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[3]

[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 *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. While an adversary is in general defined as a *non-compact* set of infinite runs, its affine task is just a finite subset of runs of the 2-round iterated immediate snapshot model. Our results generalize and improve all previously derived topological characterizations of distributed-computing models.

Regular and student paper: T. Rieutord and Y. He are full-time students

1 Introduction

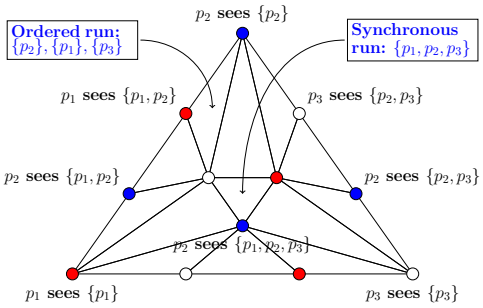
Compact topology of wait-freedom. Distributed computing is a jungle of models, parameterized by types of failures, synchrony assumptions, and employed communication primitives. Determining relative computability power of these models ("is model A more powerful than model B") is an intriguing and important problem.

In this paper, we deal 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.

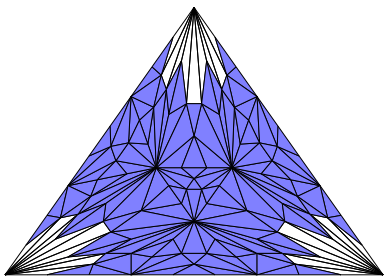
The *wait-free* model of computation [15] makes no assumptions about the number of failures that can occur, so no correct process can be prevented from making progress. Herlihy and Shavit proposed an elegant characterization of wait-free task computability through the existence of a specific continuous map from geometrical structures describing inputs and outputs of a given task [19]. 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 to the *output* simplicial complex, satisfying the specification of T. In particular, we can choose this subdivision to be the iterated *standard chromatic* subdivision (Figure 1(a)). The subdivision precisely captures the output complex of the *immediate snapshot* (IS) task [5]. By solving the IS task iteratively, proposing the view obtained in the current iteration as the input value of the next one, we obtain the *iterated* immediate snapshot (IIS) model.

Thus, the topological characterization of Herlihy and Shavit [19] can be interpreted as: the set of wait-free solvable task is precisely the set of tasks solvable in the IIS model. The ability of (iteratively) solving the IS task allows us 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, 19, 25], more general models were considered. The prominent *adversarial* failure model [8] is defined through a collection $\mathcal{A}$ of process subsets, called *live sets*, and requires that, in every run of the



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



(b) T_1, the task corresponding to 1-resilience.

Figure 1: Examples of subsets of iterated standard chromatic subdivisions of $\mathbf{s}$ in dimension 2.

corresponding *adversarial $\mathcal{A}$-model*, the set of processes taking infinitely many steps must be a live set.

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. Notice that, assuming the conventional "longest-prefix" metric [2], the model is *non-compact*. Indeed, all finite prefixes of an infinite run can be in the model, even though the run (the infinite limit of the sequence of its ever-extending finite prefixes) is not. Consider, for example, an infinite "solo" run in which exactly one process takes steps in a 1-resilient system of three processes.

Saraph et al. [26] recently proposed a direct characterization of t-resilient task computability via a specific task T_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)).

Solving a task T in the t-resilient model is then equivalent to finding a map from iterations of T_t (starting from the input complex of T) to the output complex of T. Therefore, T_t is a compact representation of a (non-compact) model of t-resilience.

Compact topology of fair adversaries. In this paper, we present a compact topological characterization of the large class of *fair* adversarial models [23] that subsumes the models of wait-freedom and t-resilience. We show that a specific task $\mathcal{R}_\mathcal{A}$ captures the task computability of an adversary $\mathcal{A}$. Informally, the task consists in solving chromatic simplex agreement [5, 19] on a specific *subcomplex* of the second iteration of the standard chromatic subdivision. Such a tasks is called *affine* [10, 13], as the geometrical representation of its output complex is a union of affine spaces (see, e.g., Figure 1(b)).

Our characterization is expressed in an abstract way via *agreement functions* [23]. For each set of *participating* processes (i.e., processes that took at least one step in the computation), the agreement function determines the best level of set consensus that can be achieved if only these processes *participate* in the computation. The agreement function corresponding to any adversary can be efficiently computed for any adversary [11, 23]. It turns out that agreement functions encode enough information to characterize the task computability of any *fair* adversary [23]. Informally, a *fair* adversary does not allow a *subset* of processes participating in a computation to achieve a better set consensus than the whole set of participants. The class of *fair* adversaries includes *superset-closed* [22] and *symmetric* [28]. In particular, the t-resilient adversary is both superset-closed and symmetric, and the *k-obstruction free* adversary, recently characterized in a compact way [10], is symmetric.

Our characterization can then be put as a generalization of the celebrated Asynchronous Computability Theorem (ACT) by Herlihy and Shavit [19]:

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 map from $\mathcal{I}$ to sub-complexes of $\mathcal{O}$, is solvable in a fair adversarial $\mathcal{A}$-model if and only if there exists a natural number ℓ and a simplicial map $\phi : \mathcal{R}^{\ell}_{\mathcal{A}}(\mathcal{I}) \to \mathcal{O}$ carried by Δ (informally, respecting the task specification Δ).

This result generalizes all topological characterizations of distributed computing models [10, 13, 19, 26], as it applies to *all* tasks (not only colorless) and *all* fair adversaries (not only t-resilient and k-obstruction-free). Furthermore, our characterization is compact: we match (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 model preliminaries and Section 3 recalls the definitions of adversaries, agreement functions, and the states the equivalence of task computability in a fair adversarial model and the corresponding α-model. In Section 4, we present the definition of the affine task $\mathcal{R}_\alpha$ corresponding to an α-model. In Section 5, we show that $\mathcal{R}_\alpha$ can be implemented in most α-model. In Section 6, we show that any task solvable in the α-model can be solved by iterating $\mathcal{R}_\alpha$. Section 7 reviews related work and Section 8 concludes the paper. Some proofs and missing details are delegated to the optional appendix.

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, 19].

Communication models. The atomic-snapshot (AS) memory is represented as a vector of shared variables, where processes are associated to distinct vector positions, and it exports two operations: *update* and *snapshot*. An *update* operation performed by p_i replaces the shared variable at position i with a new value and a *snapshot* returns the current state of the vector.

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 *immediate snapshot* 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 and runs. 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.

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 immediate snapshot operation on memory M_{r-1} is submitted as the input value for the immediate snapshot operation on memory M_r. After a certain number of such (asynchronous) rounds, 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.

Tasks. In this paper, we focus on distributed *tasks* [19]. 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. An input $\perp$ denotes a *non-participating* process and an output value $\perp$ denotes an *undecided* process. Check [16] for more details on the definition.

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 where: (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* [16, 27] to give a combinatorial representation of the IIS model. See Appendix A for missing details. 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 used to obtain the set of processes associated to a simplex.

The *standard chromatic subdivision* [19] 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, it is homeomorphic to $|\mathbf{s}|$, the geometric realization of $\mathbf{s}$ [21]. 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 [19].

The *carrier* of simplex $\sigma \in \mathrm{Chr}^m\,\mathbf{s}$ relatively to $\mathrm{Chr}^{m'}\,\mathbf{s}$, with $m' < m$, which we denote as $carrier(\sigma, \mathrm{Chr}^{m'}\,\mathbf{s})$ is the smallest simplex $\sigma' \in \mathrm{Chr}^{m'}\,\mathbf{s}$ such that in the geometric realization of σ, $|\sigma|$, is included in $|\sigma'|$. Intuitively, for a vertex v in $\mathrm{Chr}^m\,\mathbf{s}$, $carrier(v, \mathrm{Chr}^m\,\mathbf{s})$ 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 thr 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 $\mathcal{C}$. In the instances of simplex agreement considered in characterizations of wait-free task computability [5, 19], 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, 19], 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 to 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 to 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 iteration 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 to 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 processes that are correct in that run belongs to $\mathcal{A}$. An adversarial $\mathcal{A}$-model is defined as the set of $\mathcal{A}$-compliant runs.

The *agreement function* [23] of an adversarial model $\mathcal{A}$ is a function $\alpha_{\mathcal{A}} : 2^{\Pi} \to \{0, \ldots, n\}$, such that for each $P \in 2^{\Pi}$, in the set of runs of $\mathcal{A}$ in which no process in $\Pi \setminus P$ participates, $\alpha_{\mathcal{A}}(P)$-set consensus can be solved, but $(\alpha_{\mathcal{A}}(P) - 1)$-set consensus cannot. Note that by convention, if $\Pi \setminus P$ is not a valid participating set, as it does not include a live set, $\alpha_{\mathcal{A}}(P)$ is set to be equal to 0. An *α-adaptive set consensus* is an abstraction solving a set consensus protocol such that at the time k distinct output values are returned the participating set P is such that $\alpha(P) \geq k$: the abstraction takes a value as an input, returns a proposed value as an output, and ensures that at the time when k distinct output has been returned at least P processes, with $\alpha(P) \geq k$, are currently participating.

Let $\mathcal{AF}(\Pi)$ be the set of all agreement functions corresponding to some adversarial model in Π. For all $\alpha \in \mathcal{AF}(\Pi)$, we can define a natural model derived from the agreement function as follows:

Definition 1 (α-model)**.** *The* α-model *is the set of runs in which, assuming that the participating set is P, $\alpha(P) \geq 1$ and at most $\alpha(P) - 1$ participating processes are faulty.*

An adversary is *superset-closed* [22] if each superset of a set of an element of $\mathcal{A}$ is also an element of $\mathcal{A}$, i.e., $\forall S \in \mathcal{A}$, $\forall S' \subseteq \Pi$, $S \subseteq S'$: $S' \in \mathcal{A}$. Superset-closed adversaries provide an interesting non-uniform generalization of the classical t-resilience condition [17]: e.g., the *t-resilient* adversary in a system of n processes consists of all sets of $n - t$ or more processes.

An adversary $\mathcal{A}$ is *symmetric* if it does not depend on process identifiers and thus only on the size of live sets: $\forall S \in \mathcal{A}$, $\forall S' \subseteq \Pi$, $|S'| = |S| \implies S' \in \mathcal{A}$.

For $P \in 2^{\Pi}$, let $\mathcal{A}|_P$ denote the set of all elements of $\mathcal{A}$ that are subsets of P, and $csize(\mathcal{A}|_P)$ denote the size of the minimal hitting set in $\mathcal{A}|_P$, i.e., the minimal subset of P that intersects with each element in $\mathcal{A}|_P$. It is shown in [11] that the smallest k such that $\mathcal{A}$ can solve k-set consensus can be computed using a function $setcon(\mathcal{A})$, and thus that $\alpha_{\mathcal{A}}(P) = setcon(\mathcal{A}|_P)$. For any superset-closed adversary $\mathcal{A}$, we have $\alpha_{\mathcal{A}}(P) = setcon(\mathcal{A}|_P) = csize(\mathcal{A}|_P)$. Moreover, for any symmetric adversary $\mathcal{A}$, we have $\alpha_{\mathcal{A}}(P) = setcon(\mathcal{A}|_P) = |\{k \in \{1, \ldots, |P|\} : \exists S \in \mathcal{A}, |S| = k\}|$ [11].

Fair adversaries. Informally, an adversary is *fair* [23] if a subset Q of participating processes P cannot achieve better set consensus than the whole set of participants (unless $|Q|$ is smaller than $\alpha_{\mathcal{A}}(P)$). The set consensus power of a subset Q of the participating processes P corresponds to the set consensus power of the adversary $\mathcal{A}|_{P,Q} = \{S \in \mathcal{A} : (S \subseteq P) \wedge (S \cap Q \neq \emptyset)\}$. Therefore, 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 [23]. It turns out that the task computability of a fair adversary is captured precisely by the $\alpha_{\mathcal{A}}$-model, i.e., they both solve *the same* set of tasks (we say that the models are *equivalent*).

Theorem 1. *[23] 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 $\alpha_{\mathcal{A}}$-model.*

Theorem 1 was proved using the following result, which will be instrumental for us too:

Theorem 2. *[23] For any task T, if T is solvable in an α-model, then T is solvable in* any *read-write shared memory model which can solve the α-adaptive set consensus task.*

4 Defining the affine task for an α-model

Given an agreement function $\alpha \in \mathcal{AF}(\Pi)$, we define the affine task $\mathcal{R}_\alpha$, a subcomplex of the second degree of the standard chromatic subdivision $\mathrm{Chr}^2\, \mathbf{s}$. In Sections 5 and 6, we show that $\mathcal{R}_\alpha^*$, i.e., the model of IIS runs obtained by iterating $\mathcal{R}_\alpha$, is equivalent to the α-model regarding task solvability.

Two classes of affine tasks were recently defined. The class $\mathcal{R}_{t-res}$ was introduced in [26], with $\mathcal{R}_{t-res}^*$ equivalent to the t-resilient model. Similarly, the class $\mathcal{R}_k$ was introduced in [10], with $\mathcal{R}_k^*$ equivalent to the k-concurrent model. Interestingly, the models of t-resilience and k-concurrency

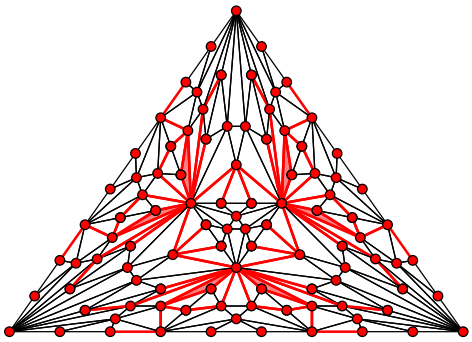


Figure 2: The 2-contention complex is shown in red in dimension 2.

correspond to two "well-behaved" sub-classes of α-models on opposite sides of the spectrum. In a sense, an α-model can be seen as a combination of resilience and concurrency conditions. Our definition of $\mathcal{R}_\alpha$ thus combines the concurrency features, expressed through the notion of *contention* simplices, and the resilience features, expressed through the notion of *critical* simplices.

Contention simplices. The carrier of a vertex v in $\mathrm{Chr}\,\mathbf{s}$ determines the view process $\chi(v)$ obtains after completing the corresponding run of immediate snapshot (IS). Thus, the *order* in which the processes obtained their immediate snapshots materializes in the resulting simplex in $\mathrm{Chr}\,\mathbf{s}$ through the inclusion-based order of the carriers of their vertices. We can therefore define a partial order on vertices of a simplex in $\mathrm{Chr}^2\,\mathbf{s}$, based on the *views* the processes obtain in the first IS and the second IS. For a vertex $v \in \mathrm{Chr}^2\,\mathbf{s}$, we define $\mathit{View}^1(v) = \mathit{carrier}(v', \mathbf{s})$ with v' such that $(\chi(v) = \chi(v')) \wedge (v' \in \mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s}))$, i.e., the first IS view that $\chi(v)$ had in the corresponding run. Similarly, $\mathit{View}^2(v) = \mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s})$, i.e., the second IS view.

Note that the more concurrency a run has, the less processes are ordered regularly. To capture the fact that a set of processes was executed concurrently in a run, we compare vertices' carriers. We say that a simplex δ in $\mathrm{Chr}^2\,\mathbf{s}$ is a 2-contention simplex, if any two vertices in δ have different ordered View^1 and View^2. Let Cont_2 denote the set of 2-contention simplices in $\mathrm{Chr}^2\,\mathbf{s}$ defined as follows[1]:

Definition 2. *[2-Contention simplices]* $\forall \sigma \in \mathrm{Chr}^2\,\mathbf{s} : \sigma \in \mathit{Cont}_2 \Leftrightarrow \forall v, v' \in \sigma, v \neq v' :$

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

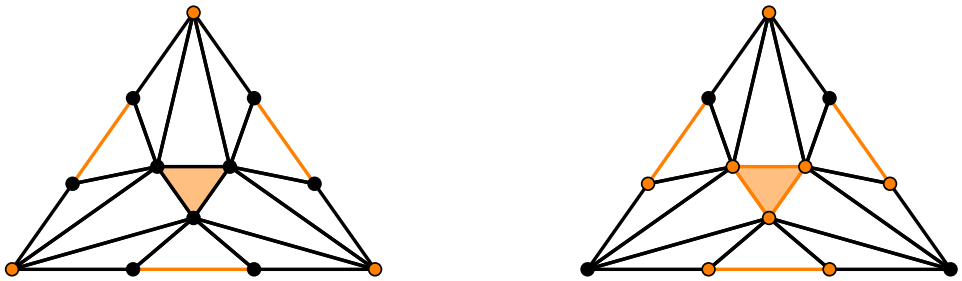
The set of 2-contention simplices is inclusion-closed: any face of a 2-contention simplex is also a 2-contention simplex. Therefore, we can define the 2-*contention complex* as the set of all 2-contention simplices. For example, the 2-contention simplices of the 3-process system is shown in Figure 2.

Critical simplices. Capturing resilience in an affine task [26] is about ensuring that every process is provided with sufficiently large views, i.e., that $\mathit{carrier}(v, \mathbf{s})$ is "large enough" for its vertex v. Providing a view P such that $\alpha(P) > 0$ is sufficient to characterize t-resilience. To see why this is the case, one can observe that if the participating set is P, then the number of distinct views V with $\alpha(V) > 0$ is exactly $\alpha(P)$. In a general α-model, however, it is not necessarily the case.

The definition of $\mathcal{R}_\alpha$ should allows us to determine which of the processes having views V with $\alpha(V) > 0$ are *prioritized*, i.e., can be used as "leaders" in solving $\alpha(P)$-set consensus. When the participation is P with $\alpha(P) > 0$, a leader with a view V with $\alpha(V) = \alpha(P)$ should be available. This is why the number "leaders" should scale according the set-consensus power of the participation.

But also, the number of set consensus proposals pushed by leaders when the participation is P should be smaller than $\alpha(P)$. Moreover, the leaders must be identifiable to other processes, in order for them to know that they have been prioritized and are not the result of a high concurrency.

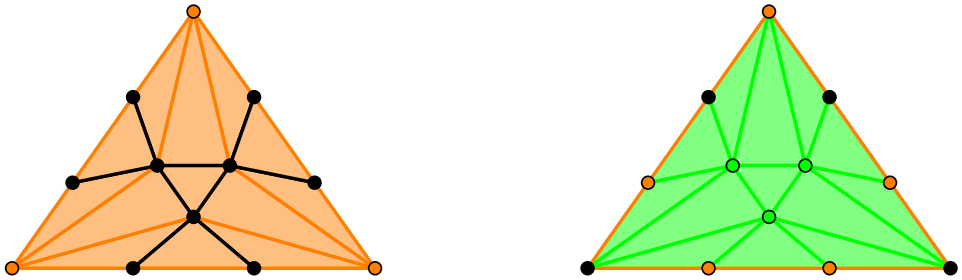
[1] In our recent paper [10] on the affine task for k-concurrency (a special case of an α-model), we introduced the notion of *contention sets*, capturing the sets of processes whose vertices in $\mathrm{Chr}^2\,\mathbf{s}$ have the same carrier in $\mathbf{s}$.



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



(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 4: 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.

To capture this intuition, we introduce the following notion. A simplex $\sigma \in \text{Chr}\,\mathbf{s}$ is a *critical simplex* if and only if: (1) all its vertices share the same carrier; and (2) the set consensus power associated to $carrier(\sigma, \mathbf{s})$ is strictly greater than the set consensus power of $\chi(carrier(\sigma, \mathbf{s})) \setminus \chi(\sigma)$.

Definition 3. $\forall \alpha \in \mathcal{AF}(\Pi), \forall \sigma \in \text{Chr}\,\mathbf{s}$*:*

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

Intuitively, the definition selects simplices which are identifiable by vertices which see them in the second IS because the selection is defined on the first subdivision, which corresponds to the first IS. Examples of critical simplices for some model of 3-process system are given in Figure 3.

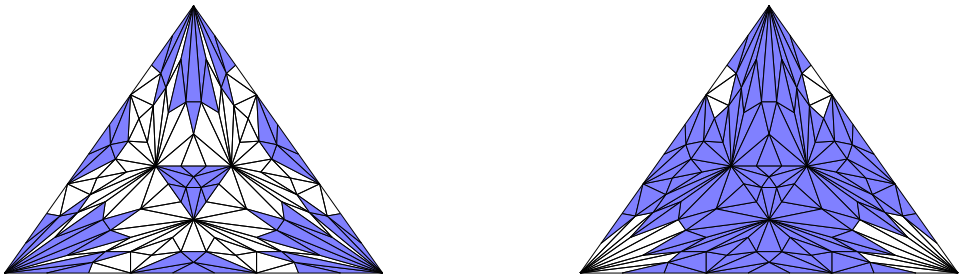
For convenience, given a simplex $\sigma \in \text{Chr}\,\mathbf{s}$, let $\mathcal{CS}_\alpha(\sigma)$ be the set of critical simplices in σ, i.e., $\mathcal{CS}_\alpha(\sigma) = \{\sigma' \subseteq \sigma : \sigma' \in Critical_\alpha(\sigma)\}$. Moreover, let $\mathcal{CSM}_\alpha(\sigma)$ be the 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 $\mathcal{CSV}_\alpha(\sigma)$ be the union of all processes observed by critical simplices in σ, i.e., $\mathcal{CSV}_\alpha(\sigma) = carrier(\mathcal{CSM}_\alpha(\sigma), \mathbf{s})$.

Concurrency map. We note that a critical simplex $\sigma \in \text{Chr}\,\mathbf{s}$ is associated to a set consensus power, i.e., $\alpha(\chi(carrier(\sigma, \mathbf{s})))$. A critical simplex associated to some k-set consensus power implies the concurrency level to be smaller than or equal to k. This allow us to define the concurrency level for every simplex in $\text{Chr}\,\mathbf{s}$ by the concurrency of its critical simplices:

Definition 4. *[Concurrency map]* $\forall \alpha \in \mathcal{AF}(\Pi), \forall \sigma \in \text{Chr}\,\mathbf{s}$*:*

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

Examples of concurrency maps for two models in a 3-process system are shown in Figure 4.



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

Affine task $\mathcal{R}_\alpha$. The affine task capturing α-models can be simply defined, based on (1) the set of critical simplices members, $\mathcal{CSM}_\alpha$, their view, $\mathcal{CSV}_\alpha$, and the concurrency map, $Conc_\alpha$, all derived from the definition of critical simplices; and (2), the definition of the 2-contention simplices.

The sub-complex $\mathcal{R}_\alpha \subseteq \text{Chr}^2\,\mathbf{s}$ is defined as follows: a $(n-1)$-dimensional simplex $\sigma \in \text{Chr}^2\,\mathbf{s}$ belongs to $\mathcal{R}_\alpha$ if and only if, every sub-simplex of σ of size k which is a 2-contention simplex and which does not contain any critical simplices members from $\text{Chr}\,\mathbf{s}$, nor are processes in their view, has a view in $\text{Chr}^2\,\mathbf{s}$ associated to a concurrency greater than or equal to k:

Definition 5. *[$\mathcal{R}_\alpha$]* $\forall \alpha \in \mathcal{AF}(\Pi), \forall \sigma \in \text{Chr}^2\,\mathbf{s}, dim(\sigma) = n-1 : \sigma \in \mathcal{R}_\alpha \Leftrightarrow \forall \theta \subseteq \sigma, \theta' = carrier(\theta, \text{Chr}\,\mathbf{s})$:

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

Some examples of affine tasks for some α-models in a 3 processes system are depicted in Figure 5.

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

Let T be any task that can be solved in $\mathcal{R}_\alpha^*$, i.e., by iterating our affine task $\mathcal{R}_\alpha$. To show that T is solvable with a *fair* adversary $\mathcal{A}$, we present an algorithm that, in the corresponding α-model, *solves* $\mathcal{R}_\alpha$, i.e., solves the chromatic simplex agreement task on the complex $\mathcal{R}_\alpha$. By iterating this task solution, we obtain a simulation of $\mathcal{R}_\alpha^*$, which implies a solution to T in the α-model.

In our solution of $\mathcal{R}_\alpha$ (Algorithm 1), every process p_i takes two immediate snapshots [5], *FirstIS* and *SecondIS*, where the input of *FirstIS* is its identifier and the input of *SecondIS* is the output of *FirstIS*. By construction, the outputs of this algorithm form a simplex in $\text{Chr}^2\,\mathbf{s}$. To ensure that this simplex is in $\mathcal{R}_\alpha$, after finishing *FirstIS*, each process p_i writes the outcome in its dedicated register $IS1[i]$, and then waits for its turn to proceed to *SecondIS*. The corresponding "waiting" conditions are given in Lines 5–11. Once *SecondIS* is completed, p_i writes its outcome in $IS2[i]$.

Intuitively, the waiting phase ensures that processes appearing in the critical simplices (maintained in the local boolean array *critical*) have higher priority to proceed with *SecondIS*. If the members of a critical simplex finishes *SecondIS*, then the processes with strictly smaller IS1 output are given priority, unless their number gets below the currently observed "concurrency level" (stored in the local variable *concurrency*). Otherwise, the process can proceed to *SecondIS*. Notice that in this case all processes with the same IS1 output will also be able to proceed to *SecondIS*.

In its waiting phase, p_i periodically goes over all processes and, based on their outputs in $IS1$ and $IS2$, updates *critical* and *concurrency*. *Concurrency* is set to the maximal set consensus power of the IS1 output of processes in a critical simplex that completed their IS2 (Line 10).

Theorem 3. *For all $\alpha \in \mathcal{AF}(\Pi)$, Algorithm 1 solves task $\mathcal{R}_\alpha$ in the α-model.*

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

1 **Shared**: $IS1[1\dots n] \in \mathcal{P}(\Pi)$, **initially** $\emptyset$; $IS2[1\dots n] \in \mathcal{P}(\Pi)$, **initially** $\emptyset$;
2 **Local**: $critical[1\dots n] \in \{true, false\}$, **initially** $false$; $concurrency \in \mathbb{N}$, **initially** 0;
3 $level[1\dots n] \in \mathcal{P}(\Pi)$;

4 $IS1[i] \leftarrow FirstIS(\textbf{InitialState})$;
5 **Do**
6 | **forall** $j \in \{1, \dots, n\}$ **do**
7 | | $level[j] \leftarrow \{p_k \in \Pi, IS1[k] = IS1[j]\}$;
8 | | $critical[j] \leftarrow (\alpha(IS1[j]) > \alpha(IS1[j] \setminus level[j]))$;
9 | | **if** $(\alpha(IS1[j]) > \alpha(IS1[j] \setminus \{p_k \in level[j], IS2[k] \neq \emptyset\})$ **then**
10 | | | $concurrency \leftarrow max(\alpha(IS1[j]), concurrency)$;
11 **While**
$(\neg critical[i]) \wedge (|\{p_j \in IS1[i] : IS2[j] = \emptyset \wedge \emptyset \subsetneq IS1[j] \subsetneq IS1[i] \wedge \neg critical[j]\}| \geq concurrency)$;
12 $IS2[i] \leftarrow SecondIS(IS1[i])$;

To convince ourselves that Algorithm 1 indeed solves $\mathcal{R}_\alpha$ in the α-model, we show that no correct process can be blocked forever in its waiting phase (Lines 5–11). This is because at most $\alpha(P) - 1$ processes can fail and therefore, at least one critical simplex will finish the *SecondIS*, enabling the remaining processes to proceed. Moreover, the more faulty processes try to block non-critical simplices members with small IS1 to finish their *SecondIS*, the less faulty processes are left to block critical simplices from finishing their *SecondIS*. But the more critical simplices finishes their *SecondIS*, the higher the value of *concurrency* is, counter-balancing which processes are faulty.

The second point we need to ensure is that the resulting simplex of $\mathrm{Chr}^2\,\mathbf{s}$ indeed belongs to $\mathcal{R}_\alpha$. One can see this by matching the conditions of the waiting phase with the definition of $\mathcal{R}_\alpha$ (Definition 5). The proof of Theorem 3 is given Appendix B.

6 From $\mathcal{R}^*_\alpha$ to α-model

In this section, we show that any task solvable in the α-model can be solved in $\mathcal{R}^*_\alpha$. More precisely, we present an algorithm that, in $\mathcal{R}^*_\alpha$, simulates a read-write shared memory system in which, additionally, α-adaptive set consensus can be solved. Then we can use Theorem 2 to obtain the desired result. The proofs for this section are delegated to Appendix C.

General structure of the simulation. In principle, our algorithm simulates read-write operations and accesses to instances α-adaptive set consensus (almost) independently. In each round of $\mathcal{R}^*_\alpha$, the input of each process to submit to the affine task $\mathcal{R}_\alpha$ is composed of a *read-write* part and an *agreement* part. Even though the simulated algorithm may have only one pending operation (read-write or propose for a set consensus instance), our algorithm requires that the process participates in all currently active agreement instances and the read-write memory simulation (possibly with "fake" inputs used to help slower processes to complete their simulated operations).

In general, such a simulation in $\mathcal{R}^*_\alpha$ can only be lock-free, i.e., ensuring that at least one process makes progress, as a slow process may never be observed by a fast process in an iterated model. Lock-freedom is good enough for us, as we are only interested in solving tasks in the simulated model: once a fast process outputs it "leaves" the computation and does not obstruct slower ones from making progress. For this, special input values are used by the decided processes to inform slower processes about the task outputs obtained by the faster ones.

Read-write simulation. To simulate a read-write shared memory, we rely upon the lock-free algorithm proposed in [14]. This algorithm, conveniently for us, uses iterated, but not necessarily immediate, snapshots. Thus, we can execute it in the global view resulting of iterations of $\mathcal{R}^*_\alpha$, i.e., on $carrier(v, \mathbf{s})$ for a vertex $v \in \mathcal{R}_\alpha$.

To make sure that every active (not yet terminated) participates in the read-write simulation in every iteration of $\mathcal{R}^*_\alpha$, even if the operations is not required by the simulated algorithm, we make the process propose the value of its last write operation. Notice that these "fake" inputs do not affect safety of the simulation, but potentially may affect liveness. We show that this, however, cannot happen.

An important feature of our algorithm, is that, even though the algorithm is lock-free but guarantees that if a process completing a simulated read-write operation at the end of a round r, has the smallest view in $\mathcal{R}_\alpha$ in that round and, thus, every other process in round r will see the it proposes to a set-consensus instance (if it has one). Hence, once a process completes a write, some progress is made: either it is a write operation of the simulated algorithm, or it is a "fake" write, but a simulated access to set consensus will be completed.

Solving α-adaptive set consensus in $\mathcal{R}^*_\alpha$. In the simulation of accesses to an α-adaptive set consensus instance, every active process adopts a decided or a proposed value as soon as another process is observed with one.

We describe below how the structure of $\mathcal{R}_\alpha$ is used to choose a decided value if no such value is seen. One issue to be resolved is that, to avoid potential deadlocks, a process is expected to take part in an agreement simulation even before it completes its first simulated write and, thus, before it is considered participating in the simulated run. Thus, we need to ensure that the number of distinct decided values in the simulated instance of agreement is allowed by α for the current participating set. For this, our decision function returns, along with the decided value, a set of participating processes. Knowing their input is however not sufficient for these processes to appear participating in the simulation. This knowledge must be shared with other processes before returning from the simulated agreement instance. Taking advantage of the read-write simulation, processes can simply write the inputs of these processes, which appears in the simulated run as a "block write."

Selecting a decision estimate. Given $Q \subseteq \Pi$, we introduce a vertex map μ_Q defined on all vertices $v \in \mathcal{R}_\alpha$ such that $\chi(v) \in Q$. Intuitively, Q is the current set of active processes and $\mu_Q(v)$ is a set of processes (intersecting with Q) that should appear participating before the decided value is returned. Formally:

$$\begin{aligned}\mu_Q(v) = \ &\textbf{if } (\chi(\mathcal{CSV}_\alpha(\mathit{carrier}(v, \operatorname{Chr}\mathbf{s}))) \cap Q \neq \emptyset)\\ &\textbf{then } \chi(\mathit{min}(\{\mathit{carrier}(\sigma', \mathbf{s}) : (\sigma' \in CS_\alpha(\mathit{carrier}(v, \operatorname{Chr}\mathbf{s})) : \chi(\mathit{carrier}(\sigma', \mathbf{s})) \cap Q \neq \emptyset)\})\\ &\textbf{else } \chi(\mathit{min}(\{\mathit{carrier}(v', \mathbf{s}) : (v' \in \mathit{carrier}(v, \operatorname{Chr}\mathbf{s})) \wedge (\mathit{dim}(v') = 0) \wedge (\mathit{carrier}(v', \mathbf{s}) \cap Q \neq \emptyset)\}).\end{aligned}$$

We list below several important properties of μ_Q.

We show first that μ_Q indeed returns a valid set of processes with inputs (a subset of the view v), which moreover intersects with Q:

Property 4. *[Validity of μ_Q]* $\forall v \in \mathcal{R}_\alpha, \mathit{dim}(v) = 0, \chi(v) \in Q$:

$$(\mu_Q(v) \subseteq \chi(\mathit{carrier}(v, \mathbf{s}))) \wedge (\mu_Q(v) \cap Q \neq \emptyset).$$

The principal guarantee μ_Q needs to ensure is that the number of distinct sets returned by μ_Q for a given simplex in $\mathcal{R}_\alpha$ cannot exceed the set consensus power of the union of these sets:

Property 5. *[Agreement of μ_Q]* $\forall Q \subseteq \Pi, (\forall \sigma \in \mathcal{R}_\alpha : \mathit{dim}(\sigma) = n-1), (\forall \theta \subseteq \sigma : \chi(\theta) \subseteq Q)$:

$$|\{\mu_Q(v) : v \in \theta\}| \leq \alpha(\cup_{v \in \theta}(\mu_Q(v))).$$

Finally, let us also observe that set Q can be determined locally based on the process view, i.e., the output of μ_Q is determined solely by the observed set of terminated processes:

Property 6. *[Robustness of μ_Q]* $\forall v \in \mathcal{R}_\alpha, \mathit{dim}(v) = 0, \forall Q \subseteq \Pi$:

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

Map μ_Q can then be used to provide an estimate of the decided value in an agreement instance. This is done by selecting any process in the set returned by μ_Q that belongs to Q and has a proposal value. Let min_Q be the function returning the minimal such process. Note that min_Q preserves the robustness property of μ_Q. It can be shown that if every process has a proposal value, then the map $min_Q \circ \mu_Q$ defines a decision map for a set consensus algorithm such that:

Property 7. $\forall Q \subseteq \Pi, (\forall \sigma \in \mathcal{R}_\alpha : dim(\sigma) = n-1), (\forall \theta \subseteq \sigma : \chi(\theta) \subseteq Q) : min_Q \circ \mu_Q(\sigma)$ *solves and* $\alpha(\cup_{v \in \theta}(\mu_Q(v)))$*-set consensus.*

Combined shared memory and α-adaptive set consensus simulation. Let us first show that if a "sufficiently fast" process is participating in our α-adaptive set consensus algorithm, then it must eventually terminate:

Lemma 8. *If a process in A with a minimal view in some iteration of $\mathcal{R}_\alpha$ shares a proposal, then the described protocol solves α-adaptive set consensus.*

Note that multiple set consensus protocols can be executed in parallel without interfering.

With such a property, we can now show that the combination of the read-write memory protocol and the α-adaptive set consensus protocols ensures progress to some active simulated process as long as there is one:

Lemma 9. *The read-write shared memory with access to set consensus simulation is lock-free.*

By Theorem 2, the resulting combination of read-write memory and α-adaptive set consensus can be used to solve any task solvable in the α-model. Finally, Theorem 3 and Theorem 1 imply:

Theorem 10. *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 solvable in $\mathcal{R}^*_\alpha$.*

As a side result, we get the following generalization of the Asynchronous Computability Theorem (ACT) [19]:

Theorem 11. *[Compact 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 $\phi : \mathcal{R}^\ell_\alpha(\mathcal{I}) \to \mathcal{O}$ carried by Δ.*

7 Related work

Inspired by the model of *dependent failures* proposed by Junqueira and Marzullo [20], Delporte et al. [8] suggested the notion of adversaries. Adversaries having the same set consensus power agree on the set of *colorless* tasks they solve [8, 12]. Intuitively, a colorless task, such as consensus or approximate agreement, can be defined as a map between a set of inputs and possible sets of outputs, without the notion of process identifiers.

Herlihy and Shavit [19] proposed a characterization of wait-free task computability through the existence of a specific continuous map from a subdivision of the input complex of a task $\mathcal{I}$ to its output complex $\mathcal{O}$. (The reader is referred to [16] for a thorough discussion of the use of combinatorial topology in distributed computability.) In particular, the characterization can consider the *iterated* standard chromatic subdivision and, thus, derive that a task is wait-free solvable if and only if it can be solved in the IIS model (capturing precisely by iterated standard chromatic subdivision [21]). Thus, the IS task is the affine task matching the wait-free model. This paper generalizes this result to any *fair* adversary.

Herlihy and Rajsbaum [18] studied colorless task computability in the special case of *superset-closed* adversaries [22], that, intuitively, do not expect any set of processes fail and are, thus, closed under the superset operation. 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 [20]. 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* way via a specific task whose iterations give an equivalent model.

Gafni et al. [13] 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, 9, 24] ("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. [26] 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 $\mathcal{R}_\alpha$ in the α-model is inspired by the t-resilient solution of T_t in [26]. Gafni et al. [10] 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 [23]. One can determine the agreement function of any given adversary using the formula suggested earlier for the set consensus power [11]. It has been shown in [23] that agreement functions encode enough information to characterize the task computability of any *fair* adversary.

8 Discussion

This paper generalizes all topological characterizations of distributed computing models known so far [10, 13, 19, 26]. It applies to all tasks (not necessarily colorless) and all *fair* adversarial models (not necessarily the model of t-resilience or k-obstruction-freedom). *Fair* adversaries include superset-closed [22] and symmetric [28] ones. Just as the wait-free characterization [19] 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 [26], 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* [16] 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 [16]. 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 5a). Instead, 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 [23]), 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 [10], such as, e.g., the model of *set consensus collections* [7] for which only special cases of k-set consensus [10] 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] C. Delporte-Gallet, H. Fauconnier, E. Gafni, and P. Kuznetsov:. Set-consensus collections are decidable. In *OPODIS*, 2016.

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

[9] 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.

[10] 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.

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

[12] E. Gafni and P. Kuznetsov. Relating L-Resilience and Wait-Freedom via Hitting Sets. In *ICDCN*, pages 191–202, 2011.

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

[14] 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.

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

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

[17] M. Herlihy and S. Rajsbaum. The topology of shared-memory adversaries. In *PODC*, pages 105–113, 2010.

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

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

[20] 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.

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

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

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

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

[25] 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.

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

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

[28] 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 [16, 27].

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 [19].

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.

B Proof of correctness of Algorithm 1

Lemma 12. *[Availability of critical simplices]:* $\forall\alpha \in \mathcal{AF}(\Pi), \forall\sigma \in \operatorname{Chr}\mathbf{s}$:

$$\chi(\sigma) = \chi(\mathit{carrier}(\sigma)) \implies \alpha(\chi(\sigma)) \leq \mathit{csize}(\mathcal{CS}_\alpha(\sigma)).$$

Proof. We proceed by induction on $\mathit{dim}(\sigma)$. The base case is trivial, as $\mathit{csize}(\emptyset) = 0$ and $\alpha(\emptyset) = 0$. Now consider a simplex $\sigma \in \operatorname{Chr}\mathbf{s}$ such that $\chi(\sigma) = \chi(\mathit{carrier}(\sigma))$ and $\alpha(\chi(\sigma)) = k$, $k \in \mathbb{N}$. Let us assume by induction that for all $\sigma' \in \operatorname{Chr}\mathbf{s}$ such that $\mathit{dim}(\sigma') < \mathit{dim}(\sigma)$ and $\chi(\sigma') = \chi(\mathit{carrier}(\sigma'))$, we have $\alpha(\chi(\sigma')) \leq \mathit{csize}(\mathcal{CS}_\alpha(\sigma'))$.

Now consider the face θ of σ consisting of all vertices of σ with the same carrier as σ, i.e., $\theta = \{v \in \sigma, \mathit{carrier}(v,\mathbf{s}) = \mathit{carrier}(\sigma,\mathbf{s})\}$. Let β be the complement of θ, i.e., $\beta = \sigma \setminus \theta$. Note that $\theta \neq \emptyset$ and that $\chi(\beta) = \chi(\mathit{carrier}(\beta))$. Therefore $\mathit{dim}(\beta) < \mathit{dim}(\sigma)$ and by assumption we have $\alpha(\chi(\beta)) \leq \mathit{csize}(\mathcal{CS}_\alpha(\beta))$. By the definition of $\mathcal{CS}_\alpha$, we have $(\mathcal{CS}_\alpha(\beta) \cup \mathcal{CS}_\alpha(\theta)) \subseteq \mathcal{CS}_\alpha(\sigma)$. Therefore if $\alpha(\chi(\beta)) = \alpha(\chi(\sigma))$, we have $\alpha(\chi(\sigma)) \leq \mathit{csize}(\mathcal{CS}_\alpha(\sigma))$.

Thus, let us assume that $\alpha(\chi(\sigma)) = \alpha(\chi(\beta)) + l$ with $l > 0$. Note that given any $P \subseteq \Pi$, $P \neq \emptyset$, and any $p \in P$ we have $\alpha(P) \geq \alpha(P \setminus \{p\}) \geq \alpha(P) - 1$[2]. Therefore, by a trivial induction we can deduce that for any set of processes $Q \subseteq P$, we have $\alpha(P) \geq \alpha(P \setminus Q) \geq \alpha(P) - |Q|$ or equivalently

[2]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 $\mathit{setcon}(\mathcal{A}|_{P\cup\{p\}}) \leq \mathit{setcon}(\mathcal{A}|_P) + 1$. Note that this might not be true for generic α models. Moreover, this is the only necessary condition for our equivalence result to hold for generic α models.

$\alpha(P \setminus Q) + |Q| \geq \alpha(P)$. So for any subset θ' of θ of size $dim(\theta) - l + 1$, $\chi(\sigma) \setminus \chi(\theta')$ contains $l - 1$ more processes than $\chi(\beta)$ and so we have:

$$\alpha(\chi(\sigma) \setminus (\chi(\theta'))) \leq \alpha(\chi(\beta)) + l - 1 < \alpha(\chi(\sigma)).$$

Therefore, $\theta' \in CS_\alpha(\sigma)$. Indeed, for every $v \in \theta'$ we have $carrier(v, \mathbf{s}) = carrier(\sigma, \mathbf{s}) = carrier(\theta', \mathbf{s})$. Moreover, as by assumption we have $\chi(\sigma) = \chi(carrier(\sigma, \mathbf{s}))$, we obtain $\alpha(\chi(carrier(\theta', \mathbf{s}))) < \alpha(\chi(carrier(\theta', \mathbf{s})) \setminus (\chi(\theta')))$.

This is true for all subsets θ' of θ of size $dim(\theta) - l + 1$. It is easy to see that $csize(\{\theta' \subseteq \theta, dim(\theta') = dim(\theta) - l + 1\}) = l$, and thus $csize(CS_\alpha(\theta)) \geq l$. As $csize(CS_\alpha(\sigma)) \geq csize(CS_\alpha(\theta)) + csize(CS_\alpha(\beta))$, we have $csize(CS_\alpha(\sigma)) \geq \alpha(\chi(\sigma))$. □

Corollary 13. *[Distribution of critical simplices]:* $\forall \alpha \in \mathcal{AF}(\Pi), \forall \sigma \in \mathrm{Chr}\,\mathbf{s}, \forall l \in \{1, \ldots, n\}$

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

Proof. Consider the face θ of σ consisting of all vertices of σ such that their carrier is associated with a set consensus power greater than $\alpha(\chi(\sigma)) - l + 1$, i.e., $\theta = \{v \in \sigma, \alpha(\chi(carrier(v, \mathbf{s}))) \geq l\}$. It is easy to see that $\{\sigma' \in \mathcal{CS}_\alpha(\sigma), \alpha(\chi(carrier(\sigma', \mathbf{s}))) \geq l\} = \mathcal{CS}_\alpha(\theta)$.

Let β be the complement of θ in σ, i.e., $\beta = \sigma \setminus \theta$. As $\chi(\sigma) = \chi(carrier(\sigma))$, we have $\chi(\beta) = \chi(carrier(\beta))$. Therefore, according to Lemma 12 we have $\alpha(\chi(\beta)) \leq csize(\mathcal{CS}_\alpha(\beta))$, hence $l - 1 \leq csize(\mathcal{CS}_\alpha(\beta))$. But as $(\mathcal{CS}_\alpha(\beta) \cup \mathcal{CS}_\alpha(\theta)) = \mathcal{CS}_\alpha(\sigma)$, we obtain $csize(\mathcal{CS}_\alpha(\theta)) = csize(\mathcal{CS}_\alpha(\sigma)) - csize(\mathcal{CS}_\alpha(\beta)) \geq \alpha(\chi(\sigma)) - l + 1$. □

Theorem 3. *For all $\alpha \in \mathcal{AF}(\Pi)$, Algorithm 1 solves task $\mathcal{R}_\alpha$ in the α-model.*

Proof. We show first that the set of outputs of the processes indeed belongs to $\mathcal{R}_\alpha$. Then we proceed to the more involved proof of termination.

The outputs belong to $\mathcal{R}_\alpha$: Except for the wait-phase, processes simply execute two rounds of immediate snapshots, therefore the set of outputs belongs to some simplex in $\mathrm{Chr}^2\,\mathbf{s}$. In order to belong to $\mathcal{R}_\alpha$, only 2-contention simplices are restricted. Let σ be the simplex resulting of the processes outputs. Consider a face θ of σ forming a 2-contention simplex which does not contain any critical simplex member, i.e., $\chi(\theta) \cap \chi(\mathcal{CSM}_\alpha(carrier(\sigma, \mathrm{Chr}\,\mathbf{s}))) = \emptyset$.

Consider the vertex v in θ, with $i = \chi(v)$, which has the smallest carrier $carrier(v, \theta)$. Note that vertices in a 2-contention simplex have distinct carriers, and so there is a smallest one. The variable $critical[i]$ can be set to true only if the parent of v in the first subdivision, i.e., $v' \in carrier(\sigma, \mathrm{Chr}\,\mathbf{s})$ such that $\chi(v') = \chi(v)$, is part in a critical set of $carrier(\sigma, \mathrm{Chr}\,\mathbf{s})$. Indeed, the output of p_i first immediate snapshot corresponds to $\chi(view^1(v))$, i.e., $\chi(Car(v', \mathbf{s}))$, and therefore $critical[i]$ was set to true only if v' belongs to a critical simplex in $carrier(\sigma, \mathrm{Chr}\,\mathbf{s})$. As p_i exited the wait phase with $critical[i] = false$, then the following condition was satisfied:

$$|\{p_j \in IS1[i], (IS2[j] = \emptyset) \wedge (\emptyset \subsetneq IS1[j] \subsetneq IS1[i]) \wedge (\neg critical[j])\}| < concurrency.$$

If $IS2[j] \neq \emptyset$, then p_j terminated its second immediate snapshot before p_i started it. This implies that p_j is associated to a vertex v_j such that $carrier(v_j, \mathrm{Chr}\,\mathbf{s}) \subsetneq carrier(v_j, \mathrm{Chr}\,\mathbf{s})$ and so $v_j \notin \theta$. Moreover, $\emptyset \subsetneq IS1[j] \subsetneq IS1[i]$ being false implies, if $j \neq i$, that $View^1(v) \subseteq View^1(v_j)$ and thus $v_j \notin \theta$ as v as $View^2(v)$ is the smallest. As $critical[j]$ also implies that $v_j \notin \theta$, we obtain that $dim(\theta) - 1 \leq concurrency$ (as p_i is not counted in the set) at the time p_i exited the wait-phase.

All that is left to show is that this value of the *concurrency* variable is smaller than or equal to $Conc(carrier(\theta, \mathrm{Chr}\,\mathbf{s})$. The value of *concurrency* is set to k if and only if a set of processes which terminated their second immediate snapshot forms a critical simplex in σ and has a set consensus power equal to k. Moreover, this critical simplex is a subset of $carrier(v, \mathrm{Chr}\,\mathbf{s})$ as the processes associated to it completed their second immediate snapshot. Thus, $Conc(carrier(v, \mathrm{Chr}\,\mathbf{s}) \geq concurrency$. Moreover as $carrier(v, \mathrm{Chr}\,\mathbf{s}) \subseteq carrier(\theta, \mathrm{Chr}\,\mathbf{s})$, we have $Conc(carrier(\theta, \mathrm{Chr}\,\mathbf{s}) \geq concurrency$.

Outputs are provided to all correct processes : Assume that the participation is equal to P. Therefore there is at most $\alpha(P) - 1$ processes which may have crashed in the α-model.

Now let us assume that some correct process did not terminate, and let p_i be such a process with the smallest first immediate snapshot output. This means that it fails infinitely often the test on line 11, and in particular after *IS1* and *IS2* were modified for the last time. Thus the following is always verified for p_i:

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

As p_i is the correct process with the smallest first immediate snapshot output, this set contains only failed processes, as $\emptyset \subsetneq IS1[j] \subsetneq IS1[i]$ is true. Let k be the number of such failed processes.

If any critical simplex with an associated set consensus power strictly greater than k was fully simulated in the second immediate snapshot, then *concurrency* would be set to a value greater than k and p_i would have terminated. But according to Corollary 13, in any completion of the first immediate snapshot outputs to include the failed processes, the minimal hitting set size of the critical sets with associated agreement power greater than or equal to $k+1$ is greater then or equal to $\alpha(P) - k$. Thus it requires at least $\alpha(P) - k$ failed processes to block all these critical simplices. This implies that the k failed processes blocking p_i and the $\alpha(P) - k$ failed processes blocking the critical simplices must intersect. A process in this intersection is not observed by p_i as a critical simplex member, therefore it cannot have written its first immediate snapshot output. But if it has not written its first immediate snapshot output, then it is not one of the k failed processes blocking p_i—a contradiction. □

C Proofs of Section 6

For convenience, let us recall the definition of μ_Q, and split it into two vertex maps γ_Q and δ_Q as follows:

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

With γ_Q and δ_Q defined as follows:

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

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

Property 4. *[Validity of μ_Q]* $\forall v \in \mathcal{R}_\alpha, dim(v) = 0, \chi(v) \in Q$:

$$(\mu_Q(v) \subseteq \chi(carrier(v, \mathbf{s}))) \wedge (\mu_Q(v) \cap Q \neq \emptyset).$$

Proof. Let us fix some vertex $v \in \mathcal{R}_\alpha$ and some $Q \subseteq \Pi$. Both γ_Q and δ_Q return the carrier of a subset of the carrier of v. By definition of the carrier, if $\sigma \subseteq \sigma'$ then $|\sigma| \subseteq |\sigma'|$ and therefore the carrier of σ is a subset of the carrier of σ'. Thus, both γ_Q and δ_Q return a subset of $carrier(carrier(v, \operatorname{Chr}\mathbf{s}), \mathbf{s})$. By transitivity of inclusion and as $\mathbf{s}$ is a subset of $\operatorname{Chr}\mathbf{s}$, we have $carrier(carrier(v, \operatorname{Chr}\mathbf{s}), \mathbf{s}) \subseteq carrier(v, \mathbf{s})$. Therefore, $\mu_Q(v) \subseteq \chi(carrier(v, \mathbf{s}))$.

If $\chi(\mathcal{CSV}_\alpha(carrier(v, \operatorname{Chr}\mathbf{s}))) \cap Q \neq \emptyset$ then there is a critical set in $carrier(v, \operatorname{Chr}\mathbf{s})$ which intersects with Q. Thus $\delta_Q(v)$, and hence $\mu_Q(v)$, returns a set of processes which intersect with Q.

Otherwise, $\chi(\mathcal{CSV}_\alpha(carrier(v, \operatorname{Chr}\mathbf{s}))) \cap Q = \emptyset$. According to the self-inclusion property of immediate snapshots transferred to carriers, $\chi(v) \in carrier(v, \operatorname{Chr}\mathbf{s})$ and thus $\chi(v) \in View^1$. Therefore, there is a process $v' \in carrier(v, \operatorname{Chr}\mathbf{s})$ such that $\chi(carrier(v', \mathbf{s})) \cap Q \neq \emptyset$. Therefore γ_Q, and hence μ_Q, returns a set of processes intersecting with Q. □

Let us show now some auxiliary lemmas that will be used for the proof of Property 5:

Lemma 14. $\forall Q \subseteq \Pi, (\forall \sigma \in \operatorname{Chr}^2 \mathbf{s} : \chi(\sigma) \subseteq Q)$:

$$(max(dim(\sigma') : \sigma' \subseteq \sigma \wedge \sigma' \in Cont_2\}) = k) \wedge (\forall v \in \sigma, \exists v' \in carrier(v, \operatorname{Chr}\mathbf{s}), \gamma_Q(v) = \chi(Car(v', \mathbf{s})))$$

$$\implies |\{\gamma_Q(v) : v \in \sigma\}| \leq k + 1.$$

Proof. Let us fix some set of processes $Q \subseteq \Pi$ and some simplex $\sigma \in \mathrm{Chr}^2\,\mathbf{s}$ such that $\chi(\sigma) \subseteq Q$. Consider the vertex v_1 of σ with one the smallest $View^2(v_1)$ and one the smallest $View^1(v_1)$ among those. According to the definition of γ_Q and to the assumption stating that γ_Q returns the $View^1(v')$ of a vertex v' of σ, we can derive that $\gamma_Q(v_1)$ returns $View^1(v_1)$.

Then let us repeat such an iterative process by selecting the vertex v_k such that v_k has the smallest $View^1(v_k)$ among the processes with the smallest $View^2$ not returning for $\gamma_Q(v)$ the $View^1(v_j)$ of a previously defined v_j with $j < k$. Like before, v_k will return $View^1(v_k)$.

This construction provides a list of vertices of the same size as the number of distinct $\gamma_Q(v)$ outputs in σ. Moreover, by construction we have $View^2(v_i) \subsetneq View^2(v_j)$ when $i < j$. But if $View^1(v_i) \subseteq View^1(v_j)$, then v_j would not have returned for $\gamma_Q(v_j)$, $View^1(v_j)$. Thus the sets of v_i forms a 2-contention simplex, proving the lemma claim. □

Lemma 15. $\forall \sigma \in \mathrm{Chr}\,\mathbf{s}, \forall \alpha \in \mathcal{AF}(\Pi), \forall k \in \mathbb{N}:$

$$|\{carrier(\sigma', \mathbf{s}), \sigma' \in \mathcal{CS}_\alpha(\sigma) \wedge \alpha(\chi(carrier(\sigma', \mathbf{s}))) = k\}| \leq 1.$$

Proof. Let us fix some simplex $\sigma \in \mathrm{Chr}\,\mathbf{s}$, some agreement function $\alpha \in \mathcal{AF}(\Pi)$ and some $k \in \mathbb{N}$. Assume that there exists two subsets of σ which are critical simplices associated with the same set consensus power k. Let C_1 and C_2 be the set of processes composing the critical simplices, and let V_1 and V_2 be the set of processes corresponding to their carriers.

By assumption we have $V_1 \neq V_2$. But by the immediacy and self-inclusion derived from immediate snapshots, we get, w.l.o.g., that $V_1 \subseteq (V_2 \setminus C_2)$. But according to the definition of critical simplices we have $\alpha(V_2 \setminus C_2) < \alpha(V_2) = \alpha(V_1)$ — A contradiction. □

Property 5. *[Agreement of μ_Q]* $\forall Q \subseteq \Pi, (\forall \sigma \in \mathcal{R}_\alpha : dim(\sigma) = n-1), (\forall \theta \subseteq \sigma : \chi(\theta) \subseteq Q):$

$$|\{\mu_Q(v) : v \in \theta\}| \leq \alpha(\cup_{v \in \theta}(\mu_Q(v))).$$

Proof. Let us fix some $Q \subseteq \Pi$, some simplex $\sigma \in \mathcal{R}_\alpha$ such that $dim(\sigma) = n-1$ and some simplex $\theta \subseteq \sigma$ such that $\chi(\theta) \subseteq Q$. Let $\{v_1, \ldots, v_k\}$ be the vertices of θ. W.l.o.g, assume that $carrier(v_1, \mathrm{Chr}\,\mathbf{s}) \subseteq \cdots \subseteq carrier(v_k, \mathrm{Chr}\,\mathbf{s})$. According to the test of μ_Q and the inclusion of the carriers, there is a $l \in \{0, \ldots, k\}$, such that $\forall v_i, i \leq l, \mu_Q(v_i) = \gamma_Q(v_i)$ and $\forall v_i, i > l, \mu_Q(v_i) = \delta_Q(v_i)$.

Assume that $l = k$. If this is the case, then none of the vertices of θ are neither $\mathcal{CSM}_\alpha(v)$, nor in $\mathcal{CSV}_\alpha(v)$. Let $\theta' \subseteq \theta$ be the greatest 2-contention simplex in θ. According to the definition of $\mathcal{R}_\alpha$ we have $dim(\theta') - 1 \leq Conc_\alpha(\theta')$. Thus there exists a critical set σ_c, observed by some process in θ', such that $dim(\theta') - 1 \leq \alpha(\chi(carrier(\sigma_c, \mathbf{s})))$. As it is observed by some process in θ', and thus in θ, it cannot intersect with Q as otherwise this vertex would have been associated to δ_Q. According to carrier inclusion, this implies that the view of every process in θ in the first subdivision, i.e., $View^1$, is greater than $carrier(\sigma_c, \mathbf{s})$. Therefore, $\alpha(\cup_{v \in \theta}(\mu_Q(v))) \geq dim(\theta') - 1$. By construction the selected processes return views provided by processes from themselves. Thus, according to Lemma 14, we obtain that $|\{\mu_Q(v) : v \in \theta\}| \leq dim(\theta') - 1 \leq \alpha(\cup_{v \in \theta}(\mu_Q(v)))$

Now let us assume that $l < k$, thus $c = \alpha(min(\{carrier(\mathcal{CS}_\alpha(\sigma)) : \chi(carrier(\mathcal{CS}_\alpha(\sigma))) \cap Q \neq \bot\})$ is well defined.

Now let us consider the processes v_i such that $i \leq l$. As they do not observe a critical set, then γ_Q necessarily maps to $View^1(v)$ with v equal to some v_i such that $i \leq l$. Let C be the set of processes which return the view of a $v_i \in \mathcal{CS}_\alpha(\sigma)$. Let us consider the set V of vertices composed of the v_i such that $i > l$ and composed of C. For the former, $\mu_Q(v_i)$ returns a set of processes which correspond to the carrier of a critical set of σ. According to Property 4, it must be a set which intersects with Q. This is also true for the latter according to carrier self inclusion. Therefore by assumption they return the view of a critical set with an associated set consensus power greater or equal to c. Let $|\{\mu_Q(v), v \in V\}| = m$. As $l < k$, we have $m > 0$. Then we have a vertex $v_i, i \in \{l+1, \ldots, k\}$ such that $\alpha(\mu_Q(v_i)) \geq m + c - 1$. Indeed, according to Lemma 15 there is at most one carrier of a critical set associated to a given set-consensus value. Therefore if there are m distinct δ_Q outputs provided, each associated to a set-consensus value greater or equal to c, then one is associated to a set consensus power of at least $m + c - 1$. Thus $\alpha(\cup_{v \in \theta}(\mu_Q(v))) \geq m + c - 1$.

Let θ' be the greatest 2-contention simplex composed of the vertices v_i such that $i \leq l$ and such that $v_i \notin \mathcal{CS}_\alpha(\sigma)$. As they did not pass the test of μ_Q, it implies that they are not in $\mathcal{CSV}_\alpha(\theta')$. Therefore, following the definition of $\mathcal{R}_\alpha$ we have $dim(\theta') - 1 \leq Conc_\alpha$. The definition implies that $dim(\theta') - 1$ is smaller than the set consensus power of a critical set σ_c observed by some process of θ'. As it is observed by some process in θ', and thus in θ, it cannot intersect with Q as otherwise this vertex would have been associated with δ_Q. Therefore, it sees a critical set associated to a set consensus power strictly smaller than c. The construction of the set of processes excludes V, the processes returning through γ_Q a view of processes which returned δ_Q. Therefore, we can apply Lemma 14 and obtain that there is at most $dim(\theta') - 1$ distinct decided values by the process not in V. Thus at most $c - 1$.

Therefore there are at most $m + c - 1$ distinct decided values, and hence: $|\{\mu_Q(v) : v \in \theta\}| \leq m + c - 1 \leq \alpha(\cup_{v\in\theta}(\mu_Q(v)))$. □

Property 6. *[Robustness of μ_Q]* $\forall v \in \mathcal{R}_\alpha, dim(v) = 0, \forall Q \subseteq \Pi :$

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

Proof. This is a direct corollary of the definition of μ_Q and γ_Q, that a given vertex $v \in \mathcal{R}_\alpha$ does not need Q to compute the result but only the processes in Q it sees. □

Property 7. $\forall Q \subseteq \Pi, (\forall \sigma \in \mathcal{R}_\alpha : dim(\sigma) = n - 1), (\forall \theta \subseteq \sigma : \chi(\theta) \subseteq Q) : min_Q \circ \mu_Q(\sigma)$ *solves and* $\alpha(\cup_{v\in\theta}(\mu_Q(v)))$*-set consensus.*

Proof. Let us consider a vertex $v \in \mathcal{R}_\alpha$, associated to a process of Q, $\chi(v) \in Q$. According to Property 4, μ_Q returns to v a non-empty subset of processes of Q. Thus $min_Q \circ \mu_Q(v)$ is well defined and returns a process in Q. Therefore, the termination property of set consensus is satisfied for all processes of Q. Moreover, Property 4 also implies that $min_Q \circ \mu_Q(v)$ is a subset of $carrier(v, \mathbf{s})$. In combination with the fact that $min_Q \circ \mu_Q(v) \in Q$, this implies that the validity property of set consensus is satisfied for all processes of Q.

Now consider any simplex $\sigma \in \sigma \in \mathcal{R}_\alpha$ composed of processes of Q, $\chi(\sigma) \subseteq Q$. According to Property 5, it implies that the number of distinct set of processes returned by μ_Q is smaller than or equal to $\cup_{v\in\theta}(\mu_Q(v))$. Thus that the number of distinct values returned by $min_Q \circ \mu_Q(v)$ is smaller than or equal to $\cup_{v\in\theta}(\mu_Q(v))$. Therefore $min_Q \circ \mu_Q$ also satisfies the agreement property of $\cup_{v\in\theta}(\mu_Q(v))$-set consensus. □

Lemma 8. *If a process in A with a minimal view in some iteration of $\mathcal{R}_\alpha$ shares a proposal, then the described protocol solves α-adaptive set consensus.*

Proof. According to Property 7 $f_A \circ \mu_A$ returns a valid set consensus proposal. Assume that the protocol does not terminate for some processes in A. This implies that there is a process in A which is not participating yet. This contradicts the assumption that there is a process with the smallest view in A in some iteration which participates.

Consider the first round at which some process returns. As views are inclusion-closed and as the process terminating must have observed all the processes of A in its view participating, then every process saw some process in A participating. Therefore every process adopts a new proposal, which is $f_A \circ \mu_A(v)$ with v the process associated vertex. According to Property 7, we have for all subset Q of A associated to the simplex σ at most $\alpha(\cup_{v\in\sigma}(\mu_Q(v)))$ distinct returned values. But before returning a value, a process writes the input state of every process in its μ_Q output. Therefore the participating set is greater than $\cup_{v\in\sigma}\mu_Q(v))$ when $\alpha(\cup_{v\in\sigma}(\mu_Q(v)))$ distinct outputs are returned. Therefore the algorithm solves an α-adaptive set consensus. □

Lemma 9. *The read-write shared memory with access to set consensus simulation is lock-free.*

Proof. Assume that eventually the simulation only completes dummy writes. This means that there is a process which infinitely often completes itself dummy writes. Thus this process is infinitely often provided with the smallest view of $\mathcal{R}_\alpha$ without completing its pending set consensus operation. According to Lemma 8, every process must have eventually returned — a contradiction. □