Geometric and combinatorial views on asynchronous computability

Eric Goubault, Samuel Mimram, Christine Tasson

▶ **To cite this version:**

Eric Goubault, Samuel Mimram, Christine Tasson. Geometric and combinatorial views on asynchronous computability. Distributed Computing, 2018, 31 (4), pp.289-316. 10.1007/s00446-018-0328-4 . hal-02155487

HAL Id: hal-02155487
https://inria.hal.science/hal-02155487

Submitted on 13 Jun 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Geometric and Combinatorial Views on Asynchronous Computability

Éric Goubault Samuel Mimram Christine Tasson

Abstract

We show that the protocol complex formalization of fault-tolerant protocols can be directly derived 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 latter 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 will extend to proving impossibility results for much more intricate fault-tolerant distributed architectures.

Contents

Introduction

Fault-tolerant distributed computing is concerned with designing 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 Paterson in 1985, who proved the existence of a simple task that cannot be solved in a message-passing (or equivalently a shared memory) system with at most one potential crash [14]. 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 [6]. 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* [29] or the *protocol complex* [28]. 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 [27], and independently by Borowsky, Gafni and Saks [8, 34]: there are simple decision tasks, such as "k-set agreement" (a weak 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 reads and writes (or equivalently, with atomic snapshots) was described by Herlihy and Shavit [28]: this relies on the central notion of chromatic (or colored) simplicial complexes, and their subdivisions. All these results stem from the contractibility of the "standard" chromatic subdivision, which was completely formalized in [29, 30] (and even for *iterated* models [23]) and corresponds to the *protocol complex* of distributed algorithms solving immediate snapshot protocols. Actually, the protocol complex that has been considered in [29, 30, 23] are all based on an atomic snapshot model of some kind; this has been later refined for atomic reads and writes in [4].

Over the years, the geometric approach to problems in fault-tolerant distributed computing has been very successful, see [26] for a fairly complete up-to-date treatment. One potential limitation however is that for some intricate models, it might be 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. One of the interests is that the semantics is easier to describe than the full combinatorics of the protocol complex, and our framework may help finding or proving these protocol complexes correct.

The other aim of this article is to make the link between two geometric theories of concurrent and distributed computations: one based on protocol 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) [33, 35], begun to suggest possible homology theories [22, 18]

and pushed the development of a specific homotopy theory, part of a general *directed algebraic topology* [24]. On the practical side, directed topological models have found applications to deadlock and unreachable state detection [12], validation and static analysis [20, 7, 10], state-space reduction (as in e.g. model-checking) [21], serializability and correctness of databases [25] (see also [19, 13] for a panorama of applications).

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 as *atomic snapshot protocols* [1, 2, 31], 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 semantics. 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.

Concretely, in this setting, we draw a precise relationship between

- execution traces up to observational equivalence,
- dihomotopy classes of paths in geometric models,
- partially ordered sets of actions,

which is the first contribution of this paper. Another major contribution is to formally show that the *full-information protocol* is the most general (i.e. initial) one, and in that the information retained by each process in this protocol, is the *view*, which we define in those different formalisms: informally, the view can be extracted from causal history as the part on which depends the last scan of a process. This allows us to provide new constructions of the protocols complex, based on traces, dipaths, and interval orders.

Contents of the paper

Section 1 begins by defining the semantics of *protocols*, which are distributed programs whose basic primitives consist in updating and scanning a shared memory: an *operational semantics* provides a formal mathematical description of the effect of those operations (Section 1.1.1) which can be extended to *interleaving traces*, describing the order in which actions occur in a particular execution (Section 1.1.2). In Section 1.2, we observe that some of those traces are observationally equivalent, in the sense that no program can distinguish between them: we axiomatize and study this notion of equivalence (we provide

normal forms for equivalence classes and identify well-bracketed traces to which we will be able to restrict without loss of generality).

The *tasks* (Section 2.1) specify the behaviors we are interested in implementing: in order to simplify the study of tasks that can be implemented, we recall some classical simplifications of the execution model which do not restrict the generality (Section 2.2), and show that it is enough to study the so-called *view protocol* because it is shown to be initial in the category of protocols (Section 2.3).

In Section 3, we give an alternative *geometric* semantics, which encodes independence of actions, as a form of *homotopy* in the semantics, and show that it coincides with the previously defined interleaving semantics: traces up to equivalence correspond to directed homotopy classes of paths in the geometric models.

We give a third characterization of traces in Section 4: those are precisely the linearizations of particular partial orders of actions called *interval orders*, which will turn out to be particularly convenient to manipulate.

In Section 5, we provide several characterizations of the *views*: those are the local states of processes in the initial protocol, thus formally encoding the communication history. We detail how they can be encoded as a particular partial order called *view order* (Section 5.1 and Section 5.2), and show how they can be extracted from traces (Section 5.3) and interval orders (Section 5.3).

This allows us in Section 6 to provide, several new constructions of the *protocol complex*, which is a classical simplicial complex whose vertices are views and simplices encode coherence between those, based on traces, directed homotopy classes of paths or interval orders.

We conclude in Section 7.

1 Concurrent semantics of asynchronous read/write protocols

In *atomic snapshot shared memory* protocols, n processes with local memory communicate through shared memory using two primitives: `update` and `scan`. Informally, the shared memory is partitioned in n cells, each corresponding to one of the n processes. A process P_i can write only on its associated memory cell, by calling `update`: this primitive writes, onto that part of memory, a value computed from the value stored in the local memory 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, 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 [31]. Generic snapshot protocols are such that all processes loop, any number of times, on the three successive actions: locally compute a *value*, `update` then `scan`, until terminating with a *decision value*. It is also known that, as far as fault-tolerant properties are concerned, an equivalent model of computation can be considered [27, 28]: 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.

1.1 Operational semantics

1.1.1 Protocols

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 we have an encoding $\langle m \rangle$ for every n-uple $m \in \mathcal{V}^n$.

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-uples 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 write $\perp^n$ for the memory m such that $m_i = \perp$ for any $i \in [n]$: this is typically the initial state of the global memory. There are two families of memories, each one containing one memory cell for each process P_i:

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

Given a memory $l \in \mathcal{V}^n$, $i \in [n]$ and $x \in \mathcal{V}$, we write $l[i \leftarrow x]$ for the memory obtained from l by replacing the i-th value by x.

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`. The effect of the actions on the state is formalized by a protocol as follows.

Definition 1. A *protocol* π 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 means "replace the contents of m_i by $\pi_{u_i}(l_i)$" and
- s_i means "replace the contents of l_i by $\pi_{s_i}(l_i, m)$".

The last condition in the definition of protocols states that once a protocol has decided upon an output value, it will not change its mind.

In order to study when a protocol can simulate another, it is convenient to use the following notion of morphism. Informally, the existence of a morphism $\phi : \pi \to \pi'$ means that the protocol π' can simulate the protocol π: given an input and a trace leading to an output from process i in π, the same input and trace should also lead to an output from process i in π'.

Definition 2. Given two protocols π and π', a *simulation* $\phi : \pi \to \pi'$ consists of two families of functions $\phi_i : \mathcal{V} \to \mathcal{V}$ and $\phi'_i : \mathcal{V} \to \mathcal{V}$, indexed by $i \in [n]$, respectively translating local and global memories, such that $\phi_i(x) = x$ for every $x \in \mathcal{I}$, $\phi_i(\mathcal{O}) \subseteq \mathcal{O}$ and the following diagrams commute:

$$\begin{array}{ccc} \mathcal{V} & \xrightarrow{\pi_{u_i}} & \mathcal{V} \\ {\scriptstyle \phi_i}\downarrow & & \downarrow{\scriptstyle \phi'_i} \\ \mathcal{V} & \xrightarrow[\pi'_{u_i}]{} & \mathcal{V} \end{array} \qquad \begin{array}{ccc} \mathcal{V} \times \mathcal{V}^n & \xrightarrow{\pi_{s_i}} & \mathcal{V} \\ {\scriptstyle \phi_i \times \prod_i \phi'_i}\downarrow & & \downarrow{\scriptstyle \phi_i} \\ \mathcal{V} \times \mathcal{V}^n & \xrightarrow[\pi'_{s_i}]{} & \mathcal{V} \end{array} \tag{1}$$

We say that π' *simulates* π when there exists such a morphism. We write **Prot** for the category with protocols as objects and simulations as morphisms.

Remark 3. This definition of morphism is not entirely satisfactory because the image of any value under ϕ_i (and ϕ'_i) has to be defined, even though this value might never occur as the local memory of process i during any execution of the protocol. The right definition consists in having ϕ_i and ϕ'_i be partial functions which are defined only on memory values which are reachable from a specified set of values (see Section 2.1). We do not detail this here in order to make this categorical part lighter, but we will implicitly suppose that memory values are reachable in the following: this is in particular required for Proposition 41 below to hold.

A simulation $\phi : \pi \to \pi'$ is *strict* when $\phi_i^{-1}(\mathcal{O}) \subseteq \mathcal{O}$. In this case, it will be clear that when π' implements a task, π also implements it. This remark can be useful to add, without loss of generality, further constraints to protocols, in order to simplify their study. An example of this is given by the following property.

Definition 4. A protocol is *full-disclosure* when $\pi_{u_i}(x) = x$ for every $x \in \mathcal{V}$.

A protocol is thus full-disclosure when each process fully discloses its local state in the global memory. The following lemma ensures that we can restrict to those protocols:

Lemma 5. *Any protocol can be simulated by a full-disclosure one.*

Proof. Suppose given a protocol π', and consider the full-disclosure protocol π defined by

$$\pi_{u_i}(x) = x \qquad \pi_{s_i}(x, m) = \pi'_{s_i}(x, (\pi'_{s_0}(m_0), \ldots, \pi'_{s_{n-1}}(m_{n-1})))$$

We can then define a morphism $\phi : \pi \to \pi'$ by $\phi_i = \mathrm{id}_{\mathcal{V}}$ and $\phi'_i = \pi'_{u_i}$, which is easily checked to satisfy the required axioms. □ □

Remark 6. In a morphism between full-disclosure protocols, the diagram on the left of (1) reduces to the fact that $\phi'_i = \phi_i$.

1.1.2 Interleaving traces

We write $\mathcal{A}_i = \{u_i, s_i, d_i\}$ and $\mathcal{A} = \bigcup_{i \in [n]} \mathcal{A}_i$ for the set of *actions*. As explained earlier, the action u_i (resp. s_i) amounts to the i-th process updating its local

memory (resp. writing in the global memory). Since we want to take into account possible failures of the protocols, we also have an action d_i meaning that the i-th protocol *dies*: in a trace containing d_i, the process P_i is said to be *dead*, and alive otherwise. We sometimes write a_i to denote an arbitrary action in $\mathcal{A}_i$.

We denote by $\mathcal{A}^*$ the free monoid over $\mathcal{A}$: its elements T are finite sequences of elements of $\mathcal{A}$, i.e. words, which will be called *interleaving traces* (or simply *traces*) in the following. We also denote by $\mathcal{A}^\omega$ the set of countably infinite sequences of actions; we always specify an *infinite* trace for an element of this set. We write ε for the empty one and $T \cdot U$, or simply TU, for the concatenation of two traces T and U. We write $\mathrm{proj}_i : \mathcal{A}^* \to \mathcal{A}_i^*$ for the projections, keeping only the actions of the i-th process, and similarly $\mathrm{proj}_{\neg i} : \mathcal{A}^* \to (\bigcup_{j \neq i} \mathcal{A}_i)^*$ for the projection keeping all actions excepting those of the i-th process. Given a trace $T \in \mathcal{A}^*$, we write $\mathrm{dead}(T)$ for the set of indices $i \in [n]$ such that d_i occurs in T, i.e. the set of *dead processes*, and $\mathrm{alive}(T) = [n] \setminus \mathrm{dead}(T)$, i.e. the set of *alive processes*. A trace is *non-dying* if $\mathrm{dead}(T) = \emptyset$, or equivalently $\mathrm{alive}(T) = [n]$.

Given a trace $T \in \mathcal{A}^*$, 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). Formally, it is defined as follows:

Definition 7. Given a protocol π and a trace T, we write $[\![T]\!]_\pi : \mathcal{V}^n \times \mathcal{V}^n \to \mathcal{V}^n \times \mathcal{V}^n$ for its *denotational semantics*, which is the function defined by induction on the length of the trace T by

$$\begin{aligned}
[\![u_i \cdot T]\!]_\pi &= [\![T]\!]_\pi \circ [\![u_i]\!]_\pi \\
[\![s_i \cdot T]\!]_\pi &= [\![T]\!]_\pi \circ [\![s_i]\!]_\pi \\
[\![d_i \cdot T]\!]_\pi &= [\![\mathrm{proj}_{\neg i}(T)]\!]_\pi \\
[\![u_i]\!]_\pi(l, m) &= (l, m[i \leftarrow \pi_{u_i}(l_i)]) \\
[\![s_i]\!]_\pi(l, m) &= (l[i \leftarrow \pi_{s_i}(l_i, m)], m) \\
[\![\varepsilon]\!]_\pi(l, m) &= (l, m)
\end{aligned}$$

This extends the functions π_{u_i} and π_{s_i} on traces, and is such that once a process has died it has no effect on the memory. Its computation is illustrated in Example 40. Notice that we do not have $[\![d_i \cdot T]\!]_\pi = [\![T]\!]_\pi \circ [\![d_i]\!]_\pi$, so that the above does not define a right monoid action from the monoid of actions onto memories in general.

1.2 Observational equivalence on traces

1.2.1 Observational equivalence

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 parts of the global memory 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 parts 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. Finally, there are similar relations expressing the fact that once a process has died, what it does afterward does not matter. This suggests introducing the following equivalence relations on traces:

Definition 8. The *strong equivalence* $\cong$ is the smallest congruence on interleaving traces such that

$$\begin{array}{lll} u_j u_i \cong u_i u_j & s_j s_i \cong s_i s_j & u_i u_i \cong u_i \\ d_i u_i \cong d_i & d_i s_i \cong d_i & d_i d_i \cong d_i \\ d_j u_i \cong u_i d_j & d_j s_i \cong s_i d_j & d_j d_i \cong d_i d_j \end{array}$$

for every $i, j \in [n]$ with $i \neq j$. The *equivalence* $\approx$ is the smallest congruence on interleaving traces containing strong equivalence and the relation

$$s_i d_i \approx d_i$$

Intuitively, the last relation expresses the fact that when a process dies, it does not really matter whether it has recently updated his knowledge or not. We separate it from other relations because, contrarily to other, it does not preserve the local memory, only the one of the alive processes, as formalized by following propositions.

Proposition 9. *The strong equivalence $\cong$ of traces induces an operational equivalence: two equivalent interleaving traces starting from the same initial state lead to the same final state.*

Proof. Direct verification that for all the generating relations of Definition 8 for $\cong$, we have for any π, $[\![T]\!]_\pi = [\![T']\!]_\pi$. For instance, the case of the relation $u_j u_i = u_i u_j$, with $i \neq j$, is

$$\begin{aligned} [\![u_j u_i]\!]_\pi(l, m) &= [\![u_i]\!]_\pi \circ [\![u_j]\!]_\pi(l, m) = [\![u_i]\!]_\pi(l, m[j \leftarrow \pi_{u_j}(l_j)]) \\ &= (l, m[j \leftarrow \pi_{u_j}(l_j)][i \leftarrow \pi_{u_i}(l_i)]) = (l, m[i \leftarrow \pi_{u_i}(l_i)][j \leftarrow \pi_{u_j}(l_j)]) \\ &= [\![u_j]\!]_\pi(l, m[i \leftarrow \pi_{u_i}(l_i)]) = [\![u_j]\!]_\pi \circ [\![u_i]\!]_\pi(l, m) \\ &= [\![u_i u_j]\!]_\pi(l, m) \end{aligned}$$

Other cases are similar. □ □

Remark 10. Note that the relation $s_i s_i \cong s_i$ is not valid, in the sense that previous proposition would not be true if we added it to the definition of strong equivalence: it is easy to come up with a protocol for which the local memory is modified each time a scan is performed.

Lemma 11. *Given two traces T and T' such that $T \approx T'$, we have* $\mathrm{dead}(T) = \mathrm{dead}(T')$ *and* $\mathrm{alive}(T) = \mathrm{alive}(T)$.

Proof. Direct verification that the set of dead processes is preserved under the relations of Definition 8 for $\approx$. □ □

Given a set $I \subseteq [n]$ of indices with $I = \{i_1, \ldots, i_k\}$ and a memory $l \in \mathcal{V}^n$, we extend previous notation and write $l[I \leftarrow \bot] = l[i_1 \leftarrow \bot] \ldots [i_k \leftarrow \bot]$.

Proposition 12. *The equivalence $\approx$ on traces preserves global memory, and local memory of alive traces: given two traces T' and T'' such that $T' \approx T''$, and memories such that $(l', m') = [\![T']\!]_\pi(l, m)$ and $(l'', m'') = [\![T'']\!]_\pi(l, m)$, writing $I = \mathrm{dead}(T') = \mathrm{dead}(T'')$, we have $m' = m''$ and $l'[I \leftarrow \bot] = l''[I \leftarrow \bot]$.*

Proof. Direct verification as in the proof of Proposition 9. □ □

Remark 13. This approach using monoid presentations for specifying the execution and failure model, seems to be quite general. For instance, the monoid corresponding to atomic read-write is given as follows. The alphabet is now $\mathcal{A} = \left\{r_i^j, w_i^j, d_i\right\}$ where r_i^j (resp. w_i^j) corresponds to the i-th process reading from (resp. writing to) the memory cell j, with the expected relations.

1.2.2 Normal forms for traces up to equivalence

In order to handle traces up to equivalence, it is sometimes convenient to use rewriting systems, which provide canonical representatives for equivalence classes and allow one to decide efficiently whether two traces are equivalent or not. These can be defined by orienting the relations defining equivalence in order to obtain rewriting rules: in the case where the obtained rewriting systems are convergent (i.e. confluent and terminating), the normal forms are canonical representatives. We study here some possible such orientations of the rules as well as the associated normal forms. This section is quite independent of the rest of the paper and can be skipped by readers not familiar with rewriting theory (see [3, 5] for an introduction to the subject).

Proposition 14. *The following rewriting system on the alphabet $\mathcal{A}$ is terminating and confluent:*

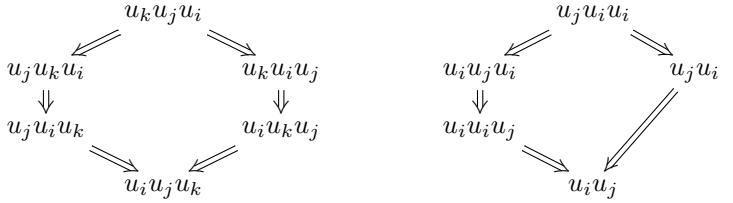
$$\begin{array}{lll} u_j u_i \Rightarrow u_i u_j & s_j s_i \Rightarrow s_i s_j & u_i u_i \Rightarrow u_i \\ d_i u_i \Rightarrow d_i & d_i s_i \Rightarrow d_i & d_i d_i \Rightarrow d_i \\ d_{i'} u_i \Rightarrow u_i d_{i'} & d_{i'} s_i \Rightarrow s_i d_{i'} & d_j d_i \Rightarrow d_i d_j \end{array} \tag{2}$$

for every $i, i', j \in [n]$ with $i < j$ and $i \neq i'$. It is thus a convergent presentation of the monoid of traces up to strong equivalence.

Proof. The termination can easily be shown by remarking that the rewriting system decreases the number of actions, puts d_i actions at the end and puts actions with low indices first. Confluence is established by checking confluence of critical pairs, which are as follows, for $i < j < k$ and $i \neq i'$ and $j \neq j'$:

$$\begin{array}{lllll} u_k u_j u_i & u_j u_i u_i & u_j u_j u_i & d_j u_j u_i & d_{j'} u_j u_i \\ s_k s_j s_i & d_j s_j s_i & d_{j'} s_j s_i & u_i u_i u_i & d_i u_i u_i \\ d_{i'} u_i u_i & d_i d_i u_i & d_j d_i u_i & d_i d_i s_i & d_j d_i s_i \\ d_i d_i d_i & d_{i'} d_{i'} u_i & d_{i'} d_{i'} s_i & d_j d_j d_i & d_j d_i d_i \\ d_j d_i u_{i'} & d_j d_i s_{i'} & d_k d_j d_i & & \end{array}$$

For instance, the confluence of the two first critical pairs is given by



Other cases are similar. □ □

The above convergent rewriting system is not the only possible one. Apart from the arbitrary ordering of processes, it tends to make processes die as late as possible. In order to illustrate this, we briefly investigate another possible orientation of the rules where processes die as early as possible. There is a corresponding convergent rewriting system, although with an infinite number of rules:

Proposition 15. *The following rewriting system on $\mathcal{A}^*$ forms a convergent presentation of the monoid of traces up to strong equivalence:*

$$\begin{array}{ccc} u_j u_i \Rightarrow u_i u_j & s_j s_i \Rightarrow s_i s_j & u_i u_i \Rightarrow u_i \\ d_i T u_i \Rightarrow d_i T & d_i T s_i \Rightarrow d_i T & d_i T d_i \Rightarrow d_i T \\ u_{i'} d_i \Rightarrow d_i u_{i'} & s_{i'} d_i \Rightarrow d_i s_{i'} & d_j d_i \Rightarrow d_i d_j \end{array}$$

for every $i, i', j \in [n]$ with $i < j$ and $i \neq i'$, and trace $T \in \mathcal{A}^*$ which does not contain any action from the process i.

Proof. The presentation can be obtained by a suitable Knuth-Bendix completion process. It can also be shown directly that the new relations are derivable and that the rewriting system is terminating and has confluent critical pairs. □ □

From previous proposition we easily deduce that in each equivalence class there is a representative such that, once the process P_i has died, it does not perform any further action. Moreover, the above axiomatization of equivalence gives rise to the same equivalence relation when applied to those representatives only. In the following, we will only consider representatives satisfying this condition.

Lemma 16. *Given an execution trace of the form $T \cdot d_i \cdot T'$, we have*

$$T \cdot d_i \cdot T' \quad \approx \quad T \cdot d_i \cdot \mathrm{proj}_{\neg i}(T')$$

A representative of a trace is called strongly properly dying *when it contains no action of process i after an action d_i for every $i \in [n]$. Two properly dying representatives are equivalent if and only if they are equivalent by applying the relations of Definition 8 between properly dying traces only.*

Proof. The fact that any trace is equivalent to a properly dying one can be shown by rewriting it using the rules of the second line of Proposition 15. The last part of the proposition follows from the fact that the axiomatization given in Proposition 15 is convergent and noticing that the rules preserve the property of being properly dying. □ □

Similar, results can be obtained for traces up to (non-strong) equivalence.

Proposition 17. *The monoid of traces up to equivalence admits a convergent presentations obtained either*

1. *by adding to the rewriting system of Proposition 14 the rules*

$$s_i T d_i \Rightarrow T d_i$$

for $i \in [n]$ and $T \in \mathcal{A}^*$ a trace not containing actions from process i,

2. *by adding to the rewriting system of Proposition 15 the rules*

$$s_i d_i \Rightarrow d_i$$

for $i \in [n]$.

Note that in both cases the rewriting systems are infinite. A generalization of Lemma 16 can also be shown:

Lemma 18. *A trace is called* properly dying *when*

- *it is strongly properly dying (it contains no action of process* i *after a* d_i*),*
- *the action of process* i *preceding an action* d_i *is not* s_i*.*

Every class of traces up to equivalence contains a properly dying one, and equivalence faithfully restricts to those traces.

The normal forms of these rewriting systems can be characterized as regular language (as for any string rewriting system) of which an explicit description can be given. Because those are much simpler in the case of well-bracketed traces, see Section 1.2.3, we will only describe them in this case, which is the one that we will use in this paper. However, a characterization similar to the one of Proposition 24 can be provided. In the following, we will be mostly interested in traces up to equivalence (by opposition to strong equivalence).

1.2.3 Well-bracketed and numbered traces

An action u_i can be thought of as an "opening bracket", and s_i or d_i as a "closing bracket". This suggests introducing the following class of words, which we will be mostly interested in the following.

Definition 19. A trace $T \in \mathcal{A}^*$ is *well-bracketed* when for every $i \in [n]$ we have $\mathrm{proj}_i(T)$ in the regular language $(u_i s_i)^*(\varepsilon + u_i d_i)$. The notion of well-bracketed infinite trace is defined similarly.

Note that a well-bracketed trace is necessarily properly dying. Also note that, with our definition, a process cannot be immediately dying (the trace d_i is not well-bracketed): we could incorporate this at the cost of adding many particular cases, moreover a process which is dying immediately can also be modeled as having $\perp$ as initial memory, which we will use when defining tasks. Usual characterizations of well-bracketed words extend to our case. For instance, one can define the set of processes which have updated, but not scanned yet as follows. We write $\wp([n])$ for the set of subsets of $[n]$.

Definition 20. We write $\mathrm{updated} : \mathcal{A}^* \to \wp([n])$ for the function defined by induction on traces by

- $\mathrm{updated}(\varepsilon) = \emptyset$,
- $\mathrm{updated}(Tu_i) = \mathrm{updated}(T) \cup \{i\}$ whenever $i \notin \mathrm{updated}(T)$,
- $\mathrm{updated}(Ts_i) = \mathrm{updated}(T) \setminus \{i\}$ whenever $i \in \mathrm{updated}(T)$,
- $\mathrm{updated}(Td_i) = \mathrm{updated}(T) \setminus \{i\}$ whenever $i \in \mathrm{updated}(T)$.

Because of the side conditions, the above function is not defined on every trace, and one can show:

Lemma 21. *A trace $T \in \mathcal{A}^*$ is well-bracketed if and only if*

1. $\mathrm{updated}(T)$ *is well-defined,*
2. $\mathrm{updated}(T) = \emptyset$, *and*
3. T *is strongly properly dying.*

The notion of equivalence restricts to the class of well-bracketed words as follows.

Proposition 22. *The equivalence on well-bracketed traces can be axiomatized by the following relations:*

$$\begin{array}{lll} u_j u_i \approx u_i u_j & s_j s_i \approx s_i s_j & d_j d_i \approx d_i d_j \\ d_{i'} u_i \approx u_i d_{i'} & d_{i'} s_i \approx s_i d_{i'} & \end{array} \tag{3}$$

for $i, i', j \in [n]$ with $i \neq j$ and $i \neq i'$. Moreover, the rewriting system obtained by orienting the relations from left to right when $i < j$ is convergent.

Proof. Suppose given two equivalent traces T and T'. By Proposition 14 they rewrite to a common normal form using the rewriting system (2). The relations (3) are the only one of (2) which can be applied to a well-bracketed trace and the confluence of (2) implies the one of (3), oriented as described above. □ □

Remark 23. As in Proposition 15, there is a variant of the orientations of rules where processes die as early as possible, which is given by

$$\begin{array}{lll} u_j u_i \Rightarrow u_i u_j & s_j s_i \Rightarrow s_i s_j & d_j d_i \Rightarrow d_i d_j \\ u_{i'} d_i \Rightarrow d_i u_{i'} & s_{i'} d_i \Rightarrow d_i s_{i'} & \end{array}$$

for $i, i', j \in [n]$ with $i < j$ and $i \neq i'$.

The normal forms for the rewriting system of Proposition 22 can be characterized as follows (similar normal forms could be given for other rewriting system, but this one is by far the most manageable one). Given a set $I \subseteq [n]$ of process indices with $I = [i_1, \ldots, i_k]$, where $i_1 < \ldots < i_k$, we write $u_I = u_{i_1} \ldots u_{i_k}$, and similarly for other actions.

Proposition 24. *The normal forms for the rewriting system of Proposition 22 on well-bracketed traces are of the form*

$$u_{I_1} s_{J_1} u_{I_2} s_{J_2} \ldots u_{I_k} s_{J_k} d_K \tag{4}$$

where the sets of indices $I_i, J_i, K \subseteq [n]$ are such that, for every $i \in [n]$,

1. *opening brackets were closed:*

$$I_i \quad \subseteq \quad [n] \setminus U_{i-1}$$

2. *closing brackets were open:*

$$J_i \quad \subseteq \quad U_{i-1} \cup I_i$$

3. dying brackets were open:

$$K \quad \subseteq \quad U_k$$

4. all the brackets are closed:

$$U_k \setminus K \quad = \quad \emptyset$$

where U_i is defined by induction on i by $U_0 = \emptyset$ and

$$U_{i+1} \quad = \quad (U_i \cup I_i) \setminus J_i$$

Proof. It is easy to show by induction on i that $U_i = \text{updated}(u_{I_1} s_{I_1} \dots u_{I_i} s_{I_i})$, from which it can be deduced that words of the form (4) are well-bracketed. Moreover, none of the rules (3) applied, and therefore those are in normal form. Finally, every well-bracketed trace will be put in this form by the rewriting system (3): informally, the rules of the second line put all the d_i in the end, and the rules of the first line order blocks of u_i (resp. s_i, resp. d_i) by increasing order of indices. □ □

Remark 25. The two last condition can of course be replaced by the only condition $K = U_k$, but our formulation makes it clearer the contribution of both inclusions.

In an execution trace, in order to distinguish between multiple instances of a same action, we will sometimes write u_i^p for the p-th occurrence of u_i in a trace T, and similarly for s_i: we call this a *numbered action*. A *numbered trace* is a trace where all the actions are decorated with their occurrence number. We generally omit the occurrence number for d_i since they occur at most once in well-bracketed traces. Notice that in a trace of the form Tu_iu_iT', considered up to equivalence, replacing u_iu_i by u_i will force us to renumber all the actions u_i in T'. For this reason, we will restrict to well-bracketed traces for which the axiomatization of Definition 8 can be reformulated as follows on numbered traces.

Proposition 26. *The equivalence of Proposition 22 corresponds to the following one on numbered well-bracketed traces:*

$$u_j^q u_i^p \approx u_i^p u_j^q \qquad s_j^q s_i^p \approx s_i^p s_j^q \qquad d_j d_i \approx d_i d_j$$
$$d_j^q u_i^p \approx u_i^p d_j^q \qquad d_j s_i^p \approx s_i^p d_j$$

for every $i, j \in [n]$ with $i \neq j$ and $p, q \in \mathbb{N}$. A convergent presentation can be obtained by considering the rewriting system whose rules rewrite the left members of the above equivalences to the corresponding right members, when $i < j$.

Remark 27. Note that two equivalent numbered well-bracketed traces contain exactly the same numbered actions since the relations preserve those.

The numbering does not bring new information on traces since it can always be computed in an unambiguous way. Therefore, we will allow ourselves to seamlessly switch between numbered and non-numbered traces. A statement similar to the above lemma of course holds if we further restrict to non-dying traces: in this case, only the relations of the first line are required. The expected properties hold for numbered well-bracketed traces, and follow easily from Lemma 21. They will be implicitly used in the following:

Lemma 28. *Given a numbered well-bracketed trace T,*

- *if the action u_i^p occurs in T then either the action s_i^p or the action d_i occurs afterward in T,*
- *given actions u_i^p and s_i^q occurring in T, with $p \leq q$, s_i^q occurs after u_i^p,*
- *given actions u_i^p and u_i^q occurring in T, with $p < q$, the actions s_i^r with $p \leq r < q$ occur in between,*
- *given actions s_i^p and s_i^q occurring in T, with $p < q$, the actions u_i^r with $p \leq r < q$ occur in between.*

Finally, we provide a convenient characterization of the equivalence of numbered well-bracketed traces: two such traces are equivalent when the relative positions of updates with respect to scans scans are the same. Given a numbered well-bracketed trace T and numbered actions a and b occurring in T, we write $a \leq_T b$ whenever the action a occurs before b in T: this relation is relation is a total order on the actions of T.

Proposition 29. *Suppose given two numbered well-bracketed traces T and T' with the same set of numbered actions. Then T and T' are equivalent if and only if*

$$s_j^q \leq_T u_i^p \qquad \textit{iff} \qquad s_j^q \leq_{T'} u_i^p \tag{5}$$

for every process numbers $i, j \in [n]$ with $i \neq j$, and round numbers p and q.

Proof. The left-to-right implication is obtained by induction on the number of equivalence steps between T and T', and then by examining each equivalence step. We now show the right-to-left implication. By the convergent rewriting system of Proposition 26, the numbered traces T and T' rewrite to their respective normal forms $\hat{T}$ and $\hat{T'}$. From the previous implication, we deduce that, for every process numbers $i \neq j$ and round numbers p and q, $s_j^q \leq_{\hat{T}} u_i^p$ iff $s_j^q \leq_{\hat{T'}} u_i^p$; and by contraposition, we also have that $u_i^p \leq_{\hat{T}} s_j^q$ iff $u_i^p \leq_{\hat{T'}} s_j^q$. Finally, by Lemma 28 the ordering of actions of a given process is also the same in $\hat{T}$ and $\hat{T'}$. Thus the traces $\hat{T}$ and $\hat{T'}$ only differ by commuting some consecutive actions of the form u_i^p (resp. s_i^p, resp. d_i), but since their relative order is fixed in a normal form (they are sorted by increasing order of process number), we have $\hat{T} = \hat{T'}$. One can also observe more directly that the relations between scans an update of (5) determine a unique normal form of the form given by Proposition 24. Finally, since a trace is equivalent to its normal form, we conclude that $T \approx T'$. □ □

2 Solving tasks

2.1 Decision tasks and protocols

We are going to consider the possibility of solving a particular task with an asynchronous protocol. A task is formalized as a relation expressing, for each possible input, the acceptable outputs from a protocol. Since we are considering a setting where processes may fail, the tasks have to take this in account. A process which has never taken part of the computation (or has died immediately)

is represented here as one having $\bot$ in its initial local memory. Moreover, if a process dies during the execution, we consider that task specifications are such that a process that never chooses an output does not constrain the outputs for processes that do.

Definition 30. A *task* Θ is a non-empty relation $\Theta \subseteq \mathcal{I}^n \times \mathcal{O}^n$ such that for every $(l, l') \in \Theta$ and $i \in [n]$,

- $l_i = \bot$ if and only if $l'_i = \bot$,
- there exists $l'' \in \mathcal{O}^n$ such that $(l, l'') \in \Theta$ and $(l[i \leftarrow \bot], l''[i \leftarrow \bot]) \in \Theta$.

The *domain* of a task Θ is

$$\operatorname{dom}\Theta \quad = \quad \{l \in \mathcal{I}^n \mid \exists l' \in \mathcal{O}^n, (l, l') \in \Theta\}$$

and its *codomain* is

$$\operatorname{codom}\Theta \quad = \quad \{l' \in \mathcal{O}^n \mid \exists l \in \mathcal{I}^n, (l, l') \in \Theta\}$$

Remark 31. As one of the referees observed, in our definition, a "consensus" task where one process starts with 1 and eventually dies before deciding, and all other processes start with 0 and eventually decide 1, would be incorrect. In more standard definitions of task solvability, processes that would start their execution would not fail, hence our definition of task is slightly more general.

Example 32. In the *binary consensus* problem each process starts with a value in $\{0, 1\}$ and should end in the same set, thus $\mathcal{I} = \mathcal{O} = \{0, 1, \bot\}$, in such a way that in the end all the values chosen by the different processes are the same, and chosen among the initial values of the alive processes. For instance, with $n = 2$, the corresponding task is

$$\Theta \quad = \quad \{(\bot\bot, \bot\bot), (b\bot, b\bot), (\bot b, \bot b), (bb', bb), (b'b, bb) \mid b, b' \in \{0, 1\}\}$$

In the case $n = 2$, we can also consider the variant called *binary quasi-consensus*, which restricts the output so that it cannot happen that p_1 decides 0 and p_0 decides 1 at the same time: the corresponding task is

$$\{(\bot\bot, \bot\bot), (b\bot, b'\bot), (\bot b, \bot b'), (bb', cc') \mid b, b', c, c' \in \{0, 1\}, c \neq 1 \vee c' \neq 0\}$$

Definition 33. A protocol π *solves* a task Θ when for every $l \in \operatorname{dom}\Theta$, and well-bracketed infinite sequence of actions $T \in \mathcal{A}^\omega$ which is *fair* (i.e. the projection on $\mathcal{A}_i$ is infinite or contains d_i for each $i \in [n]$) there exists a finite prefix T' of T such that $(l[I \leftarrow \bot], l'[I \leftarrow \bot]) \in \Theta$, where $I = \operatorname{dead}(T)$ and l' is the local memory and m' is the global memory after executing T', i.e. $(l', m') = [\![T']\!]_\pi(l, \bot^n)$.

Given a task Θ, a memory state (l, m) is *reachable* when $(l, m) = [\![T]\!]_\pi(l', \bot^n)$ for some finite execution trace T and $l' \in \operatorname{dom}\Theta$.

It can be shown [28] that, w.r.t. task solvability, the most important case is the following one:

Definition 34. A task Θ has *standard input* (or is *inputless*) when $\operatorname{dom}\Theta$ contains only the memory l such that $l_i = i$, and its "faces", i.e. memories of the form $l[I \leftarrow \bot]$ for some $I \subseteq [n]$.

For simplicity we will do so in Section 6. All other cases can be deduced from this one by suitably renaming the indices and gluing multiple copies. Given an execution trace T, we simply write $[\![T]\!]_\pi$ instead of $[\![T]\!]_\pi(l, \bot^n)$, were l is the standard input.

2.2 Variants of the execution model

Many variants of the execution model are considered in the literature, in order to tame the combinatorial complexity of the execution traces. For instance, we have already seen in Lemma 5 that we can restrict, without loss of generality to full-disclosure protocols. Here, we briefly mention further possible classical restrictions, and refer to [31, 26] for details (in particular for the proof that they do not restrict solvability). First, it can be shown that one can consider traces which are well-bracketed as the only possible executions:

Proposition 35. *One can, without changing task solvability, restrict to protocols which operate on well-bracketed traces only: those are called* well-bracketed protocols*.*

In the well-bracketed setting, a *round* of a process P_i is a sequence of actions $u_i s_i$, or $u_i d_i$, and we write $r_i \in \mathbb{N}$ for the number of rounds executed by a process P_i. For instance, in the trace $u_0 s_0 u_0 u_1 s_1 s_0$ we have $r_0 = 2$ and $r_1 = 1$. One can suppose that all the rounds of the process are synchronized, thus forbidding traces such as the previous one, where the process P_0 starts its second round (the second u_0) before P_1 has performed its first round. A protocol is *iterated* if the loops of the various processes are synchronized: no process starts its $(k+1)$-th round before every process has ended its k-th round or is dead.

Proposition 36. *Restricting to iterated well-bracketed protocols does not restrict task solvability.*

In a well-bracketed execution, rounds are either in "parallel" (such as in the trace $u_0 u_1 s_0 s_1$) or in "sequence" (such as in the trace $u_0 s_0 u_1 s_1$). A protocol is *immediate snapshot* when in a parallel execution, all the updates of parallel rounds are performed concurrently, and then all the scans are. Formally, this means that we restrict to the executions such that after a scan has been performed, no update can be performed unless no other scan can be performed:

Definition 37. A well-bracketed trace $T \in \mathcal{A}^*$ is *immediate snapshot* when for every prefix of the form $T' s_i T'' u_j$, where T'' contains only actions of the form d_k, one has $\mathrm{updated}(T' s_i T'') = \emptyset$.

For instance, with three processes, the execution $u_0 u_1 s_1 s_0 u_2 s_2$ is immediate snapshot, but $u_0 u_1 s_0 u_2 s_1 s_2$ is not: after the prefix $u_0 u_1 s_0$ the scan s_1 can be performed so that doing u_2 is not allowed.

Proposition 38. *Restricting to immediate snapshot executions does not affect protocol solvability.*

In the following, unless otherwise stated, we only restrict to protocols which are full information and well-bracketed, and explicitly state so if we consider other of the restrictions mentioned above.

2.3 Views and the view protocol

Of particular interest is the following, very general, protocol:

Definition 39. The *view protocol* $\pi^{\lhd}$ is the full-disclosure (well-bracketed) protocol such that $\pi^{\lhd}_{s_i}(x, m) = \langle x, \langle m \rangle \rangle$ 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. Because, it remembers all history and shares it with others (it is full-disclosure), this protocol is often called a *full-information* protocol. This protocol will allow us to formulate a definition of view (Definition 42), which is shown to coincide with the usual one in Section 6. In order to simplify notations, we write $\langle x, m\rangle$ instead of $\langle x, \langle m\rangle\rangle$ in the following (this notation never brings in ambiguities).

Example 40. For instance, with two processes, consider the trace

$$u_0u_1s_1u_1s_0s_1u_0s_0$$

Writing $\begin{array}{|c|c|}\hline l_0 & l_1 \\ \hline m_0 & m_1 \\ \hline \end{array}$ for the (local and global) memory, the memory will evolve as follows when the trace is executed:

$$\begin{array}{|c|c|}\hline 0 & 1 \\ \hline \bot & \bot \\ \hline \end{array} \xrightarrow{u_0} \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 0 & \bot \\ \hline \end{array} \xrightarrow{u_1} \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 0 & 1 \\ \hline \end{array} \xrightarrow{s_1} \begin{array}{|c|c|}\hline 0 & \langle 1,01\rangle \\ \hline 0 & 1 \\ \hline \end{array} \xrightarrow{u_1} \begin{array}{|c|c|}\hline 0 & \langle 1,01\rangle \\ \hline 0 & \langle 1,01\rangle \\ \hline \end{array} \xrightarrow{s_0} \begin{array}{|c|c|}\hline \langle 0,0\langle 1,01\rangle\rangle & \langle 1,01\rangle \\ \hline 0 & \langle 1,01\rangle \\ \hline \end{array}$$

$$\xrightarrow{s_1} \begin{array}{|c|c|}\hline \langle 0,0\langle 1,01\rangle\rangle & \langle\langle 1,01\rangle, 0\langle 1,01\rangle\rangle \\ \hline 0 & \langle 1,01\rangle \\ \hline \end{array}$$

$$\xrightarrow{u_0} \begin{array}{|c|c|}\hline \langle 0,0\langle 1,01\rangle\rangle & \langle\langle 1,01\rangle, 0\langle 1,01\rangle\rangle \\ \hline \langle 0,0\langle 1,01\rangle\rangle & \langle 1,01\rangle \\ \hline \end{array}$$

$$\xrightarrow{s_0} \begin{array}{|c|c|}\hline \langle\langle 0,0\langle 1,01\rangle\rangle, \langle 0,0\langle 1,01\rangle\rangle\langle 1,01\rangle\rangle & \langle\langle 1,01\rangle, 0\langle 1,01\rangle\rangle \\ \hline \langle 0,0\langle 1,01\rangle\rangle & \langle 1,01\rangle \\ \hline \end{array}$$

The fact that this protocol is the "most general" one, can be formalized as follows.

Proposition 41. *The view protocol $\pi^{\triangleleft}$ is initial in the category of full-disclosure well-bracketed protocols.*

Proof. Suppose given a protocol π. We have to construct a morphism $\phi : \pi^{\triangleleft} \to \pi$ and show that this is the only possible such morphism. Notice that since we are considering morphisms between full-disclosure protocols, the diagram on the left of (1) reduces to the fact that $\phi_i = \phi'_i$ for every $i \in [n]$. By Remark 3, we only have to define ϕ on reachable states, i.e. those of the form $[\![T]\!]_{\pi^{\triangleleft}}$ for some execution trace T. A local memory of the i-th process is thus either of the form i, in which case we must have $\phi_i(i) = i$ because morphisms are required to preserve initial values, or of the form $\langle x, m\rangle$, in which case we should have $\phi_i(\langle x, m\rangle) = \pi'_{s_i}(\phi_i(x), (\prod_i \phi_i)(m))$ by the second diagram of (1), which is well-defined by induction since the states x and m have been produced by prefixes of T. Conversely, suppose given a reachable memory x for the process i. Since the memory is reachable there exists a trace T such that $l_i = x$ with $l = [\![T]\!]_{\pi^{\triangleleft}}$, and we define $\phi_i(x) = l'_i$ where $(l', m') = [\![T]\!]_{\pi}$. By definition of $[\![T]\!]_{\pi}$, it satisfies the above requirements for the uniqueness part of the proof. We only have to check that it does not depend on the choice of the trace T. By Proposition 9, it does not depend on the representative of T in its equivalence class. Showing that it only depends on l_i (which we will call the i-view of T in the following) is a much more delicate task, for which we will need the tools of Section 5. We will see in Proposition 84 that T leads to $\langle x, m\rangle$ for process i if and only if its

i-restriction is a given trace T' (i.e. T contains T' as a particular subtrace) and that the local memory for process i resulting from the execution of the trace T only depends on T' (Proposition 83). □ □

This property says that protocols π are in bijection with morphisms $\phi : \pi^{\lhd} \to \pi$. This is akin to the use of *generic protocols in normal form* [28], where protocols only exchange their full history of communication for a fixed given number of rounds, and then apply a local decision function (corresponding to our morphism ϕ). Those protocols are moreover restricted to traces which are well-bracketed, see below. For this reason, we will be satisfied with describing the potential sets of histories of communication between processes, without having to encode the decision values: this is the basis of the geometric semantics of Section 3. 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 [28, 26].

Definition 42. Given an execution trace T in which process P_i is alive, the *view* of the i-th process is l_i, where $(l, m) = [\![T]\!]_{\pi^{\lhd}}$ is the state reached by the view protocol after the execution of T, also called an *i-view*.

3 Directed geometric semantics

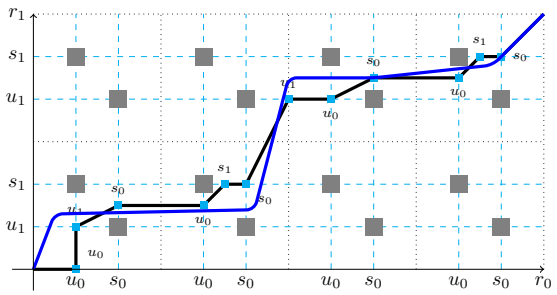
In this section, we give an alternative semantics to atomic snapshot protocols, using a geometric encoding of the state space, together with a notion of "time direction". One of the most simple settings in which this can be performed is the one of pospaces [32, 17]: a *pospace* is a topological space $\mathbb{X}$ 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 [10, 11]. 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 sound and complete with respect to the interleaving semantics, in Section 3.4 : to dipaths, we will associate interleaving traces and show that equivalence of dipaths give rise to equivalence of interleaving traces. Then, we will associate to an interleaving trace a dipath such that its associate trace is equivalent to the starting one.

3.1 Dipaths and dihomotopies

A *dipath* (or *directed path*) in a pospace $(\mathbb{X}, \leq)$ is a continuous map $\alpha : [0, 1] \to \mathbb{X}$ which is 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 $\alpha \cdot \alpha'$ such that $\alpha \cdot \alpha'(x)$ is $\alpha(x)$ (resp. $\alpha'(2x - 1)$) when $x \leq 0.5$ (resp. $x \geq 0.5$).

The continuous setting allows us to use the classical concepts of (endpoint-preserving) (di)homotopy, which is the natural notion of equivalence between

paths with compatible endpoints, and to use some tools from algebraic topology to derive properties of protocols (and more generally programs [19]). 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*, 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. For instance, the following figure pictures two dipaths that are dihomotopic in the geometric space $\mathbb{X}^2_{(4,2)}$ representing protocols with 2 processes, 4 rounds for process 0 and 2 rounds for process 1.



(6)

3.2 The case of fault-free processes

As two traces are equivalent only when they have the same set of actions, we focus on well-bracketed traces with the same number of rounds. Therefore, we will consider pospaces associated to a number of rounds and inextendible dipaths in this pospaces.

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 iterated 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

$$\mathbb{X}^n_{(r)} \quad = \quad \prod_{i\in[n]} [0, r_i] \setminus \bigcup_{\substack{i,j\in[n] \\ p\in[r_i],\ q\in[r_j]}} U^p_i \cap S^q_j \tag{7}$$

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 $0 < u < s < 1$,
- the space

$$U^p_i \quad = \quad \left\{ x \in \prod_{i\in[n]} [0, r_i] \;\middle|\; x_i = p + u \right\}$$

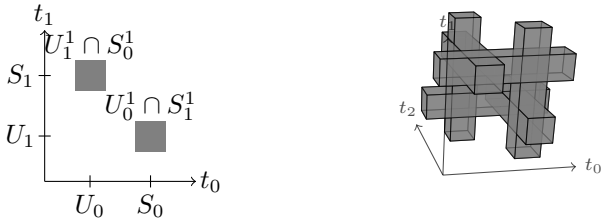
stands for the region where the i-th process updates the global memory into its local memory for the p-th time,

- the space

$$S_j^q \quad = \quad \left\{ x \in \prod_{i\in[n]} [0, r_i] \;\middle|\; x_j = q + s \right\}$$

stands for the region where the i-th process scans the global memory with its local memory for the q-th time.

The meaning of (7) 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 except when it is in $U_i^p \cap S_j^q$ (for $i, j \in [n]$ and $p \in [r_i]$, $l \in [r_j]$). These forbidden states are precisely the states for which there is a `scan` and `update` conflict. Namely, states in $U_i^p \cap S_j^q$ are states for which process P_i updates (for the p-th time) while process P_j scans (for the q-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 on the left of (8) for two processes with one round each, and in (6) for two processes with several rounds each. Notice that the holes should be points since the operations are atomic. Here they are depicted as squares instead of points to improve the visibility on the diagram. In higher-dimensions, the holes exhibit a complicated combinatorics. For instance, for three processes, and one round each, as in the right diagram of (8), shows forbidden regions that intersect one another.



(8)

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 on the left of (8). 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 on the right of (8).

3.3 Processes with faults

The model of fault we are studying is the one of crash failures (which are dying failures). At any point in time, any number of processes P_i can crash, stopping its execution abruptly right after local time t_i. In terms of geometric semantics, this amounts to forbidding all states $(x_0, \ldots, x_{n-1})$ in $\mathbb{R}^n$ with $x_i > t_i$.

There are two kinds of times at which a process can fail. The first is when it fails even before doing its first update. The second one is when a process fails after its last update. Notice that the relative position of the fails to the scans is not relevant as nobody else will see the effect of the scan and the concerned faulty process will not help with solving the task. So that we can consider that the concerned process halt just before scanning. This implies to erase the hole due to the conflict between this scan and the updates of other processes and stop the faulty process at the corresponding round. Let us denote

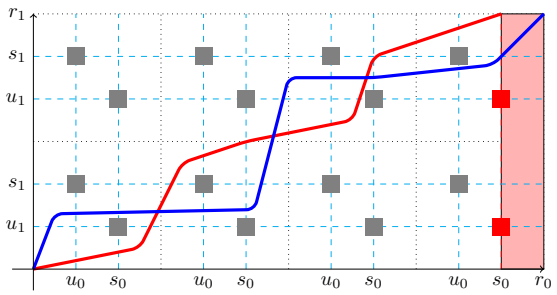
$$\begin{aligned} D_j^0 &= \prod_{i<j}[0,r_i] \times \{0\} \times \prod_{i>j}[0,r_i] \\ D_j^{p+1} &= \prod_{i<j}[0,r_i] \times [0,p+1+s[\times \prod_{i>j}[0,r_i] \end{aligned}$$

D_j^0 corresponds to the failure of the jth process before it first update and D_j^{p+1} corresponds to the failure of the jth process after its $p+1$th update. Now, let F be the set of faulty processes. Then the corresponding pospace is

$$\mathbb{X}^n_{(r),F} = \mathbb{X}^n_{(r)} \cap \left(\bigcup_{j \in F} D_j^{r_j} \right) \tag{9}$$

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

On the figure below, the blue path represents a trace with no failure whereas, the red path represent a trace with a failure of process 0 after its fourth update. The blue belongs to $\mathbb{X}^2_{(4,2)}$ and the red one belongs to $\mathbb{X}^2_{(4,2),\{0\}} = \mathbb{X}^2_{(4,2)} \cap D_0^4$ where D_0^4 is the red region. Notice that red points are excluded from $\mathbb{X}^2_{(4,2),\{0\}}$ as they were points of intersection between update and scan hyperplanes and they belong to D_0^4.



(10)

3.4 Equivalence of the standard and geometric semantics

3.4.1 From dipaths modulo dihomotopy 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.

To any inextendible dipath $\alpha : [0,1] \to \mathbb{X}^n_{(r),F}$, we associate its projection α_i on the ith coordinate and the real numbers u_i^p and s_i^p, respectively corresponding to the event "α enters an update or scan hyperplane":

$$u_i^p = \inf\{t \in [0,1] \mid \alpha(t) \in U_i^p\} \qquad s_i^p = \inf\{t \in [0,1] \mid \alpha(t) \in S_i^p\}. \tag{11}$$

Wlog, we assume that $u_i^p < s_i^p$ (indeed, any dipath can be parameterized in such a way that this condition holds without changing the graph of the dipath). To a dipath α, we associate the following interleaving trace T_α. The u_i^p and s_i^p form a finite total sub-order in $(\mathbb{R}, \leq)$, hence is isomorphic, as an order, to the order on $\{1, \dots, k\}$ for some integer k. Under this isomorphism, some $j \in \{1, \dots, k\}$ is mapped onto $a(j)$ which is one of the u_i^p or s_i^p. The trace T_α is constructed as the concatenation $a(1) \dots a(k)$ followed by the d_i for $i \in F$.

For any $i \in [n]$, since α_i is non-decreasing, the order in which α enters update or scan hyperplanes induces a total order on the actions of process i in T_α such that $u_i^p \leq_T s_i^p$. We can therefore check that T_α is well-bracketed.

Remark 43. One should keep in mind that a dipath α satisfies:

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

Lemma 44. *Let α and β be two inextendible dipaths in $\mathbb{X}^n_{(r),F}$. They intersect the update and scan hyperplanes in the same order if and only if they are dihomotopic.*

Proof. Let us first prove the left-to-right implication. 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^p and s_j^q intersect are the same for α and β. Then, the function defined by $H : x, t \mapsto t\,\alpha(x) + (1-t)\beta(x)$ is a dihomotopy. Let us prove that H takes its value in $\mathbb{X}^n_{(r),F}$, that is, for all $x, t \in [0,1]$, $H(x,t) \notin U_i^p \cap S_j^q$. Assume for instance that $u_i^p > s_j^q$. If $H(x,t) \in U_i^p$, then $H(x,t)_i = p + u$ and, since $\alpha, \beta \in \mathbb{X}^n_{(r),F}$,

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

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

Let us now prove the right-to-left implication. Let $H : [0,1] \times [0,1] \to \mathbb{X}^n_{(r),F}$ be a dihomotopy between $\alpha = H(-,1)$ and $\beta = H(-,0)$. Let u_i^p (resp. v_i^p) and s_i^p (resp. t_i^p) be the defined as in (11) for α (resp. β). Let us fix $i, j \in [n]$ and prove that, for any $p \in [r_i]$ and $l \in [r_j]$, the dipaths α and β intersect U_i^p and S_j^q in the same order. More precisely, we want to prove that:

$$u_i^p < s_j^q \qquad \text{iff} \qquad v_i^p < t_j^q. \tag{12}$$

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^p and S_j^q are parallel to the direction along which we project, H_{ij}, α_{ij} and β_{ij} are taking their values in the pospace:

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

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: $\alpha(u_i^p)_j < q+s$ iff $\beta(v_i^p)_j < q+s$. Finally, the equivalence (12) follows. □ □

Theorem 45. *Dihomotopic dipaths induce equivalent traces.*

Proof. This results from Proposition 29 which characterizes equivalent traces through the order of their update and scan actions and from preceding Lemma 44. □ □

3.4.2 From equivalence classes of interleaving traces to dipaths modulo dihomotopy

In this section, we start by showing the equivalence of the interleaving semantics modulo equivalence of interleaving traces with the geometric semantics and dihomotopy of directed paths, in the case when there are no crash failures.

To any interleaving trace T with n processes and (r) rounds, we associate a dipath α_T in $\mathbb{X}^n_{(r),F}$. This dipath accurately reflects the whole computation of T, e.g. if T' extends T, then $\alpha_{T'}$ also extends α_T. For example, the black path of (6) is the dipath associated to the trace $u_0u_1s_0u_0s_1s_0u_1u_0s_0u_0s_1s_0$: the points along it correspond to actions and the path consists of a linear interpolation between those. The dipath α_T is built by induction on the length of trace T: when T is of length 0, α_T is the constant dipath staying at the origin; when T is the concatenation of a trace T_1 with an action A, we concatenate the dipath α_{T_1} and a dipath β which is defined according to the previous actions in T_1 as in the proof of the following lemma:

Lemma 46. *Let T be a well-bracketed trace. There exists a dipath α_T in $\mathbb{X}^n_{(r),F}$ such that α_T intersects update and scan hyperplanes in the same order as in T.*

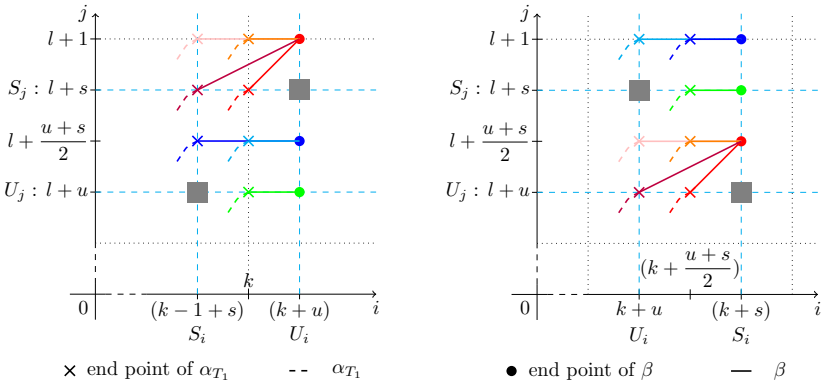
Proof. We build a (not necessarily inextendible) dipath $\alpha_T \in \mathbb{X}^n_{(r),F}$ by induction on T, such that for any $i \in [n]$, $\alpha_T(0)_i = 0$; if the last action in T is the $(p+1)$-th update of process i, then $\alpha_T(1) \in U_i^p$, that is $\alpha_T(1)_i = p+u$; if the last action in T is the $(p+1)$-th scan of process i, then $\alpha_T(1) \in S_i^p$, that is $\alpha_T(1)_i = p+s$. Moreover, if the last action of process is its

- $(p+1)$-th update, then $\alpha_T(1)_i \in \left\{p+u, p+\frac{u+s}{2}\right\}$;
- $(p+1)$-th scan, then $\alpha_T(1)_i \in \{p+s, p+1\}$.

The lemma follows indeed, if $T = T_0u_i^pT_1$, then $\alpha_{T_0u_i^p} \in U_i^p$ and similarly, if $T = T_0s_i^pT_1$, then $\alpha_{T_0s_i^p} \in S_i^p$.

First, when T is of length 0, α_T is the constant dipath staying at the origin 0. Otherwise, let $T = T_1a_i$ be the concatenation of a trace T_1 with action a_i (being

either update u_i, scan s_i or death d_i of process 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 46. Now, construct a dipath β, which is a line, as pictured on figure below,



(13)

starting at $\beta(0) = \alpha_{T_1}(1)$ and ending at $\beta(1)$. Assume i is alive in T_1.

- If a_i is an update, say the $(p+1)$-th update of process i, as partly represented on the left part of (13), by Lemma 46, since the previous action was a scan or nothing, $\alpha_{T_1}(1)_i \in \{0, p-1+s, p\}$ and we set $\beta(1)_i = p+u$. For any other process $j \neq i$, if j is alive and its the last action is its say $(q+1)$-th scan, then $\alpha_{T_1}(1)_j \in \{q+s, q+1\}$ and we set $\beta(1)_j = q+1$ (in red tones), otherwise we set $\beta(1)_j = \alpha_{T_1}(1)_j$ (in blue tones).

- If a_i is a scan, say the $(p+1)$-th scan of process i then, as represented on the right part of (13). since the action of i before was the $(p+1)$-th update, $\alpha_{T_1}(1)_i \in \left\{p+u, p+\frac{u+s}{2}\right\}$ and we set $\beta(1)_i = p+s$. For any other process j, if j is alive and its last action is its $(q+1)$-th update, then we have $\alpha_{T_1}(1)_j = \left\{q+u, q+\frac{u+s}{2}\right\}$ and we 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).

- If a_i is a death, the we set $\beta(1)_i = r_i - 1 + \frac{u+s}{2}$. For any other alive process j, if $\alpha_{T_1}(1)_j = q+u$ then, we set $\beta(1)_j = q + \frac{u+s}{2}$ otherwise, we set $\beta(1)_j = \alpha_{T_1}(1)_j$.

We then define the dipath $\alpha_{T_1 a_i} = \alpha_{T_1} \cdot \beta$. To a maximal interleaving trace T, we associate an inextendible dipath α'_T by further extending α_T: 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 $\gamma(1)_i = r_i - 1 + s$ for $i \in F$ and $\gamma(1)_i = r_i$ otherwise, the point $\gamma(1)$ being the end of all inextendible dipaths in $\mathbb{X}^n_{(r),F}$. We shall not distinguish in the sequel α'_T from α_T since we will only consider maximal interleaving traces and their inextendible counterparts. □ □

Theorem 47. *For any $T \approx T'$ well-bracketed traces, the induced dipaths α_T and $\alpha_{T'}$ are dihomotopic. Moreover, the dipath α_T, built from a well-bracketed trace T, induces a trace T_{α_T} such that $T \approx T_{\alpha_T}$.*

Proof. Indeed, by construction α_T (resp. $\alpha_{T'}$) intersects update and scan hyperplanes in the same order as T (resp. T'). Since $T \approx T'$, by Proposition 29,

α_T and $\alpha_{T'}$ intersect the update and scan hyperplanes in the same order. By Lemma 44 they are dihomotopic. For the second point, by construction (see Paragraph 3.4.1) the order of update and scans in T_{α_T} are the same as the order of intersection of α_T with update and scan hyperplanes. So that T_{α_T} and T are equal by construction. □

4 Interval orders

In this section we provide a convenient combinatorial representation of execution traces up to equivalence as interval orders, encoding the relative execution of rounds. We begin by considering the case where the processes are not dying, i.e. the traces do not contain actions of the form d_i. The usefulness of using partial order to specify concurrent objects has already been observed [16, 9]; here, we make precise the relationship with traces. We will only need basic facts, recalled here, about this notion which was introduced by Fishburn [15].

4.1 From traces to interval orders

Definition 48. 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

$$x \prec y \qquad \text{if and only if} \qquad \forall s \in I_x, \forall t \in I_y, s < t \tag{14}$$

meaning that every element of the first interval is below the second. Such a poset is called an *interval order* and a family of intervals giving rise to it is called an *interval representation* of the poset. A *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.

We use the standard terminology for posets. In particular, two elements x and y are *independent*, what we write $x \parallel y$, whenever neither $x \prec y$ nor $y \prec x$ holds. A *predecessor* of an element y is an element x with $x \prec y$ and such that there is no element x' with $x \prec x' \prec y$. A poset is often depicted by its Hasse diagram, which is the graph with the elements of the poset as vertices and there is an edge $x \to y$ whenever x is a predecessor of y, as on the left below. We do complete the Hasse diagrams that we present below by adding some arrows that come from transitivity of the order, when we feel that is is necessary for the understanding.

$$\begin{array}{ccc} x' & \leftarrow & y' \\ \uparrow & \nwarrow & \uparrow \\ x & & y \end{array} \qquad\qquad \begin{array}{ccc} 0 & \leftarrow & 1 \\ \uparrow & \nwarrow & \uparrow \\ 0 & & 1 \end{array} \tag{15}$$

In the case where we considered a labeled interval order, we picture the labels instead of the elements. For instance, the previous poset labeled by $\ell(x) = \ell(x') = 0$ and $\ell(y) = \ell(y') = 1$ will be pictured as on the right above. This can be formally justified by the fact that we consider colored interval orders up to color-preserving isomorphism: the name of the elements do not really matter, only their labels do. In fact, the elements of a colored interval order can always be named canonically as follows, which will be useful in the following (in fact, we

will generally be implicitly supposing that the colored interval orders that we manipulate are of this form).

Lemma 49. *Suppose given a colored interval order $(X, \preceq, \ell)$ and $i \in [n]$. The elements x of X such that $\ell(x) = i$ are totally ordered, with cardinal denoted r_i, and, given such an element x, its index in the chain is called its* occurrence number. *We can thus unambiguously denote by x_i^p an element of X where i is its label and p its occurrence number and therefore, up to isomorphism, we can suppose that the elements of X are of this form, i.e. that we have*

$$X \quad = \quad \{x_i^p \mid i \in [n], 0 \leq p < r_i\}$$

Example 50. The elements of the colored interval order (15) can be named as

$$\begin{array}{ccc} x_0^1 & \leftarrow & x_1^1 \\ \uparrow & \nwarrow & \uparrow \\ x_0^0 & & x_1^0 \end{array}$$

Remark 51. A purely combinatorial description of interval orders (without referring to the real line) can also be given [15]: a partial order $(X, \preceq)$ is an interval order if and only if it does not contain "$2+2$" as induced suborder, i.e. a subset $\{x, y, z, t\}$ of X with $x < y$ and $z < t$ and no more comparisons. Equivalently, for any $x, y, z, t \in X$, ($x \leq y$ and $z \leq t$) implies ($x \leq t$ or $z \leq y$). And a similar characterization can of course be given for colored interval orders.

Interval orders are now going to be used to encode execution traces, up to equivalence, of non-dying processes. The idea is that, for each numbered execution trace $a^0 \dots a^k$ with an action a^j being either of the form $u_{i_j}^{p_j}$ or $s_{i_j}^{p_j}$, we associate the interval of "global times" (on the corresponding trace) $[k, l]$ where $a^k = u_i^p$ and $a^l = s_i^p$ are corresponding update and scans (in the "bracket" system they are defining), this interval being labeled by i.

Proposition 52. *Execution traces up to equivalence of well-bracketed protocols, in which no process dies, are in bijection with colored interval orders.*

Proof. To every non-dying well-bracketed numbered trace T, we associate the interval order whose set of elements is

$$X \quad = \quad \{x_i^p \mid i \in [n], 0 \leq p < r_i\} \tag{16}$$

where r_i is the number of rounds of process i in T. It thus contains elements of the form x_i^p, with indices such that u_i^p (and thus also s_i^p) occurs in T. To an element x_i^p, we associate the interval $[m_i^p, n_i^p]$, where $m_i^p \in \mathbb{N}$ (resp. $n_i^p \in \mathbb{N}$) is the index of the occurrence of u_i^p (resp. s_i^p) in T, thus defining an interval order as in Definition 48. We label the elements by the function $\ell : X \to [n]$ such that $\ell(x_i^p) = i$. Note that $[m_i^p, n_i^p]$ is a representation of the interval order X and is such that $n_i^p < m_j^q$ iff $s_i^p \leq_T u_i^q$. So that thanks to Proposition 29, two equivalent well-bracketed traces will generate the same interval order.

Conversely, suppose given a colored interval order $(X, \preceq, \ell)$. By Lemma 49, we can suppose that the elements of X are of the form given in (16). There is an interval representation of the interval order associating to each element x_i^p an interval $[m_i^p, n_i^p] \subseteq \mathbb{R}$. Since X is finite, it is easy to see that we can

suppose that $m_i^p \neq m_{i'}^{p'}$, $m_i^p \neq n_{i'}^{p'}$ and $n_i^p \neq n_{i'}^{p'}$ whenever $i \neq i'$ or $p \neq p'$, that the m_i^p and n_i^p are integers, and that the set $M = \{m_i^p, n_i^p \mid i \in [n], 0 \leq p < r_i\}$ is an initial segment of $\mathbb{N}$. Therefore, it induces a word $a_0 a_1 \dots a_{k-1}$, with $k = \mathrm{card}(M)$, such that $a_j = u_i^p$ if $m_i^p = j$ and $a_j = s_i^p$ if $n_i^p = j$. Note that $x_i^p < x_j^q$ iff $n_i^p < m_i^q$ iff $s_i^p \leq_T u_i^q$. Thus, thanks to Proposition 29, we have that two representations of the interval order will induce equivalent traces. □ □

Example 53. The numbered trace $u_0^0 u_1^0 s_1^0 u_1^1 s_0^0 s_1^1 u_0^1 s_0^1$ induces an interval order which is $X = \left\{x_0^0, x_0^1, x_1^0, x_1^1\right\}$ with the interval representation of x_i^p given by the positions of u_i^p and s_i^p in the word:

$$x_0^0 = [0,4] \qquad x_0^1 = [6,7] \qquad x_1^0 = [1,2] \qquad x_1^1 = [3,5]$$

and therefore the corresponding poset is (15). Conversely, the poset (15) admits the above interval representation, which induces the above trace.

We finally mention a useful technical property, showing that the correspondence of Proposition 52 between traces up to equivalence and colored interval orders is compatible with restriction.

Lemma 54. *Suppose given an interval order $(X, \preceq, \ell)$ and a subset $Y \subseteq X$ which is downward and independent closed: for every $x \in Y$ and $y \in X$, $y \preceq x$ or $y \parallel x$ implies $y \in Y$. Any trace T corresponding to X by Proposition 52 is of the form $T = T'T''$ where T' is associated to the interval order Y (with order and labeling inherited from X) by the same proposition.*

Proof. Consider the last action of the form s_i^p in T such that $x_i^p \in Y$. The trace T is of the form $T = T'T''$, where the last action of T' is s_i^p and T'' does not contain actions of the form a_j^q with $x_j^q \in Y$. The trace T' cannot contain an action of the form a_j^q with $x_j^q \in X \setminus Y$: if it contained such an action, then it would contain u_j^q occurring before s_i^p, so that $x_i^p \preceq x_j^q$ and by downwards closure of Y we would have $x_i^p \in Y$ which contradicts the hypothesis. Finally, the trace T' is easily shown to correspond to Y by Proposition 52. □ □

Under the equivalence of classes of interleaving traces with dipaths modulo dihomotopy, (see Section 3.4), we know that we have a correspondence as well between the dipaths modulo dihomotopies and interval orders, that is easy to picture. For instance, to any non-faulty inextendible dipath $\alpha : [0,1] \to \mathbb{X}^n_{(r)}$, we associate an interval order $\preceq_\alpha$ on the set

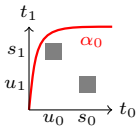
$$X^n_{(r)} \quad = \quad \{x_i^p \mid i \in [n], p \in [r_i]\}$$

where x_i^p is labeled by i and represents the interval $x_i^p = [u_i^p, s_i^p]$ where we recall that u_i^k (resp. s_i^k) corresponds to the event "α enters an update (resp. scan) hyperplane":

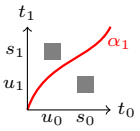
$$u_i^p = \inf\{t \in [0,1] \mid \alpha(t)_i \in U_i^p\} \qquad s_i^p = \inf\{t \in [0,1] \mid \alpha(t)_i \in S_i^p\}$$

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 $\alpha(u_i^p)_i = p + u$, $\alpha(s_i^p)_i = p + s$, hence for all $p \in [r_i]$, $u_i^p < s_i^p$ and if $p \neq 0$, $s_i^{p-1} < u_i^p$.

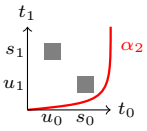
Let us give simple examples of this in dimension 2 and 3. In dimension 2, and for one round, consider the three following inextendible dipaths in $\mathbb{X}^2_{(1,1)}$:



$[u_1,s_1] \prec_{\alpha_0} [u_0,s_0]$



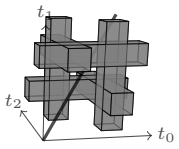
$[u_0,s_0] \|_{\alpha_1} [u_1,s_1]$



$[u_1,s_1] \prec_{\alpha_2} [u_0,s_0]$

(we are not writing the round number as upper index since we are considering here only one round). Those are representatives of the three dihomotopy classes of dipaths in this pospace. The dipath α_0, on the above left figure, 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 dipath α_2 is symmetric: the corresponding interval order is $[u_0, s_0] \prec_{\alpha_2} [u_1, s_1]$. The dipath on the middle corresponds to an execution in which the two processes are running synchronously, updating at the same time, and scanning at the same time: the corresponding interval order is $[u_0, s_0] \|_{\alpha_1} [u_1, s_1]$.

In dimension 3, there are more dipaths that one can draw. Consider, for instance, the synchronous execution of the three processes (i.e. the pospace $\mathbb{X}^3_{(1,1,1)}$), shown on the right. It corresponds to the interval order where the intervals $[u_0, s_0]$, $[u_1, s_1]$ and $[u_2, s_2]$ are not comparable. The path figured corresponds to a synchronous execution:



4.2 The effect of processes dying

The notion of interval order can be generalized in order to extend the correspondence described in Proposition 52. The idea is that we now have two kinds of intervals: those of the form $x_i^p = [u_i^p, s_i^p]$ and those of the form $y_i^p = [u_i^p, d_i]$, respectively called alive and dying intervals. We will distinguish the two by adding another kind of label to colored interval orders.

Definition 55. A *colored interval order with death* $(X, \preceq, \ell, \delta)$ consists of a colored interval order $(X, \preceq, \ell)$, in the sense of Definition 48, together with a function $\delta : X \to \{\heartsuit, \dagger\}$ indicating for each element x if it is *alive* $(\delta(x) = \heartsuit)$ or *dying* $(\delta(x) = \dagger)$, such that a dying element is maximal among those with the same label.

In the following we simply call those colored interval orders and specify when we consider "non-dying" ones.

Proposition 56. *Execution traces up to equivalence for well-bracketed protocols are in bijection with colored interval orders.*

Proof. The proof is similar to the one of Proposition 52 except that a pair u_i^p, s_i^p (resp. u_i^p, d_i) in an execution trace corresponds to an alive (resp. dying) element of the interval order. □ □

Remark 57. Note that we really need to consider traces up to equivalence (as opposed to strong equivalence) for this correspondence to hold: otherwise, we would have to distinguish between $u_0 s_0 d_0$ and $u_0 d_0$, which there is no easy way to encode in interval orders.

5 Views of interval orders

We study here the views, as introduced in Definition 42, generated by an interval order. For simplicity, we only handle the case of non-dying interval orders here.

Definition 58. Given an element x of a poset, we write $\updownarrow x$ for the set of elements which are not strictly greater than x, i.e. those which are lower than x or independent from x.

5.1 Interval orders and their views

By the correspondence given by Proposition 52, the i-view of an interval order can be defined as for traces:

Definition 59. Given a colored interval order X, its *i-view* $\llbracket X \rrbracket_i$ is defined as the i-view of T, in the sense of Definition 42, where T is the trace corresponding to X by Proposition 52.

It will be convenient, more generally, to consider the view associated to any element x_i^p of a colored interval order X: we write $\llbracket x_i^p \rrbracket$ for the local view of the i-th process after executing all the actions which the scan s_i^p can see, i.e. such that the update occurs before this scan. Formally, by Proposition 52, the colored interval suborder $\updownarrow x_i^p$ (obtained by restricting X to $\updownarrow x_i^p$) corresponds to a trace T (up to equivalence) and we define $\llbracket x_i^p \rrbracket = l_i$ where $(l, m) = [\![T]\!]_{\pi^\triangleleft}$. Notice that if we write T' for a trace corresponding to X (by Proposition 52), the trace T is a prefix of T' up to equivalence, by Lemma 54. Moreover, we recover Definition 59 as a particular case: we have $\llbracket X \rrbracket_i = \llbracket x_i^p \rrbracket$, where x_i^p is the maximal element of X labeled by i. We now show that we can reconstruct a colored interval order from its views, starting by giving an inductive characterization of the views.

Proposition 60. *Suppose given $i \in [n]$ and write x_i^p for the maximal element labeled by i of a colored interval order X (by convention $p = -1$ when there is no such element). Then the i-view $\llbracket X \rrbracket_i = \llbracket x_i^p \rrbracket$ can be computed by induction on the (well-founded) poset X by*

$$\llbracket x_i^p \rrbracket \quad = \quad \left\langle \llbracket x_i^{p-1} \rrbracket, l_0 l_1 \dots l_{n-1} \right\rangle$$

where, by convention, $\llbracket x_i^{-1} \rrbracket = i$, and $l_j = \llbracket x_j^{q-1} \rrbracket$ with x_j^q the maximal element of $\updownarrow x_i^p$ with label j, where by convention, $l_j = \bot$ when no such element exists.

Proof. By induction on the size of X and p. If $p = -1$ then the process i does not perform any action and its local memory $[\![x_i^{-1}]\!]$ is the initial one, i.e. $[\![x_i^{-1}]\!] = i$ by definition of the standard input. Otherwise, since x_i^p is maximal, the trace corresponding to $\natural\, x_i^p$ is, up to equivalence, of the form Ts_i. Writing $(l, m) = [\![T]\!]_{\pi^\triangleleft}$, we have

$$[\![Ts_i]\!]_{\pi^\triangleleft} \quad = \quad (l[i \leftarrow \langle l_i, m\rangle], m)$$

and therefore

$$[\![x_i^p]\!] \quad = \quad \langle l_i, m\rangle$$

Above, l_i is the local memory which was last modified by the action s_i^{p-1} in T. Given an element x_j^q in X with $x_j^q > x_i^{p-1}$, the action u_i^q occurs after x_i^{p-1} in T, and therefore l_i is the local memory of the i-th process after the execution of the prefix of T corresponding to $\natural\, x_i^{p-1}$ (this is a prefix by Lemma 54). By induction hypothesis, we thus have $l_i = [\![x_i^{p-1}]\!]$. We can proceed by a similar reasoning to determine m_j. Writing x_j^q for the maximal element of $\natural\, x_i^p$ with label j, the contents of m_j is the one which was written by u_j^q. Since the view protocol is full-disclosure, this value corresponds to the local memory of j-th process after the execution of s_j^{q-1} (the preceding scan). Therefore, $m_j = [\![x_j^{q-1}]\!]$. □ □

Example 61. Consider the interval order X pictured in (15) again (with elements named as in Example 50). Its 0-view can be computed as follows:

- $[\![x_1^0]\!] = \left\langle [\![x_1^{-1}]\!], [\![x_0^{-1}]\!][\![x_1^{-1}]\!] \right\rangle = \langle 1, 01\rangle$
- $[\![x_0^0]\!] = \left\langle [\![x_0^{-1}]\!], [\![x_0^{-1}]\!][\![x_1^0]\!] \right\rangle = \langle 0, 0\,\langle 1, 01\rangle\rangle$
- $[\![x_0^1]\!] = \left\langle [\![x_0^0]\!], \left\langle [\![x_0^0]\!][\![x_1^0]\!] \right\rangle \right\rangle = \langle \langle 0, 0\,\langle 1, 01\rangle\rangle\,, \langle 0, 0\,\langle 1, 01\rangle\rangle\,\langle 1, 01\rangle\rangle$

This result is precisely the one we have obtained in Example 40 by simulating the trace $u_0u_1s_1u_1s_0s_1u_0s_0$ which corresponds to the interval order $\natural\, x_0^1 = X$, see Example 50 and 53.

A number of interesting remarks can be made on the inductive definition of the view provided by the previous proposition. The views from previous rounds can be extracted by iteratively considering the first component of the view. Formally, the previous view can be recovered as follows.

Definition 62. Given a view l of the form $l = \left\langle l', l'_0 l'_1 \dots l'_{n-1} \right\rangle$, the *previous view* is $\mathrm{pr}(l) = l'$.

Lemma 63. *Given x_i^p in a colored interval order X, we have*

$$\mathrm{pr}([\![x_i^p]\!]) \quad = \quad [\![x_i^{p-1}]\!]$$

Moreover, the number of rounds executed by an action can be recovered as the number of times the previous view is defined. Formally,

Definition 64. Given a view l, we define is *occurrence number* $\mathrm{on}(l)$ by induction by

$$\mathrm{on}(l) \quad = \quad \begin{cases} 1 + \mathrm{on}(l') & \text{if } l = \left\langle l', l'_0 l'_1 \dots l'_{n-1} \right\rangle \\ -1 & \text{if } l \in [n] \\ -\infty & \text{if } l = \bot \end{cases}$$

Lemma 65. *Given x_i^p in a colored interval order X, we have*

$$\mathrm{on}(\llbracket x_i^p \rrbracket) \quad = \quad p$$

This suggests introducing a relation $\lhd$ which expresses when a process can be "seen" by another one, i.e. $x_j^q \lhd x_i^p$ (which is read the q-th round of process j is seen by the p-th round of process i) means that the $(q+1)$th update of process j occurs before the pth scan of process i, so that process i see the observations of process j. Suppose given an colored interval order $(X, \preceq)$ and $x_i^p \in X$ (by convention, we always consider that x_i^{-1} is an element of X for $i \in [n]$). Given the view $\llbracket x_i^p \rrbracket$, we write $x_j^q \lhd x_i^p$ when $\llbracket x_i^p \rrbracket$ is of the form

$$\llbracket x_i^p \rrbracket \quad = \quad \langle l, l_0 l_1 \dots l_{n-1} \rangle \tag{17}$$

with $-\infty < q \leq \mathrm{on}(l_j)$. By the preceding remarks, it is easy to show that

Lemma 66. *We have $x_j^q \lhd x_i^p$ if and only if $x_j^{q+1} \not\succ x_i^p$, i.e. either $x_j^{q+1} \preceq x_i^p$ or $x_j^{q+1} \parallel x_i^p$ in X.*

Proof. Indeed, $x_j^q \lhd x_i^p$ means by definition that u_j^{q+1} happens before s_i^p which is equivalent to s_i^p does not happen before u_j^{q+1} which means by definition that $x_j^{q+1} \not\succ x_i^p$. □

Remark 67. The careful reader will have noticed the shift of 1 in exponents q in previous lemma. This is necessary for the construction of the view complex below to work and can be explained as follows. When we have $x_j^{q+1} \not\succ x_i^p$, this means that in a corresponding execution the action u_j^{q+1} occurs before s_i^p and therefore, the process i will know the contents of the value obtained during the preceding scan s_j^q of process j. In the same vein, an element of the form x_i^{-1} stands for the initial value of the process i and is necessary to determine whether another process sees it or not.

We have seen in Proposition 9 that the relations defining equivalence of traces are correct: two equivalent traces lead to the same local memory state. The above considerations allow us to show a completeness result: two traces which are *indistinguishable*, in the sense that they lead to the same local memory state in every protocol, are equivalent (in the sense of Definition 8). We begin by showing the result considering the view protocol only.

Proposition 68. *Two non-dying well-bracketed traces T and T' are equivalent if and only if we have $l = l'$ where $(l, m) = [\![T]\!]_{\pi^\lhd}$ and $(l', m') = [\![T']\!]_{\pi^\lhd}$.*

Proof. The left-to-right implication is given by Proposition 9, we show the reciprocal. We write X and X' for the colored interval orders respectively corresponding to T and T'. By Remark 27, we know that X and X' are isomorphic as sets and moreover the labels coincide. We thus have to show that the (interval) order relations coincide. From the view l_i, we can reconstruct the views of all the $x_i^p \in X$: l_i is precisely the view $\llbracket x_i^p \rrbracket$ with p maximal and others can be recovered from l_i by Lemma 63. We can thus compute the relations $- \lhd x_i^p$ for every $x_i^p \in X$ (the resulting relation will be detailed in Definition 70 and called the view order). By Lemma 66 and the following discussion, this uniquely determines the order on X. Since, by hypothesis, we have $l = l'$, we deduce that the orders on X and X' are the same, i.e. the traces T and T' are equivalent. □ □

By Proposition 41, two traces are indistinguishable if and only if they lead to the same memory state in the view protocol, so that previous proposition immediately implies:

Theorem 69. *Two non-dying well-bracketed traces T and T' are equivalent if and only if, for every protocol π, we have $l = l'$ where $(l, m) = [\![T]\!]_\pi$ and $(l', m') = [\![T']\!]_\pi$.*

5.2 View orders

The preceding developments show that the information contained in the views is precisely the order $\lhd$ they induce on elements x_i^p. We have seen in (17) that a view induces a relation, and we now introduce a relation which is the union of all such relations for all the possible views of actions in an interval order: the views of the maximal elements can be obtained as the final local memory in the execution of the interval order with the view protocol, and the views of non-maximal elements can be deduced by iteratively constructing the previous views (Definition 62) as explained in Lemma 63.

Definition 70. Suppose given a colored interval order X. We write $x_i^{p_i}$ for the maximal element of X labeled by i. We also write l for the local memory obtained by executing a trace corresponding to X (by Proposition 52). The associated *view order* $[\![X]\!]$ is the set

$$X^- \quad = \quad X \cup \left\{ x_i^{-1} \;\middle|\; i \in [n] \right\}$$

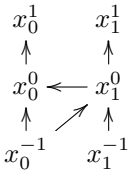
equipped with the relation $\lhd$ such that $x_j^q \lhd x_i^p$ whenever

$$\mathrm{pr}^{p_i - p}(l_i) \quad = \quad \left\langle l', l'_0 l'_1 \dots l'_{n-1} \right\rangle$$

with $q \leq \mathrm{on}(l'_j)$ (above, pr^{p_i-p} denotes the function pr of Definition 62 iterated $p_i - p$ times).

By Proposition 68, the operation $[\![-]\!]$ which to a colored interval order associates its view order is injective.

Example 71. The view order associated to (15) is



(we do not figure edges which can obtained by transitivity, i.e. picture the Hasse diagram of the relation).

In an execution trace T if u_i^{p+1} occurs before s_j^q and u_j^{q+1} occurs before s_k^r, we know that u_i^{p+1} occurs before s_k^r, because we always have that s_j^q occurs before u_j^{q+1}, see Lemma 28:

$$\dots u_i^{p+1} \dots s_j^q \dots u_j^{q+1} \dots s_k^r \dots$$

Using this reasoning, and Proposition 68, one deduce the following properties of the relation $\lhd$:

Proposition 72. *Given a colored interval order X, the relation $\lhd$ of $\llbracket X \rrbracket$ is always irreflexive, transitive and acyclic.*

Proof. Irreflexivity corresponds to the fact that in an execution trace u_i^{p+1} never occurs before s_i^p. Transitivity can be shown using the above reasoning. Acyclicity follows by absurd from transitivity and irreflexivity. □ □

The most interesting feature of view orders is that one can formulate a definition of views directly on them. Suppose given a colored interval order $(X, \preceq)$. Writing $X^- = X \cup \left\{x_i^{-1} \;\middle|\; i \in [n]\right\}$, consider the view order $(X^-, \lhd)$ associated to it as in Definition 70. This set is implicitly labeled by $\ell(x_i^{-1}) = i$. Given a subset $Y \subseteq X^-$, we write $\downarrow Y$ for the *downward closure* of Y: it contains Y, the x_j^{-1} for $j \in [n]$, and the elements x_j^p such that $x_j^p \lhd x_i^p$ for some $x_i^p \in Y$. This set can be equipped with the restriction of the relation $\lhd$.

Definition 73. The *i-view* of the view order $(X^-, \lhd)$ is the view order $(\downarrow \{x_i^p\}, \lhd)$, which will be denoted $\llbracket X^- \rrbracket_i$, where x_i^p is the greatest element labeled by i.

Example 74. The 0-view (on left) and 1-view (on right) associated to the view order of Example 71 are respectively

(again, we do not figure transitive edges).

This definition is consistent with the one of Definition 59, thus justifying the use of the same notation, in the following sense. Given the interval order X, consider its i-view $\llbracket X \rrbracket_i$. We can add to it elements of the form x_i^{-1}, for $i \in [n]$, and equip it with the relation $\lhd$ as defined in Definition 70. The view order we obtain in this way is then precisely $\llbracket X^- \rrbracket_i$. Moreover, the "traditional" view $\llbracket X \rrbracket_i$ can be reconstructed from $(\llbracket X^- \rrbracket_i, \lhd)$ as the view $\langle\!\langle x_i^p \rangle\!\rangle$, definition follows, by induction. We define $\langle\!\langle x_i^{-1} \rangle\!\rangle = i$ and

$$\langle\!\langle x_i^p \rangle\!\rangle \quad = \quad \left\langle \langle\!\langle x_i^{p-1} \rangle\!\rangle, \langle\!\langle x_0^{p_0} \rangle\!\rangle \langle\!\langle x_1^{p_1} \rangle\!\rangle \dots \langle\!\langle x_{n-1}^{p_{n-1}} \rangle\!\rangle \right\rangle$$

where $x_j^{p_j}$ is the predecessor of x_i^p labeled by j, by convention $\langle\!\langle x_j^{p_j} \rangle\!\rangle = \bot$ when no such predecessor exists. It is routine to check that the two transformations are mutually inverse to each other. To sum it up, the view order is simply a convenient way to represent views. However, there is an advantage to this new notation: one can consider the "view of several processes at once".

Definition 75. Given a set $I \subseteq [n]$ of process indices, the *I-view* of a view order $(X^-, \lhd)$ is the view order $(\downarrow \{x_i^{p_i} \mid i \in I\}, \lhd)$, which will be denoted $\llbracket X^- \rrbracket_I$, where $x_i^{p_i}$ is the greatest element labeled by $i \in I$.

Given a view order X^- and two elements x_i^p and x_j^q, with $i \neq j$ and $p, q \geq 0$, which are maximal with their label, the views $\llbracket X^- \rrbracket_i = \downarrow \{x_i^p\}$ and $\llbracket X^- \rrbracket_j = \downarrow \{x_j^q\}$ are distinct since otherwise we would have both $x_j^q \lhd x_i^p$ and $x_i^p \lhd x_j^q$, which would

contradict the acyclicity of $\lhd$. Moreover, $\downarrow\{x_i^p, x_j^q\} = \downarrow\{x_i^p\} \cup \downarrow\{x_j^q\}$ since $x \in \downarrow\{x_i^p, x_j^q\}$ is equivalent to $x \lhd x_i^p$ or $x \lhd x_j^q$. This observation generalizes into:

Lemma 76. *With the above notations, given a set $I \subseteq [n]$, we have that*

$$\llbracket X^- \rrbracket_I \quad = \quad \bigcup_{i \in I} \llbracket X^- \rrbracket_i$$

and the relation on $\llbracket X^- \rrbracket_I$ is also the union of the relations $\llbracket X^- \rrbracket_i$.

This will turn out to be very useful in next section, in order to provide an alternative definition to the protocol complex.

5.3 View orders and traces

We have seen in Section 4.1 that colored interval orders are in bijection with traces, which are well-bracketed, up to equivalence (Proposition 52). A natural question is then to which traces correspond i-view orders? We show here that those correspond to traces, up to equivalence, satisfying a variant of the well-bracketing condition (Definition 19), where only the "brackets" of the i-th process is closed. We omit most proofs since those are easy adaptations of those presented in Sections 1.2.3 and 4.1 to the variant of the well-bracketing condition.

The previous definitions on traces take all the processes in account. We first generalize those in order to account for the fact that we are now interested in the views of a specific set of processes $I \subseteq [n]$.

Definition 77. In a trace T, an action is *I-relevant* if it is

- u_j when it occurs before an action s_i with $i \in I$,
- s_j when there is an action u_j afterward which is I-relevant,
- s_i or d_i with $i \in I$.

The *I-restriction* of T is the trace obtained from T by keeping only I-relevant actions, and is denoted $\lceil T \rceil_I$.

Definition 78. A trace T is *I-well-bracketed* when

- $\mathrm{proj}_i(T) \in (u_i s_i)^*(\varepsilon + u_i d_i)$ for $i \in I$,
- $\mathrm{proj}_i(T) \in \varepsilon + ((u_i s_i)^* u_i)$ for $i \in [n] \setminus I$, and
- all the actions of T are I-relevant.

In particular a well-bracketed trace is the same as an $[n]$-well-bracketed one. In the following, we simply write i-well-bracketed instead of $\{i\}$-well-bracketed. The characterization of Lemma 21 can easily be adapted:

Lemma 79. *A trace $T \in \mathcal{A}^*$ is I-well-bracketed if and only if*

1. $\mathrm{updated}(T)$ *is well-defined,*
2. $\mathrm{updated}(T) = [n] \setminus I$,

3. T is strongly properly dying, and

4. all the actions of T are I-relevant.

Notice that the last scan of a process j, with $j \notin I$, is never I-relevant. More generally, one can show:

Lemma 80. *The I-restriction of a trace is I-well-bracketed.*

Similarly, the notion of equivalence can be adapted in order to distinguish when two traces lead to the same result when we observe only the local memory cells of processes in I.

Definition 81. Two traces T and T' are *I-equivalent*, what we write $T \approx_I T'$ when their I-restrictions are equivalent, i.e. $\lceil T \rceil_I \approx \lceil T \rceil'_I$.

Example 82. For instance, with $I = \{1\}$, the following traces are I-equivalent but not equivalent:

$$u_0 u_1 s_0 s_1 \qquad\qquad u_0 s_0 u_1 s_1$$

Namely, they only differ by the relative orderings of u_1 and s_0, which has no influence on the view of 1, which is $\langle 1, 01 \rangle$ in both cases.

We have seen in Theorem 69 that two traces are equivalent if and only if they give rise to the same views in every protocol. This can be generalized without difficulty in order to show a variant "relative to the set I of processes":

Proposition 83. *Two non-dying well-bracketed traces T and T' are I-equivalent if and only if for every protocol π and every $i \in I$, we have $l_i = l'_i$, where $(l, m) = [\![T]\!]_\pi$ and $(l', m') = [\![T']\!]_\pi$.*

From now on, we only consider non-dying traces for simplicity. With the previous definitions at hand, the correspondence described in Proposition 52 can be adapted in order to show the following. The formulation is a bit contrived because we do not have (at least for now) a characterization of the i-view orders, i.e. of those which come from traces or colored interval orders.

Proposition 84. *Suppose given a colored interval order X corresponding to a trace T by Proposition 52. Then there is a bijection between*

(i) the I-restrictions of T up to equivalence,

(ii) view orders of the form $\downarrow\{x_i^{p_i} \mid i \in I\}$ (the downward closure is taken in the view order X^- associated to X) where $x_i^{p_i}$ is the maximal element labeled by $i \in I$.

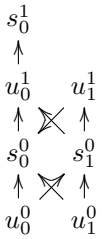
Moreover, this bijection does not depend on T (or X).

By the fact that the bijection "does not depend on T", we mean that given two (possibly non-equivalent) traces T' and T'' having a common I-restriction T the view orders by the above bijection will be the same (and similarly for the other side of the bijection). The proof is very similar to the one of Proposition 52. Instead of going over it once again, we illustrate it on an example.

Example 85. Consider the 0-view X^- depicted on the left in Example 74. The 0-well-bracketed trace corresponding to it is an interleaving of the traces $u_0^0 s_0^0 u_0^1 s_0^1$ and $u_1^0 s_1^0 u_1^1$ (notice that the last one is not well-bracketed in the traditional sense, but the presence of u_1^1 is encoded in the 0-view). We have that

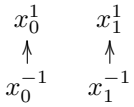
- the relations $x_i^p \lhd x_i^{p+1}$ imply that u_i^{p+1} occurs before s_i^{p+1} (which we already knew anyway),
- the relation $x_1^0 \lhd x_0^0$ implies that u_1^1 occurs before s_0^0,
- the relation $x_1^{-1} \lhd x_0^0$ implies that u_1^0 occurs before s_0^0,
- the relation $x_0^{-1} \lhd x_1^0$ implies that u_0^0 occurs before s_1^0,
- the absence of the relation $x_0^0 \lhd x_1^0$ implies that we do not have u_0^1 before s_1^0 (i.e. we have u_0^1 after s_1^0),

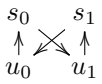
The relative order of the actions is thus



and we see that the only possible execution trace is $u_0^0 u_1^0 s_1^0 u_1^1 s_0^0 u_0^1 s_0^1$, up to permuting consecutive updates or consecutive scans, i.e. up to equivalence.

Remark 86. Again, not every set with a transitive order relation is a view. For instance, if we applied the construction of the previous example to the set on the left



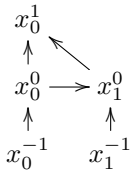


we obtain the constraints on the right for a trace, which obviously cannot be satisfied since they are cyclic.

Example 87. To illustrate the other side of the bijection, consider the 0-well-bracketed trace $u_0^0 s_0^0 u_0^1 u_1^0 s_1^0 u_1^1 s_0^1$. We have that

- since u_1^1 occurs before s_0^1 we have $x_1^0 \lhd x_0^1$,
- since u_0^1 occurs before s_1^0 we have $x_0^0 \lhd x_1^0$

(other relations are redundant or obvious). Therefore the associated 0-view order is



5.4 View orders and interval orders

We would like to also very briefly explain how the views can be encoded directly in interval orders. A first idea is that given an interval order $(X, \preceq)$ and $i \in [n]$, writing x_i^p for the maximal element labeled by i, the i-view should be the restriction of X to elements which are not above (i.e. below or independent from) x_i^p. This is however not the case. For instance, consider the two following colored interval orders

$$x_0^0 \quad x_1^0 \qquad\qquad x_0^0 \nrightarrow x_1^0$$

taking the "1-view" as described above, i.e. restricting to elements not above x_1^0 leaves the interval orders unchanged, and thus distinct. However, their views (in the sense of Definition 59) are the same: they are both $\langle 1, 01 \rangle$. The discrepancy comes from the fact that interval orders encode well-bracketed traces (by opposition to 1-well-bracketed traces). Namely, the two interval orders respectively correspond to the traces

$$u_0 u_1 s_0 s_1 \qquad\qquad u_0 s_0 u_1 s_1$$

whereas, by Section 5.3, the view correspond to the 1-well-bracketed trace $u_0 u_1 s_1$ (this observation is essentially the same as the one in Example 82). Another way to state this is that the interval order encodes the relative positions of u_1 and s_0, whereas this is irrelevant since s_0 does not play a role in the view. This suggests introducing the following definition.

Definition 88. Given $I \subseteq [n]$, a *colored I-interval* order is an interval order such that, for $i \in [n] \setminus I$, a maximal element labeled by i is maximal (among all elements, even those with different labels). The *I-restriction* of a colored interval order $(X, \preceq)$ is the interval order, on the same elements, obtained by removing dependencies from any element x_i^p, with $i \in [n] \setminus I$, which is maximal among elements labeled by i.

Remark 89. The fact that the I-restriction of an interval order is still an interval order is not immediate, but can be shown using the characterization mentioned in Remark 51.

Finally, views can be defined as follows.

Definition 90. Suppose given a colored interval order $(X, \preceq)$ and $I \subseteq [n]$. For $i \in I$, we write $x_i^{p_i}$ for the greatest element of X which is labeled by i. The *I-view* $[\![X]\!]_I$ of X is the interval order obtained by

1. restricting X to elements which are below or independent from an element $x_i^{p_i}$ with $i \in I$,
2. taking the I-restriction of the resulting interval order.

It can be shown that this construction coincide with the previous ones, in a way which is compatible with the various isomorphisms established. We do not detail it further here, because it does not play an important role and is less convenient to manipulate than the description in terms of view order. For instance, reconstructing the I-view from the i-views is less direct than for view orders, as described in Lemma 76.

6 Protocol complexes, derived from the concurrent semantics

In this section, we are going to define the protocol complex [26] associated to a protocol, in equivalent ways. First, in Section 6.1, we define it from the operational semantics of Section 1.1.1. Equivalently, based on the results of Section 5, this can be defined using interval orders, or the geometric semantics: this is made formal using the notion of view order of Section 5 in the form of a *view complex*, in Section 6.2, and also, equivalently, in the form of a *interval order complex*, and a *trace complex*. They are shown all equivalent to the (standard) protocol complex in Proposition 99. Finally, in Section 6.3, we will particularize this construction to the simpler case of the immediate snapshot protocol and the protocol complex constructed through chromatic subdivisions [29].

6.1 The protocol complex

The protocol complex [28] is a simplicial complex which has been designed 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 91. Given numbers $(r_i)_{i\in[n]}$ of rounds, 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 its local memory in a reachable state,
- maximal simplices are $\{(0, l_0), \ldots, (n, l_n)\}$ where $\langle i, l_i\rangle$ is the local view by process i at the end of the execution with r rounds represented by this simplex.

Example 92. The local views in each vertex are determined by the operational semantics of Section 1, as in the following example, using the same notations as in Example 40:

$$\begin{array}{|c|c|} \hline 0 & 1 \\ \hline \bot & \bot \\ \hline \end{array} \xrightarrow{u_0} \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 0 & \bot \\ \hline \end{array} \xrightarrow{u_1} \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 0 & 1 \\ \hline \end{array} \xrightarrow{s_1} \begin{array}{|c|c|} \hline 0 & \langle 1, 01\rangle \\ \hline 0 & 1 \\ \hline \end{array} \xrightarrow{s_0} \begin{array}{|c|c|} \hline \langle 0, 01\rangle & \langle 1, 01\rangle \\ \hline 0 & 1 \\ \hline \end{array}$$

leading to the local views

$$l_0 = \langle 0, 01\rangle \qquad l_1 = \langle 1, 01\rangle$$

Similarly, the trace $u_0 s_0 u_1 s_1$ leads to the local views

$$l_0 = \langle 0, 0\bot\rangle \qquad l_1 = \langle 1, 01\rangle$$

and there is a third potential outcome of the computation, symmetric to this last case, in which process 1 updates and scans before process 0 does. Putting

this together, according to Definition 91, we get the protocol complex for one round and two processes [28]:

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

For concision, we do not figure the external brackets, i.e. write $0, 0\bot$ instead of $\langle 0, 0\bot \rangle$. The identifier of the process whose local view is written is the number before the comma, e.g. the state $0, 0\bot$ above is the local view of processor 0.

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 (Proposition 52), and a maximal simplex in the protocol complex. Furthermore, we can deduce the local view of the i-th process by using the i-view of this interval order (by Definition 59, or equivalently using the i-view of the view order of Definition 73). These local views will identify the interval orders seen as maximal simplices of the protocol complex as convex hulls of the $n + 1$ local views, hence will encode the full simplicial complex structure.

We encode here local views restricting to the full information generic protocol in normal form with initial local state $l_i = i$ for $i \in [n]$, i.e. with standard input, see Definition 34 (this only changes the naming of local states, and not the structure of the protocol complex). This can be generalized to more general input complexes, as hinted in Section 6.4.

By Theorem 69, we know that two non-dying well-bracketed traces are equivalent if and only if they are non distinguished by the full information generic protocol in normal form, which is initial in the category of protocols by Proposition 41. Hence local views of process i, on a trace T, corresponds to the i-view of the interval order corresponding to T (Section 5).

These observations lead to the equivalent descriptions of the protocol complex using interval orders, views and traces in Section 6.2. Before formally defining them, let us give a few examples first.

Example 93. Consider again the one round, two processes case. We have represented below the protocol complex already depicted in Example 92, and decorated its maximal simplices, i.e. edges, with the corresponding dipaths modulo dihomotopy above, and the corresponding interval order, below:

$$0, 0\bot \xrightarrow[0 \prec 1]{} 1, 01 \xrightarrow[0 \quad 1]{} 0, 01 \xrightarrow[0 \succ 1]{} 1, \bot 1$$

The local view (at the leftmost part of the figure above) of process 0 which is $0, 0\bot$ comes from the 0-view $[\![X]\!]_0$ of the interval order $X = {}_{0 \prec 1}$, subscript of the leftmost edge in the graph above: an interleaving trace corresponding to $\natural\, x_0^0 = {}_0$, under Proposition 52 (and the remark at the end of Section 4) is $u_0 s_0$ leading to local state $\langle 0, 0\bot \rangle$ on process 0. Similarly, $1, 01$ corresponds to the local state for process 1, which is both the 1-view of $[\![X]\!]_1$ which corresponds to the local view of the interval order $\natural\, x_1^0 = {}_{0 \prec 1}$ (corresponding to a trace $u_0 s_0 u_1 s_1$, as in the trace $\overset{\frown}{\cdot\,\cdot}$ superscript of the edge on the left of the graph above) and to the 0-view $[\![Y]\!]_0$, i.e. the local view of $\natural\, x_0^0 = {}_0 \quad {}_1$, where $Y = {}_0 \quad {}_1$ (corresponding to a trace $u_0 u_1 s_0 s_1$ for instance, as in the trace $\cdot\,\cdot$ superscript of the middle edge of the graph above). Note that $[\![X]\!]_1 = [\![Y]\!]_0$ but $\natural\, x_1^0$ in X is not the same interval order as $\natural\, x_0^0$ in Y, as remarked already in Section 5.4.

Example 94. An example of interval order complex with the traces corresponding to the execution for 2 processes, 2 rounds is depicted at Figure 1. 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 [26]. This is because we are considering more executions than the classical *iterated immediate snapshot protocols* [26]: we allow round 2 of process 0 to begin while process 1 is still in round 1 for instance. Consider the interval order $X = \begin{smallmatrix} 0 \to 1 \\ \uparrow \times \uparrow \\ 0 \to 1 \end{smallmatrix}$ labeling the upper left edge of the protocol complex in Figure 1, where an arrow $x \to y$ 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 $\begin{smallmatrix} 0 \\ \uparrow \\ 0 \end{smallmatrix}$ An interleaving trace corresponding to this is e.g. $u_0 s_0 u_0 s_0$, which, by the semantics of Section 1, leads to the local state $\langle 0, \langle 0, 0\bot\rangle \bot\rangle$ of process 0 (which is the 0-view of X, $\llbracket X \rrbracket_i$) of Proposition 60), written in condensed form as the upper left local state $0, ((0\_)\_)$ in Figure 1.

Example 95. In Figure 2, 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 [26]: to the 13 triangles of [26], 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: $u_0 u_2 s_2 u_1 s_1 s_0$.

6.2 Alternative descriptions of the protocol complex

Alternative descriptions of the protocol complex can be handled, using the results of Section 5 in three equivalent ways : through views (and the view complex, Definition 96), through interval orders (and the interval order complex, Definition 97) and through traces modulo equivalence, Definition 98.

Definition 96. The *view complex* for atomic snapshot protocols on $n+1$ processes and (r) rounds is the abstract simplicial complex constructed as follows:

- maximal simplices are $\{(0, \llbracket X^- \rrbracket_0), \ldots, (n, \llbracket X^- \rrbracket_n)\}$ where X is a colored interval order on the set $X^n_{(r)}$ and $\llbracket X^- \rrbracket_i$ (Definition 73) is the i-view on the view order generated by X (Definition 70)
- the boundaries of these maximal simplices are their subsets: the iterated boundary $\{(i_1, \llbracket X^- \rrbracket_{i_1}), \ldots, (i_k, \llbracket X^- \rrbracket_{i_k})\}$ of $\{(0, \llbracket X^- \rrbracket_0), \ldots, (n, \llbracket X^- \rrbracket_n)\}$ can be identified with the $(I, \llbracket X^- \rrbracket_I)$, where $I = \{i_1, \ldots, i_k\}$ (Definition 75).

Definition 97. The *interval order complex* for atomic snapshot protocols on $n+1$ processes and (r) rounds is the abstract simplicial complex constructed as follows:

- maximal simplices are $\{(0, \llbracket X \rrbracket_0), \ldots, (n, \llbracket X \rrbracket_n)\}$ where X is a colored interval order on the set $X^n_{(r)}$ and $\llbracket X \rrbracket_i$ is the i-view of X (Definition 90),

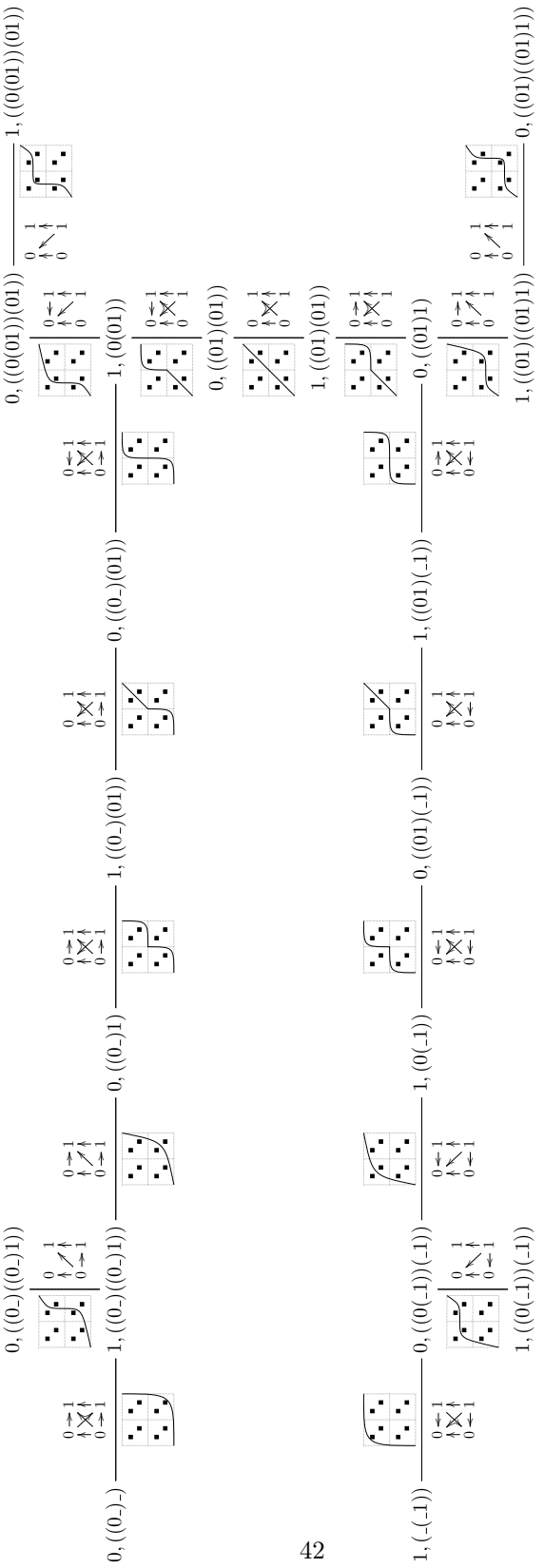


Figure 1: The protocol complex, decorated with corresponding traces and interval orders, of 2 processes, 2 rounds.

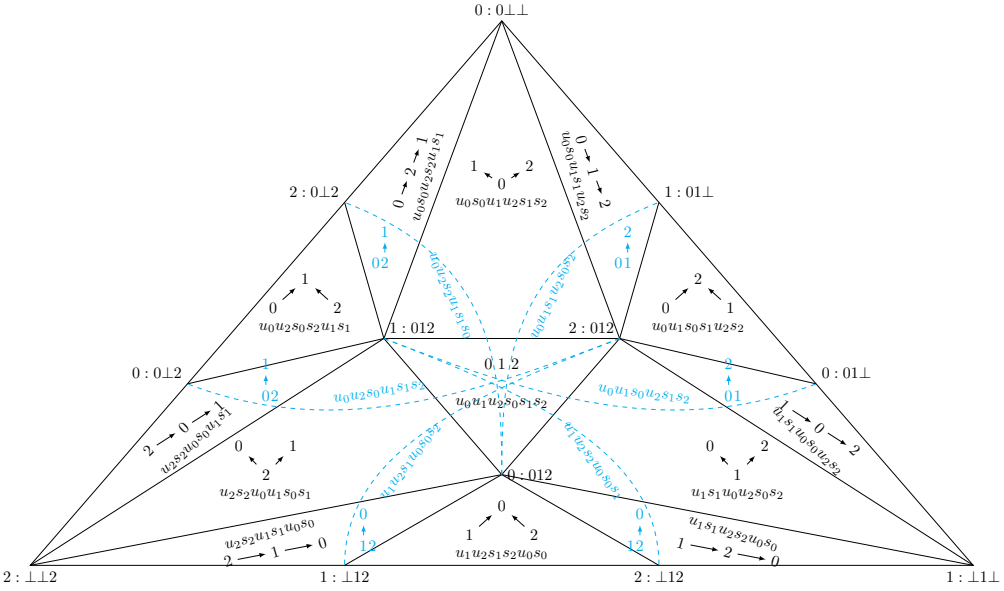


Figure 2: The protocol complex decorated with interval orders and corresponding traces, of 3 processes and 1 round.

- the boundaries of these maximal simplices are their subsets: the iterated boundary $\{(i_1, \llbracket X \rrbracket_{i_1}), \ldots, (i_k, \llbracket X \rrbracket_{i_k})\}$ of $\{(0, \llbracket X \rrbracket_0), \ldots, (n, \llbracket X \rrbracket_n)\}$ can be identified with $(I, \llbracket X \rrbracket_I)$, where $I = \{i_1, \ldots, i_k\}$ (see Definition 90).

Definition 98. The *trace complex* for atomic snapshot protocols on $n+1$ processes and (r) rounds is the abstract simplicial complex constructed as follows:

- maximal simplices are $\{(0, T_0), \ldots, (n, T_n)\}$ where T is a maximal trace of the view protocol for $n+1$ processes and (r) rounds (Definition 39) and T_i is the $\{i\}$-restriction of T up to equivalence (Proposition 84),
- the boundaries of these maximal simplices are their subsets: the iterated boundary $\{(i_1, T_{i_1}), \ldots, (i_k, T_{i_k})\}$ of $\{(0, T_0), \ldots, (n, T_n)\}$ can be identified with the pair composed of I and the I-restriction of T, where $I = \{i_1, \ldots, i_k\}$ (Proposition 84).

Note that this trace complex could have been equivalently defined from the dipaths modulo dihomotopy thanks to the equivalence between the trace semantics and the geometric semantics, see Section 3.4.

Proposition 99. *The view complex, the interval order complex and the trace complex are isomorphic to the protocol complex of Definition 91.*

Proof. We know by Section 5 that $\llbracket X^- \rrbracket_i$ corresponds to the view of $\llbracket X \rrbracket_i$ by Proposition 60 and Section 5.2. Hence $\llbracket X^- \rrbracket_i$ can be identified with the local memory state l_i of processor i for the execution corresponding to the interval order X. The maximal simplices of the protocol complex of Definition 91 and of the view complex of Definition 96 are then the same. Similarly for the boundary

operations. Similarly for the interval order complex, by Section 5.4, the i-views defined directly on interval orders are isomorphic to the ones defined on views. Now for the trace complex, this stems from the equivalence (Proposition 84) between $\{i\}$-restrictions of a trace T modulo equivalence with i-views of the corresponding view order. □ □

Example 100. Consider the protocol complex for 2 processes and 2 rounds of Figure 1, and the 1-simplex corresponding to the interval order X below (left). Its local views are shown on the right hand side of the following table:

$$X = \begin{array}{ccc} 0 & \leftarrow & 1 \\ \uparrow & \nwarrow\!\!\nearrow & \uparrow \\ 0 & \rightarrow & 1 \end{array} \qquad [\![X]\!]_0 = \begin{array}{ccc} 0 & & 1 \\ \uparrow & \nwarrow\!\!\nearrow & \uparrow \\ 0 & \rightarrow & 1 \end{array} \qquad [\![X]\!]_1 = \begin{array}{ccc} & & 1 \\ & & \uparrow \\ 0 & & 1 \end{array}$$

Let us explain the calculation of $[\![X]\!]_1$: we first eliminate the maximal 0 in X since it is greater than the maximal 1, giving

$$\begin{array}{ccc} & & 1 \\ & \nearrow & \uparrow \\ 0 & \rightarrow & 1 \end{array}$$

Now, we take its 1-restriction which eliminates the arrow from 0 to the upper 1, but also the arrow from 0 to the lower 1 (because, otherwise, by transitivity, 0 would still be lower than the upper 1!).

Consider now the 1-simplex encoded by the interval order Y below (left). Its local views are shown on the right hand side of the following table :

$$Y = \begin{array}{ccc} 0 & & 1 \\ \uparrow & \nwarrow\!\!\nearrow & \uparrow \\ 0 & \rightarrow & 1 \end{array} \qquad [\![Y]\!]_0 = \begin{array}{ccc} 0 & & 1 \\ \uparrow & \nwarrow\!\!\nearrow & \uparrow \\ 0 & \rightarrow & 1 \end{array} \qquad [\![Y]\!]_1 = \begin{array}{ccc} 0 & & 1 \\ \uparrow & \nwarrow\!\!\nearrow & \uparrow \\ 0 & \rightarrow & 1 \end{array}$$

As we see from the above, $[\![X]\!]_0 = [\![Y]\!]_0$, linking the 2 1-simplices together in Figure 1. Indeed, the view of processor 0 for both X and Y is encoded, by the view protocol, by $0, ((0\_)(01))$ as shown on the same figure.

Now consider Z as below, and its views:

$$Z = \begin{array}{ccc} 0 & \leftarrow & 1 \\ \uparrow & \nwarrow & \uparrow \\ 0 & & 1 \end{array} \qquad [\![Z]\!]_0 = \begin{array}{ccc} 0 & & 1 \\ \uparrow & \nwarrow & \uparrow \\ 0 & & 1 \end{array} \qquad [\![Z]\!]_1 = \begin{array}{ccc} & & 1 \\ & & \uparrow \\ 0 & & 1 \end{array}$$

and T and its views:

$$T = \begin{array}{ccc} 0 & \leftarrow & 1 \\ \uparrow & \nwarrow\!\!\nearrow & \uparrow \\ 0 & & 1 \end{array} \qquad [\![T]\!]_0 = \begin{array}{ccc} 0 & & 1 \\ \uparrow & \nwarrow\!\!\nearrow & \uparrow \\ 0 & & 1 \end{array} \qquad [\![T]\!]_1 = \begin{array}{ccc} & & 1 \\ & & \uparrow \\ 0 & & 1 \end{array}$$

We have indeed $[\![T]\!]_1 = [\![Z]\!]_1 = [\![X]\!]_1$ glueing together these 3 1-simplices as show in the upper right of Figure 1.

Example 101. Consider the protocol complex for 3 processes and 1 round of Figure 2, and consider the 2-simplex, corresponding to the interval order X below (left). The local views of each processor is indicated on the right:

$$X = \begin{array}{ccc} 0 & & 1 \\ & \nwarrow\ \nearrow & \\ & 2 & \end{array} \qquad [\![X]\!]_0 = \begin{array}{ccc} 0 & & 1 \\ & & \uparrow \\ & & 2 \end{array} \qquad [\![X]\!]_1 = \begin{array}{ccc} 0 & & 1 \\ \uparrow & & \\ 2 & & \end{array} \qquad [\![X]\!]_2 = \ 2$$

As a matter of fact, restricting X to the elements below or independent from $i = 0, 1$ still gives X, but then, taking the 0-restriction (respectively 1-restriction) implies forgetting about the dependency between 2 and 0 (respectively between 2 and 1). Finally, restricting X to the elements below or independent of 2 gives just the singleton 2.

Now, the view $[\![X]\!]_{\{0,1\}}$ is obtained from X by $\{0,1\}$-restriction, for which we obtain:

$$[\![X]\!]_{\{0,1\}} = \begin{array}{ccc} 0 & 1 & 2 \end{array}$$

This is clearly the same as the $\{0,1\}$-view of the central 2-simplex encoded by the interval order

$$Y = \begin{array}{ccc} 0 & 1 & 2 \end{array}$$

hence X and Y share a common face.

Finally, the $\{0,2\}$-view of X is just the $\{0,2\}$-restriction of X, which is

$$\begin{array}{ccc} 0 & & 1 \\ & \nwarrow & \\ & & 2 \end{array} \tag{18}$$

Note that the $\{0,2\}$-view of

$$2 \twoheadrightarrow 1 \twoheadrightarrow 0$$

is the $\{0,2\}$-restriction of itself, which is (18) again, showing that X and Z share a common face indeed.

6.3 The particular case of 1-round immediate snapshot protocols

We recall that an (iterated, for multi-round protocols) *immediate snapshot* protocol [26] 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 $u_{i_1} \dots u_{i_k} s_{i_1} \dots s_{i_k}$. Of course, there is some difference with the protocol complex of Definition 91, in that the latter accounts for non necessarily layered by rounds, nor "immediate" protocols. It is the aim of this section to make the connection between the subcomplex generated by some interval orders only, describing iterated immediate snapshot protocol executions, and the equivalent two definitions of standard chromatic subdivision [29, 23] that describe combinatorially the protocol complex in that case.

The standard chromatic subdivision $\chi(\Delta^{[n]})$ of the standard colored simplicial complex $\Delta^{[n]}$ is defined as follows (see [23], where an equivalence with the Definition in [29] is also shown):

Definition 102. The *standard chromatic subdivision* $\chi(\Delta^{[n]})$ of $\Delta^{[n]}$ is the colored simplicial complex whose vertices are pairs (V, i) with $V \subseteq [n]$ and $i \in V$ and simplices are sets of the form $\sigma = \{(V_0, i_0), \dots, (V_d, i_d)\}$ 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: $\ell(V, i) = i$.

Proposition 103. *Layered immediate snapshot executions (for any number of rounds) correspond to the colored interval orders such that: $J \prec K$ and I is not comparable with J implies $I \prec K$. The subcomplex of the protocol complex of Definition 91 on one round that contains only such immediate snapshot executions is isomorphic to the standard chromatic subdivision of Definition 102.*

Proof. For the first part, suppose that we have an interval order $\preceq$, representing a maximal simplex in the protocol complex of Definition 91, such that $J \prec K$ and I is not comparable with J and K. I, J and K correspond to some intervals of update and scan 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 103.

Conversely, we suppose that for I not comparable to J and $J \prec K$, then $I \prec K$. We prove now that all execution paths are layered and immediate snapshot ones. Suppose we have an interleaving path (up to equivalence) of the form: $T u_j^{l_j} U s_j^{l_j} V u_k^{l_k} W s_k^{l_k} X$ 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:

$$\sigma \quad = \quad \{(V_0, i_0), \ldots, (V_d, i_d)\}$$

with $d \geq 0$ (the case $d = -1$ is trivial) in the standard chromatic subdivision of Definition 102. We associate to σ the following interval order: 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 an interval order, and that the condition of Proposition 103 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

$$(V_x, i_x) \prec_\sigma (V_y, i_y) \qquad\qquad (V_z, i_z) \prec_\sigma (V_t, i_t)$$

Then, as σ is "ordered" (see Definition 102), 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 [15]. Now consider again σ in the standard chromatic 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 102, $V_y \subsetneq V_x$ which implies $(V_y, i_y) \prec_\sigma (V_x, i_x)$. This is impossible since (V_x, i_x) and (V_y, i_y) are supposed incomparable. Finally, note that well-coloredness of σ implies that the labeling we define is indeed a labeling function of an interval order.

Conversely, suppose we have a 1-round colored interval order $(X, \preceq)$ on $d+1$ elements which satisfies the property from Proposition 103. 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 standard chromatic subdivision of Definition 102 by defining k-simplices (for all $k \leq n$) $\sigma_X = ((|V_i^{k_i}|, i))_{i \in [k]}$ (where $|V|$ is the set of elements of the 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 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 102 holds as well. □

□

6.4 Input, output, protocol complexes and the solvability of tasks

Given a task Θ as in Section 2.1, i.e. a relation $\Theta \subseteq \mathcal{I}^n \times \mathcal{O}^n$ between input and output values, we note first that $\operatorname{dom}\Theta$ can be seen as a presimplicial set such that the dimension of $l \in \operatorname{dom}\Theta$ is the number of entries different from $\perp$, and the i-th face is given by $\partial_j(l)$ where j is the index of the i-th entry different from $\perp$. It can also be seen as a simplicial complex with $[n] \times (\mathcal{I} \setminus \{\perp\})$ as vertices, and simplices are of the form $\{(i,x) \in [n] \times \mathcal{V} \mid l_i = x \neq \perp\}$, 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 have seen how to construct the protocol complex from a unique global state, i.e. one maximal dimensional simplex. From the input complex, we can construct the corresponding protocol complex, by gluing together the protocol complexes obtained from each separate initial simplices, according to the same gluing scheme as for the input complex. We do not detail this here since this is completely standard (see [26]). Now, a (pre-)simplicial map from it to the output complex will necessary exist as a (necessary and sufficient) condition for solvability of the task Θ in (r) rounds. Most of this is out of the scope of this paper, which is concerned with the semantics of scan-update protocols and the construction and characterization of the protocol complex, and we refer the interested reader to [26].

Still, we expect that the existence of such a (pre-)simplicial map will stem from the initiality of the view protocol : we believe that the decision map should be obtained using the universal morphism derived from the initial character of the view protocol. This paves the way towards proving computability results directly from the semantics, and without constructing explicitly the protocol complex. This is left for future work.

7 Conclusion and future work

We have revealed 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 iterated immediate snapshot model, but also on the more complicated (non immediate) iterated snapshot model. This, combined

with the results of [30, 23], entirely classifies geometrically the computability of wait-free iterated 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 program. Fault-tolerant distributed models, whose protocol complex are more complex to guess combinatorially, may be handled by going through the very same steps we went through, starting with the geometric semantics of the communication primitives, and classifying dipaths modulo dihomotopy. We shall apply 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), Sept. 1993.

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

[3] F. Baader and T. Nipkow. *Term rewriting and all that*. Cambridge university press, 1999.

[4] F. Benavides and S. Rajsbaum. The read/write protocol complex is collapsible. In *Latin American Symposium on Theoretical Informatics*, pages 179–191. Springer, 2016.

[5] M. Bezem, J. W. Klop, and R. de Vrijer. *Term rewriting systems*. Cambridge University Press, 2003.

[6] 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 *PoDC*. ACM, 1988.

[7] 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*, 2011.

[8] E. Borowsky and E. Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In *STOC*, 1993.

[9] A. Castañeda, S. Rajsbaum, and M. Raynal. Specifying concurrent problems: beyond linearizability and up to tasks. In *International Symposium on Distributed Computing*, pages 420–435. Springer, 2015.

[10] L. Fajstrup, É. Goubault, E. Haucourt, S. Mimram, and M. Raussen. Trace spaces: An efficient new technique for state-space reduction. In *ESOP*, 2012.

[11] L. Fajstrup, É. Goubault, E. Haucourt, S. Mimram, and M. Raussen. *Directed Algebraic Topology and Concurrency.* Springer International Publishing, 2016.

[12] L. Fajstrup, É. Goubault, and M. Raussen. Detecting deadlocks in concurrent systems. In *CONCUR*, number 1466 in LNCS. Springer-Verlag, 1998.

[13] L. Fajstrup, M. Raussen, and É. Goubault. Algebraic topology and concurrency. *TCS*, 357(1), 2006.

[14] 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.

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

[16] E. Gafni. Snapshot for time: the one-shot case. *arXiv preprint arXiv:1408.3432*, 2014.

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

[18] É. Goubault. *The Geometry of Concurrency.* Ph.D. dissertation, ENS, 1995.

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

[20] É. Goubault and E. Haucourt. A practical application of geometric semantics to static analysis of concurrent programs. In *CONCUR 2005.* Springer, 2005.

[21] É. Goubault, T. Heindel, and S. Mimram. A geometric view of partial order reduction. *MFPS, Electr. Notes Theor. Comput. Sci.*, 298, 2013.

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

[23] É. Goubault, S. Mimram, and C. Tasson. Iterated chromatic subdivisions are collapsible. *Applied Categorical Structures*, 2014.

[24] M. Grandis. *Directed Algebraic Topology : Models of Non-Reversible Worlds*, volume 13 of *New Mathematical Monographs.* Cambridge University Press, 2009.

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

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

[27] 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.

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

[29] D. Kozlov. Chromatic subdivision of a simplicial complex. *Homology, Homotopy and Appl.*, 14(2), 2012.

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

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

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

[33] V. Pratt. Modeling concurrency with geometry. In *POPL*. ACM Press, 1991.

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

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