation equivalent to the 1-dimensional automaton resulting from ignoring its higher dimensional cells, and to the unfolding of that 1-dimensional automaton into a tree.

If I compare the models up to ST-bisimulation equivalence, the model of synchronisation trees is still less expressive than that of event structures, as explained in Section 9.2. I conjecture that all models of Figure 1 other than synchronisation trees are equally expressive, in the sense that any process representable in any of the models can be translated into an ST-bisimulation

equivalent prime event structure. For rooted finitary configuration structures this is Theorem 1 in [12]. My conjecture is that this result extends to higher dimensional automata; in other words, that up to ST-bisimulation equivalence the prime event structures have universal expressivity.

References

[1] Baeten, J.C.M. and W.P. Weijland, "Process Algebra," Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, 1990.

[2] Bednarczyk, M., "Categories of asynchronous systems," Ph.D. thesis, Computer Science, University of Sussex, Brighton (1987).
Available at ftp://ftp.ipipan.gda.pl/marek/phd.ps.gz.

[3] Bednarczyk, M., *Hereditary history preserving bisimulation, or what is the power of the future perfect in program logics*, Technical report, Polish Academy of Sciences, Gdansk (1991).
Available at ftp://ftp.ipipan.gda.pl/marek/historie.ps.gz.

[4] Best, E., R. Devillers, A. Kiehn and L. Pomello, *Concurrent bisimulations in Petri nets*, Acta Informatica **28** (1991), pp. 231–264.

[5] Boudol, G., *Flow event structures and flow nets*, in: I. Guessarian, editor, *Semantics of Systems of Concurrent Processes, Proceedings LITP Spring School on Theoretical Computer Science,* La Roche Posay, France, LNCS **469** (1990), pp. 62–95.

[6] Engelfriet, J., *Branching processes of Petri nets*, Acta Informatica **28(6)** (1991), pp. 575–591.

[7] Glabbeek, R.J. van, *An operational non-interleaved process graph semantics of CCSP* (abstract), in: E.-R. Olderog, U. Goltz and R.J. van Glabbeek, editors, *Combining Compositionality and Concurrency,* Summary of a GMD-Workshop, Königswinter, March 1988, Arbeitspapiere der GMD 320 (1988), pp. 18–19.

[8] Glabbeek, R.J. van, *The refinement theorem for ST-bisimulation semantics*, in: M. Broy and C.B. Jones, editors, Proceedings IFIP TC2 Working Conference on *Programming Concepts and Methods,* Sea of Gallilee, Israel (1990), pp. 27–52, available at http://kilby.stanford.edu/~rvg/pub/STbisimulation.pdf.

[9] Glabbeek, R.J. van, *Bisimulations for higher dimensional automata*, email message (July 7, 1991), available at http://theory.stanford.edu/~rvg/hda.

[10] Glabbeek, R.J. van, *History preserving process graphs*, draft, available at http://kilby.stanford.edu/~rvg/pub/history.draft.dvi (1996).

[11] Glabbeek, R.J. van and U. Goltz, *Refinement of actions and equivalence notions for concurrent systems*, Acta Informatica **37** (2001), pp. 229–327, available at http://boole.stanford.edu/pub/refinement.ps.gz.

[12] Glabbeek, R.J. van and G.D. Plotkin, *Configuration structures (extended abstract)*, in: D. Kozen, editor, Proceedings 10^{th} Annual IEEE Symposium on *Logic in Computer Science*, LICS'95, San Diego, USA (1995), pp. 199–209, available at http://boole.stanford.edu/pub/conf.ps.gz.

[13] Glabbeek, R.J. van and G.D. Plotkin, *Event structures for resolvable conflict*, in: V. Koubek and J. Kratochvil, editors, Proceedings 29^{th} International Symposium on *Mathematical Foundations of Computer Science*, MFCS 2004, Prague, Czech Republic, LNCS (August 2004), available at http://boole.stanford.edu/pub/resolv.ps.gz.

[14] Glabbeek, R.J. van and F.W. Vaandrager, *Petri net models for algebraic theories of concurrency (extended abstract)*, in: J.W. de Bakker, A.J. Nijman and P.C. Treleaven, editors, Proceedings *PARLE, Parallel Architectures and Languages Europe*, Eindhoven, The Netherlands, June 1987, Vol. II: Parallel Languages, LNCS **259** (1987), pp. 224–242, available at http://kilby.stanford.edu/~rvg/pub/petri.pdf.

[15] Glabbeek, R.J. van and F.W. Vaandrager, *The difference between splitting in n and $n+1$*, Information and Computation **136** (1997), pp. 109–142, available at http://boole.stanford.edu/pub/split.pdf.

[16] Glabbeek, R.J. van and F.W. Vaandrager, *Bundle event structures and CCSP*, in: R. Amadio and D. Lugiez, editors, Proceedings *CONCUR 2003*, 14^{th} International Conference on *Concurrency Theory*, Marseille, France, September 2003, LNCS **2761** (2003), pp. 57–71, available at http://boole.stanford.edu/pub/bundle.ps.gz.

[17] Goltz, U. and W. Reisig, *The non-sequential behaviour of Petri nets*, Information and Computation **57** (1983), pp. 125–147.

[18] Langerak, R., "Transformations and Semantics for LOTOS," Ph.D. thesis, Department of Computer Science, University of Twente (1992).

[19] Meseguer, J., U. Montanari and V. Sassone, *On the semantics of Petri nets*, in: W. Cleaveland, editor, Proceedings CONCUR '92, Second International Conference on *Concurrency Theory*, Stony Brook, NY, USA, LNCS **630** (1992), pp. 286–301.

[20] Milner, R., "A Calculus of Communicating Systems," LNCS **92**, Springer, 1980.

[21] Nielsen, M., G.D. Plotkin and G. Winskel, *Petri nets, event structures and domains, part I*, Theoretical Computer Science **13** (1981), pp. 85–108.

[22] Nielsen, M. and G. Winskel, *Petri nets and bisimulation*, Theoretical Computer Science **153** (1996), pp. 211–244.

[23] Papadimitriou, C., "The Theory of Database Concurrency Control," Computer Science Press, 1986.

[24] Pratt, V.R., *Modeling concurrency with geometry*, in: *Proc. 18th Ann. ACM Symposium on Principles of Programming Languages* (1991), pp. 311–322, available at http://boole.stanford.edu/pub/cg.ps.gz.

[25] Pratt, V.R., *Chu spaces*, a summary and large collection of papers available at http://chu.stanford.edu/ (1993–2002).

[26] Rabinovich, A. and B.A. Trakhtenbrot, *Behavior structures and nets*, Fundamenta Informaticae **11** (1988), pp. 357–404.

[27] Serre, J., *Homology singulière des espaces fibrés. application*, Ph.D. thesis, École Normale Supérieure (1951).

[28] Shields, M.W., *Concurrent machines*, The Computer Journal **28** (1985), pp. 449–465.

[29] Stark, E.W., *Concurrent transition systems*, Theoretical Computer Science **64** (1989), pp. 221–269.

[30] Winskel, G., *Event structures*, in: W. Brauer, W. Reisig and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, Part II; Proceedings of an Advanced Course,* Bad Honnef, September 1986, LNCS **255** (1987), pp. 325–392.

[31] Winskel, G., *An introduction to event structures*, in: J.W. de Bakker, W.P. d. Roever and G. Rozenberg, editors, *REX School and Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency,* Noordwijkerhout, The Netherlands, May/June 1988, LNCS **354** (1989), pp. 364–397.

[32] Winskel, G. and M. Nielsen, *Models for concurrency*, in: *Handbook of Logic in Computer Science*, Oxford University Press, 1995 pp. 1–148.

Bibliography on Higher Dimensional Automata (besides [24] and [9])

[33] Badouel, E. *Splitting of actions, higher-dimensional automata, and net synthesis*, Rapport de recherche RR-3013, INRIA, France (1996).
Available at http://www.inria.fr/rrrt/rr-3013.html.

[34] Buckland, R., *Choice as a first class citizen*, in: M. Orgun and E. Ashcroft, editors, Proceedings *Intensional Programming I* (1996), pp. 249–259.

[35] Buckland, R., M. Johnson and D. Verity, *On the specification of higher dimensional automata*, Electronic Notes in Theoretical Computer Science **68(1)** (2002), pp. 1–11.

[36] Cattani, G.L. and V. Sassone, *Higher dimensional transition systems*, in: Proceedings LICS '96, Eleventh Annual IEEE Symposium on *Logic in Computer Science*, New Brunswick, USA (1996), pp. 55–62.
Available at ftp://ftp.cl.cam.ac.uk/users/glc25/hdts.dvi.gz.

[37] Cridlig, R., *Semantic analysis of shared-memory concurrent languages using abstract model-checking*, in: Proceedings PEPM 1995, ACM SIGPLAN Symposium on *Partial Evaluation and Semantics-Based Program Manipulation*, La Jolla, USA, June 1995 (1995), pp. 214–225.
Available at http://portal.acm.org/citation.cfm?doid=215465.215577.

[38] Cridlig, R., *Implementing a static analyzer of concurrent programs: Problems and perspectives*, in: M. Dam, editor, Selected Papers of *Analysis and Verification of Multiple-Agent Languages*, 5th LOMAPS Workshop, Stockholm, Sweden, June 1996, LNCS **1192** (1997), pp. 244–259.

[39] Cridlig, R. and E. Goubault, *Semantics and analysis of Linda-based languages*, in: Proceedings WSA '93, 3rd International Workshop on *Static Analysis*, Padova, Italy, September 1993, LNCS **724** (1993), pp. 72–86.
Available at http://www.di.ens.fr/~goubault/papers/Linda.ps.gz.

[40] Fajstrup, L., *Loops, ditopology and deadlocks*, Mathematical Structures in Computer Science **10(4)** (2000), pp. 459–480.

[41] Fajstrup, L., E. Goubault and M. Raussen, *Detecting deadlocks in concurrent systems*, in: D. Sangiorgi and R. de Simone, editors, Proceedings CONCUR '98, 9th International Conference on *Concurrency Theory*, Nice, France, September 1998, LNCS **1466** (1998), pp. 332–347.
Available at http://www.di.ens.fr/~goubault/papers/concur98.ps.gz.

[42] Gaucher, P., *From concurrency to algebraic topology*, Electronic Notes in Theoretical Computer Science **39(2)** (2000).
Available at http://www.pps.jussieu.fr/~gaucher/expose.ps.gz.

[43] Gaucher, P., *Homotopy invariants of higher dimensional categories and concurrency in computer science*, Mathematical Structures in Computer Science **10(4)** (2000), pp. 481–524.
Available at http://www.pps.jussieu.fr/~gaucher/homotopie_cat.ps.gz.

[44] Gaucher, P., *Combinatorics of branchings in higher dimensional automata*, Theory and Applications of Categories **8(12)** (2001), pp. 324–376.
Available at http://www.pps.jussieu.fr/~gaucher/coin.ps.gz.

[45] Gaucher, P., *About the globular homology of higher dimensional automata*, Cahiers de Topologie et Géométrie Différentielle Catégoriques **XLIII(2)** (2002), pp. 107–156. At http://www.pps.jussieu.fr/~gaucher/sglob.ps.gz.

[46] Gaucher, P., *Investigating the algebraic structure of dihomotopy types*, Electronic Notes in Theoretical Computer Science **52(2)** (2002).
Available at http://www.pps.jussieu.fr/~gaucher/dihomotopy.ps.gz.

[47] Gaucher, P., *The branching nerve of HDA and the Kan condition*, Theory and Applications of Categories **11(3)** (2003), pp. 75–106.
Available at http://www.pps.jussieu.fr/~gaucher/fibrantcoin.ps.gz.

[48] Gaucher, P., *A model category for the homotopy theory of concurrency*, Homology, Homotopy and Applications **5(1)** (2003), pp. 549–599.
Available at http://www.pps.jussieu.fr/~gaucher/modelflow.ps.gz.

[49] Gaucher, P., *Comparing globular complex and flow*, New-York Journal of Mathematics **11** (2005), pp. 97–150.
Available at http://www.pps.jussieu.fr/~gaucher/glcompflow.ps.gz.

[50] Gaucher, P., *Homological properties of non-deterministic branchings and mergings in higher dimensional automata*, Homology, Homotopy and Applications **7(1)** (2005), pp. 51–76.
Available at http://www.pps.jussieu.fr/~gaucher/exbranching.ps.gz.

[51] Gaucher, P., *Flow does not model flows up to weak dihomotopy*, to appear in Applied Categorical Structures (2005).
Available at http://www.pps.jussieu.fr/~gaucher/nonexistence.ps.gz.

[52] Gaucher, P. and E. Goubault, *Topological deformation of higher dimensional automata*, Homology, Homotopy and Applications **5(2)** (2003), pp. 39–82.
Available at http://www.pps.jussieu.fr/~gaucher/diCW.ps.gz.

[53] Glabbeek, R.J. van, *The individual and collective token interpretations of Petri nets*, in: M. Abadi and L. de Alfaro, editors, Proceedings *CONCUR 2005*, 16^{th} International Conference on *Concurrency Theory*, San Francisco, USA, August 2005, LNCS **3653** (2005), pp. 323–337, available at http://boole.stanford.edu/pub/individual.ps.gz.

[54] Goubault, E., *Domains of higher-dimensional automata*, in: E. Best, editor, Proceedings CONCUR '93, 4th International Conference on *Concurrency Theory*, Hildesheim, Germany, August 1993, LNCS **715** (1993), pp. 293–307.
Available at http://www.di.ens.fr/~goubault/papers/Domains.ps.gz.

[55] Goubault, E., *The geometry of concurrency*, Ph.D. thesis, École Normale Supérieure (1995), http://www.di.ens.fr/~goubault/papers/these.ps.gz.

[56] Goubault, E., *Schedulers as abstract interpretations of higher-dimensional automata*, in: Proceedings PEPM 1995, ACM SIGPLAN Symposium on *Partial Evaluation and Semantics-Based Program Manipulation*, La Jolla, USA, June 1995 (1995), pp. 134–145.
Available at http://www.di.ens.fr/~goubault/papers/schedule2.ps.gz.

[57] Goubault, E., *Durations for truly-concurrent actions*, in: H. R. Nielson, editor, Proceedings *Programming Languages and Systems* - ESOP'96, 6th European Symposium on *Programming*, Linköping, Sweden, April 1996, LNCS **1058** (1996), pp. 173–187. Available as *Transitions take time* at
http://www.di.ens.fr/~goubault/papers/Timed_HDA.ps.gz.

[58] Goubault, E., *A semantic view on distributed computability and complexity*, in: Proceedings 3rd *Theory and Formal Methods* Workshop (1996).
Available at http://www.di.ens.fr/~goubault/papers/Imperial.ps.gz.

[59] Goubault, E., *Geometry and concurrency: a user's guide*, Mathematical Structures in Computer Science **10(4)** (2000), pp. 411–425.
Available at http://www.di.ens.fr/~goubault/papers/introMSCS2.ps.gz
and as *A historical note on "Geometry and Concurrency"* at
http://www.di.ens.fr/~goubault/index1.html.

[60] Goubault, E., *Labelled cubical sets and asynchronous transition systems: an adjunction* (2002),
Available at http://www.di.ens.fr/~goubault/papers/cmcim02.ps.gz.

[61] Goubault, E., *Some geometric perspectives in concurrency theory*, Homology, Homotopy and Applications **5(2)** (2003), pp. 95–136.
Available at http://www.di.ens.fr/~goubault/papers/stanford.ps.gz.

[62] Goubault, E. and T.P. Jensen, *Homology of higher dimensional automata*, in: R. Cleaveland, editor, Proceedings CONCUR '92, Third International Conference on *Concurrency Theory*, Stony Brook, NY, USA, August 1992, LNCS **630** (1992), pp. 254–268.
Available at http://www.di.ens.fr/~goubault/papers/Homology.ps.gz.

[63] Goubault, E. and M. Raussen, *Dihomotopy as a tool in state space analysis*, in: Proceedings LATIN '02, 5th Latin American Symposium on *Theoretical Informatics*, Cancun, Mexico, 2002, pp. 16–37.
Available at http://www.di.ens.fr/~goubault/papers/final.ps.gz.

[64] Gunawardena, J., *Homotopy and concurrency*, Bulletin of the EATCS **54** (1994), pp. 184–193, also in: G. Paun, G. Rozenberg and A. Salomaa, editors, *Current trends in Theoretical Computer Science: Entering the 21st Century*, World Scientific, 2001.
Available at http://www.jeremy-gunawardena.com/papers/hac.pdf.

[65] Lanzmann, E., *Automates d'ordre supérieur*, Master's thesis, Université d'Orsay (1993).

[66] Pratt, V.R., *Higher dimensional automata revisited*, Mathematical Structures in Computer Science **10(4)** (2000), pp. 525–548.
Available at http://boole.stanford.edu/pub/hda.ps.gz.

[67] Pratt, V.R., *Transition and cancellation in concurrency and branching time*, Mathematical Structures in Computer Science **13(4)** (2003), pp. 485–529.
Available at http://boole.stanford.edu/pub/seqconc.ps.gz.

[68] Raussen., M., *On the classification of dipaths in geometric models for concurrency*, Mathematical Structures in Computer Science **10(4)** (2000), pp. 427–457.

[69] Sokolowski, S., *A case for po-manifolds: in chase after a good topological model for concurrency*, Electronic Notes in Theoretical Computer Science **81** (2003), at ftp://ftp.ipipan.gda.pl/stefan/reports/73-pomanif.ps.gz.

[70] Takayama, Y., *Parallelization of concurrent processes in higher dimensional automata*, in: Proceedings RIMS Workshop on *Term Rewriting Systems and its Applications*, RIMS Kyoto University, 1995.

[71] Takayama, Y., *Extraction of concurrent processes from higher dimensional automata*, in: H. Kirchner, editor, Proceedings CAAP '96, 21st International Colloquium on *Trees in Algebra and Programming*, Linköping, Sweden, April 1996, LNCS **1059** (1996), pp. 72–86.

[72] Takayama, Y., *Towards cycle filling as parallelization*, in: Proceedings 4th International RIMS Workshop on *Concurrency Theory and Applications*, RIMS Kyoto University, 1996.