From Geometric Semantics to Asynchronous Computability

Éric Goubault Samuel Mimram
LIX, École Polytechnique
Eric.Goubault@polytechnique.edu
Samuel.Mimram@polytechnique.edu

Christine Tasson
PPS, Paris 7
Christine.Tasson@pps.univ-paris-diderot.fr

Abstract

We show that the protocol complex formalization of fault-tolerant protocols can be derived from first principles, i.e. directly from a suitable semantics of the underlying synchronization and communication primitives, based on a geometrization of the state space. By constructing a one-to-one relationship between simplices of the protocol complex and (di)homotopy classes of (di)paths in the later semantics, we describe a connection between these two geometric approaches to distributed computing: protocol complexes and directed algebraic topology. This is exemplified on atomic snapshot, iterated snapshot and layered immediate snapshot protocols, where a well-known combinatorial structure, interval orders, plays a key role. We believe that this correspondence between models paves the way to proving impossibility results for much more intricate fault-tolerant distributed architectures.

Categories and Subject Descriptors F.1.1 [*Computation by Abstract Devices*]: Models of Computation–*computability theory*; F.1.2 [*Computation by Abstract Devices*]: Models of Computation–*parallelism and concurrency*

Keywords fault-tolerant distributed computing, atomic snapshot protocol, protocol complex, directed algebraic topology, dihomotopy, interval order

1. Introduction

Fault-tolerant distributed computing is concerned with determining algorithms, and, when possible, solving so-called *decision tasks* on a given distributed architecture, in the presence of faults. The seminal result in this field was established by Fisher, Lynch and Patterson in 1985, who proved that there exists a simple task that cannot be solved in a message-passing (or equivalently a shared memory) system with at most one potential crash [10]. In particular, there is no way in such a distributed system to solve the very fundamental consensus problem: each processor starts with an initial value in local memory, typically an integer, and should end up with a common value, which is one of the initial values. Later on, Biran, Moran and Zaks developed a characterization of the decision tasks that can be solved by a (simple) message-passing system in the presence of one failure [3]. The argument uses a "similarity chain", which can be seen as a connectedness result of a representation of the space of all reachable states, called the *view complex* [24] or the *protocol complex* [22]. Of course, this argument turned out to be difficult to extend to models with more failures, as higher-connectedness properties of the protocol complex matter in these cases. This technical difficulty was first tackled, using homological considerations, by Herlihy and Shavit [21] (and independently [5, 29]): there are simple decision tasks, such as k-set agreement, a weaker form of consensus, that cannot be solved in the wait-free asynchronous model, i.e. shared-memory distributed protocols on n processors, with up to $n-1$ crash failures. Then, the full characterization of wait-free asynchronous decision tasks with atomic read and writes (or equivalently, with atomic snapshots) was described by Herlihy and Shavit [22]: this relies on the central notion of chromatic (or colored) simplicial complexes, and their subdivisions. All results above are deduced from the contractibility of the so-called "standard" chromatic subdivision, which was completely formalized in [24, 25] (and even for *iterated* models [18]) and corresponds to the *protocol complex* of distributed algorithms solving layered immediate snapshot protocols.

Over the year, the geometric approach to problems in fault-tolerant distributed computing has been very successful, see [23] for a fairly complete up-to-date treatment. One potential limitation however is that for some intricate models, it is extremely difficult to produce their corresponding protocol complex. In this paper, we are exploring the links between the semantics of the synchronization and communication primitives we are considering on a given distributed architecture, and the protocol complex. The interest is that the semantics of such synchronization primitives is much simpler to write down than the protocol complex, which is very error-prone to describe, as we will see in Section 3.2. We advocate in this paper the calculation of protocol complexes *from the first principles*, i.e. directly from the formal semantics of the underlying synchronization primitives.

The other aim of this article is to make the link between two geometric theories of concurrent and distributed computations: one based on protocols complexes, and the other, based on *directed algebraic topology*. Actually, the semantics of concurrent and distributed systems can be given by topological models, as pushed forward in a series of seminal papers in concurrency, in the early 1990s. These papers have explored the use of precubical sets and *Higher-Dimensional Automata* (which are labeled precubical sets equipped with a distinguished beginning vertex) [28, 30], begun to suggest possible homology theories [13, 16] and pushed the development of a specific homotopy theory, part of a general *directed algebraic topology* [19]. On the practical side, directed topological models have found applications to deadlock and unreachable state detection [6], validation and static analysis [4, 8, 15] state-space reduction (as in e.g. model-checking) [17], serializability and correctness of databases [20] (see also [7, 14] for a panorama of applications).

PODC 2015, Month d–d, 20yy, City, ST, Country.

http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn

In order to instantiate this link, we will be considering the simple model of shared-memory concurrent machines with crash failures, where processors compute and communicate through shared locations, and where reads and writes are supposed to be atomic. This model can also be presented [26] as *atomic snapshot protocols* [1, 2], where processors are executing the following instructions: scanning the entire shared memory (and copying it into their local memory), computing in its local memory, and updating its "own value", i.e. writing the outcome of its computation in a specific location in global memory, assigned to him only. The methodology we are describing here is by no means limited to this simple model: we have provided in this paper a general framework that builds protocol complexes from the semantics of communication primitives. However, what is more difficult is determining the set of *directed homotopy classes* of directed paths in this semantics. This is one of the reasons why we chose to exemplify the method on a well-known and simple case in fault-tolerant distributed computing. In general, this step is by no means trivial, reinforcing the need for formally deriving protocol complexes *from first principles*. The other reason is that the reader will be more familiar with the model and the expected result, and will be able to focus on the new technical (directed algebraic topological) aspects of the paper.

Contents of the paper and main contributions. Section 2.1 begins by defining the *standard semantics* (or interleaving semantics) of atomic read/write protocols, and more precisely of atomic snapshot protocols where read and writes primitives are replaced by update and (global) scan ones. In Section 2.2, we give an alternative *geometric* semantics, which encodes also independence of actions, as a form of *homotopy* in a geometric model. The very basics of *directed algebraic topology* have been introduced for this purpose, but we refer the reader to [9, 19] for more details. Yet, for the wider picture, we prove the fact that (directed) homotopy encodes commutation of actions, in the form of an equivalence between the standard semantics and the geometric semantics. It is shown in Section 2.3.1, Proposition 6 that two traces in the interleaving semantics modulo commutation of actions induce dihomotopic (directed) paths in the geometric model. The converse is shown in Section 2.3.3, Proposition 11, using the combinatorial notion of *interval order* [11]. We then combine these results with the semantic equivalence of Proposition 9, Section 2.3.3; this is the first main contribution of the paper.

In Section 3.1, we turn to the other geometric model of distributed systems: protocol complexes. The second main contribution of the paper is developed in Section 3.2: the protocol complex for atomic snapshot protocols (possibly iterated) is derived from the geometric semantics of Section 2.2, through interval orders again. We particularize this construction in Section 3.3 to the case of *layered immediate snapshot* which is generally studied by most authors, since it is much simpler to study, and is enough to prove the classical impossibility theorems, as e.g. [21]. Our explicit description of the protocol complex in the latter case is the same as the one of [18] (linked as well to the equivalent presentation of [24]), hence, combined with the result of [18] it proves that the layered immediate snapshot protocols produce collapsible protocol complexes, for any number of rounds. Therefore, it implies the asynchronous computability theorem of [21] all the way from the semantics of the communication primitives is used in these protocols.

2. Concurrent semantics of asynchronous read/write protocols

2.1 Interleaving semantics of atomic read/write protocols

In *atomic snapshot* protocols, n processes communicate through shared memory using two primitives: `update` and `scan`. Informally, the shared memory is partitioned in n parts, each one corresponding to one of the n processes. The part of the memory associated with process P_i, with $i \in \{0, \dots, n-1\}$, is the one on which process P_i can write, by calling `update`. This primitive writes, onto that part of memory, a value computed from the value stored in a local register of P_i. Note that as the memory is partitioned, there are never any write conflicts on memory. Conversely, all processes can read the entire memory through the `scan` primitive. Note also that there are never any read conflicts on memory. Still, it is well known that atomic snapshot protocols are equivalent [26], with respect to their expressiveness in terms of fault-tolerant decision tasks they can solve, to the protocols based on atomic registers with atomic reads and writes. Generic snapshot protocols are such that all processes loop, any number of times, on the three successive actions: locally compute a *decision value*, `update` then `scan`. It is also known [21, 22] that, as far as fault-tolerant properties are concerned, an equivalent model of computation can be considered: the full-information protocol where, for each process, decisions are only taken at the end of the protocol, i.e. after rounds of `update` then `scan`, only remembering the history of communications.

2.1.1 Interleaving semantics and trace equivalence

Formally, we consider a fixed set $\mathcal{V}$ of *values*, together with two distinguished subsets $\mathcal{I}$ and $\mathcal{O}$ of *input* and *output values*, the elements of $\mathcal{V} \setminus (\mathcal{I} \cup \mathcal{O})$ being called *intermediate values*, and an element $\perp \in \mathcal{I} \cap \mathcal{O}$ standing for an *unknown value*. We suppose that the sets of values and intermediate values are infinite countable, so that pairs $\langle x, y \rangle$ of values $x, y \in \mathcal{V}$ can be encoded as intermediate values, and similarly for tuples. We suppose fixed a number $n \in \mathbb{N}$ of processes. We also write $[n]$ as a shortcut for the set $\{0, \dots, n-1\}$, and $\mathcal{V}^n$ for the set of n-tuples of elements of $\mathcal{V}$, whose elements are called *memories*. Given $v \in \mathcal{V}^n$ and $i \in [n]$, we write v_i for the i-th component of v. We sometimes write $\perp^n$ for the memory l such that $l_i = \perp$ for every $u \in [n]$.

There are two families of memories, each one containing one memory cell for each process P_i:

- the *local memories*: $l = (l_i)_{i \in [n]} \in \mathcal{V}^n$,
- the *global (shared) memory*: $m = (m_i)_{i \in [n]} \in \mathcal{V}^n$.

A *state* of a program is a pair $(l, m) \in \mathcal{V}^n \times \mathcal{V}^n$ of such memories. Processes can communicate by performing actions which consist in updating and scanning the global memory, using their local memory: we denote by u_i any `update` by the i-th process and s_i any of its `scan`. We write $\mathcal{A}_i = \{u_i, s_i\}$ and $\mathcal{A} = \bigcup_{i \in [n]} \mathcal{A}_i$ for the set of *actions*. Formally, the effect of the actions on the state is defined by a *protocol* π which consists of two families of functions

$$\pi_{u_i} : \mathcal{V} \to \mathcal{V} \qquad \text{and} \qquad \pi_{s_i} : \mathcal{V} \times \mathcal{V}^n \to \mathcal{V}$$

indexed by $i \in [n]$ such that $\pi_{s_i}(x, m) = x$ for $x \in \mathcal{O}$. Starting from a state (l, m), the effect of actions is as follows:

- u_i: replace the contents of m_i by $\pi_{u_i}(l_i)$,
- s_i: replace the contents of l_i by $\pi_{s_i}(l_i, m)$.

A protocol is *full-information* when $\pi_{u_i}(x) = x$ for every $x \in \mathcal{V}$, i.e. each process fully discloses its local state in the global memory. A sequence of actions $T \in \mathcal{A}^*$ is called an *interleaving trace*, and we write $[\![T]\!]_\pi(l, m)$ for the state reached by the protocol π after executing the actions in T, starting from the state (l, m). A sequence of actions $T \in \mathcal{A}^*$ is *well-bracketed* (giving some form of generic protocol) when for every $i \in [n]$ we have $\mathrm{proj}_i(T) \in (u_i s_i)^*$, where $\mathrm{proj}_i : \mathcal{A}^* \to \mathcal{A}_i^*$ is the obvious projection which only keeps the letters in $\mathcal{A}_i$ in a word over $\mathcal{A}$. We denote by $\mathcal{A}^\omega$ the set of countably infinite sequences of actions; such a sequence is well-bracketed when every finite prefix is.

It can be noticed that different interleaving traces may induce the same final local view for any process. Indeed, if $i \neq j$, then u_i and u_j modify different part of the global memory, as we already noted informally, and thus $u_i u_j$ and $u_j u_i$ induce the same action on a given state. Similarly, s_i and s_j change different part of the local memory, and thus $s_i s_j$ and $s_j s_i$ induce the same action on a given state. On the contrary, $u_i s_j$ and $s_j u_i$ may induce different traces as u_i may modify the global memory that is scanned by s_j. We can sum up this remark by:

Proposition 1. *The* equivalence *on interleaving traces, defined as the smallest congruence $\approx$ such that*

$$\forall i \neq j, \quad u_j u_i \approx u_i u_j \qquad \textit{and} \qquad s_j s_i \approx s_i s_j$$

induces an operational equivalence: equivalent interleaving traces starting from the same initial state leads to the same final state.

This justifies that we consider traces *up to equivalence* in the following. We use the usual notions on such operational semantics: *execution traces*, *interleaving traces* or *paths* will denote any finite sequences of actions u_i and s_i in $\mathcal{A}^*$, *maximal* execution traces are traces that cannot be further extended. We also use the classical notions of *length* and *concatenation* of execution traces.

2.1.2 Decision tasks

We are going to consider the possibility of solving a particular task with an asynchronous protocol. Formally, those tasks are specified as follows:

Definition 2. A *wait-free task* Θ is a relation $\Theta \subseteq \mathcal{I}^n \times \mathcal{O}^n$ such for every $(l_1, l_2) \in \Theta$ and $i \in [n]$, we also have $(l_1', l_2') \in \Theta$ where l_1' (resp. l_2') is the memory obtained from l_1 (resp. l_2) by replacing the i-th value by $\perp$. The *domain* of a wait-free task Θ is $\operatorname{dom}\Theta = \{l \in \mathcal{I}^n \mid \exists l' \in \mathcal{O}^n, (l, l') \in \Theta\}$ and its codomain is $\operatorname{codom}\Theta = \{l' \in \mathcal{O}^n \mid \exists l \in \mathcal{I}^n, (l, l') \in \Theta\}$.

Notice that $\operatorname{dom}\Theta$ induces a simplicial complex, with $[n] \times \mathcal{I} \setminus \{\perp\}$ as vertices, and simplices are $\{(i, x) \in [n] \times \mathcal{V} \mid l_i = x \neq \perp\}$, for any $l \in \operatorname{dom}\Theta$. This simplicial complex is called the *input complex*; the *output complex* is defined similarly from $\operatorname{codom}\Theta$. We say that a protocol π *solves* a task Θ when for every $l \in \operatorname{dom}\Theta$, and well-bracketed infinite sequence of actions $T \in \mathcal{A}^\omega$, there exists a finite prefix T' of T such that $(l, l') \in \Theta$ where l' is the local memory after executing T', i.e. $(l', m') = [\![T']\!]_\pi(l, \perp^n)$. It can be shown that, w.r.t. task solvability, we can assume that $\operatorname{dom}\Theta$ contains only the memory l such that $l_i = i$, and its faces; for simplicity we will do so in Section 3.

Out of particular interest is the *view protocol* (sometimes identified with the full-information protocol in the literature) $\pi^\lhd$ such that $\pi^\lhd_{u_i}(x) = x$ for $x \in \mathcal{V}$, i.e. the protocol is full-information, and $\pi^\lhd_{s_i}(x, m) = \langle x, \langle m \rangle \rangle$ for $x \in \mathcal{V}$ and $m \in \mathcal{V}^n$: when reading the global memory, the protocol stores (an encoding as a value of) the pair constituted of its current local memory x and (an encoding as a value of) the global memory m it has read. Given a set $\mathcal{L} \subseteq \mathcal{I}^n$ of possible initial local memories, a local memory reached by this protocol by a well-bracketed trace, starting from a state $(l, \perp^n)$ with $l \in \mathcal{L}$, is called a *coherent view memory*, and the set of such memories is written $\mathrm{CViews}_\mathcal{L}$. Given $i \in [n]$, an *i-view* is a value of the form l_i for some coherent view memory $l \in \mathrm{CViews}_\mathcal{L}$ and we write $\mathrm{Views}_i(\mathcal{L})$ for the set of i-views. It can be shown that the view protocol is the "most general one" in the following sense:

Proposition 3. *Given any protocol π, there exists a unique family of functions $\phi_i : \mathrm{Views}_i(\mathcal{I}^n) \to \mathcal{V}$ indexed by $i \in [n]$, such that $\phi_i(x) = x$ for $x \in \mathcal{I}$, and such that for every $\langle x, \langle m \rangle \rangle \in \mathrm{Views}_i(\mathcal{I}^n)$,*

$$\pi_{s_i}\left(\phi_i(x), \left(\pi_{u_j} \circ \phi_j\left(m_j\right)\right)_{j \in [n]}\right) = \pi_{u_i} \circ \phi_i\left(\langle x, \langle m \rangle \rangle\right)$$

This is akin to the use of *generic protocols in normal form* [22], where protocols only exchange their full history of communication for a fixed given number of rounds, and then apply a local decision function. This also means that we will be satisfied with describing the potential sets of histories of communication between processes, without having to encode the decision values: this will be the basis of the geometric semantics of Section 2.2. As a direct consequence, we recover the usual definition of the solvability of a task as a simplicial map from some iterated protocol complex to the output complex [22, 23].

2.2 Directed geometric semantics

In this section, we give an alternative semantics to atomic snapshot protocols, using a geometric encoding of the state space, together with a notion of "time direction". One of the most simple setting in which this can be performed is the one of pospaces [12, 27]: a *pospace* is a topological space $\mathbb{X}$ endowed with a partial order $\leq$ such that the graph of the partial order is closed in $\mathbb{X} \times \mathbb{X}$ with the product topology. The intuition is that, given two points $x, y \in \mathbb{X}$ such that $x \leq y$, y cannot be reached before x. The encoding can be done in a quite general manner [8, 9]. Here, for the sake of simplicity, we define directly the pospace that gives the semantics we are looking for. It is rather intuitive and we will check this is correct with respect to the interleaving semantics, in Section 2.3.

Consider the pospace $\mathbb{X}^n_r$ below, indexed by the number n of processes and the vector of number of rounds $(r) = (r_0, \ldots, r_{n-1})$ (each $r_i \in \mathbb{N}$, with $i \in [n]$, is the number of times process P_i performs `update` followed by `scan`). Here, we use a vector to represent the number of rounds, which is rather unusual: this is because we do not want to treat only the layered immediate snapshot protocols, but more general atomic snapshot protocols. We claim now that the geometric semantics of the generic protocol, for n processes and (r) rounds, is represented by the pospace

$$\mathbb{X}^n_{(r)} \quad = \quad \prod_{i \in [n]} [0, r_i] \setminus \bigcup_{\substack{i,j \in [n] \\ k \in [r_i],\ l \in [r_j]}} U_i^k \cap S_j^l \qquad (1)$$

endowed with the product topology and product order induced by $\mathbb{R}^n$, where

- $n, r_i \in \mathbb{N}$ and $u, s \in \mathbb{R}$ with $0 < u < s < 1$,
- $U_i^k = \left\{ x \in \prod_{i \in [n]} [0, r_i] \;\middle|\; x_i = k + u \right\}$ stands for the region where the i-th process updates the global memory into its local memory for the k-th time,
- $S_j^l = \left\{ x \in \prod_{i \in [n]} [0, r_i] \;\middle|\; x_j = l + s \right\}$ stands for the region where the i-th process scans the global memory with its local memory for the l-th time.

The meaning of (1) is that a state $(x_0, \ldots, x_{n-1}) \in \prod_{i \in [n]} [0, r_i]$, i.e. a state in which each process P_i is at local time x_i, is allowed excepting when it is in $U_i^k \cap S_j^l$ (for $i, j \in [n]$ and $k \in [r_i]$, $l \in [r_j]$): these forbidden states are precisely the states for which there is a `scan` and `update` conflict. Namely, states in $U_i^k \cap S_j^l$ are states for which process P_i updates (for the k-th time) while process P_j scans (for the l-th time), which is forbidden in the semantics. Indeed, the memory has to serialize the accesses since shared locations are concurrently read and written, and either the `scan` operation will come before the `update` one, or the contrary, but the two operations cannot occur at the same time. This is reflected in the geometric semantics by a hole in the state space, as pictured in Figure 1 for two processes with one round each, and in Figure 3 for two processes with several rounds each. In higher-dimensions, the holes exhibit a complicated combinatorics. For instance, for three processes, and one round each, Figure 2

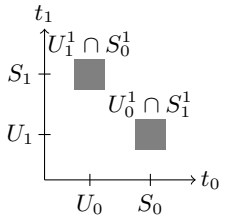


Figure 1. Pospace $X^1_{(1,1)}$.

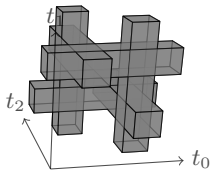


Figure 2. Pospace $X^2_{(1,1,1)}$.

shows forbidden regions that intersect one another. What happens in dimension 3 is that for all 3 pairs of processes (P,Q), we have to produce a forbidden region which has a projection, on the two axes corresponding to P and Q, similar to the one of Figure 1. Hence for all three pairs of processes, we have two cylinders with square section punching entirely the set of global states of the system. Each of these 6 cylinders correspond to a pair (P,Q) of processes, and a hole created either by a scan of P and an update of Q, or a scan of Q and an update of P. Consider the cylinder created by the conflict between the scan of P with the update of Q: it intersects exactly two cylinders (parallel to the two other axes) in a non trivial way, the one created by the scan of the third processor R and the update of Q, and the one created by the update of R and the scan of P, as shown in Figure 2.

2.3 Equivalence of the standard and geometric semantics

In the geometric semantics of Section 2.2, we can define notions analogous to equivalence of traces as for the standard interleaving semantics of Section 2.1 (Proposition 1).

A *dipath* (or *directed path*) in a pospace $(\mathbb{X}, \leq)$ is a continuous map $\alpha : [0,1] \to \mathbb{X}$ which is continuous and non decreasing when $[0,1]$ is endowed with the order and topology induced by the real line. A dipath is the continuous counterpart (as we will make clear later) of a trace in the interleaving semantics, or an execution. A dipath $\alpha : [0,1] \to \mathbb{X}$ is called *inextendible*, if there is no dipath $\beta : [0,1] \to \mathbb{X}$ such that $\alpha([0,1]) \subsetneq \beta([0,1])$. This is the analogous in our geometric setting to maximal execution traces. The *concatenation* of two dipaths $\alpha, \alpha' : [0,1] \to \mathbb{X}$ with compatible ends, i.e. $\alpha(1) = \alpha'(0)$ is the dipath $\alpha \cdot \alpha'$ given by

$$(\alpha \cdot \alpha')(x) = \begin{cases} \alpha(x) & \text{if } x \leq 0.5 \\ \alpha'(2x-1) & \text{if } x \geq 0.5 \end{cases}$$

The continuous setting allows us to use the classical concepts of (di)homotopy, which is the natural notion of equivalence between paths, and to use some tools from algebraic topology to derive properties of protocols (and more generally programs [14]). A *dihomotopy* is a continuous map $H : [0,1] \times [0,1] \to \mathbb{X}$ such that for all $t \in [0,1]$, the map $H(-,t)$ is a dipath. Two dipaths α, β such that $\alpha(0) = \beta(0)$ and $\alpha(1) = \beta(1)$ are *dihomotopic*, written $\alpha \leftrightsquigarrow \beta$, if there is a dihomotopy map $H : [0,1] \times [0,1] \to \mathbb{X}$ with $H(-,0) = \alpha$ and $H(-,1) = \beta$. We denote by $[\alpha]$ the set of inextendible dipaths dihomotopic to α and $\mathbf{dPath}(\mathbb{X})$ the set of dipaths up to dihomotopy. Figure 3 represents dihomotopic dipaths in a $\mathbb{X}^n_{(r)}$ space.

2.3.1 From equivalence classes of interleaving traces to dihomotopic dipaths

We associate a dipath α_T, starting at the origin 0, to any interleaving trace T with n processes and (r) rounds. This dipath is built by induction on the length of trace T:

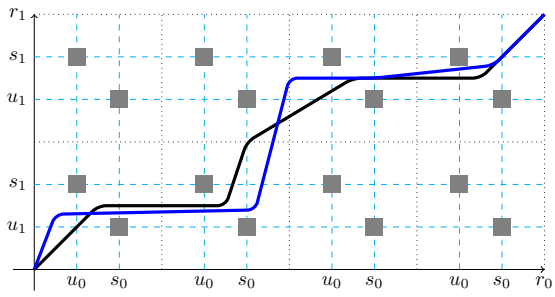


Figure 3. Two dihomotopic dipaths in the geometric semantics $X^2_{(4,2)}$.

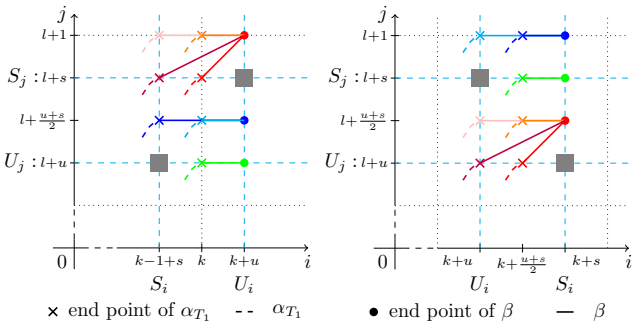


Figure 4. Extension of α_{T_1} by β according to last action of j: on left during the k-th update of i, on right during the k-th scan of i.

Lemma 4. *There exists a (not necessary inextendible) dipath α_T in $\mathbb{X}^n_{(r)}$ such that $\alpha_T(0)_i = 0$, for every $i \in [n]$, and satisfying the following. If the last action of process i in T is its k-th update, then $\alpha_T(1)_i \in \left\{k+u, k+\frac{u+s}{2}\right\}$. If it is its k-th scan, then $\alpha_T(1)_i \in \{k+s, k+1\}$. If the last action in T is the k-th update of process i, then $\alpha_T(1)_i = k+u$. If it is the k-th scan of i, then $\alpha_T(1)_i = k+s$.*

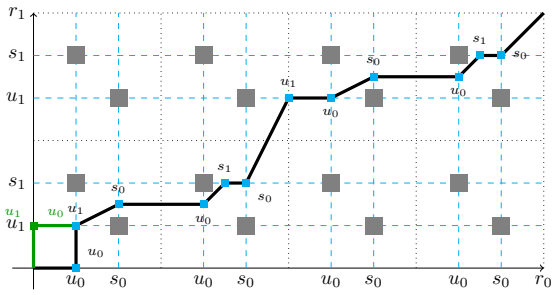
Proof. First, when T is of length 0, α_T is the constant dipath staying at the origin 0. Otherwise, let $T = T_1 \cdot A$ be the concatenation of a trace T_1 with action A (being either update u_i or scan s_i). By induction, we have a dipath α_{T_1} starting at 0 and ending at $\alpha_{T_1}(1)$, associated to T_1, that satisfies Lemma 4. Now, construct a dipath β, which is a line as described on Figure 4 starting at $\beta(0) = \alpha_{T_1}(1)$, and ending at $\beta(1)$ such that:

- Let us assume that the action A is an update, say the k-th update of process i. As partly represented on the left part of Figure 4, by Lemma 4, since the action before was a scan or nothing, $\alpha_{T_1}(1)_i \in \{0, k-1+s, k\}$ and we set $\beta(1)_i = k+u$.
  For any other process $j \neq i$, if the last action of j is its say l-th scan, then $\alpha_{T_1}(1)_j \in \{l+s, l+1\}$ and we set $\beta(1)_j = l+1$ (in red tones), otherwise we set $\beta(1)_j = \alpha_{T_1}(1)_j$ (in blue tones).
- If A is a scan, say the k-th scan of i then, see the right part of Figure 4. Since the action of i before was the k-th update, we have $\alpha_{T_1}(1)_i \in \left\{k+u, k+\frac{u+s}{2}\right\}$ and we set $\beta(1)_i = k+s$.
  For any other process j, if the last action of j is its l-th update, then $\alpha_{T_1}(1)_j = \left\{l+u, l+\frac{u+s}{2}\right\}$ and set $\beta(1)_j = l+\frac{u+s}{2}$ (in red tones), otherwise we set $\beta(1)_j = \alpha_{T_1}(1)_j$ (in blue tones).

We then define the dipath $\alpha_{T_1 \cdot A} = \alpha_{T_1} \cdot \beta$. □

To a maximal interleaving trace T, we can associate an inextendible dipath α'_T by further extending α_T as defined above: we define α'_T to be $\alpha_T \cdot \gamma$ where γ is the dipath given by (any parameterization of) the line from $\gamma(0) = \alpha_T(1)$ to $\gamma(1) = (r_i)_{i\in[n]}$, the point $\gamma(1)$ thus being the end of all inextendible dipaths in $\mathbb{X}^n_{(r)}$. We shall not distinguish in the sequel α'_T from α_T since we will only consider maximal interleaving traces and their inextendible counterparts.

Example 5. The following figure shows the dipaths associated to the trace $T = u_0u_1s_0u_0s_1s_0u_1u_0s_0u_0s_1s_0$ and the trace $T' = u_1u_0s_0u_0s_1s_0u_1u_0s_0u_0s_1s_0$ (only the beginnings differ, the second trace being in green):



We finally have:

Proposition 6. *Two equivalent interleaving traces induce homotopic dipaths.*

Proof. Recall from Proposition 1 that the equivalence of traces is generated by $u_iu_j \approx u_ju_i$ and $s_is_j \approx s_js_i$. Consider two traces T and T' and their associated dipaths α_T and $\alpha_{T'}$. Assume that T and T' are identical until the $(m-1)$-th action and only differ by the ordering of their m-th and $(m+1)$-th actions. Up to reparametrization, we can assume that these actions occur at the same time in α_T and $\alpha_{T'}$, respectively t_{m-1}, t_m, and t_{m+1}.

First assume that in T, the m-th action is the k-th update of process i and the $(m+1)$-th action is the l-th update of process j. On the left part of Figure 5, the possible paths are drawn, one color being associated to one possible point at t_{m-1}. Notice that from t_m, the paths are identical and are colored in black. Indeed, by Lemma 4,

$$\alpha_T(t_m)_i = k+u \quad \text{and} \quad \alpha_T(t_{m+1})_j = l+u.$$

These actions are in the reverse order in T', so

$$\alpha_{T'}(t_m)_j = l+u \quad \text{and} \quad \alpha_{T'}(t_{m+1})_i = k+u.$$

The action of i and j before t_m in T are respectively the $(k-1)$-th scan and the $(l-1)$-th scan or nothing. Hence,

$$\alpha_T(t_{m-1})_i = \alpha_{T'}(t_{m-1})_i \in \{0, (k-1)+s, k\},$$
$$\alpha_T(t_{m-1})_j = \alpha_{T'}(t_{m-1})_j \in \{0, (l-1)+s, l\}.$$

Besides, by construction (the scan and update region is forbidden),

$$\alpha_{T'}(t_m)_i = k \quad \text{and} \quad \alpha_T(t_m)_j = l,$$
$$\alpha_T(t_{m+1})_i = k+u \quad \text{and} \quad \alpha_{T'}(t_{m+1})_j = l+u.$$

Then $t \mapsto t\alpha_T + (1-t)\alpha_{T'}$ is a dihomotopy in $\mathbb{X}^n_{(r)}$ between α_T and $\alpha_{T'}$.

Now, assume that in T, the m-th action is the k-th scan of process i and the $(m+1)$-th action is the l-th scan of process j. The possible paths are drawn on the right part of Figure 5. Again by Lemma 4,

$$\alpha_T(t_m)_i = k+s \quad \text{and} \quad \alpha_T(t_{m+1})_j = l+s.$$

These action are in the reverse order in T', so

$$\alpha_{T'}(t_m)_j = l+s \quad \text{and} \quad \alpha_{T'}(t_{m+1})_i = k+s.$$

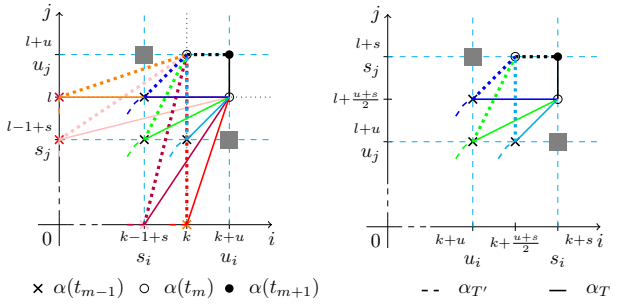


Figure 5. Dihomotopic dipaths of equivalent traces differing by swapping two updates on left and two scans on right according to their starting points.

The action of i and j before t_m in T are respectively the k-th update and the l-th update. Hence,

$$\alpha_T(t_{m-1})_i = \alpha_{T'}(t_{m-1})_i \in \{k+u, k+(u+s)/2\},$$
$$\alpha_T(t_{m-1})_j = \alpha_{T'}(t_{m-1})_j \in \{l+u, l+(u+s)/2\}.$$

Besides, by construction (the scan and update region is avoided),

$$\alpha_{T'}(t_m)_i = k+(u+s)/2 \quad \text{and} \quad \alpha_T(t_m)_j = l+(u+s)/2,$$
$$\alpha_T(t_{m+1})_i = k+s \quad \text{and} \quad \alpha_{T'}(t_{m+1})_j = l+s.$$

Then $t \mapsto t\alpha + (1-t)\alpha'$ is a dihomotopy between α_T and $\alpha_{T'}$. □

2.3.2 Equivalence between equivalence classes of interleaving traces and (colored) interval orders

In order to prove that dihomotopic dipaths are in bijection with interleaving traces modulo equivalence, we introduce a combinatorial gadget that encodes the history of events observable on both an equivalence class of interleaving traces, and a dihomotopy class of dipaths in our continuous models.

Definition 7. Let $(I_x)_{x\in X}$ be a family of intervals on the real line $(\mathbb{R}, \leq)$. This family induces a poset $(X, \preceq)$, where $\prec$ is defined as

$$x \prec y \quad \text{iff} \quad \forall s \in I_x,\ \forall t \in I_y, \quad s < t.$$

Such a poset is called an *interval order* [11].

We denote $x \| y$ the *independence* relation, that is $x \| y$ whenever $\neg(x \prec y)$ and $\neg(y \prec x)$. A colored variant can also be defined:

Definition 8. An $[n]$-*colored interval order* is given by an interval order $(X, \preceq)$ and a *labeling* function $\ell : X \to [n]$ such that two elements with the same label are comparable. Then for any $i \in [n]$, the restriction of the interval order to intervals labeled by i is a total order. Write $\mathbf{cIO}(X)$ the set of colored interval orders on a set X.

Proposition 9. *There is a bijection between $[n]$-colored interval orders and traces up to equivalence.*

Proof. We first associate a colored order interval to an interleaving trace T. For any $i \in [n]$. Let r_i be the number of occurrences of u_i in T. Let v_i^k and σ_i^k be the respective k-th occurrence of u_i and s_i. Let $X = \{(i,k) \mid k \in [r_i],\ i \in [n]\}$. Any embedding of T in the real line induces an interval order by setting $I_{(i,k)} = [v_i^k, \sigma_i^k]$. More precisely, X is then endowed with the partial order:

$$(i,k) \prec (i',k') \quad \text{iff} \quad \sigma_i^k < v_{i'}^{k'} \tag{2}$$

that is σ_i^k occurs before $v_{i'}^{k'}$. We can label this interval order $(X, \preceq)$ by $\ell : (i,k) \mapsto i$, and hence produce an $[n]$-colored interval order since T is well-bracketed.

Conversely, we associate an interleaving trace T_I to an $[n]$-colored interval order $I = (X, \preceq)$ labeled by ℓ. For any $i \in [n]$, the set $\{x \in X \mid l(x) = i\}$ is totally ordered of cardinal $[r_i]$. Then, we can assume w.l.o.g. that $X = \{(i,k) \mid k \in [r_i],\ i \in [n]\}$ and that $(i,k) \prec (i,k')$ whenever $k < k'$. Let us choose w.l.o.g. an interval representation I of $(X, \preceq)$ such that endpoints are pairwise disjoints. For any $k \in [r_i]$, $i \in [n]$, let v_i^k and σ_i^k be the left and right endpoint of the interval $I_{(i,k)}$ of the real line. The real line order induces a linear ordering of the endpoints such that the equivalence (2) is satisfied. Then T_I is obtained by substituting u_i to v_i^k and s_i to σ_i^k in the given sequence of endpoints.

Let us finally prove that two interval representations $I = (I_{k,i})$ and $J = (J_{k,i})$, indexed by $k \in [r_i]$ and $i \in [n]$, induce equivalent traces $T_I \approx T_J$. From the equivalence (2), we deduce that if $(i,k) \prec (i',k')$ then $\sigma_i^k < v_{i'}^{k'}$ and if $(i,k) \| (i',k')$ then $\sigma_i^k \not< v_{i'}^{k'}$, that is $\sigma_i^k > v_{i'}^{k'}$. Thus, the only freedom is on the ordering of the u_i's on the one side, and of the s_i's on the other side, which corresponds precisely to the equivalence of traces. □

From the last proposition and Proposition 6, we can associate to any interval order a class of dihomotopic dipaths. We denote $i : \mathbf{cIO}(X_n) \to \mathbf{dPath}(\mathbb{X}^n_{(r)})$ the maps that send an interval order to a dipath up to dihomotopy.

2.3.3 From dihomotopic dipaths to equivalence classes of interleaving traces

As already mentioned, dipaths geometrically represent execution traces, keeping in mind that dipaths which can be deformed through a continuous family of executions are operationally equivalent. This argument can be made concrete for the asynchronous model we are working on, by giving the explicit relation between dipaths and colored interval orders (Definition 8), because of Proposition 9.

To any inextendible dipath $\alpha : [0,1] \to \mathbb{X}^n_{(r)}$, we associate an interval order $\preceq_\alpha$ on

$$X^n_{(r)} \quad = \quad \{(i,k) \mid i \in [n], k \in [r_i]\}$$

through the interval collection for $i \in [n]$, $I_{(i,k)} = [u_i^k, s_i^k]$ colored by i where

$$u_i^k \quad = \quad \inf\left\{ t \in [0,1] \;\middle|\; \alpha(t)_i \in U_i^k \right\} \tag{3}$$

$$s_i^k \quad = \quad \inf\left\{ t \in [0,1] \;\middle|\; \alpha(t)_i \in S_i^k \right\} \tag{4}$$

correspond to the event "α enters an update or scan hyperplane".

Let us give simple examples of this in dimension 2 and 3. In dimension 2, and for one round, consider the three inextendible dipaths of Figure 6. (We are not writing the round number as upper index since we are considering here only one round). They are actually representatives of the three dihomotopy classes of dipaths in this pospace. The dipath α_0, on the left of Figure 6, corresponds to an execution in which process 1 does its update and scan before process 0 even starts updating. Hence, the interval of local times at which process 1 updates and scans is less than the interval of local times at which process 0 updates and scans: this is reflected by the corresponding interval order $[u_1, s_1] \prec_{\alpha_0} [u_0, s_0]$. The one on the right, α_3 is symmetric: the corresponding interval order is $[u_0, s_0] \prec_{\alpha_3} [u_1, s_1]$. The dipath on the middle of Figure 6 corresponds to an execution in which the two processes are running synchronously, hence they both update at the same time, and scan at the same time: the corresponding interval order is $[u_0, s_0]$ not comparable with $[u_1, s_1]$.

In dimension 3, there are many more dipaths that one can draw. Consider only, for instance, the synchronous execution of the three processes: this is shown in Figure 7 and corresponds to the interval order where all three intervals $[u_0, s_0]$, $[u_1, s_1]$ and $[u_2, s_2]$ are not comparable.

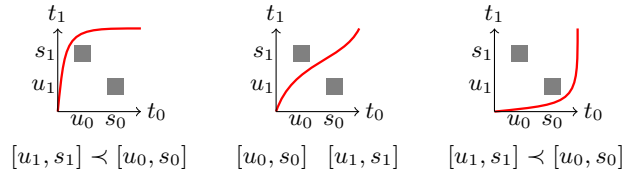


Figure 6. Dipaths in the three dihomotopy classes of inextendible dipaths in $\mathbb{X}^2_{(1,1)}$ together with corresponding interval orders.

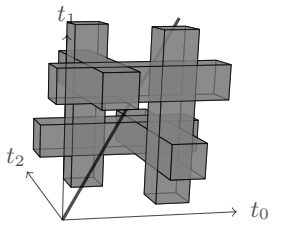


Figure 7. Pospace $\mathbb{X}^3_{(1,1,1)}$ and a synchronous execution, on the diagonal.

For any $i \in [n]$, the restriction of this order to the intervals labeled by i is a total order. Indeed, dipaths α are non decreasing, $u < s$ and $\alpha(u_i^k)_i = k+u$, $\alpha(s_i^k)_i = k+s$, hence for all $k \in [r_i]$, $u_i^k < s_i^k$ and if $k \neq 0$, $s_i^{k-1} < u_i^k$.

Remark 10. One should keep in mind that:

- $\alpha(u_i^k)_i = k + u$ and $\alpha(s_i^k)_i = k + s$,
- if $u_i^k \leq t < s_i^k$, then $k + u \leq \alpha(t)_i < k + s$,
- if $s_i^k \leq t < u_i^{k+1}$, then $k + s \leq \alpha(t)_i < (k + 1) + u$.

Moreover, notice that the trace T_α induced by the intersection of α with the update and scan hyperplanes is associated to the interval order $(X^n_{(r)}, \preceq_\alpha)$.

Proposition 11. *If two inextendible dipaths on $\mathbb{X}^n_{(r)}$ are dihomotopic, then they induce the same interval order.*

Proof. Let $H : [0,1] \times [0,1] \to \mathbb{X}^n_{(r)}$ be a dihomotopy between $\alpha = H(-,1)$ and $\beta = H(-,0)$. Let u_i^k (resp. v_i^k) and s_i^k (resp. t_i^k) be the defined as in (3) and (4) for α (resp. β). Let us fix $i, j \in [n]$ and prove that, for any $k \in [r_i]$ and $l \in [r_j]$, the dipaths α and β intersect U_i^k and S_j^l in the same order. More precisely, we want to prove that:

$$u_i^k < s_j^l \quad \text{iff} \quad v_i^k < t_j^l. \tag{5}$$

We can then deduced that the dihomotopic dipaths $H(-,0)$ and $H(-,1)$ induce the same interval order on

$$X^n_{(r)} \quad = \quad \{(i,k) \mid i \in [n],\ k \in [r_i]\}\,.$$

Let H_{ij}, α_{ij} and β_{ij} be the projections of H, α and β respectively on the plan $[0, r_i] \times [0, r_j]$ induced by the processes i and j. Notice that H_{ij}, α_{ij} and β_{ij} are continuous and that for any $t \in [0,1]$, $H_{ij}(-,t)$, α_{ij} and β_{ij} are non-decreasing. Moreover, since U_i^k and S_j^l are parallel to the direction along which we project, H_{ij}, α_{ij} and β_{ij} are taking their values in the pospace:

$$\mathbb{X}_{ij} \quad = \quad [0, r_i] \times [0, r_j] \setminus \bigcup_{k \in [r_i],\ l \in [r_j]} U_i^k \cap S_j^l.$$

Thus, H_{ij} is a dihomotopy between α_{ij} and β_{ij} in the space $\mathbb{X}_{ij}$. Since α_{ij} and β_{ij} are homotopic, the concatenation of α_{ij} and of the reverse of β_{ij} is contractible in $\mathbb{X}_{ij}$. Thus, there is no hole between α_{ij} and β_{ij}. Since moreover they are non-decreasing, we get:

$$\alpha(u_i^k)_j < l + s \quad \text{iff} \quad \beta(v_i^k)_j < l + s.$$

The equivalence (5) follows. □

Let us denote $r : \mathbf{dPath}(\mathbb{X}^n_{(r)}) \to \mathbf{cIO}(X_n)$ the map sending a dipath up to dihomotopy to an interval order. We end this section by proving that there is a retraction between traces up to equivalence and dipaths up to dihomotopy:

$$\mathbf{cIO}(X_n) \underset{r}{\overset{i}{\rightleftarrows}} \mathbf{dPath}(\mathbb{X}^n_{(r)})$$

Lemma 12. *Two inextendible dipaths α and β, which intersect the update and scan hyperplanes in the same order, are dihomotopic.*

Proof. Since α and β intersect the update and scan hyperplanes in the same order, we can reparametrize β such that the times at which u_i^k and s_j^l intersect are the same for α and β. Then, the function $H : x, t \mapsto t\,\alpha(x) + (1-t)\beta(x)$ is a dihomotopy. Let us prove that H takes its value in $\mathbb{X}^n_{(r)}$, that is, for all $x, t \in [0,1]$, $H(x,t) \notin U_i^k \cap S_j^l$. Assume for instance that $u_i^k > s_j^l$. If $H(x,t) \in U_i^k$, then $H(x,t)_i = k + u$ and, since $\alpha, \beta \in \mathbb{X}^n_{(r)}$,

- either $\alpha(x)_i > k + u$ and $\beta(x) < k + u$, then, as α and β are non decreasing, $x > u_i^k$ and $x < u_i^k$ and we get a contradiction,
- either $\alpha(x)_i < k+u$ and $\beta(x) > k+u$, this case is impossible for the same reason,
- or $\alpha(x)_i = k + u$ and $\beta(x) = k + u$, then, as α and β are non decreasing, $\alpha(x)_j \geq \alpha(u_i^k)_j > \alpha(s_j^l)_j = l + s$ and $\beta(x)_j > l + s$, thus $H(x,t) \notin S_j^l$.

If $u_i^k < s_j^l$, consider $H(x,t) \in S_j^l$ to show $H(x,t) \notin U_i^k$. □

Proposition 13. *A dipath α is dihomotopic to the dipath associated to the interval order induced by α, that is, $i \circ r(\alpha) \leftrightsquigarrow \alpha$.*

Proof. Let T be a trace representing the interval order $(X^n_{(r)}, \preceq_\alpha)$ induced by α. Let T_α be the trace induced by the sequence of intersection of α with the update and scan hyperplanes. By Remark 10, T_α is also representing the interval order $(X^n_{(r)}, \preceq_\alpha)$, so that T_α and T are equivalent interleaving traces. Thus, $\alpha_T \leftrightsquigarrow \alpha_{T_\alpha}$ by Proposition 6. Now, by construction, the dipath α_{T_α} intersects the update and scan hyperplanes in the order given by T_α, that is in the same order as α (see Remark 10). Therefore, by Lemma 12, $\alpha \leftrightsquigarrow \alpha_{T_\alpha}$. Finally, we get $\alpha \leftrightsquigarrow \alpha_{T_\alpha} \leftrightsquigarrow \alpha_T = i(r(\alpha))$. □

The key result that we have just proved can be summarized as follows:

Theorem 14. *There is a deformation retract between dihomotopy classes of dipaths over $\mathbb{X}^n_{(r)}$ and interleaving traces up to equivalence.*

3. Protocol complexes, derived from the concurrent semantics

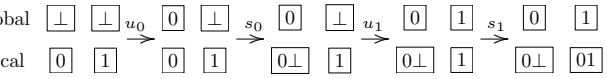
3.1 Protocol complex

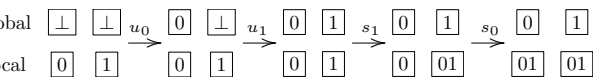
The protocol complex has been designed [22] to represent the possible reachable states, at some given round, of the generic protocol in normal form, i.e. it is going to encode all possible histories of communication between processes, and as we will prove later on, all interleaving traces up to equivalence (or equivalently the dipaths up to dihomotopy), by maximal simplices:

Definition 15. The *protocol complex* for atomic snapshot protocols is the abstract simplicial complex constructed from the generic protocol in normal form, and whose

- vertices are pairs (i, l_i) where $i \in [n]$ represents the name of a process and l_i represents its local memory,
- maximal simplices are $\{(0, l_0), \ldots, (n, l_n)\}$ where l_i is the local view by process i at the end of the execution represented by this simplex.

Example 16. The local views in each vertex are determined by the operational semantics of Section 2.1, as in the two following examples:





There is a third potential outcome of the computation, symmetric to the first case, in which process 1 would do its update and scan before process 0 does. Putting this together, according to Definition 15, we get the well known protocol complex for one round and two processes [22]:

$$0, (0\bot) \text{ ——— } 1, (01) \text{ ——— } 0, (01) \text{ ——— } 1, (\bot 1)$$

The encoding of the local states, i.e. vertices in the graph above, is as follows. The identifier of the process whose local view is the number before the comma, e.g. the state $0, (0\bot)$ above is the local view of processor 0. The group of numbers or $\bot$ within parentheses, e.g. $(0\bot)$ in the state above, is a condensed notation for the local state where $l_0 = \langle 0, \langle 0, \bot \rangle \rangle$, see Section 2.1. Similarly, state $1, (01)$ denotes the local view of processor 1, with local state such that $l_1 = \langle 1, \langle 0, 1 \rangle \rangle$.

3.2 Construction of the protocol complex from the directed geometric semantics

We can now link protocol complexes with interval orders, i.e. traces up to equivalence or dipaths up to dihomotopy: a colored interval order represents indeed an execution and we can deduce the local view of the i-th process by restricting the interval order to the last scan of i. We encode local views restricting to the full information generic protocol in normal form with initial local state $l_i = i$ for $i \in [n]$ (this only changes the naming of local states, and not the structure of the protocol complex).

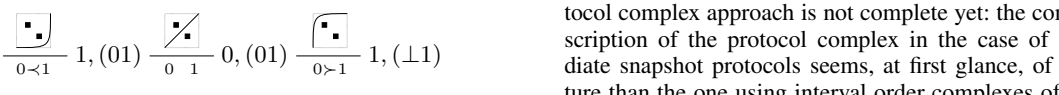
Proposition 17. *Let $(X^n_{(r)}, \preceq)$ be an $[n]$-colored interval order. Then the local memory of the i-th process at round k of its corresponding execution (in the full-information generic protocol in normal form, i.e. its view) is given by its restriction $\mathcal{V}_i^k$ to the k-th scan S_i^k of the i-th process, i.e.*

$$\mathcal{V}_i^k \quad = \quad \{(j,l) \mid (i,k) \| (j,l) \text{ or } (j,l) \prec (i,k)\}$$

meaning that it is the value of the local state l_i under the semantics of Section 2.1 for the interleaving path corresponding to the interval order $\mathcal{V}_i^k$ under the equivalence of Proposition 9.

Proof. Remember that $(i,k) \prec (j,l)$ iff S_i^k happens before U_j^l, see Equivalence (2). By contradiction, $(i,k) \| (j,l)$ or $(j,l) \prec (i,k)$ iff S_i^k happens after U_j^l. We conclude, noticing that the i-th local memory only depends on the updates preceding the last i-th scan. □

Example 18. Consider again the one round, two processes case. We have represented below the protocol complex already depicted in Example 16, and decorated its maximal simplices, i.e. edges, with the corresponding dipaths modulo dihomotopy above, and the corresponding interval order, below:



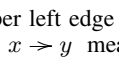
The local view of process 0 which is $0, (0\bot)$ comes from the restriction of the interval order $_{0\prec 1}$, subscript of the leftmost edge in the graph above, to 0: an interleaving trace corresponding to this interval order, under Proposition 9 is $u_0 s_0$ leading to local state $(0\bot)$ on process 0. Similarly, $1, (01)$ corresponds to the local state $l_1 = (01)$ for process 1, both for the restriction $_{0\prec 1}$ of $_{0\prec 1}$ to $\mathcal{V}_1^1$ (corresponding to a trace $u_0 s_0 u_1 s_1$, as in the trace  superscript of the edge on the left of the graph above) and for the restriction $_{0\quad 1}$ of $_{0\quad 1}$ to $\mathcal{V}_1^1$ (corresponding to a trace $u_0 u_1 s_0 s_1$ for instance, as in the trace  superscript of the middle edge of the graph above).

We are now in a position to give a combinatorial description of the protocol complex of Definition 15, using interval orders. We call the resulting equivalent complex, the *interval order complex*:

Definition 19. The *interval order complex* is the simplicial complex whose

- vertices are $((i,k), V_i^k)$ where i stands for the i-th process, k for the round number and V_i^k an interval order such that for all $(j,l) \in V_i^k$, either $(i,k) \| (j,l)$ or $(j,l) \prec (i,k)$,
- maximal simplices are $\{((0,r_0), V_0^{r_0}), \ldots, ((n,r_n), V_n^{r_n})\}$ such that there is an interval order $(X^n_{(r)}, \prec)$ whose restriction to (i,r_i) is $V_i^{r_i}$.

In that case we say that it is the interval order complex on (r) rounds and for $n+1$ processes.

Example 20. An example of interval order complex with the traces corresponding to the execution for 2 processes, 2 rounds is depicted at Figure 8. Note that this is not the classical iterated subdivision in three parts at each round, i.e. a 9 edges complex, that is depicted for atomic snapshot protocols [23]. This is because we are considering more executions that the classical *layered immediate snapshot protocols* of [23]: we allow round 2 of process 0 to begin while process 1 is still in round 1 for instance. Consider the interval order  labeling the upper left edge of the protocol complex in Figure 8, where an arrow $x \twoheadrightarrow y$ means $x \prec y$. As shown in the same figure, it corresponds to the execution  precisely where process 0 is executing its 2 rounds before process 1 even starts its first round. The local view of process 0 at (its) round 2 corresponds to the interval order  , restriction of to $\mathcal{V}_0^{(2,0)}$. An interleaving trace corresponding to this is e.g. $u_0 s_0 u_0 s_0$, which, by the semantics of Section 2.1, leads to the local state of process 0: $\langle 0, \langle 0, \langle 0, \bot\rangle\rangle \bot\rangle$ written in condensed form as the upper left local state $0, ((0\_)\_)$ in Figure 8.

In Figure 9, we show the interval order complex for 3 processes and 1 round. Note again that we do not have exactly the same picture as in [23]: to the 13 triangles that we have in [23], we have to add the 6 extra blue triangles that make the complex not faithfully representable as a planar shape and which correspond to non immediate snapshot executions. For instance, the upper left blue triangle is labeled with the interval order where 0 is not comparable to both 1 and 2, and 2 is less than 1. An interleaving trace (up to equivalence) corresponding to this interval order is given on the same figure: $u_0 u_2 s_2 u_1 s_1 s_0$.

3.3 Particular case of 1-round immediate snapshot protocols

The connections between directed algebraic topology and the protocol complex approach is not complete yet: the combinatorial description of the protocol complex in the case of layered immediate snapshot protocols seems, at first glance, of a different nature than the one using interval order complexes of Definition 19. We recall that an (layered, for multi-round protocols) *immediate snapshot* protocol [23] is a protocol where the snapshot of a given process comes "right after" its update, meaning that the allowed traces (within one round), up to equivalence, should be, of the form $u_{i_1} \ldots u_{i_k} s_{i_1} \ldots s_{i_k}$. Of course, there is some difference in that interval order complexes account for non necessarily layered nor "immediate" protocols. It is the aim of this section to make the connection between the subcomplex of interval order complexes describing layered immediate snapshot protocols, and the equivalent two definitions of chromatic barycentric subdivision [18, 24] that describe combinatorially the protocol complex in that case.

We recall that the standard chromatic subdivision $\chi(\Delta^{[n]})$ of the standard $[n]$-colored simplicial complex $\Delta^{[n]}$ is defined as follows (see [18], where an equivalence with the Definition in [24] is also shown):

Definition 21. The *standard chromatic subdivision* $\chi(\Delta^{[n]})$ of $\Delta^{[n]}$ is the $[n]$-colored simplicial complex whose vertices are pairs (V,i) with $V \subseteq [n]$ and $i \in V$ and simplices are sets of the form $\sigma = \{(V_0, i_0), \ldots, (V_d, i_d)\}$ with $d \geq -1$ ($\sigma = \emptyset$ when $d = -1$) which are

1. well-colored: for every $k, l \in [d]$, $i_k = i_l$ implies $k = l$,
2. ordered: for every $k, l \in [d]$, $V_k \subseteq V_l$ or $V_l \subseteq V_k$,
3. transitive: for every $k, l \in [d]$, $i_l \in V_k$ implies $V_l \subseteq V_k$.

This complex is colored via the second projection: $\ell(V,i) = i$.

Remark 22. The transitivity (property 3) of Definition 21 is equivalent to looking only at immediate snapshot executions. Observe the left upper blue triangle of Figure 8, which is composed of vertices $(0 : 012)$, $(1 : 012)$ and $(2 : 0\bot 2)$ (respectively meaning $(\{0,1,2\}, 0)$, $(\{0,1,2\}, 1)$ and $(\{0,2\}, 2)$ in the notations of Definition 21). It does not correspond to a layered execution: it corresponds to the equivalence class of traces $u_0 u_2 s_2 u_1 s_1 s_0$. Transitivity does not hold either: $0 \in \{0,2\}$ but $\{0,1,2\} \not\subseteq \{0,2\}$.

Proposition 23. *Layered immediate snapshot executions correspond to interval orders such that $J \prec K$ and I is not comparable with J implies $I \prec K$. The subcomplex of the interval order complex on one round, $(X^n_{(1,\ldots,1)}, \preceq)$, that contains only immediate snapshot executions is isomorphic to the chromatic barycentric subdivision of Definition 21.*

Proof. For the first part, suppose that we have an interval order $\preceq$, representing some simplex in the interval order complex such that $J \prec K$ and I is not comparable with J and K. I, J and K correspond to some intervals of updates and scans local times on some process, $[u_i^{l_i}, s_i^{l_i}]$, $[u_j^{l_j}, s_j^{l_j}]$ and $[u_k^{l_k}, s_k^{l_k}]$ respectively. Suppose that I is not comparable with K, this means that the interleaving path $\ldots u_i^{l_i} \ldots u_j^{l_j} \ldots s_j^{l_j} \ldots u_k^{l_k} \ldots s_k^{l_k} \ldots s_i^{l_i} \ldots$ is in the equivalence class represented by the interval order we are considering. This is clearly not layered nor immediate snapshot, therefore being a layered immediate snapshot execution implies the condition on $\preceq$ of Proposition 23.

Conversely, we suppose that for I not comparable to J and $J \prec K$, then $I \prec K$ and we prove that all interleaving paths are

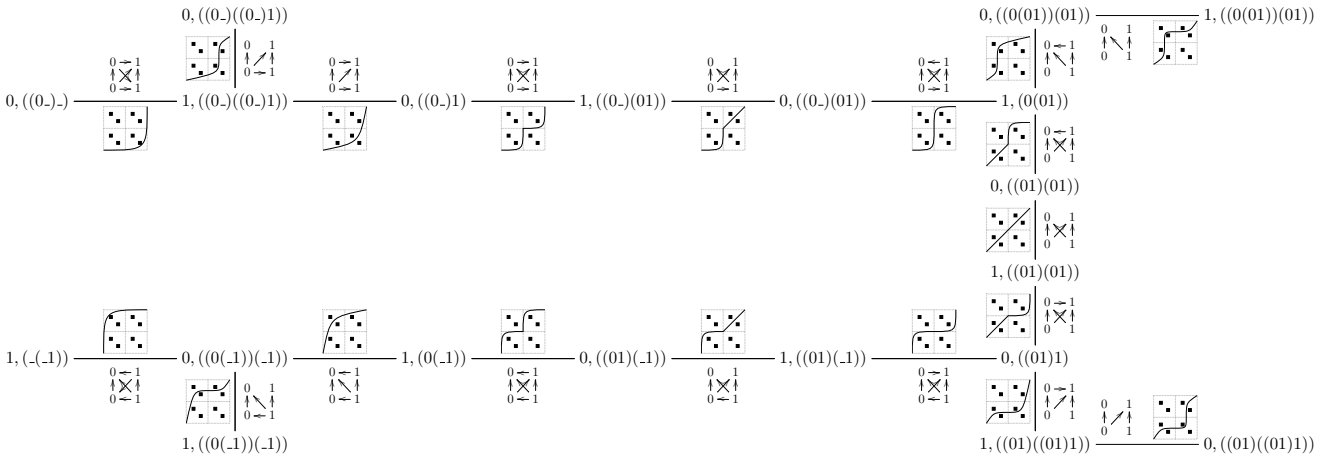


Figure 8. interval order complex, together with corresponding traces, of 2 processes, 2 rounds.

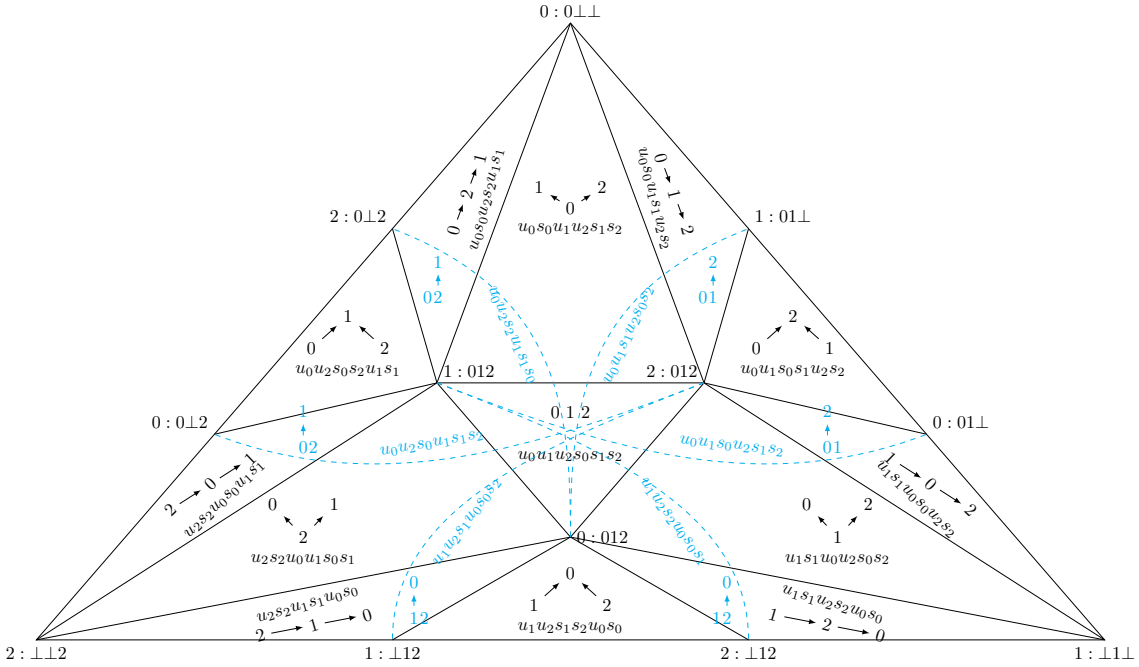


Figure 9. interval order complex with traces of 3 processes, 1 round.

layered and immediate snapshot ones. Suppose we have an interleaving path (up to equivalence) of the form: $Tu_j^{l_j}Us_j^{l_j}Vu_k^{l_k}Ws_k^{l_k}X$ where T, U, V, W and X are interleaving paths. This is a layered immediate snapshot execution except if there are update and scans $u_i^{l_i}$, $s_i^{l_i}$ such that $u_i^{l_i}$ appears in U and $s_i^{l_i}$ appears in W. But $u_i^{l_i}$ appearing in U implies $I = [u_i^{l_i}, s_i^{l_i}]$ is not comparable with J and hence, by hypothesis, I must be less that K, implying that $s_i^{l_i}$ appears in U or V.

Now, we prove the second statement. Consider a simplex $\sigma = \{(V_0, i_0), \ldots, (V_d, i_d)\}$ with $d \geq 0$ (the case $d = -1$ is trivial) in the chromatic barycentric subdivision of Definition 21. We associate to σ the following simplex in the interval order complex: we construct a partial order $\preceq_\sigma$ on $\{(V_0, i_0), \ldots, (V_d, i_d)\}$ such that $V_k \prec_\sigma V_l$ if $V_k \subsetneq V_l$ and the color of (V_l, i_l) is i_l, we just need to prove that this partial order is a colored interval order, and that the condition of Proposition 23 holds.

Let us now consider, in our partial order $\preceq_\sigma$, four elements (V_x, i_x), (V_y, i_y), (V_z, i_z) and (V_t, i_t), and suppose furthermore that $(V_x, i_x) \prec_\sigma (V_y, i_y)$ and $(V_z, i_z) \prec_\sigma (V_t, i_t)$. Then, as σ is "ordered" (see Definition 21), necessarily, either $V_x \subseteq V_z$ or $V_z \subseteq V_x$. Suppose we are in the first situation. We also have that $V_z \subseteq V_t$ and $V_z \neq V_t$ by definition of $\preceq$. Hence $V_x \prec_\sigma V_t$. We conclude that, as a partial order, $\preceq_\sigma$ is (2+2)-free, property which characterizes interval orders [11].

Now consider again σ in the chromatic barycentric subdivision, and its associated interval order $\preceq_\sigma$. Take $(V_y, i_y) \prec_\sigma (V_z, i_z)$

and (V_x, i_x) which is not comparable with (V_y, i_y). Hence, by definition of the (strict) order $\prec_\sigma$, $V_x = V_y$ or $V_x \not\subseteq V_y$. In the first case, $(V_x, i_x) \prec_\sigma (V_z, i_z)$, trivially, and in the second case, by property 2 ("ordered") of Definition 21, $V_y \subsetneq V_x$ which implies $(V_y, i_y) \prec_\sigma (V_x, i_x)$, impossible since (V_x, i_x) and (V_y, i_y) are supposed incomparable.

Finally, note that well-coloredness of σ implies trivially that the labeling we define is indeed a labeling function of a colored interval order.

Conversely, suppose we have a 1-round colored interval order $(X, \preceq)$ on $d+1$ elements which satisfies the property from Proposition 23. We consider the interval orders V_i^k, restriction of X to $\mathcal{V}_i^k = \{(j,l) \mid (i,k) \| (j,l) \text{ or } (j,l) \prec (i,k)\}$. We construct a (colored) d-simplex in the chromatic barycentric subdivision of Definition 21 by defining k-simplices (for all $k \leq n$) $\sigma_X = ((|V_i^{k_i}|, i))_{i \in [k]}$ (where $|V|$ the set of elements of an interval order V). Indeed we check easily that this is well-colored. Suppose we have $(|V_k|, i_k)$ and $(|V_l|, i_l)$ such that $i_l \in |V_k|$. As V_k and V_l are restrictions of the same interval order to both the set of elements less than or incomparable to i_k, respectively i_l, and that by definition of V_l, $i_l \in V_l$, we have $|V_l| \subseteq |V_k|$. A similar argument shows that property 2 of Definition 21 holds as well. □

4. Conclusion and future work

We have shown that there are strong connections between directed algebraic topology, with its applications to semantics and validation of concurrent systems, and the protocol complex approach to fault-tolerant distributed systems. This has been exemplified on the simple layered immediate snapshot model, but also on the more complicated (non layered, non immediate) iterated snapshot model. This, combined with the results of [18, 25], entirely classifies geometrically the computability of wait-free layered immediate snapshot protocols, directly from the semantics of the update and scan primitives. We classified combinatorially, en route, the potential schedules of executions (equivalently, the potential local views of processes) as an interesting and well-known combinatorial structure: interval orders.

This is a first step towards a more ambitious programme. Fault-tolerant distributed models, whose protocol complex are more complex to guess combinatorially, may be taken care of, by going through the very same steps we went through, starting with the geometric semantics of the communication primitives, and classifying the dipaths modulo dihomotopy. We are thinking of applying this to atomic read/write protocols with extra synchronization primitives such as test&set, compare&swap and others.

In the long run, we would like to derive impossibility results directly by observing some obstructions in the semantics, in the form of suitable directed algebraic topological invariants.

References

[1] Y. Afek, H. Attiya, D. Dolev, E. Gafni, M. Merritt, and N. Shavit. Atomic snapshots of shared memory. *J. ACM*, 40(4):873–890, Sept. 1993. .

[2] J. H. Anderson. Composite registers. In *Conference on Principles of Distributed Computing*, pages 15–30. ACM, New York, 1993.

[3] O. Biran, S. Moran, and S. Zaks. A combinatorial characterization of the distributed tasks which are solvable in the presence of one faulty processor. In *Proceedings of the seventh annual ACM Symposium on Principles of distributed computing*, pages 263–275. ACM, 1988.

[4] R. Bonichon, G. Canet, L. Correnson, É. Goubault, E. Haucourt, M. Hirschowitz, S. Labbé, and S. Mimram. Rigorous evidence of freedom from concurrency faults in industrial control software. In *SAFECOMP*, pages 85–98, 2011.

[5] E. Borowsky and E. Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In *Proc. of the 25th STOC*. ACM Press, 1993.

[6] L. Fajstrup, E. Goubault, and M. Raussen. Detecting deadlocks in concurrent systems. In *Proceedings of the 9th International Conference on Concurrency Theory*. Springer-Verlag, 1998.

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

[8] L. Fajstrup, É. Goubault, E. Haucourt, S. Mimram, and M. Raußen. Trace spaces: An efficient new technique for state-space reduction. In *ESOP*, pages 274–294, 2012.

[9] L. Fajstrup, E. Goubault, E. Haucourt, S. Mimram, and M. Raussen. *Directed Algebraic Topology and Concurrency*. Springer Verlag, 2015, to be published.

[10] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[11] P. C. Fishburn. Intransitive indifference with unequal indifference intervals. *Journal of Mathematical Psychology*, 7(1):144–149, 1970.

[12] G. Gierz. *A Compendium of continuous lattices*. Springer, 1980.

[13] E. Goubault. *The Geometry of Concurrency*. Ph.d. dissertation, Ecole Normale Supérieure and Ecole Polytechnique, 1995.

[14] É. Goubault. Some geometric perspectives in concurrency theory. *Homology, Homotopy and Applications*, 2003.

[15] É. Goubault and E. Haucourt. A practical application of geometric semantics to static analysis of concurrent programs. In *CONCUR 2005–Concurrency Theory*, pages 503–517. Springer, 2005.

[16] É. Goubault and T. P. Jensen. Homology of higher-dimensional automata. In *Proc. of CONCUR*. Springer-Verlag, Aug. 1992.

[17] É. Goubault, T. Heindel, and S. Mimram. A geometric view of partial order reduction. *Proceedings of Mathematical Foundations of Programming Semantics, Electr. Notes Theor. Comput. Sci.*, 298:179–195, 2013.

[18] E. Goubault, S. Mimram, and C. Tasson. Iterated chromatic subdivisions are collapsible. *Applied Categorical Structures*, pages 1–42, 2014. ISSN 0927-2852. .

[19] M. Grandis. *Directed Algebraic Topology : Models of Non-Reversible Worlds*, volume 13 of *New Mathematical Monographs*. Cambridge University Press, 2009. ISBN 978-0-521-76036-2.

[20] J. Gunawardena. Homotopy and concurrency. *Bulletin of the EATCS*, 54:184–193, 1994.

[21] M. Herlihy and N. Shavit. The asynchronous computability theorem for t-resilient tasks. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 111–120. ACM, 1993.

[22] M. Herlihy and N. Shavit. The topological structure of asynchronous computability. *Journal of the ACM (JACM)*, 46(6):858–923, 1999.

[23] M. Herlihy, D. Kozlov, and S. Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Elsevier, 2014.

[24] D. Kozlov. Chromatic subdivision of a simplicial complex. *Homology, Homotopy and Applications*, 14(2):197–209, 2012.

[25] D. Kozlov. Topology of the view complex. *arXiv preprint arXiv:1311.7283*, 2013.

[26] N. A. Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996.

[27] L. Nachbin. *Topology and order*. Van Nostrand mathematical studies. Van Nostrand, 1965.

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

[29] M. E. Saks and F. Zaharoglou. Wait-free k-set agreement is impossible: the topology of public knowledge. In *STOC*, pages 101–110, 1993.

[30] R. van Glabbeek. Bisimulation semantics for higher dimensional automata. Technical report, Stanford University, Manuscript available on the web as `http://theory.stanford.edu/~rvg/hda`, 1991.