

# Dihomotopy as a tool in state space analysis

Éric Goubault<sup>1</sup> and Martin Raussen<sup>2</sup>

<sup>1</sup> LIST (CEA Saclay), DTISI-SLA-LSL, 91191 Gif-sur-Yvette, France,  
Eric.Goubault@cea.fr

<sup>2</sup> Institute of Mathematical Sciences, Aalborg University, 9220 Aalborg Øst,  
Denmark, raussen@math.auc.dk

**Abstract.** Recent geometric methods have been used in concurrency theory for quickly finding deadlocks and unreachable states, see [14] for instance. The reason why these methods are fast is that they contain in germ ingredients for tackling the state-space explosion problem. In this paper we show how this can be made formal. We also give some hints about the underlying algorithmics. Finally, we compare with other well-known methods for coping with the state-space explosion problem.

## 1 Introduction

In model-checking techniques, temporal formulas, expressing important properties on traces of a concurrent system one has to verify, are checked by traversing the interleaving semantics of the program. This runs unfortunately into the “*state-space explosion problem*”: the number of paths to be traversed might be exponential in the number of processes involved. It has been very tempting for a number of authors to try to use the information about the *independence* of actions to decrease this number by a possibly exponential ratio. For instance, if all actions considered are completely independent, meaning that any interleaving of actions taken in this set of actions computes the same thing, as a function from (parallel or distributed) store to store, then there is no need to consider all the interleavings to check any kind of “interesting” properties, such as safety or deadlock properties.

But this is not always as simple as we show with the transition system of Figure 1. Here we suppose that  $a$  and  $b$  are independent or “commuting” actions. The problem in Figure 1 is that we might choose to traverse only path  $a.b$  since it is equivalent to  $b.a$  and we will have missed the branching after  $b$ , which would have lead us into transition system  $C$ , which might contain any deadlock we want for instance. In fact, there are correct ways to infer state-space reduction methods from the independence relation. A classical one explained in Section 7.1 has been originally introduced by Valmari [46] under the name of “stubborn sets”, based on a notion of independence on Petri nets. These have been ameliorated later under the name of “persistent sets” by Godefroid [21], based on the notion of independence of asynchronous transition systems. We develop in this paper new methods for finding better state-space reduction techniques, based on *global* semantical information. This is done using geometric ideas, which have

recently regained impetus after the seminal work [12] and [36]. We formalize this methodology, the “disconnected components” of the geometric semantics using a category of fractions of the fundamental category of the semantics, giving all information about all possible schedules of execution.

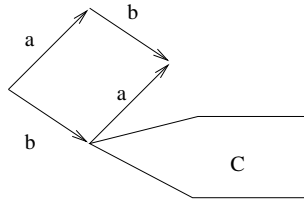


Fig. 1.

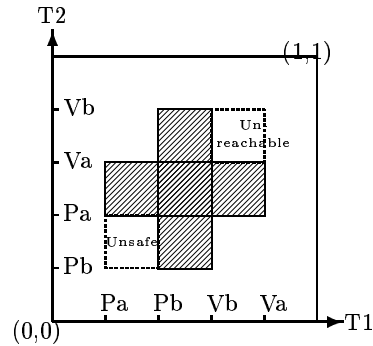


Fig. 2. Example of a progress graph

*History: Towards Higher Dimensional Automata.* The first “algebraic topological” model in the literature is called *progress graph* and has appeared in operating systems theory, in particular for describing the problem of “deadly embrace” in “multiprogramming systems”. *Progress graphs* are introduced in [10], but attributed to E. W. Dijkstra. The basic idea is to give a description of what can happen when several processes are modifying shared resources. Given a shared resource  $a$ , we see it as its associated *semaphore* that rules its behaviour with respect to processes. For instance, if  $a$  is an ordinary shared variable, it is customary to use its semaphore to ensure that only one process at a time can write on it (this is mutual exclusion). Then, given  $n$  deterministic sequential processes  $Q_1, \dots, Q_n$ , abstracted as a sequence of locks and unlocks on shared objects,  $Q_i = R^1 a_i^1 . R^2 a_i^2 \dots R^{n_i} a_i^{n_i}$  ( $R^k$  being  $P$  or  $V^1$ ), there is a natural way to understand the possible behaviours of their concurrent execution, by associating to each process a coordinate line in  $\mathbf{R}^n$ . The state of the system corresponds to a point in  $\mathbf{R}^n$ , whose  $i$ th coordinate describes the state (or “local time”) of the  $i$ th processor.

Consider a system with finitely many processes running altogether. We assume that each process starts at (local time) 0 and finishes at (local time) 1; the  $P$  and  $V$  actions correspond to sequences of real numbers between 0 and 1, which reflect the order of the  $P$ 's and  $V$ 's. The initial state is  $(0, \dots, 0)$  and the final state is  $(1, \dots, 1)$ . An example consisting of the two processes  $T_1 = Pa.Pb.Vb.Va$

<sup>1</sup> Using E. W. Dijkstra's notation  $P$  and  $V$  [12] for respectively acquiring and releasing a lock on a semaphore.

and  $T_2 = Pb.Pa.Va.Vb$  gives rise to the two dimensional *progress graph* of Fig. 2. The shaded area represents states which are not allowed in any execution path, since they correspond to mutual exclusion. Such states constitute the *forbidden region*. An *execution path* is a path from the initial state  $(0, \dots, 0)$  to the final state  $(1, \dots, 1)$  avoiding the forbidden region and increasing in each coordinate - time cannot run backwards. We call these paths *directed paths* or *dipaths*. This entails that paths reaching the states in the dashed square underneath the forbidden region, marked “unsafe” are deemed to deadlock, i.e. they cannot possibly reach the allowed terminal state  $(1, 1)$ . Similarly, by reversing the direction of time, the states in the square above the forbidden region, marked “unreachable”, cannot be reached from the initial state  $(0, 0)$ . Notice that all terminating paths above the forbidden region are “equivalent” in some sense: they are all characterized by the fact that  $T_2$  gets  $a$  and  $b$  before  $T_1$  (as far as resources are concerned, we call this a *schedule*). Similarly, all paths below the forbidden region are characterized by the fact that  $T_1$  gets  $a$  and  $b$  before  $T_2$  does.

In this picture, one can already recognize many ingredients that are at the center of algebraic topology, namely the classification of shapes modulo “elastic deformation”. As a matter of fact, the actual coordinates that are chosen to represent the times at which  $P$ s and  $V$ s occur are unimportant, and these can be “stretched” in any manner, so the properties (deadlocks, schedules etc.) are invariant under some notion of deformation, or *homotopy*. This has to be a particular kind of homotopy though causing many difficulties in later work. We call it (in subsequent sections) a *directed homotopy* or *dihomotopy* in the sense that it should preserve the direction of time.

The semantics community came back to these geometric considerations with the development of “truly-concurrent” semantics, as opposed to “interleaving” semantics. The base of the argument was that interleaving semantics, i.e. the representation of parallelism by non-determinism ignores real asynchronous behaviours:  $a \mid b$  where  $a$  and  $b$  are atomic is represented by the same transition system as the non-deterministic choice  $a$  then  $b$  or  $b$  then  $a$ . This fact creates problems in static analysis of (asynchronous) concurrent systems: Interleaving builds a lot of uninteresting states in the modelisation, hence induces a high cost in verification. This is called the *state-space explosion problem*. Quite a few models for true-concurrency have appeared (see in particular the account of [50]) but it is only in 1991 that geometry is proposed to solve the problem, in [36]. The diagnosis is that interleaving is only the boundary of the real picture.  $a \mid b$  is really the filled-in square whose boundary is the non-deterministic choice  $a$  then  $b$  or  $b$  then  $a$  (the hollow square). The natural combinatorial notion, extension of transition systems, is that of a *cubical set*, which is a collection of points (states), edges (transitions), squares, cubes and hypercubes (higher-dimensional transitions representing the truly-concurrent execution of some number of actions). This is introduced in [36] as well as possible formalizations using  $n$ -categories, and a notion of homotopy. This is actually a combinatorial view of some kind of progress graph. Look back to Figure 2. Consider all interleavings of actions  $Pa$ ,  $Pb$ ,  $Va$  and  $Vb$ : they form a subgrid of the progress graph. Take as 2-transitions

(i.e. squares in the cubical set we are building) the filled-in squares. Only the forbidden region is really interleaved. Cubical sets generalize progress graphs, in that they allow any amount of non-deterministic choices as well as dynamic creation of processes. These cubical sets are called *Higher-Dimensional Automata* (HDA) following [36] because it really makes sense to consider a hypercube as some form of transition. Actually at about the same time, a bisimulation semantics was given in [47]. Notice that 2-transitions or squares are nothing but a *local commutation relation* as in Mazurkiewicz trace theory [34], *independence relation* as in asynchronous transition systems, see [2], as in trace automata, as in transition systems with independence [40], or (indirectly) as with the “confluence” relation of concurrent transition systems [45]. There are two more ingredients with HDA: the elegance and the power of the tools of geometric formalisations, and the natural generalisation to *higher* dimensions (i.e. “higher-order independence relation” or  $n$ -ary independence relations).

*Example: Semaphores and progress graphs.* In the rest of the paper, we will stick to one particular model which is sufficiently simple to explain, and gives sufficiently many nasty example: the shared memory model, in which asynchronous processes read and write atomically onto variables which are all in a common (shared) memory. To protect writing onto shared variables, we use mutual exclusion locks, which we put explicitly before writing a variable  $x$ , by  $Px$ , and that we release explicitly after, by  $Vx$ . It is then easy to see that writing on two distinct variables are two independent actions, as well as reading two variables (even the same one) by two processes. This model can also easily include [16] counting semaphores which are weakly synchronising objects that can be shared by  $n$  but not  $n + 1$  processes at the same time (for some  $n > 1$ ). Notice that asynchronous message-passing with bounded buffers can be translated into that framework. It is therefore not only a useful example, but a quite general application indeed.

The key idea is to regard a progress graph as a topological space in which points are ordered globally through time, i.e., equipped with a (closed) partial order  $\leq$ . Traces of executions are continuous and increasing maps from the totally ordered unit segment to  $(X, \leq)$ . These are called *dipaths* for “directed paths”. A dihomotopy between two dipaths  $f$  and  $g$  on  $X$  is a deformation via dipaths interpolating continuously between  $f$  and  $g$  and fixing the endpoints. The technical definitions will be given in Sect. 3. Now we can give semantics to a very simple language in which a finite number of processes can only do a deterministic sequence of lockings  $Px$  and unlockings  $Vx$  of shared resources  $x$ . So processes are just strings of  $P$ ’s and  $V$ ’s. Suppose that each semaphore  $x$  (binary or counting) is equipped with a number  $s(x)$ , the maximal number of processes that can share it at any time. Supposing that the length of the strings  $X_i$  ( $1 \leq i \leq n$ ) are integers  $l_i$ , the semantics of *Prog* is included in  $[0, l_1] \times \cdots \times [0, l_n]$ . A description of the progress graph  $\llbracket Prog \rrbracket$  associated with *Prog* can be given by describing inductively what should be digged into this hyperrectangle. The semantics of our language can be described by the simple rule,  $[k_1, r_1] \times \cdots \times [k_n, r_n] \in \llbracket X_1 \mid \cdots \mid X_n \rrbracket_2$  if there is a partition of  $\{1, \dots, n\}$

into  $U \cup V$  with  $\text{card}(U) = s(a) + 1$  for some object  $a$  with,  $X_i(k_i) = Pa$ ,  $X_i(r_i) = Va$  for  $i \in U$  and  $k_j = 0$ ,  $r_j = l_j$  for  $j \in V$ . This language is somehow disappointing. To be able to consider looping and branching constructs, we are lead to the notion of local po-spaces in Sect. 3.1.

*Goals of the present paper.* After having explained the geometric semantics, the idea of deformation of paths of executions, and introduced the disconnected components approach to the state-space explosion problem, we compare (favorably) our technique with classical techniques such as persistent sets. We also review in Sect. 7.3 some orthogonal techniques which could still be used on top of our geometric technique.

## 2 The fundamental group of a topological space

In this section, we give a brief review of the fundamental group of a topological space, a very important concept from algebraic topology. See e.g. [1, 5, 27, 35] for details. Hereafter, we develop a variation of this notion and apply it to state space analysis.

Topological spaces are abstractions of metric spaces. For a metric space  $X$ , nearness is expressed by a metric  $d$  measuring the distance between pairs of points. For a topological space  $Y$ , nearness is expressed with the aid of a collection of *open* subsets of  $Y$ . The usual definition for a continuous map between two metric spaces has the following generalisation for topological spaces: A map  $f : Y \rightarrow Y'$  between topological spaces is *continuous* if and only if  $f^{-1}(U) \subset Y$  is open for every open subset  $U \subset Y'$ .

In this paper, we will mainly be concerned with (different types of) *paths*, i.e., continuous maps  $f : I \rightarrow X$  from an interval  $I$  into a topological space  $X$ . For the moment, we let  $I = [0, 1]$  denote the unit interval with standard metric and topology. In general, one cannot compose paths in  $X$ . But if the endpoint  $f_1(1)$  of  $f_1$  coincides with the start point  $f_2(0)$  of  $f_2$ , their *concatenation*

$$f_2 * f_1 : I \rightarrow X \text{ is defined by } (f_2 * f_1)(s) = \begin{cases} f_1(2s), & t \leq \frac{1}{2} \\ f_2(2s - 1), & t \geq \frac{1}{2}. \end{cases}$$

Both paths are thus pursued with “double speed”. Concatenation defines a (non-commutative, non-associative) operation on the space  $\mathcal{P}(X)$  of all paths on  $X$ .

Two points  $x, y \in X$  are called *path-connected*, if there exists a path  $f$  with  $f(0) = x$  and  $f(1) = y$ . The equivalence classes of this equivalence relation are called the *path components* of  $X$ . The image  $f(X_0) \subset Y$  of a path component  $X_0 \subset X$  under a continuous map  $f : X \rightarrow Y$  is path-connected. As a consequence, path components are completely independent of each other, and one can investigate them “one at a time”. A *loop* in a topological space  $X$  is a path  $f : I \rightarrow X$  such that  $f(0) = f(1)$ . Loops with the same start/end-point can be concatenated. A *homotopy* of paths (loops) is a continuous map  $H : I \times I \rightarrow X$  with  $H(t, 0) = H(0, 0)$  and  $H(t, 1) = H(0, 1)$  for all  $t \in I$ . It should be regarded as a one-parameter family of paths  $H_t : I \rightarrow X$ ,  $H_t(s) = H(t, s)$  (with fixed end

points) connecting  $H_0$  and  $H_1$ . Two paths  $f_0, f_1 : I \rightarrow X$  with the same end-points are called *homotopic* if there is a fixed end point homotopy  $H : I \times I \rightarrow X$  with  $H_0 = f_0$  and  $H_1 = f_1$ . Homotopy is an equivalence relation.

A continuous and strictly increasing map  $\varphi : I \rightarrow I$  with  $\varphi(0) = 0$  and  $\varphi(1) = 1$  can be used to reparametrise a path, i.e., to pass from a path  $f$  in  $X$  to the (reparameterised) path  $f \circ \varphi$  with the same image. Remark that  $\varphi$  is homotopic to the identity map on  $I$ ; a homotopy is given by  $H(t, s) = (1-t)\varphi(s) + ts$ . As a consequence, the paths  $f$  and its reparametrisation  $f \circ \varphi$  are homotopic via the homotopy  $\bar{H}(t, s) = f(H(t, s))$ .

A basic invariant of a topological space  $X$  is its *fundamental group*: Fix a base point  $x_0 \in X$ . The elements of the fundamental group  $\pi_1(X; x_0)$  are the homotopy classes of *loops*  $f : I \rightarrow X$  which start and end at  $f(0) = f(1) = x_0$ . Concatenation of loops at  $x_0$  factorizes over homotopy to yield a 2-adic operation on  $\pi_1(X; x_0)$ . The homotopy class of the constant map  $c : I \rightarrow X$ ,  $c(s) = x_0$ ,  $s \in I$ , serves as the neutral element – since  $f, f * c$  and  $c * f$  are homotopic to each other. The inverse to the class of the loop  $f$  is given by the the class of the loop  $f^- : I \rightarrow X$ ,  $f^-(t) = f(1-t)$ :  $f^- * f$  and  $f * f^-$  are both homotopic to  $c$ .

The size of the fundamental group has an interesting interpretation: A loop  $f$  can be regarded as a map from the unit circle  $\bar{f} : S^1 \rightarrow X$ ,  $\bar{f}(\exp(2\pi is)) = f(s)$ . The loop  $f$  represents the trivial element in  $\pi_1(X; x_0)$  if it is homotopic to the constant loop  $c$ . A homotopy  $H$  with  $H_0 = c$  and  $H_1 = f$  can be transformed into an extension  $\bar{H} : D^2 \rightarrow X$  of  $\bar{f}$ , viz.  $\bar{H}(t \exp(2\pi is)) = H(t, s)$ . Conversely, an extension of  $\bar{f}$  to a continuous map  $\bar{H} : D^2 \rightarrow X$  gives rise to a homotopy between  $f$  and  $c$ . A homotopically trivial loop can thus be “filled in”. Hence, the the fundamental group of a space “counts the numbers of holes” in it.

The fundamental group of a space does only depend on the *path component* of the base point: Let  $g$  denote an arbitrary path with  $g(0) = x_0$  and  $g(1) = x_1$ . Then the map “conjugation with  $g$ ”:  $\pi_1(X; x_0) \rightarrow \pi_1(X; x_1)$ ;  $[f] \mapsto [g^- * f * g]$  is a *group isomorphism*.

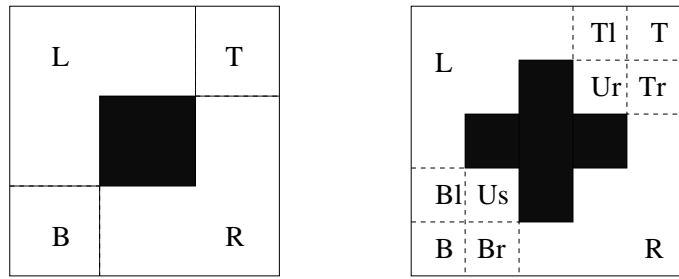
*Examples:* Proofs of the following statements can be found in almost any textbook on algebraic topology:

- The fundamental group of Euclidean space  $\mathbf{R}^n$  is trivial for all  $n$ .
- The fundamental group of the unit circle  $S^1$  is isomorphic to the integers. An explicit isomorphism  $\pi_1(S^1) \rightarrow \mathbf{Z}$  associates to a loop its “winding number”, i.e., it counts (with a sign) the number of times a particular value is attained. The fundamental group of an  $n$ -sphere  $S^n = \{x \in \mathbf{R}^n \mid \|x\| = 1\}$  is trivial for every  $n > 1$ .
- The fundamental group of “the figure 8” (two circles with only a single base point in common) is the free group on two letters representing the two directed loops.
- For every group  $G$ , there is a path-connected topological space  $BG$  with  $\pi_1(BG) \simeq G$ .

### 3 The fundamental category of an lpo-space

#### 3.1 Lpo-spaces and dipaths

There are many models for state spaces for concurrent processes and the executions on them, cf. Sect. 7. In this paper, we follow the basic idea from [16]: A po-space consists of a *topological space*  $X$  with a *partial order*  $\leq \subset X \times X$ . The partial order is assumed to be closed (as a subset of  $X \times X$ ) to ensure coherence between topology and order: this makes it possible to take limits “under the  $\leq$  sign”. For an example of such a po-space (in fact, a progress graph) see Fig. 3; the left figure represents the state space for two processes that acquire and relinquish a lock to a single shared resource; the right figure pictures the situation where locks to two shared resources have to be acquired in reverse order by the two processes. The black areas are the “forbidden regions” of the progress graph which are *not* part of the state space.



**Fig. 3.** Square with a hole and complement of a “Swiss flag”

If one or several of the processes contain loops, the resulting abstraction will no longer have a *global* partial order. Instead one requires for a local po-space (*lpo-space*) a relation  $\leq$  on  $X$  that restricts to a partial order on sufficiently small subsets of  $X$  that form a basis for the topology. Two such relations are equivalent (and define the same local partial order) if they agree on sufficiently small open sets forming a basis for the topology. For an example, consider the relation on the unit circle  $S^1 \subset \mathbf{R}^2$  given by  $x \leq y \Leftrightarrow$  the angle from  $x$  to  $y$  is less than  $\alpha$ . This relation is certainly not transitive, but it defines a local partial order if and only if  $\alpha \leq \pi$ ; for  $\alpha \leq \pi$  these are all equivalent.<sup>2</sup>

Traces of a concurrent system (executions) are modelled by so-called *dipaths*—di is an abbreviation for *directed*. A *short*, resp. *long* dipath in an lpo-space  $X$  is defined as an *order preserving continuous* map from the interval  $\vec{I} = [0, 1]$ , resp. from the non-negative reals  $\mathbf{R}_{\geq 0} = \{x \in \mathbf{R} \mid x \geq 0\}$  (with the natural order) into  $X$ . A short dipath models a concurrent process from a start point to

<sup>2</sup> This version of the definition is due to Ulrich Fahrenberg; it is in fact equivalent to the one given in [16, 17].

an end point, while a long dipath runs indefinitely (e.g., in loops) but avoiding *zeno* executions. Technically, one requires that a long dipath does not admit a limit for  $t \rightarrow \infty$ .

### 3.2 Dihomotopy

When can you be sure that two execution traces in a concurrent program provide the same result? This is the case if the corresponding dipaths  $f, g : I \rightarrow X$  are *dihomotopic*. This means, that there exists a continuous order-preserving *dihomotopy*  $H : I \times \vec{I} \rightarrow X$  with  $H_0 = f$  and  $H_1 = g$ . Remark that the parameter interval is equipped with the trivial order, i.e.,  $(t, s) \leq (t', s') \Leftrightarrow t = t' \wedge s \leq s'$ . In particular, every “intermediate” path  $H_t$  has to be a dipath. Moreover, we require a fixed start point ( $H(t, 0) = H(0, 0)$ ) and, for short dipaths, a fixed end point ( $H(t, 1) = H(0, 1)$ ); for long dipaths all the paths  $H_t$  have to be non-zeno.

### 3.3 The fundamental category

For an lpo-space, one should no longer watch out for a fundamental *group*. The reverse of a dipath is no longer a dipath. On a global po-space, there are no (non-trivial) directed loops at all. Instead, one has to work with the fundamental *category* of a local po-space  $X$ , or rather with two versions of it, depending on whether short or long dipaths are considered:

The *objects* of the fundamental category  $\bar{\pi}_1(X)$  are the points of  $X$ . The *morphisms* between elements  $x$  and  $y$  are given as the dihomotopy classes in  $\bar{\pi}_1(X; x, y)$ . Composition of morphisms

$$\bar{\pi}_1(X; x, y) \times \bar{\pi}_1(X; y, z) \rightarrow \bar{\pi}_1(X; x, z)$$

is given by concatenation of dipaths – up to dihomotopy.

The category  $\bar{\pi}_1^\infty(X)$  contains  $\bar{\pi}_1(X)$ . It has an additional maximal element  $\infty$  with  $Mor(x, \infty)$  consisting of the dihomotopy classes of long dipaths starting at  $x$  and  $Mor(\infty, y) = \emptyset$  for  $y \in X \cup \{\infty\}$ . Concatenation of a (short) dipath from  $x$  to  $y$  with a (long) dipath from  $y$  yields a (long) dipath – up to dihomotopy via long dipaths.

Compared to the fundamental group, a fundamental category is an enormous gadget and it has a much less nice algebraic structure. On the other hand, in easy examples one has the impression, that the cardinality of the set of morphisms between two points is quite robust when these points are perturbed a little.

*Example 1.* For the square with one hole (Fig. 3), there is no morphism between the regions marked  $L$ , resp.  $R$ , and no morphism from  $T$  to any other region, neither a morphism from any other region to  $B$ . There are two morphisms from any point of  $B$  to any point of  $T$ . Moreover, from any point of  $B$ , certain points of  $B, L, R$  can be reached by (exactly one) morphism. Likewise, any point of  $T$  can be reached from (certain of the points in)  $L, R$  and  $T$  in one way.

For the complement of a “Swiss flag” (Fig. 3), the situation is a bit more complicated: There is no dipath leaving the unsafe rectangle  $Us$  and there is no dipath entering the unreachable rectangle  $Ur$  from the outside. It is possible to

reach  $Us$  by a dipath from  $B \cup Bl \cup Br$ ; from  $Ur$ , one can reach  $Tl \cup Tr \cup T$ . The only possibility for *two* classes of dipaths between points occurs when the first is in  $B$  and the second in  $T$ . Moreover, these classes can be represented by dipaths along the boundary, representing the two sequential executions.

The lesson to learn is that the complete “dynamics” of these state space can be described from the decomposition into the blocks studied above. It is the aim of this paper to define and describe these “diconponents” in the general case and thus, in a realistically large model, to provide a “collapse” of the exponentially large state space into pieces that show the same behaviour with respect to execution paths between each other. It is then enough to study the “flow” between these “components” in order to capture the dynamics of the whole system.

## 4 Categories of fractions and components

### 4.1 Categories of fractions

Next, we have to invest in a construction from category theory: We invert in a systematic way all those partial dipaths that never contribute to a decision along any dipath. The resulting category will then have many “zig-zag” isomorphisms giving rise to the components. Here is a general method [18, 4]:

Let  $\mathcal{C}$  denote a category. To keep things simple, assume  $\mathcal{C}$  small, i.e., objects and morphisms are sets. Let  $\Sigma \subset Mor(\mathcal{C})$  denote a *system of morphisms*, i.e.,  $\Sigma$  includes all unit morphisms and is closed under composition. For any such system, one may construct the category of fractions  $\mathcal{C}[\Sigma^{-1}]$  and the localization functor  $q_\Sigma : \mathcal{C} \rightarrow \mathcal{C}[\Sigma^{-1}]$  [18, 4] having the following universal property:

- For every  $s \in \Sigma$ , the morphism  $q_\Sigma(s)$  is an *isomorphism*. - For any functor  $F : \mathcal{C} \rightarrow \mathcal{D}$  such that  $F(s)$  is an isomorphism for every  $s \in \Sigma$ , there is a unique functor  $\theta : \mathcal{C}[\Sigma^{-1}] \rightarrow \mathcal{D}$  with  $\theta \circ q_\Sigma = F$ .

The objects of  $\mathcal{C}[\Sigma^{-1}]$  are just the objects of  $\mathcal{C}$ . To define the morphisms of  $\mathcal{C}[\Sigma^{-1}]$ , one introduces a (formal) inverse  $s^{-1}$  to every morphism  $s \in \Sigma(x, y)$ . These inverses are collected in  $\Sigma^{-1}(y, x)$ ,  $x, y \in Ob(\mathcal{C})$  and then in  $\Sigma^{-1}$ . Consider the closure of  $Mor(\mathcal{C}) \cup \Sigma^{-1}$  under composition and the smallest equivalence relation containing  $s^{-1} \circ s = 1_x$  and  $s \circ s^{-1} = 1_y$  for  $s \in \Sigma(x, y)$  that is compatible with composition. The equivalence classes correspond then to the morphisms of  $\mathcal{C}[\Sigma^{-1}]$ . In particular, if  $t \circ \alpha = \beta \circ s$  for  $s, t \in \Sigma$ , then  $\alpha \circ s^{-1} = t^{-1} \circ \beta$ . A morphism in  $\mathcal{C}[\Sigma^{-1}]$  can always be represented in the form

$$s_k^{-1} \circ f_k \circ \dots \circ s_1^{-1} \circ f_1, \quad s_j \in \Sigma, f_j \in Mor, k \in \mathbf{N}.$$

Let  $Iso(\mathcal{C})$  denote the isomorphisms of the category  $\mathcal{C}$ , and let  $\Sigma * Iso(\mathcal{C})$  denote the system of morphisms generated by  $\Sigma$  and by  $Iso(\mathcal{C})$ . The isomorphisms in  $\mathcal{C}[\Sigma^{-1}]$  are the *zig-zag* morphisms, i.e.,

$$Iso(\mathcal{C}[\Sigma^{-1}]) = \{s_1^{-1} \circ s_2 \circ \dots \circ s_{2k-1}^{-1} \circ s_{2k}, s_j \in \Sigma * Iso(\mathcal{C}), k \in \mathbf{N}\}.$$

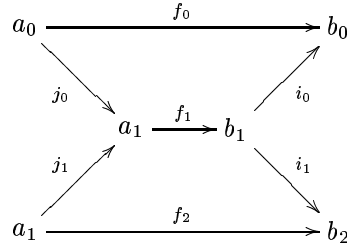
The subcategory of  $\mathcal{C}[\Sigma^{-1}]$  with all objects, the morphisms of which are given by the zig-zag morphisms in  $Iso(\mathcal{C}[\Sigma^{-1}])$ , forms in fact a *groupoid* [30].

## 4.2 Components

A “compression” of the category  $\mathcal{C}[\Sigma^{-1}]$  is achieved by dividing out all isomorphisms: Two objects  $x, y \in \text{Ob}(\mathcal{C})$  are  $\Sigma$ -isomorphic or  $\Sigma$ -connected –  $x \simeq_{\Sigma} y$  – if there exists a zig-zag-morphism from  $x$  to  $y$ . This definition corresponds to usual path connectedness *with respect to paths representing isomorphisms only – but regardless orientation*.  $\Sigma$ -connectivity is an equivalence relation; the equivalence classes are called the  $\Sigma$ -connected components – the path components with respect to  $\Sigma$ -zig-zag paths, viz. the components of the groupoid above.

Next, consider the equivalence relation on the morphisms of  $\mathcal{C}[\Sigma^{-1}]$  generated (under composition) by  $\alpha \simeq \alpha \circ s$ ,  $\alpha \simeq t \circ \alpha$  with  $\alpha \in \text{Mor}(x, y)$ ,  $s \in \text{Inv}(\mathcal{C}[\Sigma^{-1}])(x', x)$ ,  $t \in \text{Inv}(\mathcal{C}[\Sigma^{-1}])(y, y')$ . Remark that equivalent morphisms no longer need to have the same source or target. Remark moreover, that any two zig-zag morphisms from  $x$  to  $y$  are equivalent to each other; in particular, they are equivalent to the unit-morphisms in both  $x$  and  $y$ .

*Example 2.* If  $i_0, i_1, j_0, j_1 \in \text{Inv}(\mathcal{C}[\Sigma^{-1}])$ , then  $f_0, f_1, f_2 \in \text{Mor}(\mathcal{C})$  in the following diagram are equivalent to each other in  $\mathcal{C}[\Sigma^{-1}]$ :



The objects of the *component category*  $\pi_0(\mathcal{C}; \Sigma)$  are by definition the  $\Sigma$ -connected components of  $\mathcal{C}$ ; the morphisms from  $[x]$  to  $[y]$ ,  $x, y \in \text{Ob}(\mathcal{C})$ , are the equivalence classes of morphisms in  $\bigcup_{x' \simeq_{\Sigma} x, y' \simeq_{\Sigma} y} \text{Mor}_{\mathcal{C}[\Sigma^{-1}]}(x', y')$ . The composition of  $[\beta] \circ [\alpha]$  for  $\alpha \in \text{Mor}_{\mathcal{C}[\Sigma^{-1}]}(x, y)$  and  $\beta \in \text{Mor}_{\mathcal{C}[\Sigma^{-1}]}(y', z)$  is given by  $[\beta \circ s \circ \alpha]$  with  $s$  any zig-zag morphism from  $y$  to  $y'$ . The equivalence class of that composition is independent of the choices taken.

*Remark 1.* These constructions decompose the study of the morphisms of  $\mathcal{C}$  into two pieces: Firstly, the study of the groupoid  $\text{Inv}(\mathcal{C}[\Sigma^{-1}])$  which can be performed separately on each of the  $\Sigma$ -connected components. For the fundamental category, all these morphisms represent executions that can be performed and/or backtracked without global effects. Secondly, certainly more important for applications, the study of the component category, which encompasses the global effects of irreversibility. In the case of the component category of the fundamental category, representatives of all non-unit dipath classes may have (different) global effects – backtracking along such a dipath class may therefore change the result of a computation.

## 5 Applications to state space analysis

In this section, we apply the preceding constructions to our models of the state space and the space of executions (from a given initial state) of a concurrent program. The key task is to single out the relevant system of morphism  $\Sigma$  that is to be inverted. It should consist of morphisms that, *from a global point of view*, do not contribute with any *decision* to the outcome of the concurrent program. Here, we give the key definitions (in a general categorial framework), their motivation, and a few elementary examples. For algorithms in low dimensions, cf. Sect. 6.

### 5.1 Extensions of morphisms

For a small category, let  $X_0, X_1 \subset Ob(\mathcal{C})$ . Let  $Mor_{0,1} = \bigcup_{x_0 \in X_0, x_1 \in X_1} Mor(x_0, x_1)$  denote the set of all morphisms between  $X_0$  and  $X_1$ . We associate to a morphism  $f \in Mor(x, y)$  with  $x, y \in Ob(\mathcal{C})$ , the set of all its *extensions*

$$\mathcal{E}(f) = \{g \circ f \circ h \mid h \in Mor(X_0, x), g \in Mor(x, X_1)\} \subset Mor_{0,1}$$

from  $X_0$  to  $X_1$ . This set consists of all morphisms from  $X_0$  to  $X_1$  that *factor through*  $f$ . It is empty if  $Mor(X_0, x) = \emptyset$  or if  $Mor(x, X_1) = \emptyset$ . In the particular case  $f = 1_x$ , the unit at  $x \in Ob(\mathcal{C})$ , the set  $\mathcal{E}(x) = \mathcal{E}(1_x)$  consists of all morphisms from  $X_0$  to  $X_1$  factoring through  $x$ .

For concatenable morphisms  $f_1, f_2$  it is obvious that  $\mathcal{E}(f_2 \circ f_1) \subseteq \mathcal{E}(f_1) \cap \mathcal{E}(f_2)$ . The geometric example Ex. 2.1 in [38] shows that the left hand side may be a *proper* subset of the right hand side.

### 5.2 Components on the space of executions

The space of partial executions of a concurrent program is modelled as the set of morphisms from the initial point  $x_0$  in the fundamental category  $\vec{\pi}_1(X)$ . More generally, one may associate to any category  $\mathcal{C}$  and any object  $x_0 \in Ob(\mathcal{C})$  the comma category  $(x_0 \downarrow \mathcal{C})$  of *objects under*  $x_0$  [33]: Its objects are the morphisms in  $Mor(x_0, x)$ ,  $x \in Ob(\mathcal{C})$ , and its morphisms are the *commutative* triangles

$$\begin{array}{ccc} & x_0 & \\ f \swarrow & & \searrow g \\ x_1 & \xrightarrow{h} & x_2 \end{array}$$

with  $x_0$  in the top and  $h \in Mor(x_1, x_2)$ .

If  $\mathcal{C}$  is the fundamental category  $\vec{\pi}_1(X)$  and  $x_0$  an initial element, the comma categories  $(x_0 \downarrow \vec{\pi}_1(X))$  and  $(x_0 \downarrow \vec{\pi}_1^\infty(X))$  have as objects the dihomotopy classes of dipaths starting at  $x_0$ : a partial dipath  $h \in \vec{\pi}_1(x_1, x_2)$  with  $x_1 \in X$  and  $x_2 \in X \cup \{\infty\}$  induces a map  $\vec{\pi}_1(x_0, x_1) \rightarrow \vec{\pi}_1(x_0, x_2)$  by concatenation.

Assume given a (minimal) object  $x_0$  such that  $X_0 = \{x_0\}$  and a set  $X_1$  of maximal objects in a category  $\mathcal{C}$ . For the fundamental category  $\vec{\pi}_1(X)$ , this set  $X_1$  should be chosen as a discrete set of final accepting states<sup>3</sup>, for the fundamental category  $\vec{\pi}_1^\infty(X)$ , the maximal object should be chosen as  $\infty$ .

<sup>3</sup> which could include deadlocking points

**Definition 1.** A morphism  $s$  from  $f \in \text{Mor}(x_0, x)$  to  $g \in \text{Mor}(x_0, y)$  belongs to  $\Sigma_1$  if and only if  $\mathcal{E}(f) = \mathcal{E}(g) \subseteq \text{Mor}_{01}$ .

It is clear that either every or no morphism from  $f$  to  $g$  is contained in  $\Sigma_1$ . Obviously,  $\Sigma_1$  contains the units and is closed under composition. For  $\mathcal{C} = \vec{\pi}_1(X)$ , a dipath  $s$  extending  $f$  to  $g$  is contained in  $\Sigma_1$  if no “decision” has been made in between – all “careers” in  $\vec{\pi}_1(X; x_0, X_1)$  open to  $f$  are still open to  $g$ . No (globally detectable) branching occurs between  $f$  and  $g$ .

A detection of the component category wrt.  $\Sigma_1$  entails the following benefit:

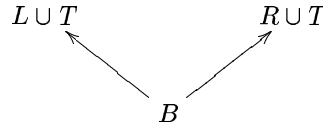
**Proposition 1.** Two dipaths  $f$  and  $g$  from  $x_0$  to  $x_1$  that proceed through the same  $\Sigma_1$ -components are dihomotopic.

We illustrate the resulting component categories by two elementary examples:

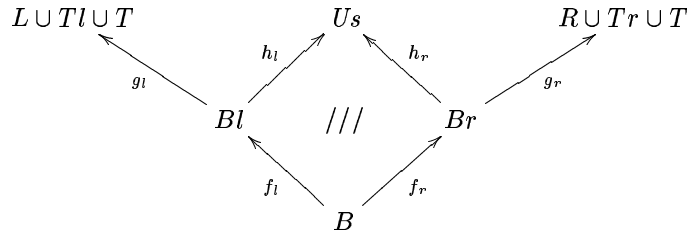
*Example 3.* Let  $x_0$ , resp.  $x_1$  denote the minimal, resp. the maximal element in the po-space  $X$ , the square with one hole from Fig. 3. Then  $\text{Mor}_{01}$  has two elements represented by dipaths  $f_L, f_R$  touching  $R$ , resp.  $L$ . Any dipath within  $B$  and any dipath within  $L \cup R \cup T$  is in  $\Sigma_1$ . No dipath starting within  $B$  and ending in  $L$  or  $R$  is in  $\Sigma_1$ .

The category  $(x_0 \downarrow \vec{\pi}_1(X))$  consists of three  $\Sigma_1$ -connected components: the dipaths ending in  $B$ ; those touching  $L$  and those touching  $R$ . Observe: There is no zig-zag path  $t^{-1} \circ s$  from a dipath to  $R$  via  $T$  to a dipath to  $L$  since there are no dipaths  $f$  from  $x_0$  to  $R$  and  $g$  from  $x_0$  to  $L$  with  $t * g$  dihomotopic to  $s * f$ .

The component category  $\pi_0(x_0 \downarrow \vec{\pi}_1(X), \Sigma_1)$  can – apart from the units – be represented by (end points in)



*Example 4.* Let  $Y$  denote the “Swiss flag” po-space from Fig. 3. Let  $x_0$  and  $x_1$  denote the minimal, resp. the maximal elements, and let  $y$  denote the deadlock point (maximal within the unsafe region  $Us$ ). The set of accepting states is  $X_1 = \{x_1, y\}$ , and  $\text{Mor}_{01}$  consists of three elements – there is also a dihomotopy class with end point in  $y$ . The component category  $\pi_0(x_0 \downarrow \vec{\pi}_1(Y), \Sigma_1)$  is represented by the diagram (with obvious morphisms between the given regions/components)



*Remark 2.* In [43], S. Sokolowski introduced a somehow similar approach resulting in the *fundamental poset*  $\Omega_1(X)$  of a po-space  $X$ . Using our terminology, a preorder on  $(x_0 \downarrow \vec{\pi}_1(X))$  is defined by:

$$f \in \vec{\pi}_1(X; x_0, x) \sqsubseteq g \in \vec{\pi}_1(X; x_0, y) \Leftrightarrow \forall h \in \vec{\pi}_1(X; y, z)$$

$$\exists a, j_1 \in \vec{\pi}_1(X; x, a), j_2 \in \vec{\pi}_1(X; z, a) \text{ with } j_1 * f = j_2 * h * g \in \vec{\pi}_1(X; x_0, a).$$

The equivalence classes given by “ $\sqsubseteq$  and  $\supseteq$ ” are the elements of the poset  $\Omega_1(X)$ , equipped with the partial order induced by  $\sqsubseteq$ .

If one considers morphisms  $Mor_{01}$  corresponding to a set  $X_1$  of maximal elements, it is easy to see that  $f \sqsubseteq g \Leftrightarrow \mathcal{E}(f) \supseteq \mathcal{E}(g)$ , and hence the  $\Sigma_1$ -connected components agree with the elements of  $\Omega_1(X)$ . The partial order between equivalence classes in  $\Omega_1(X)$  corresponds to the *existence* of morphisms in  $\mathcal{C}[\Sigma^{-1}]$  between elements of these classes. The component category contains more information. It allows to compare factorisations of two given morphisms and to discuss in which parts of the po-space they agree and in which they differ.

P. Gaucher [19] has a quite different categorical approach to branching and merging, not only for dipaths, but also for their higher-dimensional analoga.

### 5.3 Components of the state space

Next, we shift attention to the entire state space of a concurrent program, modelled by an lpo-space  $X$  with a minimal element  $x_0$ , the only element of  $X_0$ , and a (discrete) subset  $X_1$  of maximal elements. For an element  $x \in X$ , we ask: Which essentially different execution paths pass through  $x$ ? How does this information develop throughout the state space?

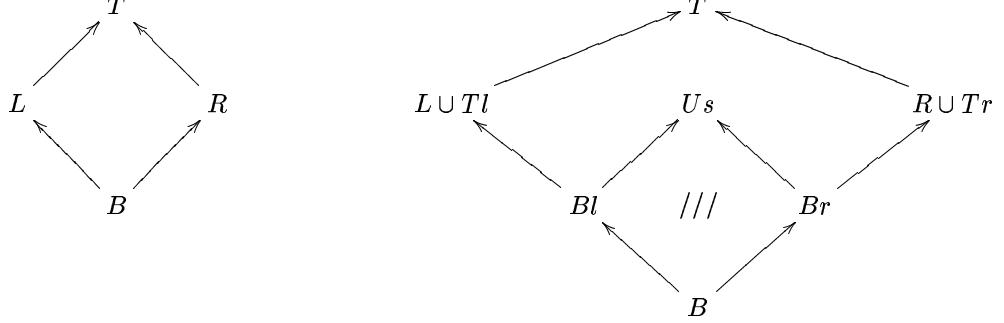
**Definition 2.** *The system  $\Sigma_2 \subset Mor(\mathcal{C})$  consists of all morphisms  $s \in Mor(x, y)$  with  $x, y \in Ob(\mathcal{C})$  satisfying*

$$\mathcal{E}(f) = \mathcal{E}(s \circ f), \mathcal{E}(g) = \mathcal{E}(g \circ s) \text{ for all } f \in Mor(-, x) \text{ and } g \in Mor(y, -). \quad (1)$$

Obviously,  $\Sigma_2$  contains the units, and it is closed under composition. It is easy to see that it suffices to require (1) for all  $f \in Mor(x_0, x)$  and  $g \in Mor(x, X_1)$ . Along a morphism  $s \in \Sigma_2(x, y)$ , no decisions with global effects are taken: concatenation with  $s$  does not alter any of the extension sets of morphisms with source in  $y$  or target in  $x$ . States in the *same*  $\Sigma_2$ -component (or *dicomponent*) cannot be distinguished by the results of executions passing through them.

*Example 5.* Let again  $X$  denote the square with one hole and  $Y$  the “Swiss flag” from Fig. 3 with minimal and maximal elements  $x_0$  (and  $y$ ), resp.  $x_1$ . The

component categories  $\pi_0(\vec{\pi}_1(X); \Sigma_2)$  and  $\pi_0(\vec{\pi}_1(Y); \Sigma_2)$  are then of the form



#### 5.4 Relation to history equivalence

The  $\Sigma_2$ -components refine the notion of a *dicomponent* of an lpo-space defined earlier in [16, 37]. Those were only defined as sets and lacked the dynamical perspective given by the component category:

**Definition 3.** *The history  $hf$  of a morphism  $f \in \text{Mor}(X_0, X_1)$  is defined as*

$$hf = \{x \in \text{Ob}(\mathcal{C}) \mid \exists f_0 \in \text{Mor}(X_0, x), f_1 \in \text{Mor}(x, X_1) \text{ with } f = f_1 \circ f_0\}.$$

*Two objects  $x, y \in \text{Ob}(\mathcal{C})$  are history equivalent if and only if  $x \in hf \Leftrightarrow y \in hf$  for all  $f \in \text{Mor}(X_0, X_1)$ .*

A history equivalence class  $C \subset \text{Ob}(\mathcal{C})$  is thus a primitive element of the Boolean algebra generated by the histories, i.e., an intersection of histories and their complements such that for all  $f \in \text{Mor}(X_0, X_1)$  either  $C \subseteq hf$  or  $C \cap hf = \emptyset$ .

The following argument shows that a morphism  $s \in \Sigma_2(x, y)$  has history equivalent source and target:

$$x \in hf \Leftrightarrow f \in \mathcal{E}(x) = \mathcal{E}(i_x) = \mathcal{E}(s) = \mathcal{E}(i_y) = \mathcal{E}(y) \Leftrightarrow y \in hf.$$

Hence, a  $\Sigma_2$ -component is contained in a path-component of a history equivalence class.

## 6 Algorithms for 2-dimensional mutual exclusion models

In this section, we confine ourselves to the progress graphs described in the introduction. Classifying dipaths up to dihomotopy in these mutual exclusion models corresponds to finding out which (and how many) schedules for a given concurrent program can potentially yield different results. An algorithm arriving at such a classification in dimension two, i.e, for semaphore programs with just *two* interacting transactions, was described in [37]; the results in this section rely on the methods described there.

In this case, the state space has  $X = I^2 \setminus \text{int}(F)$  as a model, i.e., a unit square from which a forbidden region  $F$  (e.g., the region in black in Fig. 3) is

deleted. This region is a union of rectangles that are parallel to the axes. Since we are interested in dipaths connecting the minimal point to the maximal point, we may assume that  $X$  does not contain neither unsafe nor unreachable points; this can always be achieved by a completion process, cf. [37]. As a consequence ([37], Lemma 4.1), every path-component  $F_i \subset F$  has a global minimum  $\mathbf{z}_i = (x_i, y_i)$  and a global maximum  $\mathbf{z}^i = (x^i, y^i)$ . We define line segment subspaces  $S_x^x, S_x^y, S_x^i, S_y^x, S_y^y, S_y^i \subseteq X$  emerging horizontally and vertically from these minima and maxima as follows:

$$\begin{aligned} S_x^x &= \{\mathbf{z} = (x, y) \in X \mid x \leq x_i, y = y_i\} & S_x^y &= \{\mathbf{z} = (x, y) \in X \mid x = x_i, y \leq y_i\} \\ S_x^i &= \{\mathbf{z} = (x, y) \in X \mid x \geq x^i, y = y^i\} & S_y^i &= \{\mathbf{z} = (x, y) \in X \mid x = x^i, y \geq y^i\}. \end{aligned}$$

All these subspaces consist of one or several line segments, that may be broken up into pieces by other components of the forbidden region. Let  $T_x^x \subseteq S_x^x, T_x^y \subseteq S_x^y, T_x^i \subseteq S_x^i$  and  $T_y^x \subseteq S_y^x, T_y^y \subseteq S_y^y, T_y^i \subseteq S_y^i$  denote the segment touching  $F_i$ , cf. Fig. 4. The unions of these separating line segments will be called  $T_- = \bigcup_i (T_x^x \cup T_x^y), T^- = \bigcup_i (T_x^i \cup T_y^i)$  and  $T = T^- \cup T_-$ . A dipath  $f : I \rightarrow X$  from  $x$  to  $y$  is said to *cross*  $T_-$  if there exists an  $i$  such that  $\emptyset \neq f^{-1}(T_-)$  is contained in the interior of  $I$ , i.e., if its image contains points on both sides of one of the segments. Similarly, one defines crossing wrt.  $T^-$  and to  $T$ . We can now detect which of the dipath classes in  $X$  are inverted in the two categories of fractions of Sect. 5:

**Proposition 2.** *Let  $s : I \rightarrow X$  denote a (partial) dipath with  $f(0) = x$  and  $f(1) = y$ . Its dihomotopy class  $[s] \in \tilde{\pi}_1(X; x, y)$  is contained in  $\Sigma_1$  if and only if  $f$  does not cross  $T_-$ ; in  $\Sigma_2$  if and only if  $f$  does not cross  $T$ .*

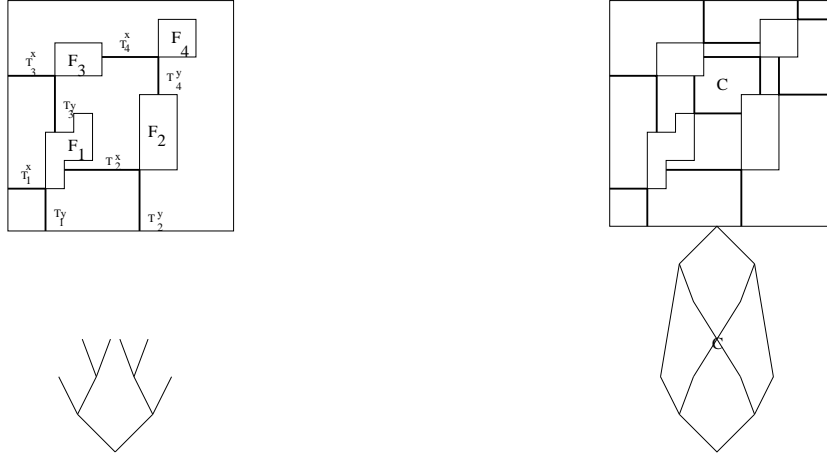
*Example 6.* In the example of Fig. 4 with a forbidden region consisting of four components  $F_i$ , there are six dihomotopy classes of dipaths between  $\mathbf{x}_0$  and  $\mathbf{x}_1$ , cf. [37], Fig. 14. The upper figures contain the line segments  $T_*^*$  that, together with boundary segments of the forbidden region, cut out the components in the two cases discussed in Sect. 5. For instance, the component marked  $C$  in the rightmost figure is characterised by two ingoing non-unit morphisms that, each have two extensions. The lower figures show the associated component categories with morphisms going upward. In this example, there are no non-trivial relations.

A similar analysis in dimensions higher than two is certainly more demanding. Not only the components of  $F$ , but also their finer topological properties will certainly play a role, cf. the discussion in [37], Sect. 5.

## 7 Classical state-space reduction techniques

### 7.1 Persistent sets

Let  $(S, i, E, Tran, I)$  be an asynchronous transition system [2]. This means that  $(S, i, E, Tran)$  is a transition system and that  $I \subseteq E \times E$  is a relation between labels  $E$ , the ‘‘independence relation’’ between two actions. We will not give a precise axiomatics for  $I$  here, and will keep on simple grounds. Basically,  $I$  should satisfy the following conditions (taken from [22]):



**Fig. 4.** Components and their categories in a 2-dimensional mutual exclusion model

- if  $t_1$  (respectively  $t_2$ ) is enabled in  $s$  and  $s \rightarrow^{t_1} s'$  (respectively  $s \rightarrow^{t_2} s'$ ) then  $t_2$  (respectively  $t_1$ ) is enabled in  $s$  if and only if  $t_2$  ( $t_1$ ) is enabled in  $s'$  (independent transitions can neither disable nor enable each other); and,

- if  $t_1$  and  $t_2$  are enabled in  $s$ , then there is a unique state  $s'$  such that both  $s \rightarrow^{t_1} s_1 \rightarrow^{t_2} s'$  and  $s \rightarrow^{t_2} s_2 \rightarrow^{t_1} s'$  (commutativity of enabled independent transitions).

In our technique,  $I$  is just a set of squares, or 2-transitions, or in the topological sense, they are elementary surfaces, enabling us to continuously deform dipaths. We extend in an intuitive manner  $I$  to sets of actions by putting  $AIB$  if for all  $a \in A$ , for all  $b \in B$ ,  $aIb$ . We identify  $a$  with the singleton  $\{a\}$ .

Let  $T$  be a set of actions,  $T \subseteq E$ , and  $p \in S$  be a state. We say that  $T$  is persistent in state  $p$  if,  $T$  contains only actions which are enabled at  $p$ , and, for all traces  $t$  beginning at  $p$  containing only actions  $q$  out of  $T$ ,  $qIT$ . Suppose we have a set of persistent actions  $T_p$  for all states  $p$  in an asynchronous transition system. Then let us look at the following set of traces  $PT$  (identified with a series of states) in  $(S, i, E, Tran, I)$  defined inductively as follows:  $(i) \in PT$ , and if  $(p_1, \dots, p_n) \in PT$ , then  $(p_1, \dots, p_n, q) \in PT$  where  $p_n \rightarrow^{t'} q \in Tran$  and  $t' \notin T_{p_n}$ . Deadlock detection can be performed on this subset  $PT$  of traces instead of the full set of traces of  $(S, i, E, Tran, I)$ . At least when  $(S, i, E, Tran, I)$  is acyclic,  $PT$  is enough for checking LTL temporal formulas (and you can modify the method so that it works generally). We exemplify the method on the process  $Pb.Pa.Vb.Va \mid Pa.Pb.Va.Vb$ . A standard interleaving semantics would be as sketched in Figure 5, showing the presence of deadlocking state 13. One set of persistent sets is  $T_1 = \{Pa\}$ ,  $T_2 = \{Pb\}$ ,  $T_5 = \{Pa, Pb\}$ ,  $T_6 = \{Pb, Va\}$ ,  $T_8 = \{Pa, Va\}$ ,  $T_{13} = \emptyset$ ,  $T_9 = \{Vb\}$ ,  $T_{12} = \{Va\}$ ,  $T_{17} = \{Pb\}$ ,  $T_{18} = \{Va\}$ ,  $T_{22} = \{Vb\}$ ,  $T_{23} = \emptyset$ ,  $T_7 = \{Pb, Vb\}$ ,  $T_{14} = \{Vb\}$ ,  $T_{15} = \{Pb\}$ ,  $T_{16} = \{Pa\}$ ,  $T_{20} = \{Vb\}$ ,  $T_{21} = \{Va\}$ , and we show the corresponding traces  $PT$  in Figure

6. We have not indicated the persistent sets corresponding to 3, 4 etc. since in a persistent set search, they will not be reached anyway, so their actual choice is uninteresting. In Figure 5 there are 16 paths from 1 to be traversed if no selective search was used. Six of them lead to the deadlock 13, and 10 (5 above the hole, 5 below the hole) are going to the final point 23. In Figure 6, one can check that there are only 8 paths to be traversed if one uses the persistent sets selective search (3 to state 13, 1 to state 23 below the hole and 4 to state 23 above the hole).

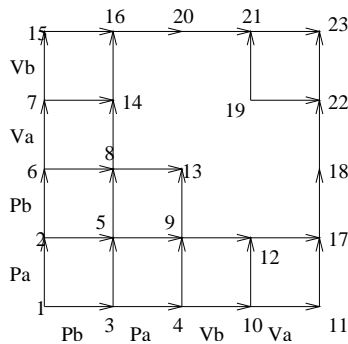


Fig. 5.

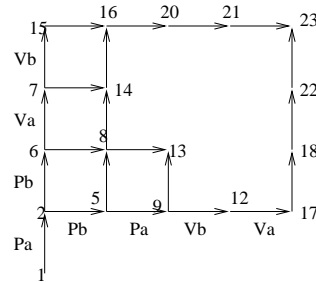


Fig. 6.

How did we find this set of persistent sets? In the *PV* case this can be done quite easily as follows. First the independence relation can be found out right away.  $Px$  and  $P_y$  stand respectively for the query for a lock on  $x$  and  $y$  (nothing is committed yet) so they are independent actions, whatever  $x$  and  $y$  are. We should rather declare  $Px$  and  $Vy$  dependent in general: if  $x = y$  this is clear, and for  $x \neq y$  this can come from the fact locks on  $x$  and  $y$  are causally related (precisely as in the case of Figure 5 with  $x = a$  and  $x = b$ ). This is slightly different from the more usual case of atomic reads and writes languages in which the independence relation can be safely determined as: actions are independent if and only if they act on distinct variables. The most elaborated technique known in this framework is that of “stubborn sets” see [46], and its adaptation to the current presentation, see [22] for a precise definition. The example of persistent set we gave in Figure 6 is in fact a stubborn set. As one can see as well, the persistent set approach here reduces the 5 paths below the hole into 1, which is a representant modulo dihomotopy of these 5 dipaths. In the disconnected components approach, one finds the set of 7 disconnected components and the corresponding graph of regions pictured in Figure 7.

## 7.2 Comparison with geometric techniques

There are 4 dipaths to be traversed in the graph of disconnected regions to determine the behaviour of this concurrent system (two of them leading to state

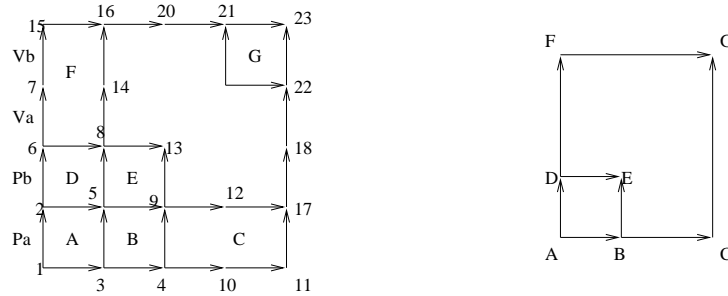


Fig. 7.

13 being dihomotopic). In fact, there are two explanations why the method of diconnected components is superior to the persistent set approach. In the latter, the independence relation does not in general depend on the current state (even if this might be changed by changing the set of labels), Our notion of independence is given by a 2-transition, which depends on the current state (see for instance [25] where the link is made formal). The second and more important reason is that the diconnected graph algorithm determines regions using *global* properties, whereas the persistent sets approach uses only (in general syntactic) local criteria for reducing the state-space. Conversely, it is relatively easy to see the following: For every state  $p$  in our asynchronous transition system (or by [25], in a 2-dimensional cubical set), all traces  $t$  composed of actions outside  $T_p$  is such that all its actions are independent with  $T_p$ . So any trace from  $p$  made up of any action (those of  $T_p$  as well as those outside  $T_p$ ) can be deformed (by dihomotopy, or “is equivalent to”) into a trace firing first actions from  $T_p$  and then actions outside  $T_p$ . Therefore the selective search approach using only actions from  $T_p$  (for all  $p$ ) is only traversing some representatives of the dihomotopy classes of paths. The persistent search approach is a particular case of dihomotopic deformation (not optimal in general).

### 7.3 Miscellaneous techniques

*Sleep sets.* The sleep sets technique can be seen as a mere amelioration of the traversal we saw, and therefore can be combined with the method of persistent sets (as well as ours). The problem we had in Section 7.1 is that quite a few of the paths we are traversing go through the same states at some point, and have a common suffix (like paths (1,2,6,8,14,16,20,21,23) and (1,2,5,8,14,16,20,21,23) in Figure 6). It is obviously not necessary to traverse again common suffixes if we want to check future tense logical formulas. The sleep sets  $S_p$  ( $p$  a state in our asynchronous transition system) use the information about the current traversals made, and not any information of any semantic kind to cope with this problem: see [21] for the precise definition. This reduces the number of transitions but not the number of states as shown in Figure 8. We now produce only 5 paths. The

same method is applied on Figure 7 to give the optimal result of Figure 9 (3 paths).

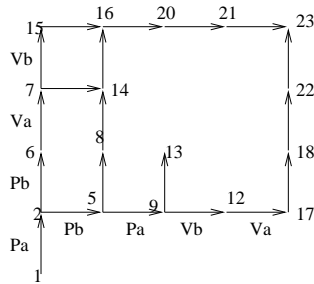


Fig. 8.

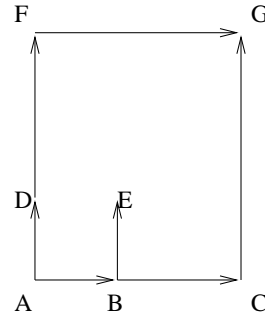


Fig. 9.

This does not entirely solve the suffix problem. A classical means to complete this method is to use “state space hashing” [20]. In this method (again orthogonal to “dihomotopy reduction”), one tries to maintain the set of states (and the set of transitions sometimes) already traversed avoiding to visit again the same sequences of states and transitions. As this database might be very big, one uses standard hashing techniques which quickly decide if two states are equal, but which might “identify” unequal states. This was used for instance in one of our simple implemented analyzers described in [14]. Again, for cyclic transition systems, this transformation does not change the deadlock(s) or LTL (future tense temporal logics) formulae that are true.

*Covering steps or Virtual Coarsening.* Another idea, called “covering step” in [49] or sometimes called in other situations “virtual coarsening”, is to group together interleavings of independent actions by “multiset” transitions or “covering steps”. For instance, when possible (basically, you should check that you are not in the situation of Figure 1), one replaces an interleaving square of two actions by the transition  $s_0 \rightarrow^{\{a,b\}} s_3$ . This is obviously subsumed by our notion of  $n$ -transition (formally, this can be done in the style of [39]).

*Algorithmics of the representation of the state-space.* A number of clever algorithmic methods have been used to reduce the representation of the state-space in memory without discarding any transition nor state, rather by compressing the representation. A very much used technique in model-checking is the representation of the transition relation with binary decision diagrams (BDDs) or QDDs as in [3] associated with symbolic representations of states [28]. Some amount of work has been devoted to “on the fly” techniques, also in model checking, see for instance [31]: Only a part of the state-space is represented during the analysis,

because there is no need in general to construct first the whole state-space and then traverse it. Last but not least some techniques involving reducing the state-space using symmetry arguments have been proposed and successfully used, see [9]. All these techniques could be equally applied to our disconnected components approach, and should be exemplified in future papers. For instance, symmetry techniques are quite well studied in geometry and should apply straightforwardly to our geometric approach.

## 8 Concluding remarks

Two further arguments in favor of our geometric techniques should be developed: We should be able to gain much more when the dimension of the problem (i.e. the number of processes involved) increases. The persistent sets types of methods basically use local transpositions, or in our geometric phrasing, faces of dimension 2, to equate some of the equivalent dipaths. Geometrically speaking, we can use sometime shorter deformation paths, like any hypercube, i.e. any cyclic permutation. The other argument is that geometric methods do cooperate well with abstraction mechanisms (in the sense of abstract interpretation [11]). It is in particular shown in [15] that the upper-approximation (or lower-approximation) of the forbidden regions can be carried out simply for a variety of languages, using classical abstract interpretation domains. These give lower (respectively upper) approximations of the “interesting” schedules or paths to be traversed.

## References

1. M.A. Armstrong, *Basic Topology*, Springer-Verlag, 1990.
2. M. A. Bednarczyk. *Categories of asynchronous systems*. PhD thesis, University of Sussex, 1988.
3. Bernard Boigelot and Patrice Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. *Formal Methods in System Design: An International Journal*, 14(3):237–255, May 1999.
4. T. Borceux, *Handbook of Categorical Algebra I: Basic category theory*, Encyclopedia of Mathematics and its Applications, Cambridge University Press, 1994.
5. Glen E. Bredon, *Topology and Geometry*, GTM, vol. 139, Springer-Verlag, 1993.
6. R. Brown and P.J. Higgins, *Colimit theorems for relative homotopy groups*, J. Pure Appl. Algebra **22** (1981), 11–41.
7. ———, *On the algebra of cubes*, J. Pure Appl. Algebra **21** (1981), 233–260.
8. S.D. Carson and P.F. Reynolds, *The geometry of semaphore programs*, ACM TOPLAS **9** (1987), no. 1, 25–53.
9. E. M. Clarke, E. A. Emerson, S. Jha, and A. P. Sistla. Symmetry reductions in model checking. In *Proc. 10th International Computer Aided Verification Conference*, pages 145–458, 1998.
10. E. G. Coffman, M. J. Elphick, and A. Shoshani. System deadlocks. *Computing Surveys*, 3(2):67–78, June 1971.

11. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction of approximations of fixed points. *Principles of Programming Languages 4*, pages 238–252, 1977.
12. E.W. Dijkstra, *Co-operating sequential processes*, Programming Languages (F. Genuys, ed.), Academic Press, New York, 1968, pp. 43–110.
13. L. Fajstrup, *Discovering spaces*, Tech. Report R-01-2023, Department of Mathematical Sciences, Aalborg University, DK-9220 Aalborg Øst, November 2001.
14. L. Fajstrup, É. Goubault, and M. Raussen, *Detecting Deadlocks in Concurrent Systems*, CONCUR '98; Concurrency Theory (Nice, France) (D. Sangiorgi and R. de Simone, eds.), Lect. Notes Comp. Science, vol. 1466, Springer-Verlag, September 1998, 9th Int. Conf., Proceedings, pp. 332 – 347.
15. L. Fajstrup, E. Goubault, and M. Raussen. Detecting deadlocks in concurrent systems. Technical report, CEA, 1998.
16. ———, *Algebraic topology and concurrency*, Tech. Report R-99-2008, Department of Mathematical Sciences, Aalborg University, DK-9220 Aalborg Øst, June 1999, conditionally accepted for publication in Theoret. Comput. Sci.
17. L. Fajstrup and S. Sokolowski, *Infinitely running concurrents processes with loops from a geometric viewpoint*, Electronic Notes Theor. Comput. Sci. **39** (2000), no. 2, 19 pp., URL: <http://www.elsevier.nl/locate/entcs/volume39.html>.
18. P. Gabriel and M. Zisman, *Calculus of fractions and homotopy theory*, Springer-Verlag New York, Inc., New York, 1967, Ergebnisse der Mathematik und ihrer Grenzgebiete, Band 35.
19. P. Gaucher, *Homotopy invariants of higher dimensional categories and concurrency in computer science*, Math. Structures Comput. Sci. **10** (2000), no. 4, 481–524.
20. P. Godefroid, G. J. Holzmann, and D. Pirottin. State space caching revisited. In *Proc. 4th International Computer Aided Verification Conference*, pages 178–191, 1992.
21. P. Godefroid and P. Wolper, *Using partial orders for the efficient verification of deadlock freedom and safety properties*, Proc. of the Third Workshop on Computer Aided Verification, vol. 575, Springer-Verlag, Lecture Notes in Computer Science, 1991, pp. 417–428.
22. P. Godefroid and P. Wolper. Partial-order methods for temporal verification. In *Proc. of CONCUR'93*. Springer-Verlag, LNCS, 1993.
23. É. Goubault, *The Geometry of Concurrency*, Ph.D. thesis, Ecole Normale Supérieure, Paris, 1995.
24. ———, *Geometry and Concurrency: A User's Guide*, Electronic Notes Theor. Comput. Sci. **39** (2000), no. 2, 16 pp., URL: <http://www.elsevier.nl/locate/entcs/volume39.html>.
25. ———, *Cubical sets are generalized transition systems*, avail. at <http://www.di.ens.fr/~goubault>, 2001.
26. J. Gunawardena, *Homotopy and concurrency*, Bulletin of the EATCS **54** (1994), 184–193.
27. A. Hatcher, *Algebraic Topology*, electronically available at <http://math.cornell.edu/~hatcher/#ATI>
28. T. A. Henzinger, O. Kupferman, and S. Qadeer. From pre-historic to post-modern symbolic model checking. In *Proc. 10th International Computer Aided Verification Conference*, pages 195–206., 1998.
29. M. Herlihy, S. Rajsbaum, and M. Tuttle, *An Overview of synchronous Message-Passing and Topology*, Electronic Notes Theor. Comput. Sci. **39** (2000), no. 2.
30. P.J. Higgins, *Categories and Groupoids*, Mathematical Studies, vol. 32, van Nostrand Reinhold, London, 1971.

31. G. Holzmann. On-the-fly model checking. *ACM Computing Surveys*, 28(4es):120–120, December 1996.
32. W. Lipski and C.H. Papadimitriou, *A fast algorithm for testing for safety and detecting deadlocks in locked transaction systems*, *Journal of Algorithms* **2** (1981), 211–226.
33. S. Mac Lane, *Categories for the working mathematician*, Graduate Texts in Mathematics, vol. 5, Springer-Verlag, New York, Heidelberg, Berlin, 1971.
34. A. Mazurkiewicz. Trace theory. In G. Rozenberg, editor, *Petri Nets: Applications and Relationship to Other Models of Concurrency, Advances in Petri Nets 1986, PART II, PO of an Advanced Course*, volume 255 of *LNCS*, pages 279–324, Bad Honnefs, September 1986. Springer-Verlag.
35. J.R. Munkres, *Elements of Algebraic Topology*, Addison-Wesley, 1984.
36. V. Pratt, *Modelling concurrency with geometry*, Proc. of the 18th ACM Symposium on Principles of Programming Languages. (1991).
37. M. Raussen, *On the classification of dipaths in geometric models for concurrency*, *Math. Structures Comput. Sci.* **10** (2000), no. 4, 427–457.
38. ———, *State spaces and dipaths up to dihomotopy*, Tech. Report R-01-2025, Department of Mathematical Sciences, Aalborg University, DK-9220 Aalborg Øst, November 2001.
39. V. Sassone and G. L. Cattani. Higher-dimensional transition systems. In *Proceedings of LICS'96*, 1996.
40. V. Sassone, M. Nielsen, and G. Winskel. Relationships between models of concurrency. In *Proceedings of the Rex'93 school and symposium*, 1994.
41. J. P. Serre, *Homologie singulière des espaces fibrés.*, *Ann. of Math.* (2) **54** (1951), 425–505.
42. A. Shoshani and E. G. Coffman. Sequencing tasks in multiprocess systems to avoid deadlocks. In *Conference Record of 1970 Eleventh Annual Symposium on Switching and Automata Theory*, pages 225–235, Santa Monica, California, Oct 1970. IEEE.
43. S. Sokolowski, *Categories of dimaps and their dihomotopies in po-spaces and local po-spaces*, Preliminary Proceedings of the Workshop on Geometry and Topology in Concurrency Theory GETCO'01 (Aalborg, Denmark) (P.Cousot et al., ed.), vol. NS-01, BRICS Notes Series, no. 7, BRICS, 2001, pp. 77 – 97.
44. J. Srba, *On the Power of Labels in Transition Systems*, CONCUR 2001 (Aalborg, Denmark) (K.G. Larsen and M. Nielsen, eds.), *Lect. Notes Comp. Science*, vol. 2154, Springer-Verlag, 2001, pp. 277–291.
45. A. Stark. Concurrent transition systems. *Theoretical Computer Science*, 64:221–269, 1989.
46. A. Valmari, *A stubborn attack on state explosion*, Proc. of Computer Aided Verification, no. 3, AMS DIMACS series in Discrete Mathematics and Theoretical Computer Science, 1991, pp. 25–41.
47. R. van Glabbeek, *Bisimulation semantics for higher dimensional automata*, Tech. report, Stanford University, 1991.
48. R. van Glabbeek and U. Goltz. Partial order semantics for refinement of actions. *Bulletin of the EATCS*, (34), 1989.
49. F. Vernadat, P. Azema, and F. Michel. Covering step graph. *Lecture Notes in Computer Science*, 1091, 1996.
50. G. Winskel and M. Nielsen. *Models for concurrency*, volume 3 of *Handbook of Logic in Computer Science*. Oxford University Press, 1994.