Formal Relationships Between Geometrical and Classical Models for Concurrency

Eric Goubault, Samuel Mimram

Formal Relationships Between Geometrical and Classical Models for Concurrency

Éric Goubault and Samuel Mimram*

June 11, 2012

Abstract

A wide variety of models for concurrent programs has been proposed during the past decades, each one focusing on various aspects of computations: trace equivalence, causality between events, conflicts and schedules due to resource accesses, etc. More recently, models with a geometrical flavor have been introduced, based on the notion of cubical set. These models are very rich and expressive since they can represent commutation between any number of events, thus generalizing the principle of *true concurrency*. While they are emerging as a central tool in concurrency, which is very promising because they make possible the use of techniques from algebraic topology in order to study concurrent computations, they have not yet been precisely related to the previous models, and the purpose of this paper is to fill this gap. In particular, we describe an adjunction between Petri nets and cubical sets which extends the previously known adjunction between Petri nets and asynchronous transition systems by Nielsen and Winskel.

A great variety of models for concurrency was introduced in the last decades: transition systems (with independence), asynchronous automata, event structures, Petri nets, etc. Each of these models focuses on modeling a particular aspect of computations, and even though their nature are very different, they are tightly related to each other as witnessed in [43]. More recently, models inspired by ideas coming from geometry, such as *cubical sets* (also sometimes called *higher dimensional automata* or HDA [30, 18]) or local po-spaces [12], have emerged as central tools to study concurrency: thanks to their nice algebraic structure, they allow one to carry on abstractly many computations, and they are very expressive because of their ability to represent commutations between multiple events. However, since their introduction, they have not been systematically and formally linked with the other models, such as transition systems, even though cubical sets contain a notion of generalized transition in their very definition.

*CEA, LIST, Point Courrier 94, 91191 Gif-sur-Yvette, France. This work has been supported by the PANDA ("Parallel and Distributed Analysis", ANR-09-BLAN-0169) French ANR project.

From a scientific point of view, the mere observation that these models are different is not satisfactory and their links with other models have to be investigated in depth. However, it turns out that their relationship is often quite subtle: the various models are usually not isomorphic, nor even one is a retract of the other. Adjunctions between the categories of models, which generalize Galois connections to categories, are the right notion to relate and compare them. This was first studied in the context of operational models for concurrency by Winskel et al. [43] and extended to geometrical models [15], but only between fairly restricted categories. In this paper, we greatly improve previous work by extending it to the full categories of transition systems (operational model of "interleaving" concurrency) and of transition systems with independence (operational model of "true" concurrency). Another approach to compare these models, based on history-preserving bisimulations, is developed in [40]. The main motivation underlying this work is that, by relating these models, we can compare the semantics of concurrent languages given in different formalisms. This also allows for reusing specific methods for statically analyzing concurrent programs in one model (such as deadlock detection algorithms for cubical sets [11], invariant generation on Petri nets [32], state-space reduction techniques such as sleep sets and persistent sets in Mazurkiewicz traces [14], or stubborn sets in Petri nets [37]) in the other.

This paper constitutes a major step towards formally relating geometric models with other models for concurrency. The links might appear as intuitive, but the formal step we are making underlines subtle differences between the models: there are many variants of the models, all of which can be embedded in the model of HDA, which allows us to precisely characterize the outcomes of choosing one of the other variant of the models. We have done our best to express in categorical terms how to construct one variant from the other. In particular, most models admits the following variations:

- events can be labeled or not,
- morphisms can be strict or partial,
- the multiplicity of an event can be taken in account or not,
- in the case where the events are labeled, morphisms between labels can be strict or not.

It turns out from this study that *strongly labeled* HDA seem to be the right notion of HDA, at least for comparing with most other common models of concurrency. This also unravels interesting phenomena (besides being necessary for being able to relate semantics given in different styles) such as the fact that persistent set types of methods for tackling the state-space explosion problem can be seen as searching for retracts of the state space, in the algebraic topological sense. We end this article by making some hypotheses on further relationships, with event structures and Petri nets in particular.

Related work. In this paper, we extend Winskel's results [43], which include adjunctions between transition systems, event structures, trace languages, asynchronous transition systems and Petri nets which are still an active research area [36]. A first step towards comparing higher-dimensional automata (a form of geometric semantics we are considering here), Petri nets, and event structures is reported in [39]. Also, an investigation of the comparison between cubical sets (another form of geometric semantics) and transition systems, as well as transition systems with independence was started in [16], but never formally published.

We describe right adjoint functors from the categories of transition systems, asynchronous transition systems, Petri nets and prime event structures of [43], to HDA. By general theorems, these functors transport limits onto limits, hence preserve classical parallel semantics based on pullbacks, by synchronized products [1], as the ones in transition systems or the ones of [43]. Cubical sets (or more generally HDA) that we take as the primary model for geometric semantics here, have appeared in numerous previous works, in algebraic topology in particular [34, 4]. A monoidal presentation can also be found in [20]. The basics of "directed algebraic topology" that is at the basis of the mathematics involved in the geometric semantics we use here can be found in [19].

Contents of the paper. We begin by recalling the geometric model provided by cubical sets in Section 1 and some well-known models for concurrent computations (transition systems, asynchronous automata, event structures and Petri nets) in Section 2. We then relate them by defining adjunctions in Section 3. HDA naturally "contain" transition systems (resp. asynchronous transition systems), which just encode the non-deterministic (resp. and pairwise independence) information. Event structures are also shown to be more abstract than HDA: they impose binary conflict relations and conjunctive dependencies (an event cannot depend on a disjunction of two events), and they do not distinguish different occurrences of the same event. Petri nets have a built-in notion of degree of parallelism, as is the case of HDA (given by cell dimension) but impose specific constraints on dynamics. We finally conclude on future works in Section 4.

1 Geometric models for concurrency

Precubical sets can be thought as some sort of generalized transition systems with higher-dimensional transitions. Similarly to transition systems there is a corresponding notion with "idle transitions", called *cubical sets*. These classical objects in combinatorial algebraic topology (see for instance [34]) have been used as an alternative *truly concurrent* model for concurrency, in particular since the seminal papers [30] and [38]. More recently, they have been used in [11] and [12] for deriving new and interesting deadlock detection algorithms. More algorithms have been designed since then, see for instance [31] and [9]. In the following, we will be mostly using *symmetric precubical sets*. However, we have done our best to introduce here the notion gradually, and recall some variants as well as important properties.

1.1 Cubical sets

A cubical set consists of a family $(C(n))_{n\in\mathbb{N}}$ of sets, the elements of $C(n)$ being called *n-cells*, together with for every pairs of integers n and i, such that $0 \leqslant i \leqslant n$, maps

$$\partial_i^-, \partial_i^+ : C(n+1) \to C(n) \qquad \text{and} \qquad \iota_i : C(n) \to C(n+1)$$

respectively called *source*, *target* and *degeneracy maps*, satisfying

$$\partial_j^\beta \partial_i^\alpha = \partial_i^\alpha \partial_{j-1}^\beta \qquad\qquad \iota_i \iota_j = \iota_{j-1} \iota_i \tag{1}$$

with $i < j$ and $\alpha, \beta \in \{-,+\}$ and, for every $\alpha \in \{-,+\}$,

$$\partial_j^\alpha \iota_i = \begin{cases} \iota_i \partial_{j-1}^\alpha & \text{if } i < j \\ \mathrm{id} & \text{if } i = j \\ \iota_{i-1} \partial_j^\alpha & \text{if } i > j \end{cases} \tag{2}$$

A morphism $\kappa : C \to C'$ between two cubical sets C and C' consists of a family $(\kappa_n : C(n) \to C'(n))_{n\in\mathbb{N}}$ of functions which is natural: for every index i and $\alpha \in \{-,+\}$,

$$\kappa_n \circ \partial_i^\alpha = \partial_i^\alpha \circ \kappa_{n+1} \qquad \text{and} \qquad \kappa_{n+1} \circ \iota_i = \iota_i \circ \kappa_n$$

and we write **CSet** for the category thus defined. The *0-source* (resp. *0-target*) of an n-cell $x \in C(n)$ is the 0-cell $\partial_0^- \ldots \partial_0^-(x)$ (resp. $\partial_0^+ \ldots \partial_0^+(x)$).

More conceptually, a *cubical set* C is a presheaf on the cubical category $\square$, that is a functor $C : \square^{\mathrm{op}} \to \mathbf{Set}$, and a morphism of cubical sets is a natural transformation between the corresponding functors. Here, the *cubical category* $\square$ is defined as the free category on the graph whose objects are natural integers $n \in \mathbb{N}$ and containing, for every integers i and n such that $0 \leqslant i \leqslant n$ and every $\alpha \in \{-,+\}$, arrows

$$\varepsilon_{i,n}^\alpha : n \to n+1 \qquad \text{and} \qquad \eta_{i,n} : n+1 \to n \tag{3}$$

quotiented by the relations expressing axioms dual to those given for cubical sets (1) and (2) – so that for every index n, the function $C(\varepsilon_{i,n}^\alpha)$ corresponds to ∂_i^α and $C(\eta_{i,n})$ corresponds to ι_i:

$$\varepsilon_{i,n+1}^\beta \varepsilon_{j,n}^\alpha = \varepsilon_{j-1,n}^\alpha \varepsilon_{i,n+1}^\beta \qquad\qquad \eta_{j,n} \eta_{i,n+1} = \eta_{i,n} \eta_{j-1,n+1} \tag{4}$$

with $i < j$ and $\alpha, \beta \in \{-,+\}$, and for every $\alpha \in \{-,+\}$,

$$\eta_{i,n} \varepsilon_{j,n}^\alpha \quad = \quad \begin{cases} \varepsilon_{j-1,n-1}^\alpha \eta_{i,n-1} & \text{if } i < j \\ \mathrm{id} & \text{if } i = j \\ \varepsilon_{j,n-1}^\alpha \eta_{i-1,n-1} & \text{if } i > j. \end{cases}$$

The *precubical category* $\boxdot$ is defined similarly with only the $\varepsilon_{i,n}^{\alpha}$ as generators and the first equations of (4) as axioms, and a *precubical set* is a presheaf on the precubical category: a precubical set consists of a family $(C(n))_{n\in\mathbb{N}}$ of sets together with a family of maps $\partial_i^-, \partial_i^+ : C(n{+}1) \to C(n)$ satisfying the equations on the left of (1). We write **PCSet** for the corresponding category.

Given an integer n, we write $\square_n$ for the full subcategory of $\square$ whose objects are the integers $k \leqslant n$. An *n-dimensional cubical set* is a presheaf on $\square_n$ and we write $\mathbf{CSet}_n$ for the category of n-dimensional cubical sets. The inclusion functor $\square_n \to \square$ induces by precomposition a functor $U_n : \mathbf{CSet} \to \mathbf{CSet}_n$ called the *n-truncation functor* (see Section 1.7).

Example 1. *The geometric intuition underlying cubical sets is the following one. An n-cell x of a cubical set should be seen as an n-dimensional cube, the $(n-1)$-dimensional cubes $\partial_i^-(x)$ and $\partial_i^+(x)$ being respectively the source and target in dimension i of x, and the degeneracy maps ι_i allowing us to see an n-dimensional cube as an $(n+1)$-dimensional one, degenerated in dimension i. So for example, a "cylinder" can be described as a precubical set C with*

$$C(0) = \{x, y\} \qquad C(1) = \{f, g, h\} \qquad C(2) = \{\alpha\} \qquad C(n) = \emptyset \quad \text{for } n > 2$$

with the following sources and targets, given by $\partial_0^-(f) = \partial_0^+(f) = \partial_0^-(h) = x$, $\partial_0^-(g) = \partial_0^+(g) = \partial_0^+(h) = y$, $\partial_0^-(\alpha) = \partial_0^+(\alpha) = h$, $\partial_1^-(\alpha) = f$ and $\partial_1^+(\alpha) = g$. This cylinder can be pictured graphically as

$$\begin{tikzcd}
& y \arrow[loop right, "g"] \\
x \arrow[ur, "h"] \arrow[loop right, "f"] \arrow[ur, phantom, "\alpha"', shift right=3]
\end{tikzcd}
\qquad \text{or} \qquad
\begin{tikzcd}
x \arrow[r, "h"] \arrow[d, "f"'] \arrow[dr, phantom, "\alpha"] & y \arrow[d, "g"] \\
x \arrow[r, "h"'] & y
\end{tikzcd}
\qquad \textit{(in an unfold representation)}$$

From a concurrency point of view, a 1-cell corresponds to the occurrence of an event (an action) and an n-cell corresponds to a commutation or an independence between the 1-cells occurring in its faces. The cubical set above representing the cylinder thus corresponds intuitively to a program constituted of two processes in parallel: a (while) loop (the actions f and g) and a single instruction (h). See also Example 9.

In previous example, the two transitions f and g are instances of a same event because they are parallel faces of the square α. This suggests that the notion of event should be reconstructed in a precubical set as an equivalence class of transitions as follows. Suppose given a precubical set C. We define a relation $\approx$ as the smallest equivalence relation on 1-cells of C, such that for every $f, g \in C(1)$, $f \approx g$ when there exists $y \in C(2)$ such that $f = \partial_i^-(y)$ and $g = \partial_i^+(y)$, for $i = 0$ or $i = 1$. An *event* is the equivalence class of a 1-cell under the relation $\approx$. Given a morphism $\kappa : C \to D$ between precubical sets, two 1-cells of D in a same event are sent to two 1-cells of D in a same event; any such morphism thus induces a function $\kappa_1/\!\approx$ from the events of C to the events of D.

1.2 A monoidal definition of the cubical category

A shorter description of the cubical category can be given if we take its monoidal structure in account: the cubical category is the free monoidal category (that is, a category equipped with a coherent tensor product and unit [24]) containing a co-cubical object [20]. This will help in defining very concisely the adjunctions we have in mind in Section 3.

Definition 2. *A* cubical object $(C, \varepsilon^-, \varepsilon^+, \eta)$ *in a monoidal category* $(\mathcal{C}, \otimes, I)$ *consists of an object* C *together with three morphisms*

$$\eta : I \to C \qquad \varepsilon^- : C \to I \qquad \varepsilon^+ : C \to I$$

such that

$$\varepsilon^- \circ \eta \quad = \quad \mathrm{id}_I \quad = \quad \varepsilon^+ \circ \eta$$

A morphism f *between two cubical objects* $(C_1, \varepsilon_1^-, \varepsilon_1^+, \eta_1)$ *and* $(C_2, \varepsilon_2^-, \varepsilon_2^+, \eta_2)$ *is a morphism* $f : C_1 \to C_2$ *such that*

$$f \circ \eta_1 = \eta_2 \qquad \varepsilon_2^- \circ f = \varepsilon_1^- \qquad \varepsilon_2^+ \circ f = \varepsilon_1^+$$

Dually, a co-cubical object $(C, \varepsilon^-, \varepsilon^+, \eta)$ *in* $\mathcal{C}$ *is a cubical object in* $\mathcal{C}^{\mathrm{op}}$*.*

In the cubical category $\square$, $(1, \varepsilon^-, \varepsilon^+, \eta)$ is a co-cubical object. The fact that $\square$ is the free monoidal category containing a co-cubical object means that all the arrows of $\square$ can be recovered from those by tensoring with identities

$$\varepsilon^\alpha_{i,n} = \mathrm{id}_i \otimes \varepsilon^\alpha \otimes \mathrm{id}_{n-i} \qquad \text{and} \qquad \eta_{i,n} = \mathrm{id}_i \otimes \eta \otimes \mathrm{id}_{n-i}$$

and that the axioms satisfied by the morphisms – the axioms dual of (1) and (2) – are precisely those imposed by the axioms of monoidal categories and those of co-cubical objects. This can be equivalently reformulated as follows:

Proposition 3. *Given a monoidal category* $\mathcal{C}$*, the category of monoidal functors* $\square \to \mathcal{C}$ *and monoidal natural transformations is equivalent to the category of co-cubical objects in* $\mathcal{C}$*.*

In other words, given a monoidal category $\mathcal{C}$, a cubical object in $\mathcal{C}$ is "the same" as a monoidal functor $\square^{\mathrm{op}} \to \mathcal{C}$. This definition of cubical sets has been known for quite some time, but no concrete application of it has been done up to now. Interestingly, we show here that it can be used to concisely define some cubical sets (see in particular Section 1.5). It is also sometimes useful to define morphisms; for instance, given integers n and i such that $0 \leqslant i \leqslant n$, and $\alpha \in \{-, +\}$, we write $\partial^\alpha_{-i} : C(n+1) \to C(1)$ for the morphism $\partial^\alpha_{-i} = C((\varepsilon^\alpha)^{\otimes i} \otimes \mathrm{id}_1 \otimes (\varepsilon^\alpha)^{\otimes(n-i)})$ where $(\varepsilon^\alpha)^{\otimes i}$ denotes the tensor product of i copies of ε^α.

Similarly, monoidal functors $\boxdot \to \mathcal{C}$ correspond to co-precubical objects in $\mathcal{C}$, where a *precubical object* $(C, \varepsilon^-, \varepsilon^+)$ is an object C of $\mathcal{C}$ together with two arrows $\varepsilon^-, \varepsilon^+ : C \to I$ (and no axiom to be satisfied), and a *co-precubical object* is defined dually.

1.3 From precubical sets to cubical sets

In this section, we formalize the intuition that morphisms between precubical sets are to morphisms between cubical sets what partial functions are to total functions. Recall that a *pointed set* (A, a) consists of a set together with a distinguished element $a \in A$, and a morphism $f : (A, a) \to (B, b)$ between two pointed sets consists of a function $f : A \to B$ such that $f(a) = b$. If we write $\mathbf{Set}^*$ for the category of pointed sets, there is a forgetful functor $U : \mathbf{Set}^* \to \mathbf{Set}$ which to every pointed set (A, a) associates the underlying set A. This functor admits a left adjoint $F : \mathbf{Set} \to \mathbf{Set}^*$ which to every set A associates the free pointed set it generates, that is the pointed set $(A \uplus \{*\}, *)$ where $\uplus$ denotes the disjoint union (we often use the notation $*$ for the newly added element). We write $? = G \circ F$ for the monad on $\mathbf{Set}$ induced by this adjunction. It is well-known [24] that

Proposition 4. *The category of sets and partial functions is isomorphic to the Kleisli category* $\mathbf{Set}_?$ *associated to the monad* $?$ *on* $\mathbf{Set}$*. Moreover, this Kleisli category is equivalent to the category* $\mathbf{Set}^*$*.*

Proof. A partial function $f : A \to B$ induces a morphism $g : A \to B$ in $\mathbf{Set}_?$ (i.e. a morphism $g : A \to ?\,B$ in $\mathbf{Set}$) defined on every $x \in A$ by $g(x) = f(x)$ if $f(x)$ is defined and $g(x) = *$ otherwise, where $?\,B = B \uplus \{*\}$. Conversely, any morphism $g : A \to B$ in $\mathbf{Set}_?$ (i.e. morphism $g : A \to ?\,B$ in $\mathbf{Set}$, with $?\,B = B \uplus \{*\}$) induces a partial function $f : A \to B$ defined on every $x \in A$ such that $g(x) \neq *$ by $f(x) = g(x)$. These two operations can easily be shown to be inverse of each other, thus exhibiting an isomorphism between the category of sets and partial functions and the Kleisli category $\mathbf{Set}_?$.

By general properties of monads (see [24], exercises p. 144), the category $\mathbf{Set}_?$ is equivalent to the full subcategory of $\mathbf{Set}^*$ whose objects are of the form FA for some set $A \in \mathbf{Set}$. Moreover, every object (A, a) of $\mathbf{Set}^*$ is isomorphic to the pointed set $F(A \setminus \{a\})$. The categories $\mathbf{Set}_?$ and $\mathbf{Set}^*$ are thus equivalent. $\square$

The proposition above formalizes the fact that a partial function $f : A \to B$ can be seen as a total function $f : A \to B \uplus \{*\}$ where f is "undefined" on an element $a \in A$ whenever $f(a) = *$. The second part of the proposition states that this partial function can also be seen as a pointed function $f : (A \uplus \{*\}, *) \to (B \uplus \{*\}, *)$.

The situation between precubical sets and cubical sets is very similar. There is an obvious inclusion functor $\boxdot \to \square$, which by precomposition, induces a forgetful functor $U : \mathbf{CSet} \to \mathbf{PCSet}$ on the corresponding presheaf categories. By general theorems (see Section 1.7), this functor admits a left adjoint $F : \mathbf{PCSet} \to \mathbf{CSet}$. As previously, we write $? = G \circ F$ for the induced monad on $\mathbf{PCSet}$ and $\mathbf{PCSet}_?$ for the Kleisli category associated to the monad. The morphisms in $\mathbf{PCSet}_?$ should be thought as "partial morphisms of precubical sets". And actually, this category can be shown to be isomorphic to a category whose objects are precubical sets and morphisms $\kappa : C \to D$ are families $(k_n : C(n) \to D(n))_{n \in \mathbb{N}}$ of *partial* functions satisfying suitable properties, which we do not need to detail here.

One of the main interests of expressing the "partial" variants of models as Kleisli constructions is that this enables us to easily lift the adjunctions between models into adjunctions between their partial variants. Namely,

Proposition 5. *Suppose that $\mathcal{C}$ and $\mathcal{D}$ are categories and with S and T monads on respectively $\mathcal{C}$ and $\mathcal{D}$. Suppose moreover that $U : \mathcal{D} \to \mathcal{C}$ is a functor such that*

$$U \circ T = S \circ U$$

and F sends the unit and the multiplication of T to the unit and the multiplication of U. Then U has a left adjoint if and only if the functor $U \circ I_D : \mathcal{D}_T \to \mathcal{C}_S$ has a left adjoint, where $I_D : \mathcal{D}_T \to \mathcal{D}$ is the canonical comparison functor between the Kleisli category $\mathcal{D}_T$ associated to T and $\mathcal{D}$.

This property, which is proved in a more general version in [27], thus enables us to *lift* an adjunction between the categories $\mathcal{C}$ and $\mathcal{D}$ into an adjunction between the corresponding Kleisli categories $\mathcal{C}_S$ and $\mathcal{D}_T$. In the following, it will be particularly useful to lift adjunction between models into adjunctions between corresponding models with partial morphisms.

1.4 Symmetric cubical sets

One sometimes needs more structure on cubical sets in order to formally express the fact that the cells of dimension $n \geqslant 2$ in cubical sets arising as models for concurrent processes are essentially not directed. This can be formalized by adjoining a notion of symmetry in cubical sets. The idea here is that given a 2-cell z in a cubical set as shown on the left of

$$\begin{tikzcd}
& x_3 & \\
x_1 \arrow[ur, "y_3"] & z & x_2 \arrow[ul, "y_4"'] \\
& x_0 \arrow[ul, "y_1"] \arrow[ur, "y_2"'] &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
& x_3 & \\
x_2 \arrow[ur, "y_4"] & z' & x_1 \arrow[ul, "y_3"'] \\
& x_0 \arrow[ul, "y_2"] \arrow[ur, "y_1"'] &
\end{tikzcd}$$

there should also be a "mirror" cell z' as shown on the right, expressing the fact that z is not really directed from $y_1 y_3$ to $y_2 y_4$. The symmetry of the cubical category will associate to each two cell a "mirror" 2-cell in this way (it actually also generalizes this principle to higher dimensions). The need for symmetry is also explained in the case of labeled cubical sets in Example 9.

The *symmetric cubical category* $\square_S$ is the free symmetric monoidal category containing a co-cubical object. The presheaves on this category are called *symmetric cubical sets* and they form a category **SCSet**. The category $\square_S$ can also be described as the free monoidal category containing a symmetric co-cubical object $(C, \varepsilon^-, \varepsilon^+, \eta, \gamma)$, which is a co-cubical object $(C, \varepsilon^-, \varepsilon^+, \eta)$ together with a morphism $\gamma : C \otimes C \to C \otimes C$ satisfying usual axioms for symmetry

$$(\gamma \otimes C) \circ (C \otimes \gamma) \circ (\gamma \otimes C) = (C \otimes \gamma) \circ (\gamma \otimes C) \circ (C \otimes \gamma) \qquad\qquad \gamma \circ \gamma = \gamma \quad (5)$$

and

$$\begin{array}{lll} \gamma\circ(\varepsilon^-\otimes C)=C\otimes\varepsilon^- & \gamma\circ(\varepsilon^+\otimes C)=C\otimes\varepsilon^+ & (\eta\otimes C)\circ\gamma=C\otimes\eta \\ \gamma\circ(C\otimes\varepsilon^-)=\varepsilon^-\otimes C & \gamma\circ(C\otimes\varepsilon^+)=\varepsilon^-\otimes C & (C\otimes\eta)\circ\gamma=\eta\otimes C \end{array}$$

see [20] for the details. Alternatively, the notion of symmetric cubical set can be equivalently reformulated as a cubical set C together with, for every integer n, an action of the symmetric group Σ_n on $C(n)$ – the action of the transposition being given by $C(\gamma) : C(2) \to C(2)$ – which satisfies the following coherence axioms: for every integers n and i such that $0 \leqslant i \leqslant n$ and every $\alpha \in \{-,+\}$,

- for every $(n+1)$-cell x and permutation $\sigma \in \Sigma_{n+1}$, $\partial^\alpha_i(\sigma x) = \partial^\alpha_{\sigma(i)}(x)$
- for every n-cell x and permutation $\sigma \in \Sigma_n$, $\iota_i(\sigma x) = \iota_{\sigma(i)}(x)$

Namely, any symmetry $\sigma : n \to n$ (i.e. a bijection on a set with n elements) can be decomposed as a product of transpositions and can therefore be seen as a morphism in $\square_S$ by sending the transposition $\sigma_i : n \to n$, which exchanges the i-th and $(i+1)$-th element, to the morphism $i \otimes \gamma \otimes (n-i-2)$. The axioms (5) imposed on γ, as well as the axioms of monoidal categories, ensure that this operation is well defined. In the following, we will thus sometimes implicitly consider a bijection as a morphism in the category $\square_S$. Given a symmetric monoidal category $\mathcal{C}$ (such as **Set** with cartesian product), any cubical object of the underlying monoidal category of $\mathcal{C}$ can be canonically equipped with a structure of symmetric cubical set, the morphism γ being given by the symmetry of the category.

Given an integer n, we write $(\square_S)_n$ for the full subcategory of $\square_S$ whose objects are integers $k \leqslant n$ and $\mathbf{SCSet}_n$ for the category of presheaves on $(\square_S)_n$, whose objects are called *n-dimensional symmetric cubical sets*. The *symmetric precubical category* $\boxdot_S$ is defined similarly as the free symmetric monoidal category containing a co-precubical object and we write **SPCSet** for the category of presheaves on $\boxdot_S$, whose objects are called *symmetric precubical sets*. Notice that many of the usual models for concurrency can be equipped with a similar, and often related, notion of symmetry: for instance event structures [42, 36], or Petri nets [21].

1.5 Labeled cubical sets

We have explained that the 1-cells of a cubical set can be seen as occurrences of events in the semantics of a concurrent computational process. One sometimes needs to remember to which instruction of the process it corresponds. Labeled (pre)cubical sets formally allows this. The presentation given here is adapted from [17], see also [13].

Suppose that we are given a set L of *labels*. The category $(\mathbf{Set}, \times, 1)$ has finite products and is thus monoidal with the cartesian product as tensor and the terminal set $1 = \{*\}$ as unit (for simplicity, we consider that the monoidal structure is strict). The set L can be canonically equipped with a structure of symmetric precubical object $(L, \varepsilon^-, \varepsilon^+, \gamma)$ where $\varepsilon^-, \varepsilon^+ : L \to 1$ are both the terminal arrow and $\gamma : L \times L \to L \times L$ is the canonical transposition. According to the preceding remarks, it thus induces a symmetric precubical set noted $!\,L$ and called the *labeling precubical set on* L. Moreover, if L' is another set of labels, any function $f : L \to L'$ induces a morphism between the corresponding co-precubical objects, and therefore induces a morphism $!\,f : !\,L \to !\,L'$, extending this operation into a functor. An explicit description of the precubical set $!\,L$ can be given as follows: its n-cells $l \in !\,L(n)$ are lists $l = (e_i)_{0 \leqslant i < n}$, of length n, of labels $e_i \in L$. The face maps $\partial^-_n, \partial^+_n : !\,L(n+1) \to !\,L(n)$ both send an $(n+1)$-cell $(e_i)_{0 \leqslant i < n+1}$ to the list obtained by removing the element at the k-th position and the action of a symmetry $\sigma : n \to n$ on $!\,L(n)$ sends a cell $(e_i)_{0 \leqslant i < n}$ to $(e_{\sigma(i)})_{0 \leqslant i < n}$.

It can be shown that $!\,L$ is the cofree precubical set generated by L in the following sense:

Proposition 6. *The functor* $E : \mathbf{SPCSet} \to \mathbf{Set}$*, which to every precubical set* C *associates its set* $C(1)/\approx$ *of events (see Section 1.1) and to every morphism* $\kappa : C \to D$ *associates the function* $(\kappa_1/\approx) : (C(1)/\approx) \to (D(1)/\approx)$*, admits* $!$ *as right adjoint.*

Proof. Suppose given a precubical set C and a set L. To every given function $f : (C(1)/\approx) \to L$, we associate the morphism $\psi(f) : C \to !\,L$ defined on an n-cell x as the n-cell $(f(\partial^-_{\neg 0}(x)), \ldots, f(\partial^-_{\neg(n-1)}(x)))$ of $!\,L$, where the function $\partial^\alpha_{\neg i}$ is defined in Section 1.2. Conversely, to every morphism $\kappa : C \to !\,L$ of cubical sets, we associate the function $\psi(\kappa) : (C(1)/\approx) \to L$ defined as $\kappa_1/\approx$: this is well defined since the events of $!\,L$ are (in bijection with) the elements of L. Finally, is it straightforward to check that the functions φ and ψ are natural in C and L, and inverse of each other. ☐

Remark 7. The cofree non-symmetric labeling precubical set on a set L could be defined in the same way, but a direct description is more difficult. It can for example be obtained from the symmetric labeling precubical set $!\,L$ on L by quotienting by the action of symmetries.

Recall that given categories $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{E}$ and functors $F : \mathcal{C} \to \mathcal{E}$ and $G : \mathcal{D} \to \mathcal{E}$, the *slice category* $F \downarrow G$ (sometimes also called *comma category*) is the category whose objects are triples (A, f, A') where A is an object of $\mathcal{C}$, A' is an object of $\mathcal{D}$ and $f : FA \to GA'$ is a morphism of $\mathcal{E}$, and whose morphisms $(h, h') : (A, f, A') \to (B, g, B')$ are the pairs of morphisms $h : A \to B$ of $\mathcal{C}$ and $h' : A' \to B'$ of $\mathcal{D}$ making the diagram

$$\begin{tikzcd} FA \arrow[r, "Fh"] \arrow[d, "f"'] & FB \arrow[d, "g"] \\ GA' \arrow[r, "Gh'"'] & GB' \end{tikzcd}$$

commute. By abuse of notation, we often write $\mathcal{D} \downarrow G$ for the category $\mathrm{Id}_{\mathcal{D}} \downarrow G$. A labeled variant of cubical sets is defined as follows.

Definition 8. *The category of* labeled symmetric precubical sets*, denoted by* **LSPCSet***, is the slice category* **SPCSet**$\downarrow$!.

By Proposition 6, a given labeled symmetric precubical set (C, ℓ, L) (defined by $C \in$ **SPCSet**, $L \in$ **Set** and $\ell : C \to !\,L$) can also be seen as a triple (C, ℓ, L) with the function $\ell : E(C) \to L$ associating a label to each event of C. In other words, the category **LSPCSet** is isomorphic to $E\downarrow!$, where E is the event functor introduced in Proposition 6.

Example 9. *The CCS processes $ab + ba$ and $(a|b)$ respectively induce labeled symmetric cubical sets of the form*

$$
\begin{tikzcd}
& z & \\
y_1 \arrow[ur, "b"] & & y_2 \arrow[ul, "a"'] \\
& x \arrow[ul, "a"] \arrow[ur, "b"'] &
\end{tikzcd}
\qquad \textit{and} \qquad
\begin{tikzcd}
& z & \\
y_1 \arrow[ur, "b"] \arrow[rr, phantom, "\alpha\ \beta"] & & y_2 \arrow[ul, "a"'] \\
& x \arrow[ul, "a"] \arrow[ur, "b"'] &
\end{tikzcd}
$$

with in the second case two squares α and β attached in the middle, respectively labeled by (a, b) and (b, a) (and none in the first case): the presence of a square indicates that the two actions a and b commute and more generally n-cubes indicate the commutation of n actions [15]. Notice that the symmetry intuitively enables us to say that the cell labeled by ab is “the same as” the cell labeled by ba. In a non-symmetric case, there would be only one cell and there is no canonical choice of naming for this cell (this is however sometimes overcome by supposing that letters are totally ordered, but supposing this is not very natural).

Notice that events provides a canonical labeling of symmetric precubical sets:

Proposition 10. *The forgetful functor U :* **LSPCSet** $\to$ **SPCSet** *which to every labeled symmetric precubical set associates the underlying symmetric precubical set (forgetting the labels) admits the functor E :* **SPCSet** $\to$ **LSPCSet** *as left adjoint, which to every symmetric precubical set C associates the labeled symmetric precubical set $(C, \ell, !\,L)$ where $L = C(1)/\approx$ is the set of events of C and ℓ is the morphism induced by the function $\ell : C(1) \to L$ which to every 1-cell associates its equivalence class under $\approx$.*

This means in particular that all following results about labeled symmetric precubical sets simply extend to the unlabeled case by considering precubical sets labeled by their events. We thus only handle labeled cases in the following, since unlabeled structures are a particular instance.

The category of labeled symmetric cubical sets is defined in a similar way. A given pointed set $(L, *)$ induces a symmetric cubical object $(L, \varepsilon^-, \varepsilon^+, \eta, \gamma)$ where $\varepsilon^-, \varepsilon^+ : L \to 1$ are both the terminal arrow, $\eta : 1 \to L$ associates $*$ to the unique element of 1 and $\gamma : L \times L \to L \times L$ is the canonical transposition function. As previously, this induces a symmetric cubical set, that we still write $!(L, *)$, and can be shown to be cofree in the sense that

Proposition 11. *The functor* $E : \mathbf{SCSet} \to \mathbf{Set}^*$*, which to every cubical set* C *associates the pointed set obtained from* $C(1)/\approx$ *by identifying all equivalence classes containing an element of the image of* ι_0 *to a single element* $*$ *chosen as distinguished element and to every morphism* $\kappa : C \to D$ *associates the morphism induced by* $\kappa_1 : C(1) \to D(1)$*, admits* $!$ *as right adjoint.*

Definition 12. *The category of* symmetric labeled cubical sets*, denoted by* $\mathbf{LSCSet}$*, is the slice category* $\mathbf{SCSet}\downarrow !$*, which is isomorphic to* $E\downarrow !$*.*

An explicit description of $!(L, *)$ is similar to the one of the labeling symmetric precubical set: its n-cells are lists $l = (e_i)_{0 \leqslant i < n}$, with the same face and symmetry maps as previously. The degeneracy maps $\iota_i : !(L, *)(n) \to !(L, *)(n+1)$ associate to every list l of length n the list of length $n+1$ obtained from l by inserting $*$ at the i-th position.

We have defined labellings in the most natural way. There is however a slight mismatch between labeled precubical and cubical sets: in the first case functions between labels are total whereas they are partial in the second case. This mismatch actually turns out to bring annoying details, as explained in Section 1.7 (see also [7]). The opposite choices can be made in both cases as follows. A slightly more general notion of labeled precubical set can be defined, by allowing partial functions between morphisms. If we write $U : \mathbf{Set}^* \to \mathbf{Set}$ for the canonical forgetful functor, the category of *weakly labeled symmetric precubical sets* $\mathbf{wLSPCSet}$ is defined as $\mathbf{wLSPCSet} = \mathbf{SPCSet}\downarrow !U$. Conversely, one can restrict labeled cubical sets by only allowing total functions between labels and imposing that only degenerate events are labeled by the distinguished element of the labeling pointed set thus defining a category of *totally labeled symmetric cubical sets* (we do not detail this construction here).

Finally, we introduce the notion of strongly labeled cubical set, which will turn out in Section 3 to be the "right" notion of labeled cubical set in order to relate them with most of the usual models of concurrency.

Definition 13. *A labeled cubical set* (C, ℓ) *is* strongly labeled *when there exists no pair of distinct* k*-cells, for some dimension* k*, whose sources and targets are equal, which have the same label: for every index* $k > 0$*, and every elements* $x, y \in C(k)$ *such that for every index* $0 \leqslant i < k$ $\partial_i(x) = \partial_i(y)$*, if* $\ell(x) = \ell(y)$ *then* $x = y$*.*

This condition can be seen as a labeled and higher dimensional analogue of Winskel's "no ravioli" condition for HDA [42], which imposes that two parallel 1-cells should be equal, and corresponds to being separated wrt a Grothendieck topology.

1.6 Higher dimensional automata

A *pointed cubical set* (C, i) is a cubical set together with a distinguished 0-cell $i \in C(0)$. The notion of higher dimensional automaton can be seen as a generalization of the classical notion of automaton to higher dimensional transition systems:

Definition 14. *A* higher dimensional automaton *(or* HDA*) is a pointed labeled symmetric cubical set C, the distinguished element i being called the* initial state*. A morphism of HDA is a morphism between the underlying labeled symmetric cubical sets which preserves the initial state.*

Given a category $\mathcal{C}$ of cubical sets, we often write $\mathcal{C}^*$ for the corresponding category of pointed cubical sets. We write $\mathbf{HDA} = \mathbf{LSCSet}^*$ for the category of HDA and $\mathbf{sHDA} = \mathbf{LSPCSet}^*$ for the category of *strict HDA*. We also write $\mathbf{HDA}_n = \mathbf{LSCSet}^*_n$ for the subcategories for truncated HDA.

A *path* $p : x \twoheadrightarrow x'$ in an HDA C is a finite sequence $(y_i)_{0 \leqslant i < n}$ of 1-cells of C such that $\partial_0^+(y_i) = \partial_0^-(y_{i+1})$, $\partial_0^-(y_0) = x$ and $\partial_0^+(y_n) = x'$. We write $s \cdot t$ for the concatenation of two paths s and t. A 0-cell x of an HDA is *reachable* when there exists a path $s : i \twoheadrightarrow x$, where i is the initial state of the HDA. Since higher dimensional cells express the fact that transitions are independent, two paths differing only by a reordering of independent transitions should be considered as equivalent from the concurrency point of view. This is formally expressed by the *homotopy* relation between paths [10, 40], which is defined as the smallest equivalence relation relating two paths $s \cdot m \cdot n \cdot t$ and $s \cdot p \cdot q \cdot t$ where m, n, p and q are 1-cells such that there exists a 2-cell z for which $m = \partial_0^-(z)$, $q = \partial_0^+(z)$, $p = \partial_1^-(z)$ and $n = \partial_1^+(z)$; graphically,

$$
\begin{tikzcd}
 & {} \\
 & \cdot \arrow[u, two heads, "t"] & \\
\cdot \arrow[ur, "n"] & z & \cdot \arrow[ul, "q"'] \\
 & \cdot \arrow[ul, "m"] \arrow[ur, "p"'] & \\
 & {} \arrow[u, two heads, "s"]
\end{tikzcd}
$$

In particular, in the situation above, m and q (resp. p and n) are part of the same event. Given two paths s and t, we write $s \sim t$ when they are homotopic. Two homotopic paths are necessarily parallel (they have the same source and target).

1.7 Relating variants of cubical sets.

Suppose given two categories $\mathcal{C}$ and $\mathcal{D}$ and a functor $I : \mathcal{C} \to \mathcal{D}$. Every presheaf $C : \mathcal{D}^{\mathrm{op}} \to \mathbf{Set}$ on $\mathcal{C}$ induces a presheaf $C \circ I^{\mathrm{op}} : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ by precomposition with I, and this operation extends into a functor $\hat{I} : \hat{\mathcal{D}} \to \hat{\mathcal{C}}$ from the presheafs on $\mathcal{D}$ to those on $\mathcal{C}$, defined on morphisms $\alpha : C \to D$ by $(\hat{I}(\alpha))_A = \alpha_{I(A)}$. These functors have many nice properties, some of which useful here are detailed below:

Proposition 15. *Suppose given two categories $\mathcal{C}$ and $\mathcal{D}$, where $\mathcal{C}$ is small, and a functor $I : \mathcal{C} \to \mathcal{D}$ between them.*

1. *The functor $\hat{I} : \hat{\mathcal{D}} \to \hat{\mathcal{C}}$ admits both a left and a right adjoint and we write T for the monad induced on $\hat{\mathcal{C}}$.*

2. *The Kleisli category $\mathcal{D}_T$ associated to the monad T embeds fully and faithfully into $\hat{\mathcal{D}}$.*

3. *When I is bijective on objects, the adjunction is monadic which means that the category $\hat{\mathcal{D}}$ is equivalent to the category $\mathcal{C}^T$ of algebras for the monad T on $\hat{\mathcal{C}}$.*

Proof. (i) and (ii) are standards properties [23]. In particular, the free presheaf in $\hat{\mathcal{D}}$ on a presheaf $C \in \hat{\mathcal{C}}$ can be computed as the left Kan extension of C along I (and similarly for the right adjoint).

(iii) This fact does not seem to be very well-known and can be found for example p. 105 of [3]. We have seen that the functor $\hat{I}$ admits a left adjoint. Since it is the bijective on objects it is conservative (it reflects isomorphisms): an isomorphism between presheaves is simply a natural transformation between them whose components are all invertible. Moreover, presheaf categories are cocomplete; in particular, they have all equalizers, these are computed pointwise and they are thus preserved by precomposition with U. We can conclude by using Beck's monadicity theorem [24]. $\square$

Notice that this generalizes in particular the situation described in Section 1.3.

This property is very interesting because, it means that all the forgetful functors between variants of categories of cubical sets admit both left and right adjoint:

– functors forgetting structure:

$$\begin{array}{c}\mathbf{SCSet} \to \mathbf{CSet}, \quad \mathbf{CSet} \to \mathbf{PCSet}, \quad \mathbf{PCSet} \to \mathbf{Set}, \quad \text{etc.}\\ \mathbf{SCSet}_n \to \mathbf{CSet}_n, \quad \mathbf{CSet}_n \to \mathbf{PCSet}_n, \quad \mathbf{PCSet}_n \to \mathbf{Set}, \quad \text{etc.}\end{array}$$

– truncation functors:

$$\mathbf{SCSet} \to \mathbf{SCSet}_n, \quad \mathbf{CSet} \to \mathbf{CSet}_n, \quad \text{etc.}$$

These adjoints will allow us to compute for example the free cubical set on a precubical set and so on, and will be used in the following. As an illustration, consider the functor $\mathbf{PCSet} \to \mathbf{PCSet}_n$. Given an n-dimensional precubical set C, the left adjoint sends C to the precubical set D whose k-cells are $D(k) = C(k)$ for $k \leqslant n$ and $D(k) = \emptyset$ otherwise. The action of the right adjoint is more subtle: it sends C to the precubical set obtained from C by "filling in" all the k-dimensional cubes, with $k > n$, by a k-cell.

The analogy between the adjunction between sets and pointed sets and the adjunction between precubical sets and sets, can be related with the construction of labeling cubical sets as follows.

Lemma 16. *The diagram*

$$\begin{tikzcd} \mathbf{Set} \arrow[r, bend left, "F"] \arrow[d, "I"'] \arrow[r, phantom, "\top"] & \mathbf{SPCSet} \arrow[l, bend left, "E"] \arrow[d, "J"] \\ \mathbf{Set}^* \arrow[r, bend left, "H"] \arrow[r, phantom, "\perp"] & \mathbf{SCSet} \arrow[l, bend left, "G"] \end{tikzcd}$$

commutes, in the sense that $J \circ F = H \circ I$ and $G \circ J = I \circ E$, where E (resp. H) is the functor which to every symmetric precubical (resp. cubical set) associates its set (resp. pointed set) of events described along with their right adjoints in Section 1.5, and I (resp. J) is the left adjoint to the forgetful functor $\mathbf{Set}^* \to \mathbf{Set}$ *(resp.* $\mathbf{SCSet} \to \mathbf{PCSet}$*).*

It could be hoped that previous Lemma would provide the starting point of a lifting of the adjunctions between **SPCSet** and **SCSet** to adjunctions between **LSPCSet** and **LSCSet**. However this is not the case: morphisms between labels have to be total or partial in both the categories. It is however easy to show that

Proposition 17. *The forgetful functor* $\mathbf{LSCSet} \to \mathbf{wLSPCSet}$ *admits both a left and a right adjoint (and other adjunctions mentioned above can be lifted to the labeled case in a similar way). Similar adjunctions also exist between the variants where functions between labels are total.*

The choice of partial or total functions between labels in the category of labeled symmetric (pre)cubical sets is thus difficult to handle in a modular way. The choice has to be made once for all and in the following, we deliberately do not explicit which one is made since all the constructions given here work in both cases.

2 Traditional models for concurrency

2.1 Transition systems

Transition systems are one of the oldest semantic models, both for sequential and concurrent systems, in which computations are modeled as the sequence of interactions that they can have with their environment. There is a convenient categorical treatment of this model, that we use in the sequel, taken from [43].

Definition 18. *A* transition system *is a quadruple* $(S, i, E, Tran)$ *where*

- S *is a set of* states *with* initial state i,
- E *is a set of* events,
- $Tran \subseteq S \times E \times S$ *is the* transition *relation.*

In other words, a transition system is a graph together with a distinguished vertex. Transition systems are made into a category by defining morphisms to be some kind of simulation (for then being able to discuss about properties modulo weak or strong bisimulation, see [22]). The idea is that a transition system T_1 simulates a transition system T_0 if as soon as T_0 can fire some action a in some context, T_1 can fire a as well in some related context. A morphism $f : T_0 \to T_1$ defines the way states and transitions of T_0 are related to states and transitions of T_1 making transition systems into a category **TS**.

Definition 19. *Let $T_0 = (S_0, i_0, E_0, Tran_0)$ and $T_1 = (S_1, i_1, E_1, Tran_1)$ be two transition systems. A* partial morphism *$f : T_0 \to T_1$ is a pair $f = (\sigma, \tau)$ where $\sigma : S_0 \to S_1$ is a function and $\tau : E_0 \to E_1$ is a* partial *function such that*

– $\sigma(i_0) = i_1$,

– $(s, e, s') \in Tran_0$ *and* $\tau(e)$ *is defined implies* $(\sigma(s), \tau(e), \sigma(s')) \in Tran_1$. *Otherwise, if* $\tau(e)$ *is undefined then* $\sigma(s) = \sigma(s')$.

As in [43], we can restrict to *strict* morphisms, i.e. the ones for which τ is a total function, by suitably completing transition systems. Partial morphisms can then be recovered by adding "idle" transitions to the systems, similarly to the construction of the category of sets and partial functions as the Kleisli category associated to the free pointed set monad ? on **Set** given in Section 1.3.

An idle transition is a transition $*$ which goes from a state s to the same state s. Consider the following completion $T_* = (S_*, i_*, E_*, Tran_*)$ of a transition system $T = (S, i, E, Tran)$, by setting $S_* = S$, $i_* = i$, $E_* = E \uplus \{*\}$ and $Tran_* = Tran \uplus \{(s, *, s) \mid s \in S\}$. Now, by the preceding remarks a total morphism (σ, τ) from $(T_0)_*$ to $(T_1)_*$ such that $\tau(*) = *$ is the same as a partial morphism from T_0 to T_1. Again, the operation $(-)_*$ induces a monad on the category **sTS** of transition systems and strict morphisms, and the category **TS** can be recovered as the Kleisli category associated to this monad. Likewise, all the models for concurrency considered in this article admit a "strict" variant, from which the "non-strict" model can be reconstructed by a Kleisli construction. For lack of space we will not detail all the variants here.

Example 20. *The CCS processes $a\cdot(b+c)$, $a\cdot(b|c)$ and $a\cdot(b\cdot c+c\cdot b)$ respectively induce the following transition systems:*

$$
\begin{tikzcd}
y \\
y \arrow[u, bend left, "b"] \arrow[u, bend right, "c"'] \\
x \arrow[u, "a"]
\end{tikzcd}
\qquad
\begin{tikzcd}
& z & \\
y_1 \arrow[ur, "c"] & & y_2 \arrow[ul, "b"'] \\
& y \arrow[ul, "b"] \arrow[ur, "c"'] & \\
& x \arrow[u, "a"] &
\end{tikzcd}
\qquad
\begin{tikzcd}
& z & \\
y_1 \arrow[ur, "c"] & & y_2 \arrow[ul, "b"'] \\
& y \arrow[ul, "b"] \arrow[ur, "c"'] & \\
& x \arrow[u, "a"] &
\end{tikzcd}
$$

2.2 Asynchronous automata

Asynchronous automata are a nice generalization of both transition systems and Mazurkiewicz traces, and have influenced a lot of other models for concurrency, such as transition systems with independence (or asynchronous transition systems). They have been independently introduced in [35] and [2]. The idea is to decorate transition systems with an *independence* relation between actions that will allow us to distinguish between true-concurrency and mutual exclusion (or non-determinism) of two actions. For example, the two last transition systems of Example 20 do not allow us to distinguish between processes which are arguably different from the concurrency point of view. We actually use a slight modification for our purposes, due to [6], called *automaton with concurrency relations*:

Definition 21. *An* automaton with concurrency relations $(S, i, E, Tran, I)$ *is a quintuple where*

- $(S, i, E, Tran)$ *is a transition system,*
- $Tran$ *is such that whenever* (s, a, s')*,* $(s, a, s'') \in Tran$*, then* $s = s''$*,*
- $I = (I_s)_{s \in S}$ *is a family of irreflexive, symmetric binary relations* I_s *on* E *such that whenever we have* $a_1 \, I_s \, a_2$ *(with* $a_1, a_2 \in E$*), there exist transitions* (s, a_1, s_1)*,* (s, a_2, s_2)*,* (s_1, a_2, r) *and* (s_2, a_1, r) *in* $Tran$*.*

A morphism of automata with concurrency relations consists of a morphism (σ, τ) between the underlying transition systems such that $a \, I_s \, b$ implies that $\tau(a) \, I'_{\sigma(s)} \, \tau(b)$ whenever $\tau(a)$ and $\tau(b)$ are both defined. This makes automata with concurrency relations into a category, written **ACR**. We also write **sACR** for the variant of this category where morphisms are strict morphisms. Again, the category **sACR** can be constructed from **ACR** by a Kleisli construction, using $*$-transitions and total morphisms (the condition on the independence relation is then that $a \, I_s \, b$ implies $\tau(a) \, I'_{\sigma(s)} \, \tau(b)$ whenever $\tau(a) \neq *$ and $\tau(b) \neq *$).

Example 22. *The CCS processes* $a \cdot (b|c)$ *and* $a \cdot (b \cdot c + c \cdot b)$ *induces the labeled asynchronous transition systems whose underlying transition system are isomorphic and shown in Example 20. The independence relation contains* $e_b I_y e_c$ *for the first process (where* e_b *and* e_c *are the events with source* y*, labeled respectively by* b *and* c*) and is empty for the second process.*

2.3 Event structures

Event structures were introduced in [28, 41] in order to abstract away from the precise places and times at which events occur in distributed systems. The idea is to focus on the notion of event and the causal ordering between them. We recall below the definition of (unlabeled prime) event structures.

Definition 23. *An* event structure $(E, \leqslant, \#)$ *consists of a poset* $(E, \leqslant)$ *of* events*, the partial order relation expressing* causal dependency*, together with a symmetric irreflexive relation* $\#$ *called* incompatibility *satisfying*

- *finite causes: for every event* e*, the set* $\{ \, e' \mid e' \leqslant e \, \}$ *is finite,*

– *hereditary incompatibility: for every events e, e' and e'', $e \# e'$ and $e' \leqslant e''$ implies $e \# e''$.*

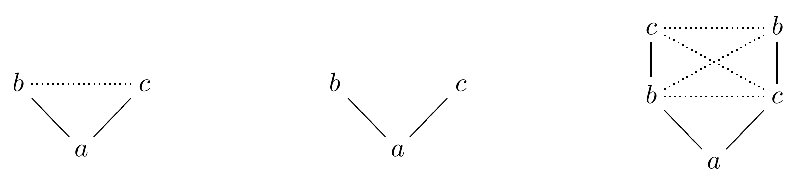
We write **ES** for the category of event structures, a morphism between two event structures $(E, \leqslant, \#)$ and $(E', \leqslant', \#')$ consisting of a partial function $f : E \to E'$ which is such that

– if $f(e)$ is defined then $\{ e' \mid e' \leqslant f(e) \} \subseteq f(\{ e'' \mid e'' \leqslant e \})$,

– and if $f(e_0)$ and $f(e_1)$ are both defined and we have either $f(e_0) \#' f(e_1)$ or $f(e_0) = f(e_1)$ then either $e_0 \# e_1$ or $e_0 = e_1$.

A *labeled event structure* consists of an event structure $(E, \leqslant, \#)$ together with a set L of *labels* and a *labeling function* $\ell : E \to L$ which to every event associates a label. A morphism $(f, \lambda) : (E, \leqslant, \#, \ell, L) \to (E', \leqslant', \#', \ell', L')$ of labeled event structure consists of a morphism $f : (E, \leqslant, \#) \to (E', \leqslant', \#')$ between the underlying event structures and a partial function $\lambda : L \to L'$ between the sets of labels such that $\ell' \circ f = \lambda \circ \ell$. We write **LES** for the category of labeled event structures. We also write **sES** (resp. **sLES**) for the category of *strict* (labeled) event structures, defined as the subcategory of **ES** (resp. **LES**) whose morphisms are total functions – these categories can also be obtained by suitable Kleisli constructions.

Example 24. *The CCS processes $a \cdot (b+c)$, $a \cdot (b|c)$ and $a \cdot (b \cdot c + c \cdot b)$ respectively induce the following labeled event structures (to be read from bottom up, the continuous lines representing the partial order and the dotted ones expressing incompatibilities):*



Notice that in the last one, b and c appear twice: this is because we have figured the labels and not the events (and two distinct events can of course have the same label).

2.4 Petri nets

Petri nets are a well-known model of parallel computation, generalizing transition systems by using a built-in notion of resource. This allows for deriving a notion of independence of events, which is much more general than the independence relation of asynchronous transition systems. They are numerous variants of Petri nets since they were introduced in [29], and we choose the definition used by Winskel and Nielsen in [43], since this is well-suited for formal comparisons with other models for concurrency:

Definition 25. *A* Petri net N *is a tuple* $(P, M_0, E, \mathrm{pre}, \mathrm{post})$ *where*

– P *is a set of* places*,*

– $M_0 \in \mathbb{N}^P$ *is the* initial marking*,*

– E *is a set of* events*,*

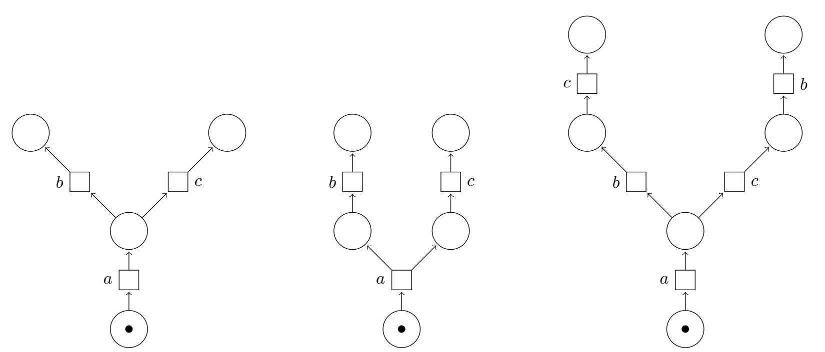
– $\mathrm{pre} : E \to \mathbb{N}^P$ *and* $\mathrm{post} : E \to \mathbb{N}^P$ *are the* precondition *and* postcondition *functions.*

When there is no ambiguity, given an event e of a Petri net N, we often write ${}^\bullet e$ for $\mathrm{pre}(e)$ and $e^\bullet$ for $\mathrm{post}(e)$. A *marking* M is a function in $\mathbb{N}^P$, which associates to every place the number of resources (or tokens) that it contains. The sum M_1+M_2 of two markings M_1 and M_2 is their pointwise sum. An event e induces a *transition* between two markings M_1 and M_2, that we write $M_1 \xrightarrow{e} M_2$, whenever there exists a marking M such that $M_1 = M + {}^\bullet e$ and $M_2 = M + e^\bullet$.

A *morphism of Petri nets* $(\varphi, \psi) : N \to N'$, between the two Petri nets $N = (P, M_0, E, \mathrm{pre}, \mathrm{post})$ and $N' = (P', M_0', E', \mathrm{pre}', \mathrm{post}')$, consists of a function $\varphi : P' \to P$ and a partial function $\psi : E \to E'$ such that for every place $p \in P'$ and event $e \in E$, $M_0' = M_0 \circ \varphi$, ${}^\bullet\psi(e) = {}^\bullet e \circ \varphi$ and $\psi(e)^\bullet = e^\bullet \circ \varphi$. We write **PNet** for the category of Petri nets and **sPNet** for the subcategory whose morphisms have total functions on events. Notice that the partial function $\varphi : P' \to P$ on places goes "backwards". This might seem a bit awkward at first sight and we explain why this is the "right" notion of morphism in Remark 37.

A *labeled Petri net* is a Petri net together with a set L of labels and a function $\ell : E \to L$ labeling events. The notion of morphism of Petri nets can be extended in a straightforward way to labeled ones and we write **LPNet** for the category of labeled Petri nets and **sLPNet** for the subcategory whose morphisms are total functions.

Example 26. *The CCS processes $a\cdot(b+c)$, $a\cdot(b|c)$ and $a\cdot(b\cdot c+c\cdot b)$ respectively induce the following labeled Petri nets:*



In the diagrams above, we have used the usual notation for Petri nets: square nodes represent transitions, circled ones represent places (with dots indicating tokens) and arrows represent pre- and postconditions.

3 Relating models for concurrency

The purpose of this section is to relate traditional models introduced in Section 2 with the geometric models of Section 1 (mainly HDA).

3.1 Transition systems and HDA

In this section, we relate labeled transition systems and HDA. We begin by relating transition systems to the category of 1-dimensional HDA by defining two adjoint functors

$$F : \mathbf{sHDA}_1 \to \mathbf{sTS} \qquad \text{and} \qquad G : \mathbf{sTS} \to \mathbf{sHDA}_1$$

We define the functor F as follows. To a 1-dimensional HDA C labeled by L, we associate the transition system $(S, i, E, Tran)$ defined by $S = C(0)$, i being the distinguished element of C, $E = L$ and the transitions being defined by $Tran = \{\ (\partial_0^-(e), \ell(e), \partial_0^+(e)) \mid e \in E\ \}$. And to any morphism $(\varphi, \lambda) : C \to D$ between labeled precubical sets, we associate the morphism (σ, τ) which is defined by $\sigma = \varphi_0 : C(0) \to D(0)$ and $\tau = \lambda$. The functor is defined in the obvious way on morphisms.

Conversely, the functor G is defined as follows. To any transition system $T = (S, i, E, Tran)$, we associate the strict 1-dimensional HDA C labeled by E whose underlying precubical set C is such that $C(0) = S$, $C(1) = Tran$, the face morphisms $\partial_0^- : C_1 \to C_0$ and $\partial_0^+ : C_1 \to C_0$ are respectively defined by $\partial_0^-(s, e, s') = s$ and $\partial_0^+(s, e, s') = s'$, the labeling function is defined by $\ell(s, e, s') = e$ and the distinguished element is the distinguished element $i \in C(0)$. To any morphism $(\sigma, \tau) : (S_1, i_1, E_1, Tran_1) \to (S_2, i_2, E_2, Tran_2)$ we associate the morphism (κ, λ) of HDA, where κ is the morphism of pointed 1-dimensional precubical set whose components are $\kappa_0 = \sigma$ and $\kappa_1 = \tau$, the morphism λ between labels being τ. The functor is defined in the obvious way on morphisms.

The functors defined above enable us to relate both models:

Theorem 27. *The functor $F : \mathbf{sHDA}_1 \to \mathbf{sTS}$ defined above is left adjoint to the functor $G : \mathbf{sTS} \to \mathbf{sHDA}_1$. The comonad $F \circ G$ on $\mathbf{sTS}$ is the identity and the adjunction restricts to an equivalence of categories between the full subcategory of $\mathbf{sHDA}_1$ whose objects are strongly labeled.*

Proof. Suppose given a transition system $T = (S, i, E, Tran)$ and a 1-dimensional HDA $C = (C, i)$. We construct a natural bijection between morphisms $FC \to T$ in $\mathbf{sTS}$ and morphisms $C \to GT$ in $\mathbf{sHDA}_1$. To every morphism $(\sigma, \tau) : FC \to T$ of transition systems we associate the morphism of HDA $\varphi_{C,T}(\sigma, \tau) : C \to GT$ defined as (κ, λ) where $\kappa_0 = \sigma$ and $\kappa_1 = \lambda = \tau$. Conversely, to every morphism $(\kappa, \lambda) : C \to GT$ of HDA we associate the morphism $\psi_{C,T}(\kappa, \lambda) : FC \to T$ of transition systems defined as (κ_0, λ). These operations are mutually inverse and can easily be shown to be natural. The second part of the proposition can be checked directly. $\square$

Now, recall that the category **TS** can be defined as the Kleisli category associated to the monad $(-)_*$ on **sTS**. Similarly, the adjunction between $\mathbf{sHDA}_1$ and $\mathbf{HDA}_1$ given in Proposition 15 induces a monad ? on $\mathbf{sHDA}_1$ which "replaces" the underlying precubical set of an HDA by the cubical set it generates.

Theorem 28. *The adjunction of Theorem* (27) *lifts to an adjunction between* **TS** *and* $\mathbf{HDA}_1$*, which induces an equivalence if we restrict* $\mathbf{HDA}_1$ *to strongly labeled cubical sets.*

Proof. Consider a strict 1-dimensional HDA consisting of a precubical set C labeled by ℓ into L. Its image under the left adjoint $F : \mathbf{sHDA}_1 \to \mathbf{HDA}_1$ to the forgetful functor $\mathbf{HDA}_1 \to \mathbf{sHDA}_1$ is the 1-dimensional HDA whose underlying cubical set is D defined by $D(0) = C(0)$, $D(1) = C(1) \uplus C(0)$ with face maps being $\partial_i^\alpha \uplus \mathrm{id}_{C(0)} : D(1) \to D(0)$ as face maps and the canonical injection $\iota_0 : D(0) \to D(1)$ as degeneracy maps, whose labeling is obtained by extending $\ell_1 : C(1) \to (!\,L)_1$ to $D(1)$ by $\ell_1(x) = (*)$ for $x \in C(0) \subseteq C(1)$. From this concrete description, it can easily be checked that $?\circ G = G \circ (-)_*$ and that the unit and the multiplication of $(-)_*$ are sent by G to the unit and multiplication of T. Finally, we deduce that the adjunction of Theorem 27 lifts to an adjunction between the Kleisli categories **TS** and $\mathbf{HDA}_1$ respectively associated to the monads $(-)_*$ and ? using Proposition 5. $\square$

Remark 29. The fact that we have to restrict to a subcategory of $\mathbf{sHDA}_1$ in Theorem 27 in order to obtain an equivalence of categories can be explained intuitively by remarking that in transition systems there is no distinction between events and labels: in particular, a transition system cannot contain two distinct transitions with the same event between the same source and the same target. For example, the following labeled (pre)cubical set cannot be represented in transition systems:

$$\begin{tikzcd} y \\ x \arrow[u, bend left, "a"] \arrow[u, bend right, "a"'] \end{tikzcd}$$

More generally, in higher dimensions most models do not have the possibility to "count" the number of commutations between events: usually, two transitions either commute or not. This contrasts with cubical sets where a tile

$$\begin{tikzcd} & x_3 & \\ x_1 \arrow[ur, "y_3"] & & x_2 \arrow[ul, "y_4"'] \\ & x_0 \arrow[ul, "y_1"] \arrow[ur, "y_2"'] & \end{tikzcd}$$

can be filled with many 2-cells. This explains why in the following most of the nice adjunctions will be obtained by restricting cubical sets to strongly labeled ones.

By Proposition 15, the truncation functor $\mathbf{HDA} \to \mathbf{HDA}_1$ admits a right adjoint. By composing this adjunction with the one of previous theorem, we obtain an adjunction between **TS** and **HDA**.

Remark 30. The HDA associated by the right adjoint to a transition system $T = (S, i, E, Tran)$ can be described in a more direct way using Proposition 35 as generated by the cubical transition system $(S, i, E, \ell, E \uplus \{*\}, t)$ where $\ell : E \to E \uplus \{*\}$ is the canonical injection and $t(x, l) = y$ if l is reduced to an event e and $(x, e, y) \in Tran$, see Section 3.4 for details.

3.2 Asynchronous automata and HDA

The adjunction given in previous section, can be extended to an adjunction between the category of strict asynchronous automata **sACR** and the category of strict 2-dimensional HDA $\mathbf{sHDA}_2$.

To any strict 2-dimensional HDA C, the left adjoint $F : \mathbf{sHDA}_2 \to \mathbf{sACR}$ associates the asynchronous automaton whose underlying transition system is induced by the underlying 1-dimensional HDA of C and such that $a_1 \, I_s \, a_2$ when there exists transitions (s, a_1, s_1), (s, a_2, s_2), (s_1, a_2, r) and (s_2, a_1, r) and a 2-cell y such that $\partial_0^-(y) = (s, a_1, s_1)$, $\partial_0^+(y) = (s_2, a_1, r)$, $\partial_1^-(y) = (s, a_2, s_2)$ and $\partial_1^+(y) = (s_1, a_2, r)$:

$$\begin{tikzcd} & r & \\ s_1 \arrow[ur, "a_2"] & y & s_2 \arrow[ul, "a_1"'] \\ & s \arrow[ul, "a_1"] \arrow[ur, "a_2"'] & \end{tikzcd} \tag{6}$$

The functor is defined in the obvious way on morphisms.

Conversely, an asynchronous automaton $A = (S, i, E, Tran, I)$ is sent by the right adjoint $G : \mathbf{sACR} \to \mathbf{sHDA}_2$ to a strict 2-dimensional HDA C, whose underlying 1-dimensional HDA is induced by the underlying transition system of A. The 2-cells are $C(2) = I$, where I is seen as a subset of $E \times S \times E$. Given a pair of events a_1 and a_2 related by I_s for some state s, there exist transitions (s, a_1, s_1), (s, a_2, s_2), (s_1, a_2, r) and (s_2, a_1, r) and these are uniquely defined by the second property of Definition 21 as in (6), face maps are defined on elements $y = a_1, s, a_2$ of I by

$$\partial_0^-(y) = (s, a_1, s_1) \quad \partial_0^+(y) = (s_2, a_1, r) \quad \partial_1^-(y) = (s, a_2, s_2) \quad \partial_1^+(y) = (s_1, a_2, r)$$

and the labeling function is defined by $\ell(a_1, s, a_2) = (a_1, a_2)$. The requirement that I is symmetric induces the symmetry of the HDA. The functor is defined in the obvious way on morphisms.

Theorem 31. *These functors form an adjunction between* **sACR** *and* $\mathbf{sHDA}_2$*. The induced comonad on* **sACR** *is the identity and the adjunction induces an equivalence of categories if we restrict* $\mathbf{sHDA}_2$ *to the full subcategory of strongly labeled HDA. Moreover, this adjunction lifts to an adjunction between* **ACR** *and* $\mathbf{HDA}_2$ *with similar properties.*

By composing with an adjunction given by Proposition 15, this induces an adjunction between **ACR** and **HDA**.

3.3 Event structures and HDA

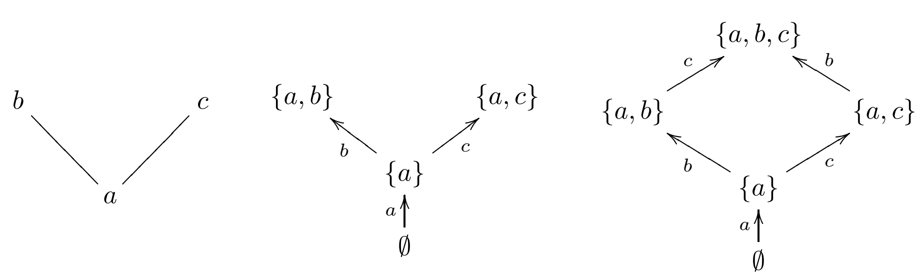
We construct here an adjunction between **sLES** and **sHDA**. This adjunction reformulates in the framework of HDA some well-known relations between event structures and transition systems with independence [33]. The study of relations between the two models was initiated in [5] and a similar connection is described in [36].

A *configuration* of an event structure $(E, \leqslant, \#)$ is a finite downward closed subset of compatible events in E. An event e is *enabled* at a configuration x if $e \notin x$ and $x \uplus \{e\}$ is a configuration. A functor $F : \mathbf{sLES} \to \mathbf{sHDA}_2$ can be defined as follows. To any labeled event structure $(E, \leqslant, \#, \ell, L)$, it associates the 2-dimensional HDA C labeled by L whose 0-cells are the configurations of the event structure with the empty configuration as initial state, 1-cells are the pairs (x, e) where x is a configuration and e is an event enabled at x, and 2-cells are the pairs (x, e_1, e_2) where x is a configuration and e_1, e_2 are both enabled at x and such that e_2 is enabled at $x \uplus \{e_1\}$ and e_1 is enabled at $x \uplus \{e_2\}$, graphically:

$$\begin{tikzcd}
 & x \uplus \{e_1, e_2\} & \\
x \uplus \{e_1\} \arrow[ur, "(x\uplus\{e_1\}, e_2)"] & (x, e_1, e_2) & x \uplus \{e_2\} \arrow[ul, "(x\uplus\{e_2\}, e_1)"'] \\
 & x \arrow[ul, "(x, e_1)"] \arrow[ur, "(x, e_2)"'] &
\end{tikzcd}$$

Notice for every 2-cell (x, e_1, e_2), (x, e_2, e_1) is also a 2-cell thus inducing a symmetry on the precubical set. The functor is defined in the obvious way on morphisms.

Example 32. *Consider the event structure $(E, \leqslant, \#, \ell, L)$, with $E = \{e_1, e_2, e_3\}$, with $e_1 \leqslant e_2$ and $e_1 \leqslant e_3$, labeled in $L = \{a, b, c\}$ by $\ell(e_1) = a$, $\ell(e_2) = b$ and $\ell(e_3) = c$. This event structure is represented on the left and induces the two HDA on the right when b is respectively incompatible and compatible with c*



(for simplicity we simply write e for a 1-cell (x, e) since x can be determined as the source of the cell). The square on the right diagram is filled with two 2-cells: $(\{a\}, b, c)$ and $(\{a\}, c, b)$.

Conversely, we define a functor $G : \mathbf{sHDA}_2 \to \mathbf{sLES}$. The intuition is that given an HDA C, the elements of $G(C)$ should be the events of C in the sense of Section 1.1. However, event structures cannot express loops, which should therefore be unfolded [43, 40, 8]. For example, an HDA of the form

$$\begin{tikzcd} x \arrow[loop above, "a"] \end{tikzcd} \tag{7}$$

with only one 0-cell and one looping 1-cell should have as image an event structure with a countable totally ordered set of events. A 2-dimensional HDA is *unfolded* when it is

- *reachable*: every 0-cell x is reachable,
- *acyclic*: any path $s : x \twoheadrightarrow x$ with the same source and target is empty,
- *unshared*: any two parallel paths $s, t : x \twoheadrightarrow x'$ are homotopic.

This reformulates the notion of occurrence transition system with independence. To any 2-dimensional HDA C with i as initial state and $\ell : C \to !\,L$ as labeling function, one can associate an unfolded 2-dimensional HDA $U(C)$ whose

- 0-cells are the paths $s : i \twoheadrightarrow x$ of C modulo homotopy,
- 1-cells are the pairs (s, m) where $s : i \twoheadrightarrow x$ is a path and m is a 1-cell such that $\partial_0^-(m) = x$, with $\partial_0^-(s, m) = s$ and $\partial_0^+(s, m) = s \cdot m$ as source and target,
- 2-cells are pairs (s, z) where $s : i \twoheadrightarrow x$ is a path and z is a 2-cell such that $\partial_0^-\partial_0^-(z) = x$, with its faces defined by $\partial_i^-(s, z) = (s, \partial_i^-(z))$ and $\partial_i^+(s, z) = (s \cdot \partial_{1-i}^-(z), \partial_i^+(z))$,
- the labeling function labels a 1-cell (s, m) by $\ell(m)$ and a 2-cell (s, z) by $\ell(z)$.

This operation can easily be extended into a comonad on the category $\mathbf{sHDA}_2$. For example, the image of the HDA (7) is

$$x_0 \xrightarrow{a} x_1 \xrightarrow{a} x_2 \xrightarrow{a} \ldots$$

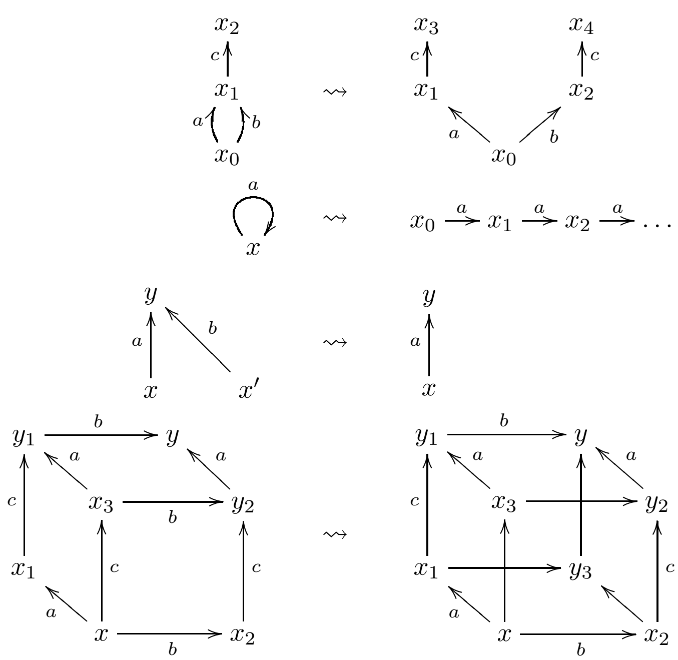
Now, to every unfolded 2-dimensional HDA C, one can associate a labeled event structure $V(C) = (E, \leqslant, \#, \ell, L)$ such that E is the set of events of C in the sense of Section 1.1. We say that an event e *occurs* in a path s when s contains a 1-cell m such that $m \in e$. Two events e and e' are such that $e \leqslant e'$ when for every path $s \cdot n : i \twoheadrightarrow x$ with $n \in e'$ the event e occurs in s. Two events e

and e' are such that $e\#e'$ when there is no path $s : i \longrightarrow\!\!\!\!\!\rightarrow x$ such that both e and e' occur in s. The labeling function is the labeling function of C (recall that we have shown in Section 1.5 that every labeled cubical set induces a labeling function on its events). The operation V is easily extended as a functor V from the category of unfolded 2-dimensional HDA to the category of labeled event structures. Finally, we define the functor $G : \mathbf{sHDA}_2 \to \mathbf{sLES}$ as the composite $G = V \circ U$.

Theorem 33. *The composite functor $G \circ F$ is isomorphic to the identity functor on* **sLES**. *Thus* **sLES** *embeds fully and faithfully into* $\mathbf{sHDA}_2$.

Proof. The adjunction between labeled event structure and transition systems with independence described [33, 25] can be straightforwardly adapted to asynchronous transition systems and one obtains the result by composing with the adjunction described in previous section. □

Notice that we did not claim that F and G are part of an adjunction, because it is not the case. Namely, consider the effect of the endofunctor $F \circ G : \mathbf{sHDA}_2 \to \mathbf{sHDA}_2$: we have pictured some HDA (on the left) together with their image under $F \circ G$ (on the right):



$$
\begin{tikzcd}
& x_3 & & & & x_3 & \\
x_1 \arrow[ur, bend left, "b"] \arrow[ur, bend right, "b"'] & & x_2 \arrow[ul, "a"'] & \rightsquigarrow & x_1 \arrow[ur, "b"] & & x_2 \arrow[ul, "a"'] \\
& x_0 \arrow[ul, "a"] \arrow[ur, "b"'] & & & & x_0 \arrow[ul, "a"] \arrow[ur, "b"'] &
\end{tikzcd}
$$

In the third example, x is the initial position and in the last two examples all the squares for which it makes sense are filled with 2-cells. These examples are representative of various kinds of behaviors that can happen:

- the first two examples show that "shared transitions" are "unshared", and in particular loops are unrolled;
- the third example shows that the unreachable 0-cells of the HDA are removed,
- the fourth example shows that if the HDA contains half of a cube then the other half of the cube is created, completing the cube – this is related to the *cube axiom* which is often used to characterize asynchronous transition systems generated by an event structure [33];
- the last example shows that HDA are made strongly labeled.

Notice, if we write C for the HDA in the left of examples, in the first three examples there is a natural arrow $TC \to C$ (but not in the other direction), whereas in the last two examples there is a natural arrow $C \to TC$ (but not in the other direction). So there is no hope that T would be either a monad or a comonad, and thus that F and G either form an adjunction in either direction.

It can however be shown that G is right adjoint to F if we restrict $\mathbf{sHDA}_2$ to the full subcategory whose objects are strongly labeled and satisfy the *cube axioms* (which state that if an asynchronous transition system contains half of a cube as in fourth example then it also contains the other half of the cube, as well as two other variants of this property). As previously, this adjunction can be extended to the non-strict variants of the models, as well as the whole category **HDA**. This adjunction can also be extended to an adjunction between general event structures (in which conflict is not necessarily a binary relation) and HDA.

3.4 Petri nets and HDA

This section constitutes perhaps the most novel part of the paper. We extend here previously constructed adjunctions between 1-bounded Petri Nets and asynchronous transition systems [43, 6, 26, 39] to an adjunction between general Petri Nets and HDA. For similar reasons as previously, one needs to restrict to strongly labeled HDA in order to obtain a well-defined adjunction. We thus implicitly only consider strongly labeled HDA in the following.

Cubical transition systems. We introduce here a general methodology for associating a symmetric precubical set to a model for concurrent processes, that we will use in order to associate a strict HDA to a Petri net. Since monoidal functors preserve the unit of monoidal categories, all cubical sets generated by cubical objects in **Set** (i.e. by the functor ! introduced in Section 1.5) contain only one 0-cell. Cubical sets with multiple 0-cells can be generated by *actions* of the labeling cubical set on the 0-cells, formalized as follows, in the same way that a transition system can be seen as an action of the free monoid on labels over the states. The resulting notion of *cubical transition system* (or *CTS*) generalizes to the setting of cubical set the notion of *step transition system* [26] which is a variant of transition systems in which multiple events can occur simultaneously.

Definition 34. *A* cubical transition system (S, i, E, t, ℓ, L) *consists of*

- *a set S of* states*,*
- *a state $i \in S$ called the* initial state*,*
- *a set E of* events*,*
- *a* transition function *which is a partial function $t : S \times {!E} \to S$,*
- *a set L of* labels*,*
- *a* labeling function $\ell : E \to L$*,*

such that for every state x and every n-cell l of ${!E}$ for which $t(x, l)$ is defined,

1. *if $l = l_1 \cdot l_2$ for some cells l_1 and l_2 then $t(x, l_1)$ and $t(t(x, l_1), l_2)$ are both defined and we have $t(x, l) = t(t(x, l_1), l_2)$,*
2. *$t(x, ())$ is defined and equal to x (where $()$ denotes the 0-cell of ${!E}$),*
3. *for every symmetry $\sigma : n \to n$, $t(x, {!E}(\sigma)(l))$ is defined and equal to $t(x, l)$.*

Cubical transition systems are thus generalized transition systems, which modify state upon incoming events. These differ from traditional transition systems in that they may accept a transition under n events $e_1, \ldots, e_n$, specified by a transition under the word $e_1 \cdots e_n \in {!E}$. With this understanding in mind, the axioms have simple interpretations: for example the first one states that the state reached under two simultaneous events e_1 and e_2 is the same as the state reached under e_1 followed by e_2.

An n-cell l of ${!E}$ is *enabled* at a position x if $t(x, l)$ is defined. Every such CTS defines a strict HDA C labeled by L whose n-cells are pairs (x, l) where x is a state and l is an n-cell of ${!E}$ which is enabled at x. Source and target functions are defined by $\partial_i^-(x, l) = (x, \partial_i^-(l))$ and $\partial_i^+(x, l) = t(t(x, e_i), \partial_i^+(l))$ where e_i is the i-th element of l and symmetries by $\sigma(x, l) = (x, {!E}(\sigma)(l))$. The labeling function is ${!\ell}$ and the initial state is i.

A *morphism* $(\sigma,\tau,\lambda):(S_1,i_1,E_1,\ell_1,L_1,t_1)\to(S_2,i_2,E_2,\ell_2,L_2,t_2)$ between two CTS consists of

– a function $\sigma:S_1\to S_2$,

– a function $\tau:E_1\to E_2$,

– a function $\lambda:L_1\to L_2$,

such that $i_2=\sigma(i_1)$, $\ell_2\circ\tau=\lambda\circ\ell_1$, and for every state $x\in S_1$ and cell l of $!\,E_1$, $t_2(\sigma(x),!\,\tau(l))=\sigma\circ t_1(x,l)$. Every such morphism induces a morphism $(\kappa,\lambda):C_1\to C_2$ between the corresponding HDA C_1 and C_2 defined on n-cells (x,l) of C_1 by $\kappa_n(x,l)=(\sigma(x),!\,\tau(l))$. We write **CTS** for the category thus defined.

Proposition 35. *The functor* **CTS** $\to$ **sHDA** *defined above is well-defined.*

Remark 36. A variant of the notion of cubical transition system can easily be defined in order to generate symmetric cubical sets.

From Petri nets to HDA. Suppose that we are given a labeled Petri net $N=(P,M_0,E,\mathrm{pre},\mathrm{post},\ell,L)$. The pre and post operations can be extended to the cells of $!\,E$ by ${}^\bullet()={}^\bullet(*)=0$, ${}^\bullet(l_1\cdot l_2)={}^\bullet l_1+{}^\bullet l_2$, $()^\bullet=(*)^\bullet=0$ and $(l_1\cdot l_2)^\bullet=l_1^\bullet+l_2^\bullet$. This enables us to see elements of $!\,E$ as generalized events. We also generalize the notion of transition and given two markings M_1 and M_2 and an event $l\in\,!\,E$, we say that there is a transition $M_1\xrightarrow{l}M_2$ whenever there exists a marking M such that $M_1=M+{}^\bullet l$ and $M_2=M+l^\bullet$. In this case, the event l is said to be *enabled* at the marking M_1. The marking M_2 is sometimes denoted M_1/l. A marking M is *reachable* if there exists a transition l such that $M=M_0/l$ where M_0 is the initial marking of N.

Remark 37. As in [43], we have chosen to define morphisms in the opposite direction on places. With the adjunction with HDA in mind, this can be explained as follows. Morphisms of Petri nets should, just as morphisms of HDA, preserve independence of events: if two events e and e' of a net N are independent and $(\varphi,\psi):N\to N'$ is a morphism of nets, then their images $\psi(e)$ and $\psi(e')$ should also be independent. By contraposition, this means that if both events $\psi(e)$ and $\psi(e')$ depend on a common place p, then the events e and e' should depend on a corresponding common place $\psi^{-1}(p)$.

Every labeled Petri net N induces a CTS (S,i,E,t,ℓ,L) whose states S are the reachable markings of the net, with the initial marking M_0 as initial state, events E are the events of the net, transition function $t(M,l)$ is defined if and only if l is enabled at M and in this case $t(M,l)=M/l$, with the set L as set of labels and $\ell:E\to L$ as labeling function.

It is routine to verify that this actually defines a CTS and thus a strict HDA. The n-cells of $\mathrm{hda}(N)$ consisting of a marking M of the net and a list l of events which is enabled at M. Moreover, any morphism $(\varphi,\psi):N\to N'$ between labeled Petri nets induces a morphism (σ,τ,λ) between the corresponding CTS defined by $\sigma(M) = M \circ \varphi$ for any reachable marking M of N, $\tau = \psi$, and $\lambda = \psi$. We denote by $\mathrm{hda} : \mathbf{sLPNet} \to \mathbf{sHDA}$ the functor thus defined.

From HDA to Petri nets. We first introduce the notion of region of an HDA, which should be thought as a way of associating a number of tokens to each 0-cell of the HDA and a pre- and postcondition to every transition of the HDA, in a coherent way. A pre-region R of a precubical set C is a sequence $(R_i)_{i \in \mathbb{N}}$ of functions $R_i : C(i) \to \mathbb{N} \times \mathbb{N}$ such that

- for every $x \in C(0)$, $R_0(x) = (0, 0)$
- for every $x \in C(i+1)$ and $\alpha_k \in \{-, +\}$,

$$R_{i+1}(x) \quad = \quad \sum_{k=0}^{i} R_1(\partial^{\alpha_k}_{\neg k}(x))$$

  where the sum is computed coordinate by coordinate on pairs of integers.

Notice that, by the second property, a region is uniquely determined by the image of 1-dimensional cells in $x \in C(1)$. We sometimes omit the index i since it is determined by the dimension of the cell in argument and respectively write $R'(x)$ and $R''(x)$ for the first and second components of $R(x)$, where x is a cell of C. It can be remarked that two 1-cells which are part of the same event necessarily have the same image under a pre-region; a pre-region R thus induces a function from the events of C to $\mathbb{N} \times \mathbb{N}$, that we still write R. A *region* of a precubical set consists of a pre-region R together with a function $S : C(0) \to \mathbb{N}$ such that for every i-cell $y \in C(i)$ whose 0-source is x and 0-target is x', there exists an integer n such that $(S(x), S(x')) = (n + R'(y), n + R''(y))$.

Example 38. *Consider the following precubical set*

$$\begin{tikzcd}
 & x_3 & & \\
x_1 \arrow[ur, "y_1"] & z & x_2 \arrow[ul, "y_3"'] & \\
 & x_0 \arrow[ul, "y_0"] \arrow[ur, "y_2"'] \arrow[rr, "y_4"] & & x_4
\end{tikzcd}$$

A region (R, S) for this cubical set is for example

$$R(y_0) = (2,1) \quad R(y_1) = (3,1) \quad R(y_2) = (3,1) \quad R(y_3) = (2,1) \quad R(y_4) = (0,2)$$

and

$$R(z) = (5,2) \quad S(x_0) = 6 \quad S(x_1) = 5 \quad S(x_2) = 4 \quad S(x_3) = 3 \quad S(x_4) = 8$$

Graphically,

$$
\begin{tikzcd}
 & 3 & & \\
5 \arrow[ur, "{(3,1)}"] & (5,2) & 4 \arrow[ul, "{(2,1)}"'] & \\
 & 6 \arrow[ul, "{(2,1)}"] \arrow[ur, "{(3,1)}"'] \arrow[rr, "{(0,2)}"] & & 8
\end{tikzcd}
$$

To every strict HDA C, we associate a labeled Petri net $\mathrm{pn}(C)$ whose

- places are the regions of C,
- events are the events of C, labeled as in C,
- pre and post functions are given on any event e and any place (R, S) by ${}^\bullet e(R, S) = R'(e)$ and $e^\bullet(R, S) = R''(e)$,
- initial marking M_0 is $M_0(R, S) = S(x_0)$, where x_0 is the initial state of C.

Suppose that $(\kappa, \lambda) : C \to D$ is a morphism of HDA. We define a morphism of labeled Petri nets $\mathrm{pn}(\kappa, \lambda) : \mathrm{pn}(C) \to \mathrm{pn}(D)$ as follows: $\mathrm{pn}(\kappa, \lambda) = (\varphi, \psi, \lambda)$, where

- φ maps every region (R, S) of D to the region $\varphi(R, S) = (R \circ \kappa, S \circ \kappa_0)$, where $R \circ \kappa$ denotes the pre-region $(R_i \circ \kappa_i)_{i \in \mathbb{N}}$,
- ψ is the map induced on events by κ_1 (two 1-cells which are part of the same event are sent to 1-cells which are part of the same event by κ_1).

This thus defines a functor $\mathrm{pn} : \mathbf{sHDA} \to \mathbf{sLPNet}$.

3.4.1 The adjunction.

Suppose that we are given an HDA C labeled by ℓ into L, and a labeled net $N = (P, M_0, E, \mathrm{pre}, \mathrm{post}, m, M)$. We want to exhibit a bijection between morphisms $\mathrm{pn}(C) \to N$ in $\mathbf{sLPNet}$ and morphisms $C \to \mathrm{hda}(N)$ in $\mathbf{sHDA}$.

To any morphism $(\varphi, \psi, \lambda) : \mathrm{pn}(C) \to N$ of labeled Petri nets, we associate a morphism $(\kappa, \lambda) : C \to \mathrm{hda}(N)$ of HDA defined as follows. Given an n-cell x of C, $\kappa_n(x)$ should be an n-cell of $\mathrm{hda}(N)$, that is a pair $(M_{\kappa_n(x)}, l_{\kappa_n(x)})$ where $M_{\kappa_n(x)}$ is a marking of N and $l_{\kappa_n(x)}$ is a list of events of N which is enabled at $M_{\kappa_n(x)}$. These are defined for every place p of N by $M_{\kappa_n(x)}(p) = S_{\varphi(p)}(y)$, where y is the 0-source of x, and $l_{\kappa_n(x)} = !\,\psi(\overline{\partial^-_{\neg 0}(x)} \cdots \overline{\partial^-_{\neg(n-1)}(x)})$ where $\overline{y}$ denotes the event associated to a 1-cell x.

Conversely, to any morphism $(\kappa, \lambda) : C \to \mathrm{hda}(N)$ of HDA, we associate a morphism of labeled Petri nets $(\varphi, \psi, \lambda) : \mathrm{pn}(C) \to N$ defined as follows. Given an n-cell x, $\kappa_n(x)$ is an n-cell of $\mathrm{hda}(N)$, that is a pair $(M_{\kappa_n(x)}, l_{\kappa_n(x)})$ as above. For every place p, $\varphi(p)$ is the region $(R_{\varphi(p)}, S_{\varphi(p)})$ of C which is defined on

0-cells x by $S_{\varphi(p)}(x) = M_{\kappa_n(x)}(p)$ and on n-cells x by $R_{\varphi(p)} = ({}^\bullet l_{\kappa_n(x)}, l^\bullet_{\kappa_n(x)})$. Given a 1-cell x, its image under $\kappa_1(x)$ is a pair $(M_{\kappa_1(x)}, l_{\kappa_1(x)})$ where $l_{\kappa_1(x)}$ is reduced to one 1-cell y. It is immediate to check that for any other 1-cell x' such that $x \approx x'$, we have that $l_{\kappa_1(x)} \approx l_{\kappa_1(x')}$: it thus makes sense to extend $x \mapsto l_{\kappa_1(x)}$ into a function which to an event e of C associates an event $l_{\kappa_1(e)}$. Given an event e of C, we define $\psi(e) = l_{\kappa_1(e)}$.

It can be shown that these transformations are well defined, are natural in C and N, and are mutually inverse. Therefore,

Theorem 39. *The functor* hda : **sLPNet** $\to$ **sHDA** *is right adjoint to the functor* pn : **sHDA** $\to$ **sLPNet**.

Proof. It is routine to check that the transformations given above are well-defined and natural in C and N. We now show that they are mutually inverse.

Suppose that $(\varphi, \psi, \lambda) : \mathrm{pn}(C) \to N$ is a morphism of Petri nets and consider the associated morphisms

$$(\kappa, \lambda) : C \to \mathrm{hda}(N) \qquad \text{and} \qquad (\varphi', \psi', \lambda) : \mathrm{pn}(C) \to N$$

obtained by successively applying the two transformations above. For any place p of N, $\varphi'(p)$ is a place of $\mathrm{pn}(C)$, that is a region $(R_{\varphi'(p)}, S_{\varphi'(p)})$ of C. By definition of the transformations, we have that for every 0-cell x of C, $S_{\varphi'(p)}(x) = M_{\kappa_n(x)}(p) = S_{\varphi(p)}(x)$ and for every n-cell y of C, the first component of $R_{\varphi'(p)}(x)$ is

$${}^\bullet l_{\kappa_n(x)}(p) = \sum_{i=0}^{n-1} \mathrm{pre} \circ \psi(\overline{\partial_i^-(x)})(p) = \sum_{i=0}^{n-1} \mathrm{pre}(\overline{\partial_i^-(x)})(\varphi(p)) = R'_{\varphi(p)}$$

and similarly $l^\bullet_{\kappa_n(x)}(p) = R''_{\varphi(p)}$, thus $R_{\varphi'(p)} = R_{\varphi(p)}$. Moreover, for every event e of C, $\psi'(e) = \psi(e)$.

Conversely, suppose that $(\kappa, \lambda) : C \to \mathrm{hda}(N)$ is a morphism of cubical sets and consider the associated morphisms

$$(\varphi, \psi, \lambda) : \mathrm{pn}(C) \to N \qquad \text{and} \qquad (\kappa', \lambda) : C \to \mathrm{hda}(N)$$

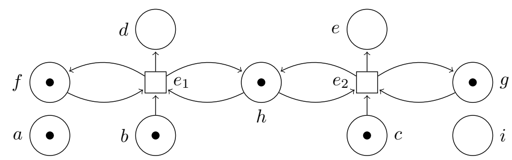
obtained by successively applying the two transformations above. For any n-cell x of C, the n-cell $\kappa'_n(x)$ is an n-cell of $\mathrm{hda}(N)$ consisting of a pair $(M_{\kappa'_n(x)}, l_{\kappa'_n(x)})$ as above. By definition of $\mathrm{hda}(N)$, we have $M_{\kappa'_n(x)} = M_{\kappa'_n(y)}$, where y is the 0-source of x. Moreover, for every place p of N, we have $M_{\kappa'_n(y)}(p) = S_{\varphi(p)}(y) = M_{\kappa_n(y)}(p)$. And finally,

$$l_{\kappa'_n(x)} = !\,\psi(\overline{\partial^-_{\neg 0}(x)} \cdots \overline{\partial^-_{\neg(n-1)}(x)}) = (\kappa_1(\overline{\partial^-_{\neg 0}(x)}) \cdots \kappa_1(\overline{\partial^-_{\neg(n-1)}(x)})) = l_{\kappa_n(x)}$$

which concludes the construction of the adjunction. $\square$

Example 40. *If we restrict to 1-bounded nets, which are nets a place can contain either* 0 *or* 1 *token, we can recover the constructions of [43] for constructing* *an adjunction between asynchronous transition systems and nets. Since the net associated to an HDA by the functor hda is generally infinite, we will give an example in the case of 1-bounded nets. Consider the asynchronous automaton, depicted on the left of* (8)*, with an empty independence relation.*

$$\begin{tikzcd} & z & \\ y_1 \arrow[ur, "e_2"] & & y_2 \arrow[ul, "e_1"'] \\ & x \arrow[ul, "e_1"] \arrow[ur, "e_2"'] & \end{tikzcd}$$



(8)

The associated 1-bounded Petri net is shown on the right. In this automaton the place d corresponds to the region (R, S) *such that* $R(e_1) = (1, 0)$*,* $R(e_2) = (0, 0)$*,* $S(x) = S(y_2) = 1$ *and* $S(y_1) = S(z) = 0$*. Graphically,*

$$\begin{tikzcd} & 0 & \\ 0 \arrow[ur, "(0,0)"] & & 1 \arrow[ul, "(1,0)"'] \\ & 1 \arrow[ul, "(1,0)"] \arrow[ur, "(0,0)"'] & \end{tikzcd}$$

Now, if we consider the same automaton with $e_1 \; I_x \; e_2$*, we obtain the same Petri net with the place h removed. The general (i.e. non-bounded) net associated to an HDA is generally infinite (even for very simple examples) and thus difficult to describe, which is why we did not provide an example in the general case.*

This adjunction can easily be lifted into an adjunction between **LPNet** and **HDA**.

4 Conclusion and future work

In this paper, we have made completely formal the relation between HDA and various classical models of concurrent computations: transition systems, asynchronous automata, event structures and Petri nets. This is not only interesting for comparison purposes, between different semantics of parallel languages, but also, for practical reasons, which will be detailed in a subsequent article.

Stubborn sets [37], sleep sets and persistent sets [14] are methods used for diminishing the complexity of model-checking using transition systems. They are based on semantic observations using Petri nets in the first case and Mazurkiewicz trace theory in the other one. We believe that these are special forms of "homotopy retracts" when cast (using the adjunctions we have hinted) in the category of higher-dimensional transition systems. We shall make this statement more formal through these adjunctions, which will allow for new state-space reduction methods.

Last but not least, in [22] is defined an abstract notion of bisimulation. Given a model for concurrency, i.e. a category of models **M** and a “path category” (a subcategory of **M** which somehow represents what should be thought of as being paths in the models), then we can define two elements of **M** to be bisimilar if there exists a span of special morphisms linking them. These special morphisms have a path-lifting property that, we believe, would be in higher-dimensional transition systems a (geometric) fibration property. We thus hope that homotopy invariants could be useful for the study of a variety of bisimulation equivalences (some work has been done in that direction in [5, 7]).

References

[1] A. Arnold. *Systèmes de transitions finis et sémantique des processus communicants*. Masson, 1992.

[2] M. A. Bednarczyk. *Categories of asynchronous systems*. PhD thesis, 1988.

[3] F. Borceux and G. Janelidze. *Galois theories*. Cambridge Univ Press, 2001.

[4] R. Brown and P. J. Higgins. On the algebra of cubes. *JPAA*, (21):233–260, 1981.

[5] G.L. Cattani and V. Sassone. Higher dimensional transition systems. In *Eleventh Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 55–62, 1996.

[6] M. Droste and RM Shortt. Petri nets and automata with concurrency relations—an adjunction. In *Sem. of Prog. Lang. and Model Theory*, pages 69–87, 1993.

[7] U. Fahrenberg. A category of higher-dimensional automata. *Foundations of Software Science and Computational Structures*, pages 187–201, 2005.

[8] U. Fahrenberg. *Higher-Dimensional Automata from a Topological Viewpoint*. PhD thesis, Aalborg University, 2005.

[9] L. Fajstrup. Loops, ditopology, and deadlocks. *Math. Struct. Comput. Sci.*, 2000.

[10] L. Fajstrup. Dipaths and dihomotopies in a cubical complex. *Advances in Applied Mathematics*, 35(2):188–206, 2005.

[11] L. Fajstrup, E. Goubault, and M. Raußen. Detecting deadlocks in concurrent systems. *CONCUR'98*, pages 332–347.

[12] L. Fajstrup, M. Raußen, and E. Goubault. Algebraic topology and concurrency. *Theoretical Computer Science*, 357(1-3):241–278, 2006.

[13] P. Gaucher. Combinatorics of labelling in higher-dimensional automata. *Theoretical Computer Science*, 411(11-13):1452–1483, 2010.

[14] P. Godefroid and P. Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. volume 575, pages 417–428. LNCS, 1991.

[15] E. Goubault. *The Geometry of Concurrency*. PhD thesis, 1995.

[16] E. Goubault. Cubical sets are generalized transition systems. Technical report, 2001.

[17] E. Goubault. Labelled cubical sets and asynchronous transistion systems: an adjunction. In *Presented at CMCIM'02*, volume 2, page 2002, 2001.

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

[19] M. Grandis. *Directed Algebraic Topology; models of non-reversible worlds*. CUP, 2009.

[20] M. Grandis and L. Mauri. Cubical sets and their site. *TAC*, 11(8):185–211, 2003.

[21] J. Hayman and G. Winskel. Symmetry in petri nets. *Perspectives in Concurrency Theory*, 2008.

[22] A. Joyal, M. Nielsen, and Winskel G. Bisimulation and open maps. In *LICS*, 1993.

[23] S. Mac Lane and I. Moerdijk. *Sheaves in geometry and logic: A first introduction to topos theory*. Springer, 1992.

[24] S. MacLane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer Verlag, 1971.

[25] S. Mimram. *Sémantique des jeux asynchrones et réécriture 2-dimensionnelle*. PhD thesis, PPS, CNRS – Université Paris Diderot, 2008.

[26] M. Mukund. Petri nets and step transition systems. *International Journal of Foundations of Computer Science*, 3(4):443–478, 1992.

[27] P. Mulry. Lifting theorems for Kleisli categories. In *MFPS*, pages 304–319, 1994.

[28] M. Nielsen, G. D. Plotkin, and G. Winskel. Petri nets, event structures and domains. In *Semantics of Concurrent Computation*, pages 266–284, 1979.

[29] C. Petri. Communication with automata, 1966.

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

[31] M. Raussen. On the classification of dipaths in geometric models for concurrency. *Mathematical Structures in Computer Science*, August 2000.

[32] Sankaranarayanan S., H. Sipma, and Z. Manna. Petri net analysis using invariant generation. volume 2772 of *LNCS*, pages 682–701, 2003.

[33] V. Sassone, M. Nielsen, and G. Winskel. Relationships between models of concurrency. In *Proceedings of the Rex'93 school and symposium*, 1994.

[34] J.P. Serre. *Homologie Singulière des Espaces Fibrés. Applications.* PhD thesis, École Normale Supérieure, 1951.

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

[36] S. Staton and G. Winskel. On the expressivity of symmetry in event structures. In *Logic in Computer Science*, pages 392–401. Citeseer, 2010.

[37] A. Valmari. A stubborn attack on state explosion. In *Proc. of CAV'90.* LNCS, 1990.

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

[39] R. van Glabbeek. Petri nets, configuration structures and higher dimensional automata. *Lecture notes in computer science*, pages 21–27, 1999.

[40] R. van Glabbeek. On the expressiveness of higher dimensional automata. *Theoretical computer science*, 356(3):265–290, 2006.

[41] G. Winskel. Event structures. In *Advances in Petri Nets*, pages 325–392, 1986.

[42] G. Winskel. Event structures with symmetry. *Electronic Notes in Theoretical Computer Science*, 172:611–652, 2007.

[43] G. Winskel and M. Nielsen. Models for concurrency. In *Handbook of Logic in Computer Science*, volume 3, pages 1–148. Oxford University Press, 1995.