Compact Topology of Shared-Memory Adversaries

Petr Kuznetsov, Thibault Rieutord, Yuan He

Compact Topology of Shared-Memory Adversaries*

Petr Kuznetsov[1], Thibault Rieutord[1], and Yuan He[2]

[1]LTCI, Télécom ParisTech, Université Paris-Saclay, Paris, France,
{petr.kuznetsov,thibault.rieutord}@telecom-paristech.fr
[2]UCLA, Los Angeles, USA, yuan.he@cs.ucla.edu

Abstract

The paper proposes a simple topological characterization of a large class of fair adversarial distributed-computing models via *affine tasks*: sub-complexes of the second iteration of the standard chromatic subdivision. We show that the task computability of a model in the class is precisely captured by iterations of the corresponding affine task. Our results generalize and improve all previously derived topological characterizations of distributed-computing models.

1 Introduction

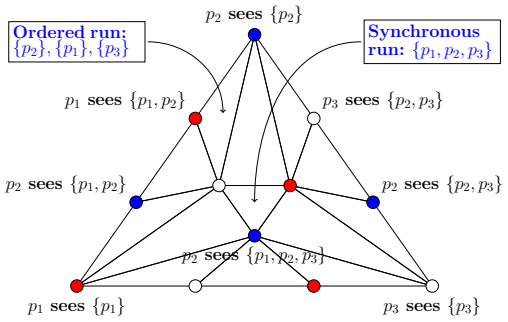
Distributed computing deals with a jungle of models, parameterized by types of failures, synchrony assumptions and communication primitives. Determining relative computability power of these models ("is model A more powerful than model B") is an intriguing and important problem. This paper deals with a large class of *shared-memory* models in which a set of *crash-prone asynchronous* processes communicate via invoking operations on a collection of shared objects. By default, we assume that the shared objects include atomic read-write registers.

Topology of wait-freedom. The *wait-free* model of computation [17] makes no assumptions on when and where failures might occur. Herlihy and Shavit proposed an elegant characterization of wait-free (read-write) task computability via the existence of a specific *simplicial* map from geometrical structures describing inputs and outputs [20]. A task T has a wait-free solution using read-write registers if and only if there exists a simplicial, chromatic map from a *subdivision* of the *input simplicial complex*, describing the inputs of T to the *output simplicial complex*, describing the outputs of T. In particular, we can choose this subdivision to be the iterated *standard chromatic* subdivision (Chr, Figure 1(a)). This subdivision precisely captures the output complex of the *immediate snapshot* (IS) task [5]. By solving the IS task iteratively, where the current iteration output is used as the input value for the next one, we obtain the *iterated* immediate snapshot (IIS) model. Therefore, we can give the following formulation of the celebrated *Asynchronous Computability Theorem (ACT)* [20]:

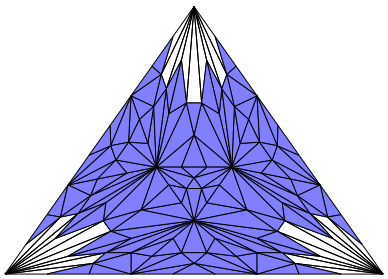
> A task $T = (\mathcal{I}, \mathcal{O}, \Delta)$, where $\mathcal{I}$ is the input complex, $\mathcal{O}$ is an output complex, and Δ is a carrier map from $\mathcal{I}$ to sub-complexes of $\mathcal{O}$, is wait-free solvable if and only if there exists a natural number ℓ and a simplicial map $\phi : \mathrm{Chr}^\ell(\mathcal{I}) \to \mathcal{O}$ carried by Δ (informally, respecting the task specification Δ).

The theorem can be interpreted as: the set of wait-free (read-write) solvable task is precisely the set of tasks solvable in the IIS model. The ability of (iteratively) solving the IS task allows us

*This is the full version of a submitted article entitled "An Asynchronous Computability Theorem for Fair Adversaries". This work has been supported by the Franco-German DFG-ANR Project DISCMAT (14-CE35-0010-02) devoted to connections between mathematics and distributed computing.



(a) $\mathrm{Chr}^1(\mathbf{s})$, the output complex of IS



(b) $\mathcal{R}_1$, the affine task of 1-resilience

Figure 1: Subsets of the iterated standard chromatic subdivision

to solve any task in the wait-free model. Hence, from the task computability perspective, the IS task is a finite (or, as we explain below, *compact*) representation of the wait-free model.

Adversaries. Given that many fundamental tasks are not solvable in the wait-free way [3, 20, 27], more general models were considered. Delporte et al. [9] introduced the notion of an *adversary*, a a collection $\mathcal{A}$ of process subsets, called *live sets*. A run is in the corresponding *adversarial $\mathcal{A}$-model* if the set of processes taking infinitely many steps in it is in $\mathcal{A}$.

For example, the *t-resilient* n-process model is defined via an adversary $\mathcal{A}_t$ that consists of all live sets of size $n - t$ or more. $\mathcal{A}_t$ is *superset-closed* [23], as it contains all supersets of its elements.

Notice that, assuming the conventional "longest-prefix" metric [2], the t-resilient model is *non-compact*: in particular, it does not contain its limit points. Consider, for example, the 1-resilient 3-process system and an infinite "solo" run in which exactly one process takes steps. All finite prefixes of this run are in the model, but the run itself is not.

Saraph et al. [28] recently proposed a direct characterization of t-resilient task computability via a specific task $\mathcal{R}_t$, defined as a restriction of the *double* immediate snapshot task: the output complex of the task is a subcomplex consisting of *all* simplices of the second degree of the standard chromatic subdivision of the task's input complex, except the simplices adjacent to the $(n - t - 1)$-skeleton of the input complex (Figure 1(b) describes the 1-resilient 3-process case). The task consists in solving chromatic simplex agreement [5, 20] on a the corresponding subcomplex of $\mathrm{Chr}^2\,\mathbf{s}$. Such tasks are called *affine* [13, 15], as the geometrical representation of their output complexes are unions of affine spaces.

Solving a task T in the t-resilient model is then equivalent to finding a map from iterations of $\mathcal{R}_t$ (applied to the input complex of T) to the output complex of T. Therefore, $\mathcal{R}_t$ is a compact representation of the model of t-resilience.

Similarly, the affine task of the *k-obstruction free* adversary, consisting of all process subsets of size at most k, was recently determined by Gafni et al. [13]. Note that such an adversary is *symmetric* [30], as it only depends on the *sizes* of live sets, and not on process identifiers.

Topology of fair adversaries. In this paper, we present a compact topological characterization of the large class of *fair* adversarial models [24]. Informally, an adversary is fair if its ability to solve set consensus does not change if only a subset of the processes are participating. Fair adversaries subsume symmetric and superset-closed ones. We define an affine task $\mathcal{R}_\mathcal{A}$ that captures the task computability of any fair adversary $\mathcal{A}$. Our characterization can be put as the following generalization of the ACT [20]:

> A task $T = (\mathcal{I}, \mathcal{O}, \Delta)$ is solvable in a fair adversarial $\mathcal{A}$-model if and only if there exists a natural number ℓ and a simplicial map $\phi : \mathcal{R}_\mathcal{A}^\ell(\mathcal{I}) \to \mathcal{O}$ carried by Δ.

This result generalizes all existing topological characterizations of distributed computing models [13, 15, 20, 28], as it applies to *all* fair adversaries (and not only t-resilient and k-obstruction-free) and *all* tasks (and not only *colorless*). Furthermore, our characterization is compact: we match

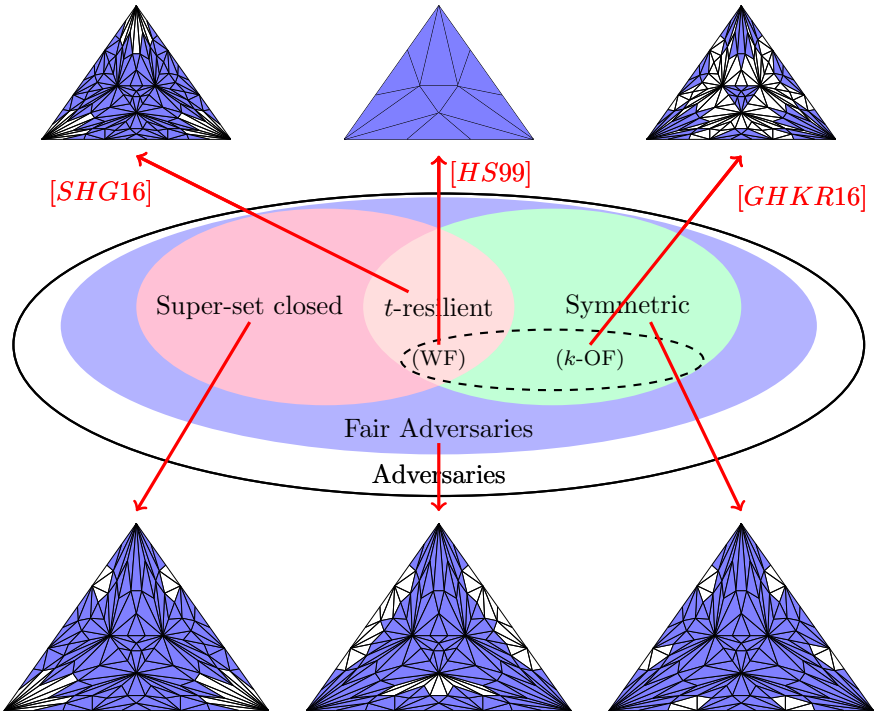


Figure 2: Earlier topological characterizations (wait-free [20], t-resilient [28], and k-obstruction-free [13]) and our contribution: e.g., affine tasks for a superset-closed adversary, a symmetric adversary, and a general fair adversary

(potentially complicated and non-compact) fair adversarial models with simple finite affine tasks, defined as sub-complexes of the second-degree standard chromatic subdivisions.

Given that there are only finitely many such affine tasks, we conclude that there can only be finitely many equivalence classes of fair adversarial models. We believe that the results can be extended to all "practical" restrictions of the wait-free model of computations, beyond fair adversaries, which may potentially result in a complete computability theory for distributed computing.

Roadmap. Section 2 gives preliminary definitions, Section 3 recalls the notion an adversary, and Section 4 defines task $\mathcal{R}_\mathcal{A}$ for a fair adversary $\mathcal{A}$. In Section 5, we show that $\mathcal{R}_\mathcal{A}^*$, the iterated $\mathcal{R}_\mathcal{A}$ model, can be simulated in $\mathcal{A}$. In Section 6, we show that any task solvable in the adversarial $\mathcal{A}$-model can be solved in $\mathcal{R}_\mathcal{A}^*$. Section 7 reviews related work and Section 8 concludes the paper.

2 Preliminaries

Let Π be a system composed of n asynchronous processes, $p_1, \ldots, p_n$. We consider two models of communication: (1) *atomic snapshots* [1] and (2) *iterated immediate snapshots* [5, 20].

Communication models. The atomic-snapshot (AS) memory is represented as a vector of shared variables, where each process p_i is associated with a distinct position i. The memory can be accessed with two operations: *update* and *snapshot*. An *update* operation performed by p_i modifies the value at position i and a *snapshot* returns the current state of the memory.

In the iterated immediate snapshot (IIS) model, processes proceed through a sequence of independent memories $M_1, M_2, \ldots$. Each memory M_r is accessed by a process with a single *WriteSnapshot* operation [4]: the operation performed by p_i takes a value v_i and returns a set V_{ir} of values submitted by the processes (w.l.o.g, we assume that the values of different

processes are distinct), so that the following properties are satisfied: (self-inclusion) $v_i \in V_{ir}$; (containment) $(V_{ir} \subseteq V_{jr}) \vee (V_{jr} \subseteq V_{ir})$; and (immediacy) $v_i \in V_{jr} \Rightarrow V_{ir} \subseteq V_{jr}$.

Protocols, runs and models. A *protocol* here is a deterministic distributed automaton that, for each local state of a process, stipulates which operation and which state transition the process is allowed to perform. A *run* of a protocol is defined as a possibly infinite sequence of alternating states and operations. A *model* is a set of runs.

In the IIS communication model, we assume that processes run the *full-information* protocol: the first value each process writes is its *initial state*. For each $r > 1$, the outcome of the WriteSnapshot operation on memory M_{r-1} is submitted as the input value for the WriteSnapshot operation on memory M_r. After a certain number of iterations, a process may gather enough information to produce an irrevocable *output* value.

Failures and participation. In an infinite run of the AS model, a process that takes only finitely many steps is called *faulty*, otherwise it is called *correct*. We assume that in its first step, a process writes its initial state in the shared memory using the *update* operation. If a process completed this first step in a given run, it is said to be *participating*, and the set of participating processes is called the *participating set*. Note that in an infinite run, every correct process is participating.

In contrast, the IIS model does not have the notion of a faulty process. Instead, a process may appear "slow" [6, 11, 26], i.e., be late in accessing iterated memories from some point on so that some "faster" processes do not see them.

Tasks. In this paper, we focus on distributed *tasks* [20]. A process invokes a task with an *input* value and the task returns an *output* value, so that the inputs and the outputs across the processes, respect the task specification. Formally, a *task* is defined through a set $\mathcal{I}$ of input vectors (one input value for each process), a set $\mathcal{O}$ of output vectors (one output value for each process), and a total relation $\Delta : \mathcal{I} \mapsto 2^{\mathcal{O}}$ that associates each input vector with a set of possible output vectors. We require that Δ is a *carrier* map: $\forall \tau, \sigma \in \mathcal{I}$, $\tau \subset \sigma$: $\Delta(\tau) \subseteq \Delta(\sigma)$. An input $\perp$ denotes a *non-participating* process and an output value $\perp$ denotes an *undecided* process. Check [18] for more details on the definition.

In the task of *k-set consensus* [7], input values are in a set of values V ($|V| \geq k+1$), output values are in V, and for each input vector I and output vector O, $(I, O) \in \Delta$ if the set of non-$\perp$ values in O is a subset of values in I of size at most k. The special case of 1-set consensus is called *consensus* [10].

A protocol solves a task $T = (\mathcal{I}, \mathcal{O}, \Delta)$ in a model M, if it ensures that in every infinite run of M in which processes start with an input vector $I \in \mathcal{I}$, there is a finite prefix R of the run in which: (1) decided values form a vector $O \in \mathcal{O}$ such that $(I, O) \in \Delta$, and (2) all correct processes decide.

Standard chromatic subdivision and IIS. We use the standard language of *simplicial complexes* [18, 29] to give a combinatorial representation of the IIS model. A *simplicial complex* is defined as a set of *vertices* and an inclusion-closed set of vertex subsets, called *simplices*. The dimension of a simplex is the number of its vertices minus one. Any subset of these vertices is called a *face*.

A simplicial complex is *pure* (of dimension n) if each of its simplices is contained in a simplex of dimension n. A simplicial complex is *chromatic* if it is equipped with a *coloring function*—a non-collapsing simplicial map χ from its vertices to the *standard* $(n-1)$*-simplex* $\mathbf{s}$ of n vertices, in one-to-one correspondence with n *colors* $1, 2, \ldots, n$. With some abuse of notation, processes may be referred to by their identifiers and χ may be used to obtain the set of processes associated with a simplex.

The *standard chromatic subdivision* [20] of $\mathbf{s}$, denoted $\mathrm{Chr}\,\mathbf{s}$ and depicted in Figure 1(a), is a complex where vertices of $\mathrm{Chr}\,\mathbf{s}$ are couples (v, σ), where v is a vertex of $\mathbf{s}$ and σ is a face of $\mathbf{s}$ containing v, and simplices are sets of vertices (v_1, σ_1), ..., (v_m, σ_m) satisfying the properties of immediate snapshot. $\mathrm{Chr}\,\mathbf{s}$ is indeed a *subdivision* of $\mathbf{s}$: in particular, its geometric realization $|\mathrm{Chr}\,\mathbf{s}|$ is homeomorphic to $|\mathbf{s}|$, the geometric realization of $\mathbf{s}$ [22]. If we *iterate* this subdivision m times, each time applying Chr to each of the simplices, we obtain the m^{th} chromatic subdivision, $\mathrm{Chr}^m\, C$.

$\mathrm{Chr}^m\,\mathbf{s}$ precisely captures the m-round (full-information) IIS model, denoted IS^m [20].

Consider a simplex of $\sigma \in \mathrm{Chr}^m\,\mathbf{s}$ and take any $1 \leq m' < m$. The *carrier* of σ with respect to $\mathrm{Chr}^{m'}\,\mathbf{s}$, denoted by $carrier(\sigma, \mathrm{Chr}^{m'}\,\mathbf{s})$, is the smallest simplex $\sigma' \in \mathrm{Chr}^{m'}\,\mathbf{s}$ such that the geometric realization of σ, $|\sigma|$, is included in $|\sigma'|$. For example, given a vertex v of $\mathrm{Chr}^m\,\mathbf{s}$, $carrier(v, \mathrm{Chr}^m\,\mathbf{s})$ is the set of all processes *seen* by the process $\chi(v)$ in the corresponding run of IS^m.

Simplex agreement and affine tasks. In a general simplex agreement task, every process is given, as an input, a vertex of its color in the standard simplex $\mathbf{s}$, and is expected to output a vertex of $\mathcal{C}$ of its color, so that the outputs form a simplex of $\mathcal{C}$ respecting the inputs. In the instances of simplex agreement considered in characterizations of wait-free task computability [5,20], inputs were vertices of $\mathbf{s}$ and $\mathcal{C}$ was a chromatic subdivision of $\mathbf{s}$. Affine tasks can be seen as a generalization of simplex agreement tasks considered in [5, 20], where the output complex is no longer a subdivision but a subset of some iteration of the standard chromatic subdivision.

Formally, let L be a pure subcomplex of $\mathrm{Chr}^l\,\mathbf{s}$ for some $l \in \mathbb{N}$. The affine task associated with L is then defined as $(\mathbf{s}, L, \Delta)$, where, for every face $\mathbf{t} \subseteq \mathbf{s}$, $\Delta(\mathbf{t}) = L \cap \mathrm{Chr}^l\,\mathbf{t}$. Notice that $L \cap \mathrm{Chr}^l(\mathbf{t})$ can be empty, in which case no participating process is required to output.

With a slight abuse of notations, we use L to denote the affine task associated with L. By running m iterations of this task, we obtain L^m, a subcomplex of $\mathrm{Chr}^{lm}\,\mathbf{s}$, corresponding to a subset of IS^{lm} runs (each of the m iterations includes l IS rounds). We denote by L^* the set of infinite runs of the IIS model where every prefix restricted to a multiple of l IS rounds belongs to the subset of IS^{lm} runs associated with L^m.

3 Adversaries and agreement functions

An *adversary* $\mathcal{A}$ is a set of subsets of Π, called *live sets*, $\mathcal{A} \subseteq 2^\Pi$. An AS run is *$\mathcal{A}$-compliant* if the set of correct processes in that run belongs to $\mathcal{A}$. The *(adversarial) $\mathcal{A}$-model* is defined as the set of $\mathcal{A}$-compliant runs.

Consider an adversary $\mathcal{A}$ and a function $\alpha : 2^\Pi \to \{0, \ldots, n\}$. We say that α is the *agreement function* of $\mathcal{A}$ if for each $P \in 2^\Pi$, in the set of $\mathcal{A}$-compliant runs in which no process in $\Pi \setminus P$ participates, $\alpha(P)$-set consensus can be solved, but $(\alpha(P)-1)$-set consensus cannot [24]. Intuitively, $\alpha(P)$ is the best level of set consensus that can be reached in the $\mathcal{A}$-model when only processes in P are allowed to participate. By convention, if P does not contain a live set, we set $\alpha(P) = 0$. Note that for any adversary $\mathcal{A}$, its agreement function α is *monotonic*: $P \subseteq P' \Rightarrow \alpha(P) \leq \alpha(P')$.

For any monotonic function $\alpha : 2^\Pi \to \{0, \ldots, n\}$, we can define a natural model (a subset of AS runs) as follows:

Definition 1 (α-model)**.** *The* α-model *is the set of infinite runs satisfying the following property: if P is the participating set, then $\alpha(P) \geq 1$ and at most $\alpha(P) - 1$ processes in P are faulty.*

An adversary is *superset-closed* [23] if each superset of a set of an element of $\mathcal{A}$ is also an element of $\mathcal{A}$, i.e., $\forall S \in \mathcal{A}, \forall S' \subseteq \Pi, S \subseteq S'$: $S' \in \mathcal{A}$. An adversary $\mathcal{A}$ is *symmetric* if it only depends on the *sizes* of live sets, and not on process identifiers.: $\forall S \in \mathcal{A}, \forall S' \subseteq \Pi, |S'| = |S| \implies S' \in \mathcal{A}$.

For $P \in 2^\Pi$, let $\mathcal{A}|_P = \{S | S\mathcal{A}, S \subseteq P\}$ and let $csize(\mathcal{A}|_P)$ denote the size of the *minimal hitting set* of $\mathcal{A}|_P$, i.e., the minimal subset of P that intersects with each element in $\mathcal{A}|_P$. It is shown in [24] that the corresponding agreement function α can be defined using the *set consensus* function *setcon* [14]: $\alpha(P) = setcon(\mathcal{A}|_P)$. Moreover, for any superset-closed adversary $\mathcal{A}$, we have $\alpha(P) = setcon(\mathcal{A}|_P) = csize(\mathcal{A}|_P)$ and if $\mathcal{A}$ is symmetric, we have $\alpha(P) = setcon(\mathcal{A}|_P) = |\{k \in \{1, \ldots, |P|\} : \exists S \in \mathcal{A}, |S| = k\}|$.

α-adaptive set consensus. The abstraction of *α-adaptive set consensus* [24] can be accessed with a *propose*(v) operation, where $v \in V$, and ensures that (termination) every operation invoked by a correct process eventually returns, (validity) every returned value is the argument of a preceding *propose* invocation, and (α-agreement) at any point of the execution, the number of distinct

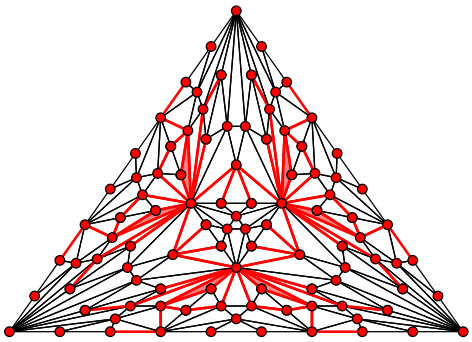


Figure 3: The 2-contention complex is shown in red in dimension 2.

returned values does not exceed $\alpha(P)$, where P is the set of processes that took at least one step up to that point. We also consider the following variant of the α-model:

Definition 2 (α-set-consensus model)**.** *The* α-set-consensus model *is the set of infinite runs in which processes can access atomic-snapshot memory and* α*-adaptive set-consensus objects and the following property is satisfied: if* P *is the participating set, then* $\alpha(P) \geq 1$.

The following result will be instrumental in our proofs:

Theorem 1. *[24] For any* α*, an agreement function and any task* T*, if* T *is solvable in the* α*-model, then* T *is solvable in the* α*-set-consensus model.*

Fair adversaries. Informally, an adversary is *fair* [24] if a subset Q of participating processes P cannot achieve better set consensus than the whole set of participants. More precisely, for an adversary $\mathcal{A}$, and $Q \subseteq P$, we define $\mathcal{A}|_{P,Q} = \{S \in \mathcal{A} : (S \subseteq P) \wedge (S \cap Q \neq \emptyset)\}$.

Definition 3. *[Fair adversary] An adversary* $\mathcal{A}$ *is fair if and only if:*

$$\forall P \subseteq \Pi, \forall Q \subseteq P, setcon(\mathcal{A}|_{P,Q}) = min(|Q|, setcon(\mathcal{A}|_P)).$$

Superset-closed and symmetric adversaries are fair [24]. It turns out that the task computability of a fair adversary is captured precisely by the corresponding α-model, i.e., they solve *the same* set of tasks (we say that the models are *equivalent*).

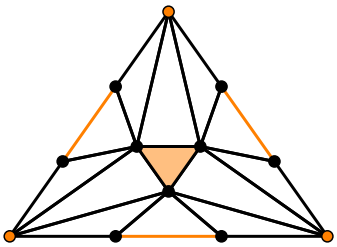
Theorem 2. *[24] For any fair adversary* $\mathcal{A}$*, a task is solvable in the adversarial* $\mathcal{A}$*-model if and only if it is solvable in the* α*-model.*

4 Affine task for fair adversarial models

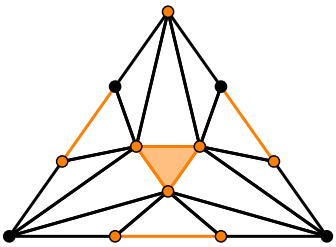
Given a fair adversary $\mathcal{A}$ and its associated agreement function α, we define the affine task $\mathcal{R}_\mathcal{A}$, a subcomplex of the second degree of the standard chromatic subdivision $\text{Chr}^2\,\mathbf{s}$. In Sections 5 and 6, we show that $\mathcal{R}_\mathcal{A}^*$, i.e., the model of IIS runs obtained by iterating $\mathcal{R}_\mathcal{A}$, is equivalent to the α-model regarding task solvability.

Agreement and contention simplices. To get an intuitive understanding of the defintion $\mathcal{R}_\mathcal{A}$, we first show how to restrict $\text{Chr}^2\,\mathbf{s}$ so that any subset of processes is able to solve k-set consensus in the resulting IIS model. For a vertex $v \in \text{Chr}^2\,\mathbf{s}$, we define $View^1(v)$ and $View^2(v)$ as the sets of processes seen by the corresponding process $\chi(v)$ in, respectively, in the first and in the second IS.

In a subcomplex of $\text{Chr}^2\,\mathbf{s}$ or, put differently, a subset of two rounds of the IIS model, a natural way to solve k-set consensus is to make every process deterministically select a process in the smallest $View^1$ it finds after completing the second IS. The solution is correct if the subcomplex



(a) Critical simplices for the α-model with $\alpha(P) = min(|P|, 1)$ (1-obstruction-freedom)



(b) Critical simplices of the α-model corresponding to the adversary $\mathcal{A} = \{\{p_1\}, \{p_2, p_3\}, \{p_1, p_2, p_3\}\}$

Figure 4: Critical simplices are displayed in orange (with p_1 associated with the vertex on the top).

of Chr^2 guarantees that in every simplex, there are at most k such *IS1* views. For example, the subcomplex that captures *consensus* (1-set consensus) can thus be defined as the set of simplices of Chr^2 adjacent to the "corners" (0-dimensional vertices of $\mathbf{s}$).

However, to solve k-set consensus among *any subset of processes*, we need to further restrict this affine task, so that the desired properties are respected for *any* processes subset. In a simplex $\delta \in \text{Chr}^2\,\mathbf{s}$, vertices v and v' are *contending* if their $View^1$ and $View^2$ are ordered in the opposite way: $View^1(v)$ is a proper subset of $View^1(v')$ and $View^2(v')$ is a proper subset of $View^2(v)$, or vice versa. If every two vertices of δ are contending, then we say that δ is a 2-*contention* simplex. Formally:

Definition 4. *[2-Contention simplices]* $\forall \sigma \in \text{Chr}^2\,\mathbf{s} : \sigma \in Cont_2 \Leftrightarrow \forall v, v' \in \sigma, v \neq v' :$

$$((View^1(v) \subsetneq View^1(v')) \wedge (View^2(v') \subsetneq View^2(v))) \vee ((View^1(v') \subsetneq View^1(v)) \wedge (View^2(v) \subsetneq View^2(v'))).$$

Note that the set of 2-contention simplices is inclusion-closed: any face of a 2-contention simplex is also a 2-contention simplex. Therefore, we can define the 2-*contention complex* as the set of all 2-contention simplices. Examples of 2-contention simplices in a 3-process system are given in Figure 3.

By restricting $\text{Chr}^2\,\mathbf{s}$ to the set of maximal simplices which do not have faces of more than k contending vertices, we get affine task $\mathcal{R}_{k-OF}$ which allows any subset if processes to solve k-set consensus and, thus, captures the k-obstruction-free adversary (see Figure 6a for $\mathcal{R}_{1-OF}$ in a 3-process system):

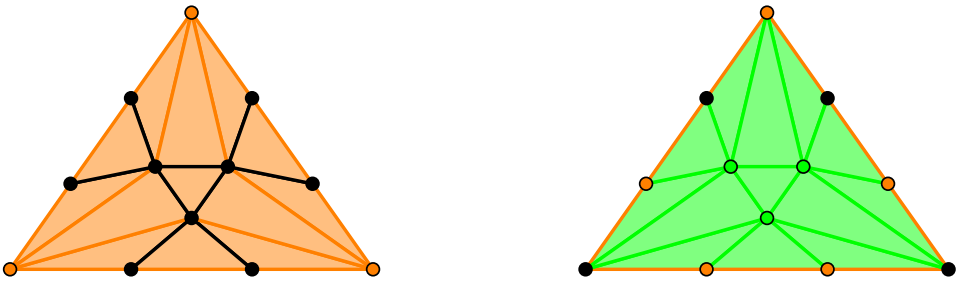
Definition 5. *[$\mathcal{R}_{k-OF}$]* $\forall \sigma \in \text{Chr}^2\,\mathbf{s}, dim(\sigma) = n-1 : \sigma \in \mathcal{R}_{\mathcal{A}} \Leftrightarrow$

$$\forall \theta \subseteq \sigma : \theta \in Cont_2 \implies dim(\theta) - 1 \leq k.$$

Agreement vs. participation. The agreement function of an adversary may define different levels of agreement for different participating sets. In iterated affine tasks, participation is captured by views of the processes, i.e., $carrier(v, \mathbf{s})$ for their vertices v. Intuitively, ensuring that "enough" processes participate requires sufficiently large views, which may conflict with selecting "leader" processes with the smallest views. Therefore, the number of potential leaders might grow with the growing participating set P. Thus, intuitively, each time a process obtains a $View^1$ associated with a higher agreement level, a new leader might arise. We thus introduce the notion of a *critical set*, i.e., the set of processes whose participation increases the agreement level of the system. More precisely, a simplex $\sigma \in \text{Chr}\,\mathbf{s}$ is a *critical simplex* if: (1) all its vertices share the same carrier; and (2) the set consensus power associated with $carrier(\sigma, \mathbf{s})$ is strictly greater than the set consensus power of $\chi(carrier(\sigma, \mathbf{s})) \setminus \chi(\sigma)$.

Definition 6. $\forall \sigma \in \text{Chr}\,\mathbf{s}$*:*

$$Critical_\alpha(\sigma) = (\forall v \in \sigma : carrier(v, \mathbf{s}) = carrier(\sigma, \mathbf{s})) \wedge (\alpha(\chi(carrier(\sigma, \mathbf{s})) \setminus \chi(\sigma)) < \alpha(\chi(carrier(\sigma, \mathbf{s})))).$$



(a) Concurrency map for the α-model with $\alpha(P) = min(|P|, 1)$. (1-obstruction-freedom)

(b) Concurrency map of the α-model corresponding to the adversary $\mathcal{A} = \{\{p_1\}, \{p_2, p_3\}, \{p_1, p_2, p_3\}\}$.

Figure 5: Examples of concurrency maps in two models of 3-process system, a color set to green corresponding to a concurrency value equal to 2, orange to 1, and black to 0. Note that p_1 is the vertex on the top.

Examples of critical simplices for 3-process models are given in Figure 4.

Given a simplex $\sigma \in \mathrm{Chr}\,\mathbf{s}$, let $\mathcal{CS}_\alpha(\sigma)$ denote the set of critical simplices in σ, i.e., $\mathcal{CS}_\alpha(\sigma) = \{\sigma' \subseteq \sigma : \mathit{Critical}_\alpha(\sigma)\}$. Moreover, let $\mathcal{CSM}_\alpha(\sigma)$ (critical simplices members) be the set of vertices of σ which belong to some critical simplex in σ, i.e., $\mathcal{CSM}_\alpha(\sigma) = \cup_{\sigma' \in \mathcal{CS}_\alpha(\sigma)} \sigma'$. Similarly, let $\mathcal{CSV}_\alpha(\sigma)$ (critical simplices view) be the union of all processes observed by critical simplices in σ, i.e., $\mathcal{CSV}_\alpha(\sigma) = \mathit{carrier}(\mathcal{CSM}_\alpha(\sigma), \mathbf{s})$.

Affine tasks for fair adversaries. Critical simplices provide a mechanism to select leaders. Agreement is solved by making every process to deterministically select a proposal in View^1 of a critical simplex. If, however, critical sets do not offer proposals, the contention level of remaining processes should be sufficiently restricted to enable the required (by α) level of set consensus. We define this restriction using the following:

Definition 7. *[Concurrency map]* $\forall \sigma \in \mathrm{Chr}\,\mathbf{s}$:

$$\mathit{Conc}_\alpha(\sigma) = \mathit{max}(0 \cup \{\alpha(\chi(\mathit{carrier}(\sigma', \mathbf{s}))), \sigma' \in \mathcal{CS}_\alpha(\sigma)\}).$$

Examples of concurrency maps for two models in a 3-process system are given in Figure 5.

Affine task $\mathcal{R}_\mathcal{A}$. The affine task $\mathcal{R}_\mathcal{A} \subseteq \mathrm{Chr}^2\,\mathbf{s}$ is defined as follows:

Definition 8. *[$\mathcal{R}_\mathcal{A}$]* $\forall \sigma \in \mathrm{Chr}^2\,\mathbf{s}, \mathit{dim}(\sigma) = n-1 : \sigma \in \mathcal{R}_\mathcal{A} \Leftrightarrow \forall \theta \subseteq \sigma, \theta' = \mathit{carrier}(\theta, \mathrm{Chr}\,\mathbf{s})$:

$$(\theta \in \mathit{Cont}_2) \wedge (\chi(\theta) \cap (\chi(\mathcal{CSM}_\alpha(\mathit{carrier}(\sigma, \mathrm{Chr}\,\mathbf{s}))) \cup \chi(\mathcal{CSV}_\alpha(\theta'))) = \emptyset \implies \mathit{dim}(\theta) + 1 \leq \mathit{Conc}_\alpha(\theta').$$

Intuitively, a simplex $\sigma \in \mathrm{Chr}^2\,\mathbf{s}$ is in $\mathcal{R}_\mathcal{A}$ if any of its "non-critical" subsets that cannot "rely" on the critical simplices in achieving α-adaptive set consensus has a sufficiently low contention level to solve α-adaptive set consensus on their own.

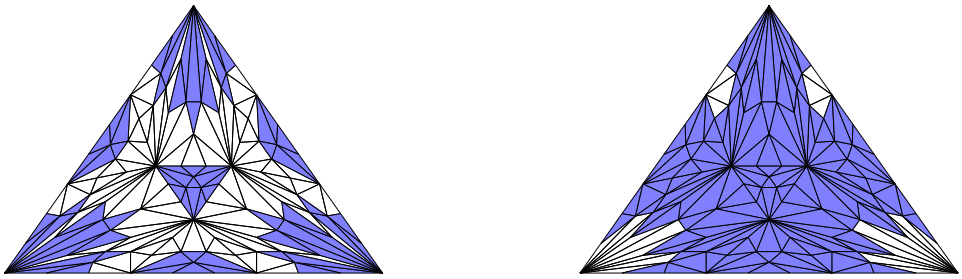
Some examples of affine tasks for some α-models in a 3-process system are depicted in Figure 6.

5 From α-model to $\mathcal{R}_\mathcal{A}$

Let us fix a fair adversary $\mathcal{A}$. Let α be its agreement function, and let T be any task that can be solved in $\mathcal{R}^*_\mathcal{A}$. To show that T is solvable with a fair adversary $\mathcal{A}$, we present an algorithm that, in the α-model, *solves* $\mathcal{R}_\mathcal{A}$, i.e., solves the chromatic simplex agreement task on the complex $\mathcal{R}_\mathcal{A}$. By iterating this task, we can simulate a run of $\mathcal{R}^*_\mathcal{A}$ and thus get a solution to T in the α-model.

5.1 Algorithm

The solution of $\mathcal{R}_\mathcal{A}$ is presented in Algorithm 1. Every process accesses two immediate snapshot objects: *FirstIS*, proposing its initial state, and *SecondIS*, proposing the outcome of *FirstIS*.



(a) Affine task for the α-model with $\alpha(P) = min(|P|, 1)$. (1-obstruction-freedom)

(b) Affine task of the α-model corresponding to the adversary $\mathcal{A} = \{\{p_1\}, \{p_2, p_3\}, \{p_1, p_2, p_3\}\}$.

Figure 6: Some examples of affine tasks $\mathcal{R}_\mathcal{A}$ in blue (with p_1 associated with the vertex on the top).

Recall that outcomes of *SecondIS* form a simplex in $\text{Chr}^2\,\mathbf{s}$ [22]. To ensure that this simplex is in $\mathcal{R}_\mathcal{A}$, after finishing *FirstIS*, processes wait for their turns to proceed to *SecondIS*.

In this *waiting phase* (Lines 5–10), processes check a specific condition on the IS outcomes that they share with each other in registers $IS1[1, \ldots, n]$ and $IS2[1, \ldots, n]$. Each process p_i periodically checks *IS1* and *IS2* to update local variables *critical* and *concurrency*. Here *critical* is the set of processes which belongs to a set of processes whose *FirstIS* outputs form a critical simplex (Line 8). The value of *concurrency* is used to limit the level of contention non-critical processes may obtain. The value of *concurrency* is initially set to the agreement power associated with the process *IS1* view (line 4). Afterwards, during the waiting phase, it may be increased to the agreement power associated with the *IS1* view of a "terminated" critical simplcex, i.e., a critical simplex for which all its processes have been provided with *SecondIS* outputs (Line 9).

Algorithm 1: Algorithm solving R_α in the α-model for process p_i.

```
1  Shared: registers IS1[1],... IS[n] ∈ 2^Π, initially ∅; IS2[1],..., IS[n] ∈ 2^(2^Π), initially ∅; immediate
     snapshot objects FirstIS, SecondIS;
2  Local: critical[1...n] ∈ {true, false}, initially false; concurrency ∈ ℕ, initially
     0, level[1...n] ∈ 2^Π;

3  IS1[i] ← FirstIS(InitialState);
4  Concurrency ← α(IS1[i]);
5  while (¬critical[i]) ∧ (|{p_j ∈ IS1[i] : IS2[j] = ∅ ∧ IS1[j] ≠ IS1[i]}| ≥ concurrency) do
6  |  forall j ∈ {1,...,n} do
7  |  |  level[j] ← {p_k ∈ Π, IS1[k] = IS1[j]};
8  |  |  critical[j] ← (α(IS1[j]) > α(IS1[j] \ level[j]));
9  |  |  if (α(IS1[j]) > α(IS1[j] \ {p_k ∈ level[j], IS2[k] ≠ ∅})) then
10 |  |  |  concurrency ← max(α(IS1[j]), concurrency);
11 IS2[i] ← SecondIS(IS1[i]);
12 return(IS2[i]);
```

Intuitively, the waiting phase is used to ensure that *critical processes*, i.e., members of critical simplices, have a higher priority in proceeding with *SecondIS*. A process may proceed to its *SecondIS* as soon as it knows that it belongs to some *critical* simplex. Non-critical processes are allowed to exit their waiting phase according to the level of *concurrency* computed. A non-critical process wait to know that the number of processes, which have or may obtain a strictly smaller *IS1* view (included in $IS1[i]$ and with an $IS1[j] \neq IS1[i]$) and which did not terminate yet ($IS2[j] = \emptyset$), is strictly smaller than the observed level of *concurrency*. This allows to ensure that the possible contention level of non-critical processes scales according to terminated critical simplex associated agreement power. This is done either explicitely, if *concurrency* has been updated on line 10, or implicitely if *concurrency* is still set to the agreement power associated with the process first IS view from line 4.

5.2 Proof of correctness.

In order to show that Algorithm 1 solves $\mathcal{R}_\mathcal{A}$ in the α-model corresponding to the fair adversary $\mathcal{A}$, we need to show that (1) every correct process eventually outputs and that (2) the set of ouputs belongs to a simplex in $\mathcal{R}_\mathcal{A}$. Note that as processes execute two consecutive immediate snapshot protocols, all outputs belong to some simplex in $\mathrm{Chr}^2\,\mathbf{s}$. Let us consider a run of the α model in which the particiation is P, hence with $\alpha(P) > 0$.

To show that outputs belong not only to $\mathrm{Chr}^2\,\mathbf{s}$ but to $\mathcal{R}_\mathcal{A}$ and that all correct processes terminate, we mostly rely on the distribution of critical simplices. We are interested in showing that the number of processes failures, required to prevent critical simplices from either appearing in *IS1* or completing their *IS2* computation, scales with the agreement power of the participation. Moreover, we want to show that the less processes fail in such a way, the higher the maximal agreement power associated with a terminated critical simplices becomes.

A process failure may prevent multiple critical simplices to terminate. Indeed, a process may be included in multiple critical simplices, and thus, its failure would prevent multiple critical simplices from terminating. This is why we are interested not in the distribution of critical processes or critical simplices, but instead, in the minimal hitting set size for the set of critical simplices. Let us recall that an hitting set of a set of sets $\mathcal{Q}$, is a set intersecting with all sets from $\mathcal{Q}$, and that *csize* denotes the minimal hitting set size. More precisely, we want to know the minimal hitting set size of (1) any subset of the participation and (2) of the set of critical simplices associated with an agreement power greater than or equal to some level l, i.e., $\{\theta \in \mathcal{CS}_\alpha(\sigma), \alpha(\chi(\mathit{carrier}(\theta, \mathbf{s}))) \geq l\}$.

Let us first look at the case in which no participating process fails before updating their *IS1* output to the memory. In this case, the set of *IS1* views form a simplex $\sigma \in \mathrm{Chr}\,\mathbf{s}$ such that $\chi(\sigma) = \chi(\mathit{carrier}(\sigma, \mathbf{s}))$: The observed processes include all participating processes (inclusion property) but no others. In this setting we can show that the minimal hitting set size of the set of critical simplices associated with an agreement power greater than or equal to some level l, is greater than or equal to the agreement power of the participation minus $l-1$, i.e., $\alpha(\chi(\sigma)) - l + 1$:

Lemma 3. *[Distribution of critical simplices]:* $\forall \sigma \in \mathrm{Chr}\,\mathbf{s}, \forall l \in \mathbb{N}$

$$\chi(\sigma) = \chi(\mathit{carrier}(\sigma, \mathbf{s})) \implies \alpha(\chi(\sigma)) - l + 1 \leq \mathit{csize}(\{\theta \in \mathcal{CS}_\alpha(\sigma), \alpha(\chi(\mathit{carrier}(\theta, \mathbf{s}))) \geq l\}).$$

Proof. Let us fix some integer $l > 0$. To show Lemma 3, we proceed by an induction on σ using the lexicographical order on $(\alpha(\chi(\sigma')), |\chi(\sigma)|)$. For any simplex σ, such that $\alpha(\chi(\sigma)) < l$, the result is trivial as for any (possibly empty) set $\mathcal{Q}$, we have $\mathit{csize}(\mathcal{Q}) \geq 0$. Now consider a simplex $\sigma \in \mathrm{Chr}\,\mathbf{s}$ such that $\chi(\sigma) = \chi(\mathit{carrier}(\sigma))$ and $\alpha(\chi(\sigma)) = k \geq l$. Let us assume by induction that for all $\sigma' \in \mathrm{Chr}\,\mathbf{s}$, if $(\alpha(\chi(\sigma')), |\chi(\sigma')|) <_{lex} (\alpha(\chi(\sigma')), |\chi(\sigma)|)$, then we have:

$$\chi(\sigma) = \chi(\mathit{carrier}(\sigma', \mathbf{s})) \implies \alpha(\chi(\sigma')) - l + 1 \leq \mathit{csize}(\{\theta \in \mathcal{CS}_\alpha(\sigma'), \alpha(\chi(\mathit{carrier}(\theta, \mathbf{s}))) \geq l\}).$$

Now consider the face τ of σ consisting of all vertices of σ with the same carrier as σ, i.e., $\tau = \{v \in \sigma, \mathit{carrier}(v, \mathbf{s}) = \mathit{carrier}(\sigma, \mathbf{s})\}$. Let β be the complement of τ, i.e., $\beta = \sigma \setminus \tau$. Note that $\tau \neq \emptyset$, due to the containment property, and that, $\chi(\mathit{carrier}(\beta, \mathbf{s})) = \chi(\mathit{carrier}(\sigma, \mathbf{s})) \setminus \chi(\tau)$, due to the immediacy property. Therefore, we obtain that $\chi(\mathit{carrier}(\beta, \mathbf{s})) = \chi(\sigma) \setminus \chi(\tau)$, and so that $\chi(\mathit{carrier}(\beta, \mathbf{s})) = \chi(\beta)$. As $(\alpha(\chi(\beta)), |\chi(\beta)|) <_{lex} (\alpha(\chi(\sigma)), |\chi(\sigma)|)$, we obtain that:

$$\alpha(\chi(\beta)) - l + 1 \leq \mathit{csize}(\{\theta \in \mathcal{CS}_\alpha(\beta), \alpha(\chi(\mathit{carrier}(\theta, \mathbf{s}))) \geq l\}). \tag{1}$$

Two cases may arise:

1. If $\alpha(\chi(\beta)) = \alpha(\chi(\sigma))$, then, as $\beta \subseteq \sigma$ we get that $\mathcal{CS}_\alpha(\beta) \subseteq \mathcal{CS}_\alpha(\sigma)$, hence, we can derive from Equation 1 that:

$$\alpha(\chi(\sigma)) - l + 1 \leq \mathit{csize}(\{\theta \in \mathcal{CS}_\alpha(\sigma), \alpha(\chi(\mathit{carrier}(\theta, \mathbf{s}))) \geq l\}).$$

2. If $\alpha(\chi(\beta)) < \alpha(\chi(\sigma))$, then let $m = \alpha(\chi(\sigma)) - \alpha(\chi(\beta)) > 0$ and let us consider any subset τ' of τ such that $|\tau'| > |\tau| - m$. By construction we have $\mathit{carrier}(\tau', \mathbf{s}) = \mathit{carrier}(\sigma, \mathbf{s})$ and by

assumption we have $\chi(carrier(\sigma, \mathbf{s})) = \chi(\sigma)$, and thus, we obtain that $\chi(carrier(\tau', \mathbf{s})) = \chi(\sigma)$. Let us recall that $\forall v \in \tau : carrier(v, \mathbf{s}) = carrier(\tau, \mathbf{s})$, and therefore $Critical_\alpha(\tau')$ if and only if $\alpha(\chi(\sigma) \setminus \chi(\tau')) < \alpha(\chi(\sigma))$.

Given a fair adversary, for any $Q \subseteq P$, we have $\alpha(P) \geq \alpha(P \setminus Q) \geq \alpha(P) - |Q|$[1]. Note that this implies that $|\chi(\tau)| \geq m$. By applying the formula for $P = \chi(\sigma) \setminus \chi(\tau')$ and for $Q = \chi(\tau) \setminus \chi(\tau')$ we get that:

$$\alpha(\chi(\sigma) \setminus \chi(\tau')) \geq \alpha(\chi(\sigma) \setminus \chi(\tau)) \geq \alpha(\chi(\sigma) \setminus \chi(\tau')) - |\chi(\tau) \setminus \chi(\tau')|.$$

But by construction $\chi(\sigma) \setminus \chi(\tau) = \chi(\beta)$ and $|\chi(\tau) \setminus \chi(\tau')| < m$, thus we obtain that:

$$\alpha(\chi(\sigma) \setminus \chi(\tau)) \geq \alpha(\chi(\sigma) \setminus \chi(\tau')) - |\chi(\tau) \setminus \chi(\tau')| \implies \alpha(\chi(\sigma) \setminus \chi(\tau')) < \alpha(\chi(\beta)) + m.$$

As $m = \alpha(\chi(\sigma)) - \alpha(\chi(\beta))$, we get $\alpha(\chi(\sigma) \setminus \chi(\tau')) < \alpha(\chi(\sigma))$, and hence, that $Critical_\alpha(\tau')$.

Since by construction $\beta = \sigma \setminus \tau$, we have $csize(\mathcal{CS}_\alpha(\sigma)) \geq csize(\mathcal{CS}_\alpha(\tau)) + csize(\mathcal{CS}_\alpha(\beta))$. Moreover, as $\alpha(\chi(\sigma)) \geq l$, we have:

$$csize(\{\theta \in \mathcal{CS}_\alpha(\sigma), \alpha(\chi(carrier(\theta, \mathbf{s}))) \geq l\}) \geq csize(\{\theta \in \mathcal{CS}_\alpha(\beta), \alpha(\chi(carrier(\theta, \mathbf{s}))) \geq l\}) + csize(\mathcal{CS}_\alpha(\tau)) \quad (2)$$

But as any subset τ' of τ, such that $|\tau'| > |\tau| - m$, is critial, we have:

$$csize(\mathcal{CS}_\alpha(\tau)) \geq csize(\{\tau' \subseteq \tau, |\chi(\tau')| > |\chi(\tau)| - m\}).$$

Moreover, since $|\chi(\tau)| \geq m$, we have $csize(\{\tau' \subseteq \tau, |\chi(\tau')| > |\chi(\tau)| - m\}) = m$, and hence, that $csize(\mathcal{CS}_\alpha(\tau)) \geq m$. With $m = \alpha(\chi(\sigma)) - \alpha(\chi(\beta))$ and Equations 1 and 2, we obtain:

$$\begin{aligned} \alpha(\chi(\sigma)) - l + 1 &= (\alpha(\chi(\beta)) - l + 1) + m \\ &\leq csize(\{\theta \in \mathcal{CS}_\alpha(\beta), \alpha(\chi(carrier(\theta, \mathbf{s}))) \geq l\}) + csize(\mathcal{CS}_\alpha(\tau)) \\ &\leq csize(\{\theta \in \mathcal{CS}_\alpha(\sigma), \alpha(\chi(carrier(\theta, \mathbf{s}))) \geq l\}) \end{aligned}$$

□

The result of Lemma 3 can be used to generalize it for cases in which not all participating processes shared their *IS1* outputs to the memory. If so, the minimal hitting set size decreases proportionnaly with the number of missing outputs:

Corollary 4. *For any $\sigma \in \mathrm{Chr}\,\mathbf{s}$, we have:*

$$\alpha(\chi(carrier(\sigma, \mathbf{s}))) - l - |\chi(carrier(\sigma, \mathbf{s})) \setminus \chi(\sigma)| + 1 \leq csize(\{\theta \in \mathcal{CS}_\alpha(\sigma), \alpha(\chi(carrier(\theta, \mathbf{s}))) \geq l\}).$$

Proof. Consider some $\sigma \in \mathrm{Chr}\,\mathbf{s}$. By construction, σ is a sub-simplex of some simplex σ' such that $\chi(carrier(\sigma, \mathbf{s})) = \chi(carrier(\sigma', \mathbf{s})) = \chi(\sigma')$. Hence, we can apply Lemma 3 on σ' and obtain that:

$$\alpha(\chi(carrier(\sigma, \mathbf{s}))) - l + 1 \leq csize(\{\theta \in \mathcal{CS}_\alpha(\sigma'), \alpha(\chi(carrier(\theta, \mathbf{s}))) \geq l\}). \quad (3)$$

But $\mathcal{CS}_\alpha(\sigma) \subseteq \mathcal{CS}_\alpha(\sigma')$ and thus given H a minimal hitting set of $\mathcal{CS}_\alpha(\sigma')$, $H \cup (\chi(\sigma) \setminus \chi(\sigma'))$ is an hitting set of $\mathcal{CS}_\alpha(\sigma')$. Therefore $csize(\{\theta \in \mathcal{CS}_\alpha(\sigma), \alpha(\chi(carrier(\theta, \mathbf{s}))) \geq l\})$ is greater than or equal to $csize(\{\theta \in \mathcal{CS}_\alpha(\sigma'), \alpha(\chi(carrier(\theta, \mathbf{s}))) \geq l\}) + |\chi(\sigma) \setminus \chi(\sigma')|$, and thus, is greater than or equal to $csize(\{\theta \in \mathcal{CS}_\alpha(\sigma'), \alpha(\chi(carrier(\theta, \mathbf{s}))) \geq l\}) + |\chi(carrier(\sigma, \mathbf{s})) \setminus \chi(\sigma)|$. Using this in Equation 3 gives us the property of Corollary 4. □

[1]Indeed, α is an agreement function derived from an adversary $\mathcal{A}$ and so $\alpha(P \cup \{p\}) \leq \alpha(P) + 1$ as by definition $setcon(\mathcal{A}|_{P \cup \{p\}}) \leq setcon(\mathcal{A}|_P) + 1$. By a trivial induction it follows for any subset. Note that this might not be true for generic α models.

Corollary 4 is a generalization of Lemma 3 to account for a partial set of first immediate snapshot outputs. This can be used to show the liveness of the algorithm:

Lemma 5. *Algorithm 1 provides outputs to all correct processes if executed in the α-model corresponding to some fair adversary $\mathcal{A}$.*

Proof. Let P be the participating set and let us assume that a correct process never terminates. Let p be one of the correct processes which do not terminates with the smallest *IS1* view among them, and let $v \in \operatorname{Chr} \mathbf{s}$ be the vertex corresponding to its *IS1* view. Let $\sigma \in \operatorname{Chr} \mathbf{s}$ be the simplex corresponding to the set of *IS1* views reached after *IS1* has been updated for the last time.

Due to the immediacy property, processes in $\mathit{carrier}(v, \mathbf{s})$ must be associated to a vertex v' such that $\mathit{carrier}(v', \mathbf{s}) \subseteq \mathit{carrier}(v, \mathbf{s})$, hence, with $\alpha(\chi(\mathit{carrier}(v', \mathbf{s}))) \leq \alpha(\chi(\mathit{carrier}(v, \mathbf{s})))$. Therefore, in any completion of σ to a simplex $\sigma' \in \operatorname{Chr} \mathbf{s}$ to include the processes which are in $\chi(\mathit{carrier}(v, \mathbf{s}))$ but not in $\chi(\sigma)$, the set of critical simplices associated with an agreement power strictly greater than $\alpha(\chi(\mathit{carrier}(v, \mathbf{s})))$ does not change. Thus applying Corollary 4 to any such completion σ' of σ, we obtaint that, for any $l > \alpha(\chi(\mathit{carrier}(v, \mathbf{s})))$:

$$\alpha(\chi(\mathit{carrier}(\sigma, s))) - l - |\chi(\mathit{carrier}(\sigma, \mathbf{s})) \setminus (\chi(\sigma) \cup \chi(\mathit{carrier}(v, \mathbf{s})))| + 1 \leq$$

$$\mathit{csize}(\{\theta \in \mathcal{CS}_\alpha(\sigma), \alpha(\chi(\mathit{carrier}(\theta, \mathbf{s}))) \geq l\}).$$

Moreover, any process in $P \setminus \chi(\mathit{carrier}(\sigma, s))$ must have failed. Thus, in $\chi(\mathit{carrier}(\sigma, s))$ at most $\alpha(P) - 1 - (|P \setminus \chi(\mathit{carrier}(\sigma, s))|)$ processes may fail. Let us recall from the proof of Lemma 3, that for the agreement function of any fair adversary, and for any $Q \subseteq P$, we have $\alpha(P) \geq \alpha(P \setminus Q) \geq \alpha(P) - |Q|$. Thus we can derive, by using $Q = P \setminus \chi(\mathit{carrier}(\sigma, s))$, that at most $\alpha(\chi(\mathit{carrier}(\sigma, s))) - 1$ processes in $\chi(\mathit{carrier}(\sigma, s))$ may fail.

Let $m_1 = |\chi(\mathit{carrier}(\sigma, \mathbf{s})) \setminus (\chi(\sigma) \cup \chi(\mathit{carrier}(v, \mathbf{s})))|$, i.e., the number of processes from $\chi(\mathit{carrier}(\sigma, \mathbf{s}))$ which (1) fail before updating their *IS1* to the memory and (2) are not included in the *IS1* view of p. Let m_2 be the number of critical processes, associated with an agreement power strictly greater than $\alpha(\chi(\mathit{carrier}(v, \mathbf{s})))$, which fail after updating their *IS1* but before updating their *IS2*. If $\alpha(\chi(\mathit{carrier}(\sigma, s))) - \alpha(\chi(\mathit{carrier}(v, s))) > m_1 + m_2$, then by selecting $l = \alpha(\chi(\mathit{carrier}(\sigma, s))) - m_2 - m_1$, we have $l > \alpha(\chi(\mathit{carrier}(v, s))))$, and hence, we obtain that:

$$\mathit{csize}(\{\theta \in \mathcal{CS}_\alpha(\sigma), \alpha(\chi(\mathit{carrier}(\theta, \mathbf{s}))) \geq \alpha(\chi(\mathit{carrier}(\sigma, s))) - m_2 - m_1\}) \geq m_2 + 1.$$

If no critical simplex in $\{\theta \in \mathcal{CS}_\alpha(\sigma), \alpha(\chi(\mathit{carrier}(\theta, \mathbf{s}))) \geq \alpha(\chi(\mathit{carrier}(\sigma, s))) - m_2 - m_1\}$ terminates, one process from each of these critical simplices failed after updating its *IS1* but before updating its *IS2*, thus an hitting set failed. As only m_2 such processes may fail and as an hitting set must be greater than $m_2 + 1$, a critical simplex associated with an agreement power greater than or equal to $\alpha(\chi(\mathit{carrier}(\sigma, s))) - m_2 - m_1$ terminates its *IS2*. Therefore eventually p updates the value of *Concurrency* (on line 10) to be at least equal to $\alpha(\chi(\mathit{carrier}(\sigma, s))) - m_2 - m_1$.

Now let us look back at p. It fails to terminate and thus never succeed to pass the test on line 5. Therefore we have that the number of processes in the *IS1* view of p which do not terminate and do not have the same *IS1* view as p are strictly more than the value of *Concurrency* for p. As p is the correct process with the smallest *IS1* view which does not terminate, it implies that there are strictly more than *Concurrency* failed processes with an *IS1* view stricly smaller than p. These failed processes are neither accounted in m_1 nor in m_2. Therefore, as at most $\alpha(\chi(\mathit{carrier}(\sigma, s))) - 1$ processes in $\chi(\mathit{carrier}(\sigma, s))$ may fail, there are at most $\alpha(\chi(\mathit{carrier}(\sigma, s))) - 1 - m_1 - m_2$ such processes which may fail. Thus $\alpha(\chi(\mathit{carrier}(\sigma, s))) - m_1 - m_2 - 1 \geq \mathit{Concurrency}$.

Two cases may arise:

- If $\alpha(\chi(\mathit{carrier}(\sigma, s))) - \alpha(\chi(\mathit{carrier}(v, s))) > m_1 + m_2$, then p sets *Concurrency* to a value greater than or equal to $\alpha(\chi(\mathit{carrier}(\sigma, s))) - m_2 - m_1$ — A contradiction.
- Otherwise, $\mathit{Concurrency} \geq \alpha(\chi(\mathit{carrier}(v, s)))$ (due to its initialization to $\alpha(\chi(\mathit{carrier}(v, s)))$ on line 4), and thus as $\alpha(\chi(\mathit{carrier}(\sigma, s))) - \alpha(\chi(\mathit{carrier}(v, s))) \leq m_1 + m_2$, we obtain that $\mathit{Concurrency} \geq \alpha(\chi(\mathit{carrier}(\sigma, s))) - m_1 - m_2$ — A contradiction.

□

Lemma 6. *The set of outputs provided by Algorithm 1 forms a valid simplex in $\mathcal{R}_\mathcal{A}$.*

Proof. Consider any execution of Algorithm 1. Except for the wait-phase, processes simply execute two rounds of immediate snapshot, therefore the set of outputs forms a simplex in $\sigma \in \mathrm{Chr}^2\,\mathbf{s}$. Without loss of generality we can consider that no processes fails and thus that $dim(\sigma) = n-1$. Indeed, if $\sigma \notin \mathcal{R}_\mathcal{A}$, then making failed processes resume their execution and terminate will produce $\sigma' \notin \mathcal{R}_\mathcal{A}$. Let us assume that $\sigma \notin \mathcal{R}_\mathcal{A}$, this implies that there exists $\theta \subseteq \sigma$ such that (for $\theta' = carrier(\theta, \mathrm{Chr}\,\mathbf{s})$):

$$(\theta \in Cont_2) \wedge ((\chi(\theta) \cap (\chi(\mathcal{CSM}_\alpha(carrier(\sigma', \mathrm{Chr}\,\mathbf{s}))) \cup \chi(\mathcal{CSV}_\alpha(\theta'))) = \emptyset) \wedge (dim(\theta)+1 > Conc_\alpha(\theta')).$$

As $\theta \in Cont_2$, we can order the processes associated with vertices from θ according to their *IS2* view (or $carrier(v, \mathrm{Chr}\,\mathbf{s})$). Let $q_1, \ldots, q_k$ be this ordered set of processes. As q_1 has the smallest *IS2* view, and as $\theta \in Cont_2$, q_1 has also the greatest *IS1* view.

Consider the state of the execution at the time p_1 successfully passed the test on line 5. To pass the test, q_1 must have witness *IS1*, *critical* and *IS2* states such that (with $q_1 = p_i$):

$$(\neg critical[i]) \wedge (|\{p_j \in IS1[i] : IS2[j] = \emptyset \wedge IS1[j] \neq IS1[i]\}| \geq concurrency)$$

If $critical[i]$, then q_1 is in a critical simplex and thus $\chi(\theta) \cap \chi(\mathcal{CSM}_\alpha(carrier(\sigma, \mathrm{Chr}\,\mathbf{s}))) \neq \emptyset$ as it would include q_1. Therefore we have:

$$|\{p_j \in IS1[i] : IS2[j] = \emptyset \wedge IS1[j] \neq IS1[i]\}| \geq concurrency$$

Note that as $\theta \in Cont_2$, all processes $q_2, \ldots, q_k$ are included in $IS1[i]$ (as they have a smaller *IS1* view), $IS2[j] = \emptyset$ (as they have a larger *IS2* view) and $IS1[j] \neq IS1[i]$ (as they have a *strictly* smaller *IS1* view). Therefore, $concurrency \geq k$.

Two cases may arise:

- $concurrency \neq \alpha(IS1[i])$: In this case, *Concurrency* was set on line 10 to a value greater than $\alpha(IS1[i])$. This implies that a critical simplex associated with an agreement level equal to *concurrency* terminated its computation and thus is included in $carrier(q_1, \mathrm{Chr}\,\mathbf{s})$. Hence we have $Conc_\alpha(\theta') \geq concurrency$. A contradiction with θ being a counter exemple since:

  $$dim(\theta) - 1 = k \leq concurrency \leq Conc_\alpha(\theta')).$$

- $concurrency = \alpha(IS1[i])$: Let c be the highest agreement power associated with a terminated critical simplex (with $c = 0$ if no critical simplex is terminated). Therefore we have $Conc_\alpha(\theta') \geq c$. Let $\lambda \in \mathrm{Chr}\,\mathbf{s}$ be the simplex corresponding to the set of *IS1* views of processes in $IS1[i]$ which are terminated. According to Corollary 4 applied to λ with $l = c+1$, as $concurrency = \alpha(\chi(carrier(\lambda, \mathbf{s})))$, we obtain that:

  $$concurrency - c - |\chi(carrier(\lambda, \mathbf{s})) \setminus \chi(\lambda)| \leq csize(\{\theta \in \mathcal{CS}_\alpha(\lambda), \alpha(\chi(carrier(\theta, \mathbf{s}))) \geq c+1\}).$$

  Note that since no critical simplex associated with an agreement power greater than or equal to $c+1$ terminated, it implies that one process of each critical simplex identified in *IS1* did not terminate its *IS2*. Thus there are at least a minimal hitting set which did not terminate its *IS*2, and hence, at least $concurrency - c - |\chi(carrier(\lambda, \mathbf{s})) \setminus \chi(\lambda)|$. But as they are in *IS1* and do not have any *IS2* output, they are counted in $|\{p_j \in IS1[i] : IS2[j] = \emptyset \wedge IS1[j] \neq IS1[i]\}|$. Therefore as there are nocritical simplex members in θ, we have $concurrency \geq k + csize(\{\theta \in \mathcal{CS}_\alpha(\lambda), \alpha(\chi(carrier(\theta, \mathbf{s}))) \geq c+1\})$. Hence, we obtain that:

  $$concurrency - c - |\chi(carrier(\lambda, \mathbf{s})) \setminus \chi(\lambda)| \leq concurrency - k.$$

  Thus, we get $c + |\chi(carrier(\lambda, \mathbf{s})) \setminus \chi(\lambda)| \geq k$. As $Conc_\alpha(\theta') \geq c$ and as a set size is positive, we have $Conc_\alpha(\theta') \geq k$ — A contradiction.

□

Using Lemmata 5 and 6, we can directly derive the validity of Algorithm 1:

Theorem 7. *Algorithm 1 solves task $\mathcal{R}_\mathcal{A}$ in the α-model corresponding to the fair adversary $\mathcal{A}$.*

6 From $\mathcal{R}^*_{\mathcal{A}}$ to the fair adversarial $\mathcal{A}$-model

In this section, we show that any task solvable in the fair adversarial $\mathcal{A}$-model can be solved in $\mathcal{R}^*_{\mathcal{A}}$. This reduction is much more intricated than in the other direction. Indeed, to show that a model is as strong as an affine task based model, it only suffices to show that any number of iterations of the affine task can be solved. In the general case, it is necessary to show that any task solvable in the target model can be solved and thus that we can emulate an algorithm solving any given task.

To simplify the simulation complexity, we are going to show that we can simulate an execution of a shared memory model in which the participation P is such that $\alpha(P) > 0$ and in which α-adaptive set consensus can be solved. Using the results from [24] (Theorem 2), we are able to deduce from it that any task solvable in a fair adversarial model can be solved in $\mathcal{R}^*_{\mathcal{A}}$.

6.1 Simulation Description.

The main difficulty of the simulation comes from the combination of the failure-freedom and the iterative structure of $\mathcal{R}^*_{\mathcal{A}}$. A process obtaining small outputs in all iterations, often denominated as a "fast" process, may never observe the values shared by other processes with larger views, comparatively denominated as a "slow" processes. But as there are no processes failures, eventually, all processes must obtain a task output. It requires that fast processes make progress with the simulation without waiting for slower processes. Slow processes must thus wait for faster processes to terminate their simulation before being able to make progress with modifying operations.

This first difficulty is resolved by making processes which obtained a task output in the simulation to use the special value $\perp$ as input for all further iterations of $\mathcal{R}_{\mathcal{A}}$. Slower processes are then aware that processes using $\perp$ do not interfere anymore and that they no longer need to witness their modifications of the simulated system state.

Another difficulty relies in the fact that processes may shift between making shared memory operations and accessing α-adaptive set consensus abstractions. Moreover, processes may be accessing distinct α-adaptive set consensus abstractions and may access them in different orders. Fortunately, set consensus abstractions are independant of each others and multiple instances can be simulated in parallel. But memory operations interact with each others and a write operation can be safely terminated only once the write value is known to be observed by all other processes. Thus a fast process must ensure that slower processes are not able to complete write operations as long as they did not terminate, even when they do not currently have a write operation to perform.

Atomic-snapshot simulation. To simulate the atomic-snapshot memory, we rely upon the algorithm proposed in [16] that simulates a lock-free atomic-snapshot algorithm in the *iterated* atomic-snapshot model. We run the simulation using the *global views* that the processes obtain at the end of $\mathcal{R}^*_{\mathcal{A}}$ iterations, i.e., $carrier(v, \mathbf{s})$ for their vertices $v \in \mathcal{R}_{\mathcal{A}}$. Recall that these global views satisfy the properties of atomic snapshots, but not necessarily the properties of immediate snapshots.

In the simulation, every new update performed by a process is assigned a monotonically growing *sequence number*. A terminated process simply stops incrementing its sequence number, which allows active (non-terminated) processes to make progress. Without loss of generality, we assume that in the simulated algorithm, every active process always has a pending memory operation to perform (intuitively, if there is nothing to write, the process rewrites its last written value).

Simulating α-adaptive set consensus in $\mathcal{R}^*_{\mathcal{A}}$. The α-adaptive set consensus simulation in $\mathcal{R}^*_{\mathcal{A}}$ submits in all iterations input, a decision estimate for all known set consensus simulations. For all pending and newly discovered set consensus simulations for which processes are involved (i.e., for which they are allowed to participate), processes update their decision estimate after each iteration of $\mathcal{R}_{\mathcal{A}}$. Processes adopt a deterministically chosen estimate from, if available, an *IS1* view associated to a critical simplex, and otherwise, from the smallest *IS1* view they see. Note that only IS1 views including a process which may participate to the agreement are considered. Most of the complexity of the α-adaptive set consensus simulation lies in this selection of which *IS1* view to adopt from. This is described extensively in the next section.

A desicion value is committed only when all processes which are involved in the α-adaptive set consensus abstraction and which are observed in a given iteration of $\mathcal{R}^*_{\mathcal{A}}$ posses a decision estimate. Once, the value is committed, the decision estimate will no longer change and will eventually be returned as output for the α-adaptive set consensus, but processes need to check that the participation in the simulated run is high enough before returning the value.

In order to ensure a high enough participation, processes make sure that all processes that they witnessed in preceding iterations of $\mathcal{R}^*_{\mathcal{A}}$ have completed their first simulated write operation. If not, processes simulate this write operation themselves. The content of this first write operation simply consists of the process initial state. Therefore, any process p may simulate this write operation (by using the shared memory simulation) for any other process q as soon as p knows the initial state of q. Once all processes for which the initial state is know are participating in the simulated run, processes can safely terminate their α-adaptive set consensus with the committed value.

6.2 α-adaptive leader election in $\mathcal{R}_{\mathcal{A}}$: the μ_Q map

Let us consider some α-adaptive set consensus and let Q be the set of processes which (1) may participate in the agreement protocol, and, (2) did not terminate yet the main simulation. Using the structure of $\mathcal{R}_{\mathcal{A}}$, we construct a map μ_Q which returns to each vertex $v \in \mathcal{R}_{\mathcal{A}}$, corresponding to a process from Q (i.e., with $\chi(v) \in Q$), a leader selected among Q for the given iteration of $\mathcal{R}_{\mathcal{A}}$.

The map μ_Q is constructed in two stages. The first stage consists on selecting an *IS1* view which includes a process from Q. Two cases may happen depending on whether the process observes in $\mathcal{R}_{\mathcal{A}}$ a critical simplex associated with an *IS1* view including a process from Q or not:

If the process observes such a critical simplex (i.e., $\chi(\mathcal{CSV}_\alpha(\mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s}))) \cap Q \neq \emptyset$), it then simply returns the smallest *IS1* view of a critical simplex which includes a process from Q, using the map δ_Q:

$$\delta_Q = \chi(\min(\{\mathit{carrier}(\sigma', \mathbf{s}) : (\sigma' \in CS_\alpha(\mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s})) : \chi(\mathit{carrier}(\sigma', \mathbf{s})) \cap Q \neq \emptyset)\}).$$

Otherwise (if $\chi(\mathcal{CSV}_\alpha(\mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s}))) \cap Q = \emptyset$), the process returns the smallest observed *IS1* view which includes a process from Q, using the map γ_Q:

$$\gamma_Q = \chi(\min(\{\mathit{carrier}(v', \mathbf{s}) : (v' \in \mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s})) \wedge (\mathit{dim}(v') = 0) \wedge (\mathit{carrier}(v', \mathbf{s}) \cap Q \neq \emptyset)\}).$$

The second stage then simply consists in selecting, from the selected *IS1* view, the process from Q associated with the smallest identifier, let $\min_Q(V) = \min\{p \in V : p \in Q\}$ be this map. The map μ_Q is therefore defined as follow:

$$\mu_Q(v) = \textbf{ if } (\chi(\mathcal{CSV}_\alpha(\mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s}))) \cap Q \neq \emptyset) \textbf{ then } \min_Q \circ \delta_Q \textbf{ else } \min_Q \circ \gamma_Q.$$

Let us first show that, for any vertex $v \in \mathcal{R}_A$ corresponding to a process in Q, the map μ_Q returns a process from Q observed in $\mathcal{R}_{\mathcal{A}}$ (i.e., a process in $\chi(\mathit{carrier}(v, \mathbf{s}))$) :

Property 8. *[Validity of μ_Q]* $\forall v \in \mathcal{R}_{\mathcal{A}}, \mathit{dim}(v) = 0, \chi(v) \in Q$:

$$\mu_Q(v) \in \chi(\mathit{carrier}(v, \mathbf{s})) \wedge \mu_Q(v) \in Q.$$

Proof. Let us fix some vertex $v \in \mathcal{R}_{\mathcal{A}}$ such that $\chi(v) \in Q$.

Let us assume that $\chi(\mathcal{CSV}_\alpha(\mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s}))) \cap Q \neq \emptyset$, and hence, $\mu_Q(v) = \min_Q \circ \delta_Q(v)$. Let us recall that given $\sigma \in \mathrm{Chr}\,\mathbf{s}$, $\mathcal{CSV}_\alpha(\sigma)$ is equal to $\mathit{carrier}(\cup_{\sigma' \in \mathcal{CS}_\alpha(\sigma)} \sigma', \mathbf{s})$. But due to carriers inclusion, the carrier of a simplex is equal to the carrier of one of its vertices, and so, of any sub-simplex which includes this vertex. Thus, as $\chi(\mathcal{CSV}_\alpha(\mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s}))) \cap Q \neq \emptyset$, we have:

$$\exists \sigma' \in CS_\alpha(\mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s})) : \chi(\mathit{carrier}(\sigma', \mathbf{s})) \cap Q \neq \emptyset.$$

This implies that δ_Q has a valid choice for v and can return the minimal one, and so that:

$$\exists \sigma' \in CS_\alpha(\mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s})) : (\delta_Q(v) = \chi(\mathit{carrier}(\sigma', \mathbf{s}))) \wedge (\chi(\mathit{carrier}(\sigma', \mathbf{s})) \cap Q \neq \emptyset).$$

Since $CS_\alpha(carrier(v, \mathrm{Chr}\,\mathbf{s})) \subseteq \{\sigma \in \mathrm{Chr}\,\mathbf{s}; \sigma \subseteq carrier(v, \mathrm{Chr}\,\mathbf{s})\}$, and as $\mu_Q(v) = \min_Q \circ \delta_Q(v)$, we obtain that:

$$\exists \sigma' \subseteq carrier(v, \mathrm{Chr}\,\mathbf{s}) : (\mu_Q(v) = \min_Q \circ \chi(carrier(\sigma', \mathbf{s}))) \wedge (\mu_Q(v) \in Q).$$

As for any simplex $\sigma \in \mathrm{Chr}^2\,\mathbf{s}$, $carrier(carrier(v, \mathrm{Chr}\,\mathbf{s}), \mathbf{s}) = carrier(v, \mathbf{s})$, Property 8 is verified if $\chi(\mathcal{CSV}_\alpha(carrier(v, \mathrm{Chr}\,\mathbf{s}))) \cap Q \neq \emptyset$.

Now let us assume that $\chi(\mathcal{CSV}_\alpha(carrier(v, \mathrm{Chr}\,\mathbf{s}))) \cap Q = \emptyset$. Due to the self-inclusion property, $\exists v' \in carrier(v, \mathrm{Chr}\,\mathbf{s}))$ such that $\chi(v') = \chi(v)$. The self-inclusion property again implies that $\exists v'' \in carrier(v', \mathbf{s})$ such that $\chi(v'') = \chi(v') = \chi(v)$. Hence, as $\chi(v) \in Q$, $\exists v' \in carrier(v, \mathrm{Chr}\,\mathbf{s})$ such that $\chi(carrier(v', \mathbf{s})) \cap Q \neq \emptyset$. Thus γ_Q has a valid choice for v and can return the minimal one. As before, by the transitivity of carriers inclusion, the set returned by γ_Q, and so the process returned by μ_Q, is a subset of $\chi(carrier(v, \mathbf{s}))$ which intersects with Q. □

Now that we have checked that μ_Q is well defined, let us show that μ_Q returns a number of distinct leaders (processes) limited by the agreement power associated with processes views in $\mathcal{R}_\mathcal{A}$:

Property 9. *[Agreement of μ_Q]* $\forall Q \subseteq \Pi, (\forall \sigma \in \mathcal{R}_\mathcal{A} : dim(\sigma) = n-1), (\forall \theta \subseteq \sigma : \chi(\theta) \subseteq Q)$:

$$|\{\mu_Q(v) : v \in \theta\}| \leq \alpha(\chi(carrier(\theta, \mathbf{s}))).$$

Let us first check the following observation stating that for any simplex $\sigma \in \mathrm{Chr}\,\mathbf{s}$, if two critical simplices in σ are associated with the same agreement power then they share the same *IS1* view:

Lemma 10. $\forall \sigma \in \mathrm{Chr}\,\mathbf{s}$, $\forall \theta_1, \theta_2 \in \mathcal{CS}_\alpha(\sigma)$*:*

$$\alpha(\chi(carrier(\theta_1, \mathbf{s}))) = \alpha(\chi(carrier(\theta_2, \mathbf{s}))) \implies carrier(\theta_1, \mathbf{s}) = carrier(\theta_2, \mathbf{s}).$$

Proof. Let $\sigma \in \mathrm{Chr}\,\mathbf{s}$ and let $\theta_1, \theta_2 \in \mathcal{CS}_\alpha(\sigma)$ such that $\alpha(\chi(carrier(\theta_1, \mathbf{s}))) = \alpha(\chi(carrier(\theta_2, \mathbf{s})))$. The inclusion property implies that, w.l.o.g., $carrier(\theta_1, \mathbf{s}) \subseteq carrier(\theta_2, \mathbf{s})$. The immediacy property implies either that $carrier(\theta_1, \mathbf{s}) = carrier(\theta_2, \mathbf{s})$ (and thus Lemma 10 is verified) or else that $\chi(\theta_2) \cap \chi(carrier(\theta_1, \mathbf{s})) = \emptyset$.

Let us assume that $\chi(\theta_2) \cap \chi(carrier(\theta_1, \mathbf{s})) = \emptyset$. Together with $carrier(\theta_1, \mathbf{s}) \subseteq carrier(\theta_2, \mathbf{s})$, it implies that $carrier(\theta_1, \mathbf{s}) \subseteq carrier(\theta_2, \mathbf{s}) \setminus \theta_2$. Since agreement functions are regular (i.e., the agreement power can only grow with a participation increase), we obtain that $\alpha(\chi(carrier(\theta_1, \mathbf{s})) \leq \alpha(\chi(carrier(\theta_2, \mathbf{s}) \setminus \theta_2)$. But as θ_2 is a critical simplex, we have $\alpha(\chi(carrier(\theta_2, \mathbf{s}) \setminus \theta_2)) < \alpha(\chi(carrier(\theta_2, \mathbf{s})))$, and we obtain a contradiction:

$$\alpha(\chi(carrier(\theta_1, \mathbf{s})) \leq \alpha(\chi(carrier(\theta_2, \mathbf{s}) \setminus \theta_2) < \alpha(\chi(carrier(\theta_2, \mathbf{s}))) = \alpha(\chi(carrier(\theta_1, \mathbf{s}))).$$

□

Let us now prove Property 9:

Proof. Let σ be a maximal simplex of $\mathcal{R}_\mathcal{A}$, i.e., $dim(\sigma) = n-1$, and let $\theta \subseteq \sigma$ such that $\chi(\theta) \subseteq Q$.

Note that for both γ_Q and δ_Q, processes returns the *IS1* view of a vertex of $carrier(\theta, \mathrm{Chr}\,\mathbf{s})$. Assume that γ_Q and δ_Q return, for vertices in θ, $k \geq 0$ distinct *IS1* views which are not the *IS1* views of some critical simplex in $carrier(\sigma, \mathrm{Chr}\,\mathbf{s})$. As δ_Q only returns *IS1* views associated with a critical simplex, they have been returned by γ_Q. Let β be the subset of θ including all vertices for which γ_Q returns such *IS1* views. As they are returned by γ_Q, we have $\mathcal{CSV}_\alpha(carrier(\beta, \mathrm{Chr}\,\mathbf{s})) \cap Q = \emptyset$.

Consider any two processes p_1 and p_2 which obtained two distinct such *IS1* views, V_1 and V_2 respectively (w.l.o.g., let $V_1 \subsetneq V_2$). As γ_Q returns the minimal *IS1* view intersecting with Q, a vertex from β sees V_2 but not V_1, and thus, p_2 has a smallest *IS2* view than p_1. Therefore p_1 and p_2 satisfy the condition to be part of a contention simplex, and so, any k processes carrying these k distinct returned *IS1* views form a contention simplex. Let τ be this contention simplex in σ.

As a vertex in β saw all these k distinct *IS1* views, we have $carrier(\tau, \mathrm{Chr}\,\mathbf{s}) \subseteq carrier(\beta, \mathrm{Chr}\,\mathbf{s})$. But as $\mathcal{CSV}_\alpha(carrier(\beta, \mathrm{Chr}\,\mathbf{s})) \cap Q = \emptyset$, we obtain that $\mathcal{CSV}_\alpha(carrier(\tau, \mathrm{Chr}\,\mathbf{s})) \cap Q = \emptyset$.

By assumption, these k processes are not critical simplices members ($\chi(\tau) \cap \mathcal{CSM}_\alpha(\sigma) = \emptyset$). Therefore, the definition of $\mathcal{R}_\mathcal{A}$ implies that we have $Conc_\alpha(carrier(\tau, \mathrm{Chr}\,\mathbf{s})) \geq k$, and hence, that $Conc_\alpha(carrier(\beta, \mathrm{Chr}\,\mathbf{s})) \geq k$.

Having $Conc_\alpha(carrier(\beta, \mathrm{Chr}\,\mathbf{s})) \geq k$ implies that $\exists \sigma_c \in \mathcal{CS}_\alpha(carrier(\beta, \mathrm{Chr}\,\mathbf{s}))$ such that $\alpha(\chi(carrier(\sigma_c, \mathbf{s}))) \geq k$. Note that we have $\chi(carrier(\sigma_c, \mathbf{s})) \subseteq \mathcal{CSV}_\alpha(carrier(\beta, \mathrm{Chr}\,\mathbf{s}))$, and thus that $\chi(carrier(\sigma_c, \mathbf{s})) \cap Q = \emptyset$. As the inclusion property implies that any *IS1* view must be stricly larger to intersect with Q, and as there are at most one *IS1* view associated with a critical simplex by agreement level (Lemma 10), all *IS1* views corresponding to some critical simplex in $carrier(\sigma, \mathrm{Chr}\,\mathbf{s})$ must be associated with an agreement power strictly greater than k.

Let $l \geq 0$ be the number of distinct *IS1* views corresponding to some critical simplex in $carrier(\sigma, \mathrm{Chr}\,\mathbf{s})$ which are returned by δ_Q or γ_Q for vertices in θ. Lemma 10 implies that they must be associated with l distinct agreement powers. As they must also be associated with agreement powers strictly greater than k, one of the returned *IS1* views is associated with an agreement power greater than or equal to $k + l$. Therefore, we have $\alpha(\chi(carrier(\theta, \mathbf{s}))) \geq k + l$. As the number of distinct *IS1* views returned by δ_Q or γ_Q is equal to $k + l$, and as the deterministic selection made by $\min_Q$ could only reduce the number of distinct returned values, we finally obtain that $|\{\mu_Q(v) : v \in \theta\}| \leq \alpha(\chi(carrier(\theta, \mathbf{s})))$. □

Last, let us also observe that knowing which processes terminated the main simulation is not required to compute μ_Q, i.e., that the knowledge of which processes belong to Q among the processes observed in the current iteration of $\mathcal{R}_\mathcal{A}$ is sufficient:

Property 11. *[Robustness of μ_Q]* $\forall v \in \mathcal{R}_\mathcal{A}, dim(v) = 0, \forall Q \subseteq \Pi$:

$$\mu_Q(v) = \mu_{carrier(v,\mathbf{s}) \cap Q}(v).$$

Proof. This is a direct corollary of the definition of δ_Q and γ_Q, that for a given vertex $v \in \mathcal{R}_\mathcal{A}$, to compute $\mu_Q(v)$, the knowledge of $Q \cap (carrier(v, \mathbf{s}))$ is sufficient. Indeed, Q is only used to compute intersections with either $\mathcal{CSV}_\alpha(\chi(carrier(v, \mathrm{Chr}\,\mathbf{s})))$, a subset of $carrier(v, \mathbf{s})$, or with $carrier(v', \mathbf{s})$ for a vertex $v' \in carrier(v, \mathrm{Chr}\,\mathbf{s})$, also a subset of $carrier(v, \mathbf{s})$. □

6.3 Validity of the simulation

Let us first show that all simulated operations are safe, i.e., that simulated shared memory operations are linearizable and that simulated α-adaptive set consensus satisfy the validity property (decision values are proposal values) and the α-agreement property (if k distinct values have been returned, then the current participation P is such that $\alpha(P) \geq k$):

Lemma 12. *The shared memory and α-adaptive set consensus simulation in $\mathcal{R}^*_\mathcal{A}$ is safe.*

Proof. The validity of shared memory operations is directly inherited from [16] since the simulation is only modified by incorporating extra write operations to perform (either dummy write operation or write operation of processes initial states through the α-adaptive set consensus simulation).

For α-adaptive set consensus operations, Property 11 ensures that $\mu_{carrier(v,\mathbf{s}) \cap Q}(v)$ can be used as if it was $\mu_Q(v)$ and thus that processes can indeed use μ_Q to elect a leader in any iteration of $\mathcal{R}_\mathcal{A}$. Moreover, Property 8 ensures that a decision estimate is either the process proposal value or is adopted from another process with a proposal value and thus that the validity property of α-adaptive set consensus is verified.

At the earliest iteration R of $\mathcal{R}^*_\mathcal{A}$ at which a process commits a decision estimate for an α-adaptive set consensus, since a committing process only observed processes from Q with decision estimates, all processes in Q adopt a decision estimate. Moreover, Property 9 states that among any k processes adopting k distinct decision estimates at this iteration R, one must have observed a set of processes associated with an agreement level greater or equal to k. But, given any k processes with distinct committed decision estimates, one of them, p, must have adopted the value from a process which had observed in R a set of processes associated with an agreement level

greater or equal to k. Thus by transitivity of proceses views, p has also observed a set of processes associated with an agreement level greater or equal to k.

Before completing an α-adaptive set consensus operation, processes make sure that all processes they observed are participating in the simulated run (by, if necessary, simulating for them a write operation of their initial states). Therefore, at the time a k^{th} distinct value is returned for some α-adaptive set consensus, the participation in the simulated run is associated with an agreement power greater than or equal to k, hence, the α-agreement property is verified. □

As we have shown that the simulation is safe, let us also show that it is live, i.e., that it provides outputs to all processes:

Lemma 13. *In the shared memory and α-adaptive set consensus simulation in $\mathcal{R}^*_{\mathcal{A}}$, all processes eventually terminate.*

Proof. Let us assume by contradiction that some process never terminates. The shared memory simulation ensures that a non-terminated process eventually terminates its pending memory operation. But, since eventually no processes make progress, it implies that a non-terminated process, p, eventually completes infinitely often only dummy write operations.

Recall that a process completing a write operation has the smallest view among non-terminated processes for the current iteration. Therefore, p has infinitely often the smallest view (among non-terminated processes) in $\mathcal{R}^*_{\mathcal{A}}$ and eventually never commits a decision estimate for a pending α-adaptive set consensus operation. This implies that some process from Q observed by p never shares a decision estimate for the α-adaptive set consensus abstraction accessed by p. But since p has infinitely often the smallest view, any other process from Q will eventually adopt a decision estimate from p or another process and use it in forthcoming iterations — A contradiction. □

By Lemmata 12 and 13, the simulation that we provide can be used to solve in $\mathcal{R}^*_{\mathcal{A}}$ any task solvable in a shared memory model with access to α-adaptive set consensus. Using Theorem 2, we can derive that any task solvable in the α-model and the the fair adversarial model can be solved in $\mathcal{R}^*_{\mathcal{A}}$. In combination with the result in Section 5, we obtain the following result:

Theorem 14. *Let $\mathcal{A}$ be any fair adversary, and let α be its agreement function. A task is solvable in the adversarial $\mathcal{A}$-model if and only if it is solvable in $\mathcal{R}^*_{\mathcal{A}}$.*

We thus obtain the following generalization of the ACT [20]:

Theorem 15. *[Fair ACT] Let $\mathcal{A}$ be any fair adversary, and let α be its agreement function. A task $T = (\mathcal{I}, \mathcal{O}, \Delta)$ is solvable in the adversarial $\mathcal{A}$-model if and only if there exists a natural number ℓ and a simplicial map $\phi : \mathcal{R}^\ell_{\mathcal{A}}(\mathcal{I}) \to \mathcal{O}$ carried by Δ.*

7 Related work

Inspired by the model of *dependent failures* proposed by Junqueira and Marzullo [21], Delporte et al. [9] suggested the notion of adversaries and showed that adversaries having the same set consensus power agree on the set of colorless tasks they solve.

Herlihy and Shavit [20] proposed a characterization of wait-free task computability through the existence of a simplicial map from a subdivision of the input complex of a task $\mathcal{I}$ to its output complex $\mathcal{O}$. (The reader is referred to [18] for a thorough discussion of the use of combinatorial topology in distributed computability.)

Herlihy and Rajsbaum [19] studied colorless task computability in the special case of *superset-closed* adversaries. The set consensus power of a superset-closed adversary is its minimum *core size*: the size of a smallest set of processes that intersects with every live set [21]. They show that the protocol complex of a superset-closed adversary with *minimal core size* c is $(c-2)$-connected. This result, obtained via an iterative application of the Nerve lemma, gives a combinatorial characterization of superset-closed adversaries, which is weaker than the characterization of wait-freedom through the immediate snapshot task. Unlike the results of this paper, the characterization

only applies to colorless tasks, and it does not allow us to express the adversary in a compact *affine* way.

Gafni et al. [15] introduced the notion of an affine task and characterized task computability in *iterated* adversarial models via infinite subdivisions of input complexes, assuming a limited notion of solvability that only guarantees outputs to "fast" processes [6, 12, 26] ("seen" by every other process infinitely often). The liveness property defined in this paper for iterated models, guarantees outputs for *every* process, which allowed us to establish a task-computability equivalence with conventional non-iterated ones.

Saraph et al. [28] gave a compact combinatorial characterization of t-resilient task computability. Note that $\mathcal{A}_t$ is a superset-closed (and thus *fair*) adversary. Our solution of the affine task $\mathcal{R}_\mathcal{A}$ in the α-model is inspired by the t-resilient solution of $\mathcal{R}_t$ in [28]. Gafni et al. [13] presented affine tasks for the model of k-set consensus and, thus, k-concurrency and k-obstruction-freedom, which can be expressed as a symmetric and thus *fair* adversary.

The notions of an agreement function and a fair adversary were introduced by the first two authors in [24]. One can determine the agreement function of any given adversary using the formula suggested earlier for the set consensus power [14]. It has been shown in [24] that agreement functions encode enough information to characterize the task computability of any *fair* adversary. Figure 2 relates the results of this paper to earlier affine characterizations of adversarial models. A short version of this paper appeared as a conference brief announcement [25].

8 Concluding remarks

This paper generalizes all topological characterizations of distributed computing models known so far [13, 15, 19, 20, 28]. It applies to all tasks (not necessarily colorless) and all *fair* adversarial models (not necessarily t-resilient or k-obstruction-free). Just as the wait-free characterization [20] implies that the IS task captures the wait-free model, our characterization equates any *fair* adversary with a (compact) affine task embedded in the 2-degree of standard chromatic subdivision.

Interestingly, unlike [28], we cannot rely on the assumption that the affine task $\mathcal{R}_{\mathcal{A}_t}$ corresponding to the t-resilient adversary $\mathcal{A}_t$ is *shellable* [18] and, thus, link connected. Link-connectivity of a simplicial complex $\mathcal{C}$ allows us to work in the *point set* of its geometrical embedding $|\mathcal{C}|$ and use continuous maps (as opposed to simplicial maps that maintain more structure). For example, the existence of a continuous map from $|\mathcal{R}_{\mathcal{A}_t}|$ to any $|\mathcal{R}^k_{\mathcal{A}_t}|$ implies that $\mathcal{R}_{\mathcal{A}_t}$ indeed captures the general task computability of $\mathcal{A}_t$. However, the existence of a continuous map onto $\mathcal{C}$ only allows us to converge on only *one* vertex [18]. If $\mathcal{C}$ is not link-connected, converging on one vertex allow us to compute only one output in a task solution and not more, which is not sufficient to solve a general (colored) task.

Unfortunately, only very special adversaries, such as $\mathcal{A}_t$, have link-connected counterparts (see, e.g., the affine task corresponding to 1-obstruction freedom in Figure 6a). Instead of relying on link-connectivity, this paper takes an explicit algorithmic way of showing that iterations of $\mathcal{R}_A$ simulate $\mathcal{A}$. This raises an interesting question to which extent point-set topology and continuous maps can be applied in affine characterizations.

Furthermore, going beyond *fair* models is an important challenge. Given that some models out of this class cannot be grasped by agreement functions (some examples of these models can be found in [24]), we should find a more refined way of to capture the power of solving set consensus for subsets of participating processes. In particular, we should be able to account for models in which *coalitions* of participants can achieve better levels of set consensus than the whole set. Nailed down, this may allow us to compactly capture all "natural" models [13], such as, e.g., the model of *set consensus collections* [8] for which only special cases of k-set consensus [13] and k-test-and-set have been, in this sense, understood so far.

References

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

[2] B. Alpern and F. B. Schneider. Defining liveness. *Inf. Process. Lett.*, 21(4):181–185, Oct. 1985.

[3] E. Borowsky and E. Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In *STOC*, pages 91–100. ACM Press, May 1993.

[4] E. Borowsky and E. Gafni. Immediate atomic snapshots and fast renaming. In *PODC*, pages 41–51, New York, NY, USA, 1993. ACM Press.

[5] E. Borowsky and E. Gafni. A simple algorithmically reasoned characterization of wait-free computation (extended abstract). In *PODC '97: Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 189–198, New York, NY, USA, 1997. ACM Press.

[6] Z. Bouzid, E. Gafni, and P. Kuznetsov. Strong equivalence relations for iterated models. In *OPODIS*, pages 139–154, 2014.

[7] S. Chaudhuri. Agreement is harder than consensus: Set consensus problems in totally asynchronous systems. In *Proceedings of the 9th ACM Symposium on Principles of Distributed Computing*, pages 311–324, Québec City, Québec, Canada, Aug. 1990.

[8] C. Delporte-Gallet, H. Fauconnier, E. Gafni, and P. Kuznetsov:. Set-consensus collections are decidable. In *OPODIS*, 2016.

[9] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and A. Tielmann. The disagreement power of an adversary. *Distributed Computing*, 24(3-4):137–147, 2011.

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

[11] E. Gafni. On the wait-free power of iterated-immediate-snapshots. Unpublished manuscript, `http://www.cs.ucla.edu/~eli/eli/wfiis.ps`, 1998.

[12] E. Gafni. Round-by-round fault detectors (extended abstract): Unifying synchrony and asynchrony. In *Proceedings of the 17th Symposium on Principles of Distributed Computing*, 1998.

[13] E. Gafni, Y. He, P. Kuznetsov, and T. Rieutord. Read-write memory and k-set consensus as an affine task. In *OPODIS*, 2016. Technical report: https://arxiv.org/abs/1610.01423.

[14] E. Gafni and P. Kuznetsov. Turning adversaries into friends: Simplified, made constructive, and extended. In *OPODIS*, pages 380–394, 2010.

[15] E. Gafni, P. Kuznetsov, and C. Manolescu. A generalized asynchronous computability theorem. In *PODC*, 2014.

[16] E. Gafni and S. Rajsbaum. Distributed programming with tasks. In *Principles of Distributed Systems - 14th International Conference, OPODIS 2010, Tozeur, Tunisia, December 14-17, 2010. Proceedings*, pages 205–218, 2010.

[17] M. Herlihy. Wait-free synchronization. *ACM Trans. Prog. Lang. Syst.*, 13(1):123–149, Jan. 1991.

[18] M. Herlihy, D. N. Kozlov, and S. Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, 2014.

[19] M. Herlihy and S. Rajsbaum. Simulations and reductions for colorless tasks. In *PODC*, pages 253–260, 2012.

[20] M. Herlihy and N. Shavit. The topological structure of asynchronous computability. *J. ACM*, 46(2):858–923, 1999.

[21] F. Junqueira and K. Marzullo. A framework for the design of dependent-failure algorithms. *Concurrency and Computation: Practice and Experience*, 19(17):2255–2269, 2007.

[22] D. N. Kozlov. Chromatic subdivision of a simplicial complex. *Homology, Homotopy and Applications*, 14(1):1–13, 2012.

[23] P. Kuznetsov. Understanding non-uniform failure models. *Bulletin of the EATCS*, 106:53–77, 2012.

[24] P. Kuznetsov and T. Rieutord. Agreement functions for distributed computing models. In *NETYS*, 2017. To apper, technical report: https://arxiv.org/abs/1702.00361.

[25] P. Kuznetsov, T. Rieutord, and Y. He. Brief announcement: Compact topology of shared-memory adversaries. In *31th International Symposium on Distributed Computing*, DISC'16, pages 56:1–4, 2017.

[26] M. Raynal and J. Stainer. Increasing the power of the iterated immediate snapshot model with failure detectors. In *SIROCCO*, pages 231–242, 2012.

[27] M. Saks and F. Zaharoglou. Wait-free k-set agreement is impossible: The topology of public knowledge. *SIAM J. on Computing*, 29:1449–1483, 2000.

[28] V. Saraph, M. Herlihy, and E. Gafni. Asynchronous computability theorems for t-resilient systems. In *DISC*, pages 428–441, 2016.

[29] E. H. Spanier. *Algebraic topology*. McGraw-Hill Book Co., New York, 1966.

[30] G. Taubenfeld. The computational structure of progress conditions. In *DISC*, 2010.

A Simplicial complexes

We recall now several notions from combinatorial topology. For more detailed coverage of the topic please refer to [18, 29].

A *simplicial complex* is a set V, together with a collection C of finite non-empty subsets of V such that:

1. For any $v \in V$, the one-element set $\{v\}$ is in C;
2. If $\sigma \in C$ and $\sigma' \subseteq \sigma$, then $\sigma' \in C$.

The elements of V are called *vertices*, and the elements of C are called a *simplices*. We usually drop V from the notation, and refer to the simplicial complex as C.

A subset of a simplex is called a *face* of that simplex.

A *sub-complex* of C is a subset of C that is also a simplicial complex.

The *dimension* of a simplex $\sigma \in C$ is its cardinality minus one. The k-skeleton of a complex C, denoted $\mathrm{Skel}^k C$, is the sub-complex formed of all simplices of C of dimension k or less.

A simplicial complex C is called *pure* of dimension n if C has no simplices of dimension $> n$, and every k-dimensional simplex of C (for $k < n$) is a face of an n-dimensional simplex of C.

Let A and B be simplicial complexes. A map $f : A \to B$ is called *simplicial* if it is induced by a map on vertices; that is, f maps vertices to vertices, and for any $\sigma \in A$, we have

$$f(\sigma) = \bigcup_{v \in \sigma} f(\{v\}).$$

A simplicial map f is called *non-collapsing* (or *dimension-preserving*) if $\dim f(\sigma) = \dim \sigma$ for all $\sigma \in A$.

A map $\Phi : A \to 2^B$ (mapping simplices of A to sub-complexes of B) is called *carrier* if for all $\tau, \sigma \in A$, we have $\Phi(\tau \cap \sigma) \subseteq \Phi(\tau) \cap \Phi(\sigma)$. A simplicial map $\phi : A \to B$ is said to be *carried by a carrier map* $\Phi : A \to 2^B$ if for all $\sigma \in A$, $\phi(\sigma) \subset \Phi(\sigma)$.

Any simplicial complex C has an associated *geometric realization* $|C|$, defined as follows. Let V be the set of vertices in C. As a set, we let C be the subset of $[0,1]^V = \{\alpha : V \to [0,1]\}$ consisting of all functions α such that $\{v \in V \mid \alpha(v) > 0\} \in C$ and $\sum_{v \in V} \alpha(v) = 1$. For each $\sigma \in C$, we set $|\sigma| = \{\alpha \in |C| \mid \alpha(v) \neq 0 \Rightarrow v \in \sigma\}$. Each $|\sigma|$ is in one-to-one correspondence to a subset of $\mathcal{R}^n$ of the form $\{(x_1, \ldots, x_n) \in [0,1]^n \mid \sum x_i = 1\}$. We put a metric on $|C|$ by $d(\alpha, \beta) = \sum_{v \in V} |\alpha(v) - \beta(v)|$.

A non-empty complex C is called *k-connected* if, for each $m \leq k$, any continuous map of the m-sphere into $|C|$ can be extended to a continuous map over the $(m+1)$-disk.

A *subdivision* of a simplicial complex C is a simplicial complex C' such that:

1. The vertices of C' are points of $|C|$.
2. For any $\sigma' \in C'$, there exists $\sigma \in C$ such that $\sigma' \subset |\sigma|$.
3. The piecewise linear map $|C'| \to |C|$ mapping each vertex of C' to the corresponding point of C is a homeomorphism.

Chromatic complexes. We now turn to the chromatic complexes used in distributed computing, and recall some notions from [20].

Fix $n \geq 0$. The *standard n-simplex* $\mathbf{s}$ has $n+1$ vertices, in one-to-one correspondence with $n+1$ *colors* $0, 1, \ldots, n$. A face $\mathbf{t}$ of $\mathbf{s}$ is specified by a collection of vertices from $\{0, \ldots, n\}$. We view $\mathbf{s}$ as a complex, with its simplices being all possible faces $\mathbf{t}$.

A *chromatic complex* is a simplicial complex C together with a non-collapsing simplicial map $\chi : C \to \mathbf{s}$. Note that C can have dimension at most n. We usually drop χ from the notation. We write $\chi(C)$ for the union of $\chi(v)$ over all vertices $v \in C$. Note that if $C' \subseteq C$ is a sub-complex of a chromatic complex, it inherits a chromatic structure by restriction.

In particular, the standard n-simplex $\mathbf{s}$ is a chromatic complex, with χ being the identity.

Every chromatic complex C has a *standard chromatic subdivision* $\operatorname{Chr} C$. Let us first define $\operatorname{Chr}\mathbf{s}$ for the standard simplex $\mathbf{s}$. The vertices of $\operatorname{Chr}\mathbf{s}$ are pairs $(i,\mathbf{t})$, where $i \in \{0,1,\ldots,n\}$ and $\mathbf{t}$ is a face of $\mathbf{s}$ containing i. We let $\chi(i,\mathbf{t}) = i$. Further, $\operatorname{Chr} s$ is characterized by its n-simplices; these are the $(n+1)$-tuples $((0,\mathbf{t}_0),\ldots,(n,\mathbf{t}_n))$ such that:

(a) For all $\mathbf{t}_i$ and $\mathbf{t}_j$, one is a face of the other;

(b) If $j \in \mathbf{t}_i$, then $\mathbf{t}_j \subseteq \mathbf{t}_i$.

The geometric realization of $\mathbf{s}$ can be taken to be the set $\{\mathbf{x} = (x_0,\ldots,x_n) \in [0,1]^{n+1} \mid \sum x_i = 1\}$, where the vertex i corresponding to the point $\mathbf{x}^i$ with i coordinate 1 and all others coordinate 0. Then, we can identify a vertex $(i,\mathbf{t})$ of $\operatorname{Chr}\mathbf{s}$ with the point

$$\frac{1}{2k-1}\mathbf{x}_i + \frac{2}{2k-1}\Big(\sum_{\{j\in\mathbf{t}\mid j\neq i\}} \mathbf{x}_j\Big) \;\in |\mathbf{s}| \subset \mathcal{R}^{n+1},$$

where k is the cardinality of $\mathbf{t}$. Thus, $\operatorname{Chr}\mathbf{s}$ becomes a subdivision of $\mathbf{s}$ and the geometric realizations are identical: $|\mathbf{s}| = |\operatorname{Chr}\mathbf{s}|$. The standard chromatic subdivision, $\operatorname{Chr}\mathbf{s}$, is illustrated for a 3-process system in Figure 1(a).

Next, given a chromatic complex C, we let $\operatorname{Chr} C$ be the subdivision of C obtained by replacing each simplex in C with its chromatic subdivision. Thus, the vertices of $\operatorname{Chr} C$ are pairs (p,σ), where p is a vertex of C and σ is a simplex of C containing p. If we iterate this process m times we obtain the m^{th} chromatic subdivision, $\operatorname{Chr}^m C$.

Let A and B be chromatic complexes. A simplicial map $f : A \to B$ is called a *chromatic map* if for all vertices $v \in A$, we have $\chi(v) = \chi(f(v))$. Note that a chromatic map is automatically non-collapsing. A chromatic map has chromatic subdivisions $\operatorname{Chr}^m f : \operatorname{Chr}^m A \to \operatorname{Chr}^m B$. Under the identifications of topological spaces $|A| \cong |\operatorname{Chr}^m A|, |B| \cong |\operatorname{Chr}^m B|$, the continuous maps $|f|$ and $|\operatorname{Chr}^m f|$ are identical.

A simplicial map ϕ is carried by the carrier map Δ if $\phi(\sigma) \subset \Delta(\sigma)$ for every simplex σ in their domain.