An Asynchronous Computability Theorem for Fair Adversaries

Petr Kuznetsov, Thibault Rieutord, Yuan He

▶ **To cite this version:**

Petr Kuznetsov, Thibault Rieutord, Yuan He. An Asynchronous Computability Theorem for Fair Adversaries. 2017. hal-01572257v4

HAL Id: hal-01572257
https://hal.science/hal-01572257v4

Preprint submitted on 18 Apr 2020

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

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

An Asynchronous Computability Theorem for Fair 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

This paper proposes a simple topological characterization of a large class of *fair* adversarial 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. Fair adversaries include, but are not restricted to, the models of wait-freedom, t-resilience, and k-concurrency. Our results generalize and improve all previously derived topological characterizations of the ability of a model to solve distributed tasks.

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 *shared-memory* models in which a set of *crash-prone asynchronous* processes communicate via invoking operations on a collection of shared objects, which, by default, include atomic read-write registers.

Topology of wait-freedom. The *wait-free* model [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].

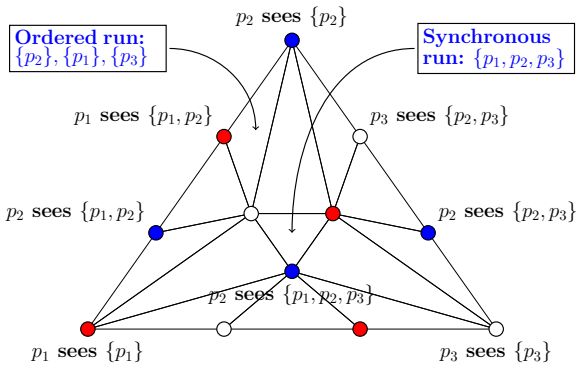
A task T has a wait-free solution using read-write registers if and only if there exists a simplicial, chromatic map from some *subdivision* of the *input simplicial complex*, describing the inputs of T, to the *output simplicial complex*, describing the outputs of T, respecting the task specification Δ. In particular, we can choose this subdivision to be a number of iterations of the *standard chromatic* subdivision (denoted Chr, Figure 1a).

Therefore, the celebrated *Asynchronous Computability Theorem (ACT)* [20] can be formulated as:

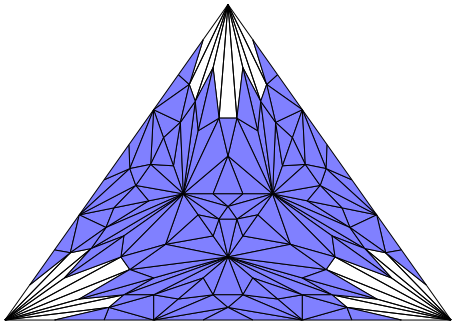
> A task $T = (\mathcal{I}, \mathcal{O}, \Delta)$, where $\mathcal{I}$ is the input complex, $\mathcal{O}$ is an output complex, and Δ is a carrier map from $\mathcal{I}$ to sub-complexes of $\mathcal{O}$, is wait-free solvable if and only if there exists a natural number ℓ and a simplicial map $\phi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

The output complex of the *immediate snapshot* (IS) task is precisely captured by Chr [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, captured by iterations of Chr. The ACT theorem can thus 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 thus allows us to solve any task in the wait-free model. Hence, from the task computability perspective, the *IS* task is a finite representation of the wait-free model.

*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) Chr(**s**), the standard chromatic subdivision of a 2-simplex, the output complex of the 3-process *IS* task.



(b) In blue, $\mathcal{R}_{1-res}$, the affine task of 1-resilience for a 3-process system.

Figure 1: Standard chromatic subdivision and affine task example.

Adversaries. Given that many fundamental tasks are not solvable in the wait-free way [3, 20, 29], more general models were considered. Delporte et al. [9] introduced the notion of an *adversary*, 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 a live set of $\mathcal{A}$. For example, the *t-resilient* n-process model is defined via an adversary $\mathcal{A}_{t-res}$ that consists of all process subsets of size $n-t$ or more. $\mathcal{A}_{t-res}$ is *superset-closed* [23], as it contains all supersets of its elements.

Saraph et al. [30] recently proposed a direct characterization of t-resilient task computability via a specific task $\mathcal{R}_{t-res}$. The task is defined as a restriction of the *double* immediate snapshot task: the output complex of the task is a sub-complex 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. Intuitively the output complex of $\mathcal{R}_{t-res}$ contains all of 2-round *IS* runs in which every process "sees" at least $n-t-1$ other processes. We call such tasks *affine* [12, 15], as the geometrical representation of their output complexes are unions of affine spaces. An affine task consists in solving a (generalized) simplex agreement [5, 20] on the corresponding sub-complex of $\text{Chr}^2\,\mathbf{s}$.

Figure 1b depicts the output complex of $\mathcal{R}_{1-res}$, the affine task for the 3-process 1-resilient model.

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

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. [12]. Note that such an adversary is *symmetric* [32], as it only depends on the *sizes* of live sets, and not on process identifiers. Unlike $\mathcal{A}_{t-res}$ (which is also symmetric), the k-obstruction-free one is not superset-closed.

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 a subset of the participating processes P cannot achieve better set consensus than P. Fair adversaries subsume, but are not restricted to, symmetric and superset-closed ones.

We define an affine task $\mathcal{R}_{\mathcal{A}}$ capturing the task computability of any fair (adversarial) $\mathcal{A}$-model. 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 [12, 15, 20, 30], as it applies to *all* fair adversaries (and not only t-resilient and k-obstruction-free) and *all* tasks (and not only *colorless*).

Figure 2 shows adversary classes and summarizes the results of this paper along with earlier affine characterizations.

We believe that the results can be extended to all "practical" restrictions of the wait-free model, beyond fair adversaries, which may result in a complete computability theory for distributed computing shared-memory models.

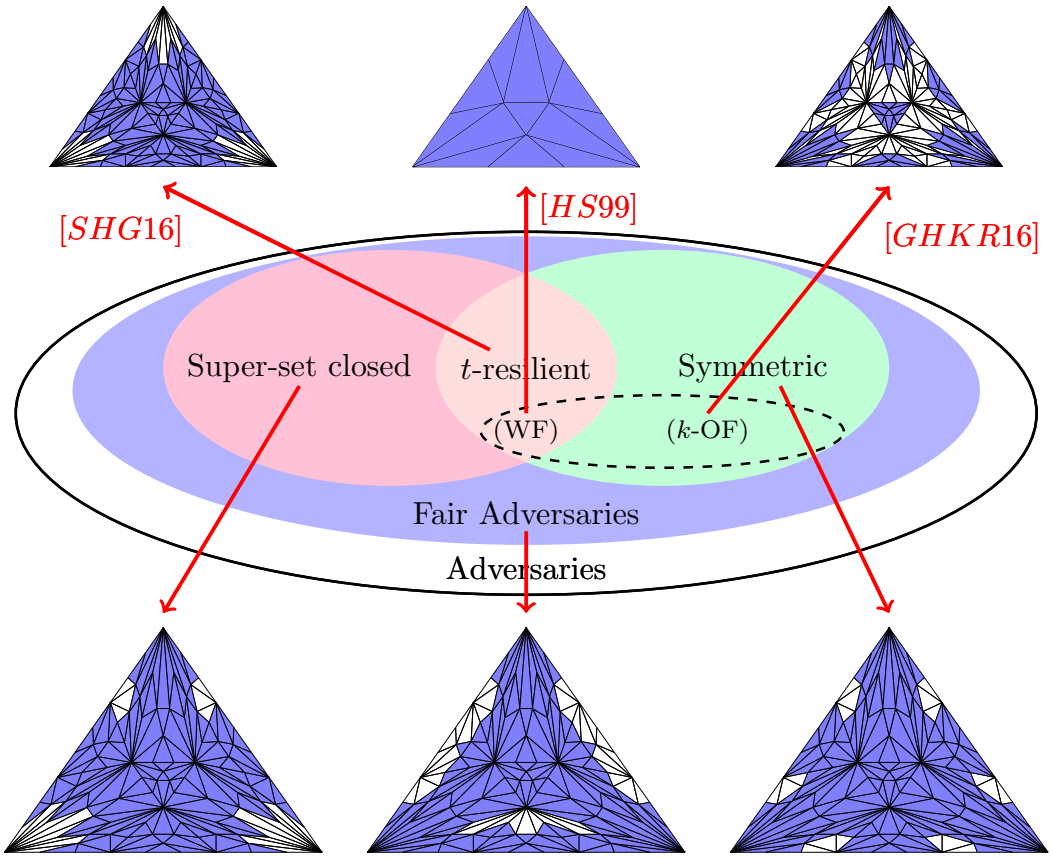


Figure 2: Earlier topological characterizations of the wait-free [20], t-resilient [30] and k-obstruction-free [12] models, and our contribution: affine tasks for all fair adversaries.

Compact models. Our affine-task formalism enables a *compact* representation of a distributed computing model. Intuitively, assuming the conventional "longest-prefix" metric [2], a model M, as a set of infinite runs, is compact if it contains its limit points: if every prefix of an infinite run *complies* with M (i.e., can be extended to a run in M), then the run is in M. M can then be viewed as a *safety property* [28, Chap. 8]: to check if a run is in M, it is sufficient to check whether each of its finite prefixes complies with M.

Most adversarial models are non-compact. For example, the 1-obstruction-free 2-process model is compliant with all finite runs, but among the infinite ones—only those in which exactly one process runs solo from some point on are in the model. Similarly, consider an infinite solo run in which exactly one process takes steps. All finite prefixes of this run complies with the 1-resilient 3-process model, but the run itself is not in the model.

In contrast, the affine model L^*, defined as the subset of infinite IIS runs resulting by iterating an affine task L is, by construction, compact. By a simple application of König's lemma, we can easily show that every task solvable in an affine model is solvable in a *bounded* number of IIS rounds, i.e., in finitely many finite runs. In a non-compact model, such as the model of 1-resilience, many tasks can only be solved in arbitrarily long runs, hence, to check if a solution is correct, we might have to consider infinitely many infinite runs. Thus, working in an equivalent affine model may be attractive from the verification viewpoint.

Roadmap. Section 2 describes our model. Section 3 recalls the definitions of adversarial models and presents agreement functions. Section 4 defines the affine task $\mathcal{R}_\mathcal{A}$ for a fair adversary $\mathcal{A}$. In Section 5, we show that $\mathcal{R}_\mathcal{A}^*$ can be simulated in the adversarial $\mathcal{A}$-model. In Section 6, we show that any task solvable in the $\mathcal{A}$-model can be solved in $\mathcal{R}_\mathcal{A}^*$. Section 7 reviews related work and Section 8 concludes the paper.

(a) Ordered run: $\{p_2\}, \{p_1\}, \{p_3\}$.

(b) Synchronous run: $\{p_1, p_2, p_3\}$.

Figure 3: Examples of valid sets of IS outputs.

2 Preliminaries

We assume a system of n asynchronous processes, $\Pi = \{p_1, \ldots, p_n\}$. Two models of communication are considered: (1) *atomic snapshots* [1] and (2) *iterated immediate snapshots* [5, 20].

Atomic snapshot models. The atomic-snapshot (AS) memory is represented as a vector of n shared variables, where each process p_i is associated with the 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 vector current state.

A *protocol* is a deterministic distributed automaton that, for each process and each its local state, stipulates which operation and state transition the process may perform. A *run* of a protocol is a possibly infinite sequence of alternating states and operations. An AS *model* is a set of infinite runs.

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 shares its initial state 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 every correct process is participating.

Iterated Immediate Snapshot model. In the iterated immediate snapshot (IIS) model, processes proceed through an infinite 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_{ir} and returns a set V_{ir} of submitted values (w.l.o.g, values of different processes are distinct), satisfying the following properties (See Figure 3 for IS examples):

- self-inclusion: $v_{ir} \in V_{ir}$;
- containment: $(V_{ir} \subseteq V_{jr}) \vee (V_{jr} \subseteq V_{ir})$;
- immediacy: $v_{ir} \in V_{jr} \Rightarrow V_{ir} \subseteq V_{jr}$.

In the IIS communication model, we assume that processes run the *full-information* protocol, in which, 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 M_r. There are no failures in the IIS model, all processes go through infinitely many *IS* instances.

Note that the wait-free AS model and the IIS model are equivalent as regards task solvability [4, 18].

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: $\forall \rho, \sigma \in \mathcal{I}$, $\rho \subseteq \sigma$: $\Delta(\rho) \subseteq \Delta(\sigma)$. 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 *k-set consensus* task [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 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 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. Hence, in the IIS model, all processes must decide.

Simplicial complexes. We use the standard language of *simplicial complexes* [18, 31] 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 σ is $|\sigma| - 1$, and any subset of σ is one of its *faces*. We denote by $\mathbf{s}$ the *standard* $(n-1)$*-simplex*: a fixed set of n vertices and all its subsets.

Given a complex K and a simplex $\sigma \in K$, σ is a *facet* of K, denoted $facet(\sigma, K)$, if σ is not a face of any stricly larger simplex in K. Let $facets(K) = \{\sigma \in K, facet(\sigma, K)\}$. A simplicial complex is *pure* (of dimension n) if all its facets have 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 $\mathbf{s}$, in one-to-one correspondence with n *colors*. In our setting, colors correspond to processes identifiers.

The *closure* of a set of simplices S, $Cl(S)$, is the complex formed by all faces of simplices in S, i.e., $\bigcup_{\sigma \in S} faces(\sigma)$. Given a complex K, the *star* of $S \subseteq K$ in K, $St(S, K)$, is the set of all simplices in K having a simplex from S as a face, i.e., $\{\sigma \in K | faces(\sigma) \cap S \neq \emptyset\}$. Given a pure complex K, we also define a new construct that we call the *pure complement* of $S \subseteq K$ in K, $Pc(S, K)$. It is the maximal pure sub-complex of K of the same dimension as K which does not intersect with S, i.e., $Cl(\{\sigma \in facets(K) | faces(\sigma) \cap S = \emptyset\})$.

Standard chromatic subdivision and IIS The *standard chromatic subdivision* [20] of a complex K, $\mathrm{Chr}\, K$ ($\mathrm{Chr}\, \mathbf{s}$ is depicted in Figure 1a), is a complex where vertices of $\mathrm{Chr}\, K$ are couples (c, σ), where c is a color and σ is a face of K containing a vertex of color c. Simplices of $\mathrm{Chr}\, K$ are the sets of vertices (c_1, σ_1), ..., (c_m, σ_m) associated with distinct colors (i.e., $\forall i, j$, $c_i \neq c_j$) such that the σ_i satisfies the containment and immediacy properties of *IS*. It has been shown that Chr is a *subdivision* [22]. In particular, the geometric realization of $\mathrm{Chr}\, \mathbf{s}$, $|\mathrm{Chr}\, \mathbf{s}|$, is homeomorphic to $|\mathbf{s}|$, the geometric realization of $\mathbf{s}$ (i.e., the convex hull of its vertices). If we *iterate* this subdivision m times, each time applying Chr to all simplices, we obtain the m^{th} chromatic subdivision, Chr^m. $\mathrm{Chr}^m\, \mathbf{s}$ captures the m-round IIS model, IS^m [5, 20].

Given a complex K and a subdivision of it $Sub(K)$, the carrier of a simplex $\sigma \in Sub(K)$ in K, $carrier(\sigma, K)$, is the smallest simplex $\rho \in K$ such that the geometric realization of σ, $|\sigma|$, is contained in $|\rho|$: $|\sigma| \subseteq |\rho|$. The carrier of a vertex $(p, \sigma) \in \mathrm{Chr}\, \mathbf{s}$ is σ. In the matching *IS* task, the carrier corresponds to the snapshot returned by p, i.e., the set of processes *seen* by p. The carrier of a simplex $\rho \in \mathrm{Chr}\, K$ is simply the union (or, due to inclusion, the maximum) of the carriers of vertices in ρ. Given a simplex $\sigma \in \mathrm{Chr}^2\, \mathbf{s}$, $carrier(\sigma, \mathbf{s})$ is equal to $carrier(carrier(\sigma, \mathrm{Chr}\, \mathbf{s}), \mathbf{s})$. $carrier(\sigma, \mathrm{Chr}\, \mathbf{s})$ corresponds to the set of all snapshots seen by processes in $\chi(\sigma)$. Hence, $carrier(\sigma, \mathbf{s})$ corresponds to the union of all these snapshots. Intuitively, it results in the set of all processes *seen* by processes in $\chi(\sigma)$ through the two successive immediate snapshots instances.

Additionnal details can be found in Appendix A.

Simplex agreement and affine tasks. In the *simplex agreement task*, processes start on vertices of some complex K, forming a simplex $\sigma \in K$, and must output vertices of some subdivision of K, $Sub(K)$, so that outputs form a simplex ρ of $Sub(K)$ respecting carrier inclusion, i.e., $carrier(\rho, K) \subseteq \sigma$. In the simplex agreement tasks considered in the characterization of wait-free task computability [5, 20], K is the standard simplex $\mathbf{s}$ and the subdivision is usually iterations of Chr.

An *affine task* is a generalization of the simplex agreement task, where $\mathbf{s}$ is fixed as the input complex and where the output complex is a pure non-empty sub-complex of some iteration of the standard chromatic subdivision, $\mathrm{Chr}^\ell\, \mathbf{s}$. Formally, let L be a pure non-empty sub-complex of $\mathrm{Chr}^\ell\, \mathbf{s}$ for some $\ell \in \mathbb{N}$. The affine task associated with L is then defined as $(\mathbf{s}, L, \Delta)$, where, for every face $\sigma \subseteq \mathbf{s}$, $\Delta(\sigma) = L \cap \mathrm{Chr}^\ell(\sigma)$. Note that $L \cap \mathrm{Chr}^\ell(\mathbf{t})$ can be empty, in which case the set of participating processes must increase before processes may produce outputs. Note that, since an affine task is characterized by its output complex, with a slight abuse of notation, we use L for both the affine task $(\mathbf{s}, L, \Delta)$ and its ouput complex.

By running m iterations of this task, we obtain L^m, a sub-complex of $\mathrm{Chr}^{\ell m}\, \mathbf{s}$, corresponding to a subset of $IS^{\ell m}$ runs (each of the m iterations includes ℓ *IS* rounds). The affine model associated with L, denoted L^*, corresponds to 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 $IS^{\ell m}$ runs associated with L^m. Note that the definition of the affine model L^* is done by satisfying a property on all its prefixes. Hence, affine models are, by construction, compact.

3 Adversaries and agreement functions

In this section, we introduce many results from [24] wich will be instrumental for our topological characterization.

Adversaries. It is convenient to model patterns in which process failures can occur using the formalism of *adversaries* [9]. An adversary $\mathcal{A}$ is defined as a set of possible correct process subsets, called *live sets*. An infinite 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* $\mathcal{A}$-model is thus defined as the set of $\mathcal{A}$-compliant runs.

An adversary is *superset-closed* [23] if each superset of a live set of $\mathcal{A}$ is also an element of $\mathcal{A}$, i.e., if $\forall S \in \mathcal{A}, \forall S' \subseteq \Pi, S \subseteq S' \implies S' \in \mathcal{A}$. Superset-closed adversaries provide a non-uniform generalization of the classical *t-resilient* adversaries. An adversary $\mathcal{A}$ is *symmetric* if it only depends on the *sizes* of live sets, and not on process identifiers: $\forall S \in \mathcal{A}, \forall S' \subseteq \Pi, |S'| = |S| \implies S' \in \mathcal{A}$. Introduced as symmetric progress conditions in [32], symmetric adversaries provide a generalization of t-resilience and k-obstruction-freedom.

The *agreement power* of a model, i.e., the smallest k such that k-set consensus is solvable, was determined for adversaries in [13] in order to characterize the power of adversaries in solving *colorless* tasks [4]. It is formalized as follows:

Definition 1. *[Agreement power]*

$$setcon(\mathcal{A}) = \begin{cases} 0 & \text{if } \mathcal{A} = \emptyset \\ \max\limits_{S \in \mathcal{A}}(\min\limits_{a \in S}(setcon(\mathcal{A}|_{S \setminus \{a\}}) + 1)) & \text{otherwise} \end{cases}$$

With $\mathcal{A}|_P$ the adversary composed of the live sets of $\mathcal{A}$ included in P. As previously shown in [14], for a superset-closed adversary $\mathcal{A}$, the agreement power of $\mathcal{A}$ is equal to $csize(\mathcal{A})$, where $csize(\mathcal{A})$ is the size of the minimal hitting set of $\mathcal{A}$, i.e., a set intersecting with each $L \in \mathcal{A}$. For a symmetric adversary $\mathcal{A}$, the agreement power formula reduces to $setcon(\mathcal{A}) = |\{k \in \{1, \ldots, n\} : \exists S \in \mathcal{A}, |S| = k\}|$.

Agreement functions. Consider an AS model M and a function α mapping subsets of Π to integers in $\{0, \ldots, n\}$. We say that α is the *agreement function* of M, if for each $P \in 2^\Pi$, $\alpha(P)$ is the agreement powe of the model $M|_P$ consisting of runs of M in which no process in $\Pi \setminus P$ participates [24]. Intuitively, $\alpha(P)$ is the best level of set consensus that can be solved adaptively in M. By convention, if $M|_P$ does not contain any run, then $\alpha(P)$ is equal to 0.

Let $P \subseteq P' \subseteq \Pi$. We can observe that, by construction, the agreement function of a model is *monotonic*, i.e., $\alpha(P) \leq \alpha(P')$ and of *bounded growth*, i.e., $\alpha(P') \leq \alpha(P) + |P' \setminus P|$. It was shown in [24] that the agreement function of $\mathcal{A}$ can be defined using the *setcon* function: $\alpha(P) = setcon(\mathcal{A}|_P)$.

Fair adversaries. Informally, an adversary is *fair* [24] if a subset Q of participating processes P cannot achieve a better set consensus than P. For an adversary $\mathcal{A}$, and $Q \subseteq P \subseteq \Pi$, we define $\mathcal{A}|_{P,Q} = \{S \in \mathcal{A} : (S \subseteq P) \wedge (S \cap Q \neq \emptyset)\}$. When solving a task, only correct processes must output. Thus, for a task restricted to processes in Q, no process has to output in a run corresponding to a live set $L \in \mathcal{A}$ with $L \cap Q = \emptyset$.

Definition 2. *[Fairness] 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], as some others. Unfortunately, not all adversaries are fair.

The α-model. Generalizing the k-active-resilient model, the α-model was introduced to capture agreement functions ability to characterize the task computability of (some) models.

Definition 3. *[α-model] The* α-model *is the AS model in which: if P is the participating set, then $\alpha(P) \geq 1$ and at most $\alpha(P) - 1$ processes in P are faulty.*

Intuitively, the α-model is the weakest model with α as its agreement function. This allows us to use agreement functions to characterize models which are as weak as the corresponding α-model. 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.

Theorem 1. *[24] For any fair adversary $\mathcal{A}$, a task is solvable in the $\mathcal{A}$-model if and only if it is solvable in the α-model.*

α-adaptive set consensus. The abstraction of *α-adaptive set consensus* [24] can be accessed with a single *propose*(v) operation. It 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 returned values does not exceed $\alpha(P)$, with P the current participating set. This abstraction allows us to define yet another family of models, equivalent with α-model, and hence, with adversarial $\mathcal{A}$-models.

Definition 4. *[α-set consensus model] The* α-set consensus *model is the AS model in which, if P is the participating set then $\alpha(P) \geq 1$, and processes have access to α-adaptive set-consensus objects.*

Theorem 2. *[24] A task is solvable in the α-model if and only if it is solvable in the α-set-consensus model.*

Hence, for a fair adversary $\mathcal{A}$ and its agreement function α the $\mathcal{A}$-model, the α-set-consensus model and the α-model can all be used interchangeably for task solvability issues.

4 Defining the affine task for a fair adversary

Given a fair adversary $\mathcal{A}$ and its agreement function α, we define the affine task $\mathcal{R}_{\mathcal{A}}$, a sub-complex of $\mathrm{Chr}^2\,\mathbf{s}$, which will be shown to capture the task computability of the $\mathcal{A}$-model.

Agreement and contention simplices. For a vertex $v \in \mathrm{Chr}^2\,\mathbf{s}$, let $\mathit{View}^1(v)$ and $\mathit{View}^2(v)$ be the sets of processes seen by the process $\chi(v)$ in, respectively, the first and the second *IS* (we call these View^1 and View^2). Formally, $\mathit{View}^2(v) = \mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s})$ and $\mathit{View}^1(v) = \mathit{carrier}(v', \mathbf{s})$ with $v' \in \mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s})$ such that $\chi(v) = \chi(v')$.

The idea behind the definition of these prohibited simplices is simple. In an execution, processes can only decide on the proposal they observed. Therefore, in an execution, if a process p sees only itself, other processes should return p's proposal to hope reaching an agreement with p. In $\mathrm{Chr}^2\,\mathbf{s}$, if p is executed alone, then it has the smallest View^1 and View^2. Thus all processes would observe p's View^1. Therefore, a natural way to try to reach an agreement among processes is to adopt the proposal from the process observed with the smallest View^1. Moreover, as processes may share the same view, it is even better to deterministically select a value from the smallest View^1 itself.

We formalize the intuitive description of contention simplices as follows: In a simplex $\delta \in \mathrm{Chr}^2\,\mathbf{s}$, we say that vertices v and v' are *contending* if their View^1 and View^2 are ordered in the opposite way: $\mathit{View}^1(v)$ is a proper subset of $\mathit{View}^1(v')$ and $\mathit{View}^2(v')$ is a proper subset of $\mathit{View}^2(v)$, or vice versa. If every two vertices of δ are contending, then we say that δ is a 2-*contention* simplex. Let Cont_2 be the set of 2-contention simplices, formally:

Definition 5. *[Cont_2]* $\sigma \in \mathrm{Chr}^2\,\mathbf{s} : \forall v, v' \in \sigma, v \neq v' :$

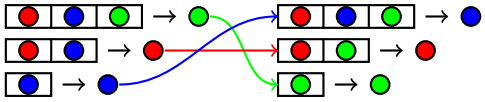
$$((\mathit{View}^1(v) \subsetneq \mathit{View}^1(v')) \wedge (\mathit{View}^2(v') \subsetneq \mathit{View}^2(v))) \vee$$

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

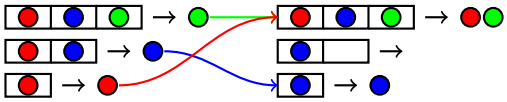
Cont_2 is inclusion-closed: any face of a 2-contention simplex is also in Cont_2. Thus, Cont_2 is a complex: the 2-*contention complex* (depicted for a 3-processes system in Figure 4c). Particular executions of two IS rounds are also represented in Figures 4a and 4b. In these executions, one can see that a couple of processes is contending if the execution "order" is strictly reversed in the two IS runs.

We first show how to restrict $\mathrm{Chr}^2\,\mathbf{s}$ to obtain an affine task $\mathcal{R}_{k-OF}$, solvable in the k-obstruction-free model, and which allows, in $\mathcal{R}^*_{k-OF}$, any set of processes to solve k-set consensus among themselves. As in [12], the idea consists in specifying prohibited simplices and take their pure complement as the affine task.

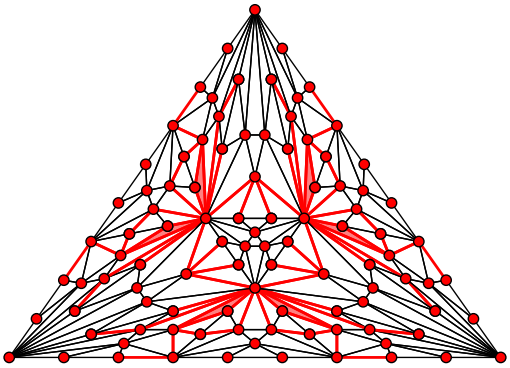
Intuitively, a contention simplex of size k is one in which, in the corresponding run, all of the k processes have distinct View^1 and each one believes it had the smallest one among them. Thus, an execution for which all processes would return distinct proposals. Hence, $\mathcal{R}_{k-OF}$ is defined by prohibiting too large contending simplices:



(a) Two reversed IS ordered runs: $\{p_2\},\{p_1\},\{p_3\}$ and $\{p_3\},\{p_1\},\{p_2\}$. Any set of processes is contending due to inverted execution orders.



(b) Two ordered runs mixed orders: $\{p_1,p_2,p_3\}$ and $\{p_2\},\{p_3,p_1\}$. The only couple of contending processes is $\{p_1,p_2\}$.



(c) The 2-contention complex shown in red.

Figure 4: Representation, in a 3-processes system, of all 2-contention simplices in $\mathrm{Chr}^2\,\mathbf{s}$ and some detailed IS runs.

Definition 6. *[Affine task $\mathcal{R}_{k-OF}$]*

$$\mathcal{R}_{k-OF} = Pc(\{\sigma \in Cont_2 | dim(\sigma) \geq k\}, \mathrm{Chr}^2\,\mathbf{s}).$$

See Figure 7a for $\mathcal{R}_{1-OF}$ in a 3-process system. To see that $\mathcal{R}_{k-OF}$ indeed captures the k-obstruction-free adversary, one can check, which is not obvious, that the latter definition of $\mathcal{R}_{\mathcal{A}}$ reduces to $\mathcal{R}_{k-OF}$ when $\mathcal{A}$ is the k-obstruction-free adversary, or, alternatively, rely on the proofs from [12].

Agreement vs. participation. Solving a desired level of agreement is no longer sufficient. 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: $carrier(v,\mathbf{s})$ is the participating set witnessed by process $\chi(v)$.

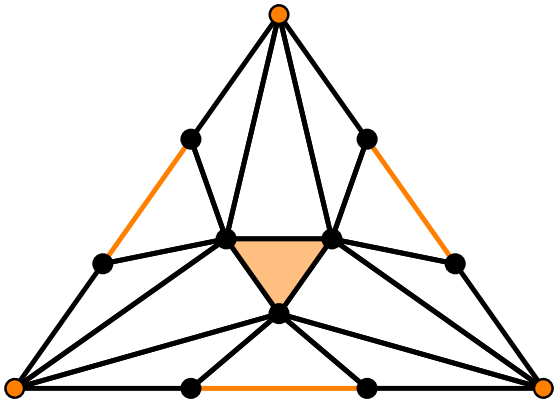
The naive approach would consists in varying the restriction on the size of contention simplices according to the carrier size. Such a restriction would indeed provide an affine task which is strong enough to solve the desired level of agreement, but it would be impossible to solve. Indeed, contention assumes that processes with the smallest $View^1$ go first. But when the agreement power is equal to 0, processes must be ensured to obtain larger views and hence to let processes with larger $View^1$ go first. But letting processes with large $View^1$ go first inherently creates contention.

The idea of the solution consists in switching between resilience and concurrency requirements. Indeed, as long as the agreement power is steady over the participation, we rely on restrictions made by limiting contention. But when the agreement power increases due to an increase of participation, we identify a "witness" of this new agreement power and require it to go first and be seen by other processes. This corresponds to changing the selection of the smallest $View^1$ by looking first on $View^1$ "witnessing" a new agreement level and otherwise, by default, selecting the smallest $View^1$. These "witnesses" of participation is what we call *critical simplices*.

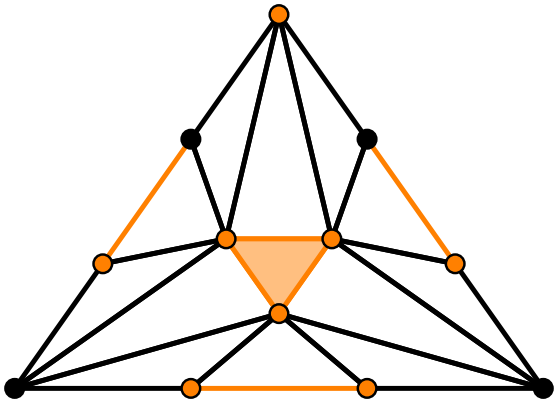
Critical simplices. The goal here is to identify for each increase of participation a new $View^1$ witnessing it. An easy requirement is that this $View^1$ should correspond to a participation level associated with the new level of agreement power. But two issues must be solved: (1) the provided $View^1$ may be irregular and there could be none for a given agreement power; and (2) distinct $View^1$ may share the same level of agreement power and the smallest one may be different depending on the executions.

The idea is to select $View^1$ which are minimal in the given execution for some level of agreement power. To do so, the value of $View^1$ is not sufficient on its own. But if we know that multiple processes all share the same $View^1$, we can deduce that all other processes with a strictly smaller view must have a $View^1$ corresponding to a lower level of agreement power. This solves the second issue, but indirectly also the first one. Indeed, if no $View^1$ exists for an agreement level, it implies that the smallest view for the next level is provided to sufficiently many processes to be able to deduce that no process with a smaller $View^1$ may obtain a $View^1$ corresponding to the "missing" level, hence this $View^1$ is a witness of both agreement levels.

A *critical set* or *critical simplex* is set of processes sharing the same $View^1$ which is sufficiently large to ensure that their $View^1$ is the smallest one for some level of agreement power. Formally, a simplex

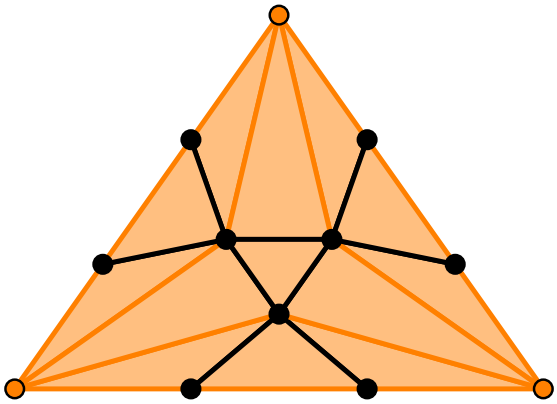


(a) Critical simplices for the α-model with $\alpha(P) = min(|P|, 1)$ (1-obstruction-freedom)

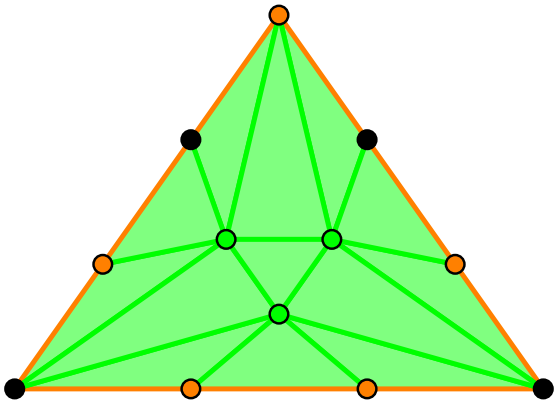


(b) Critical simplices for the adversary defined as $\{p_2\}$, $\{p_1, p_3\}$ plus all supersets.

Figure 5: Critical simplices are displayed in orange (with p_2 the top vertex, p_1 the bottom left vertex and p_3 the bottom right vertex).



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



(b) For the adversary defined as $\{p_2\}$, $\{p_1, p_3\}$ plus all supersets.

Figure 6: Simplices in black, orange and green are mapped to concurrency levels of 0, 1 and 2 respectively (with p_2 the top vertex, p_1 the bottom left vertex and p_3 the bottom right vertex).

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

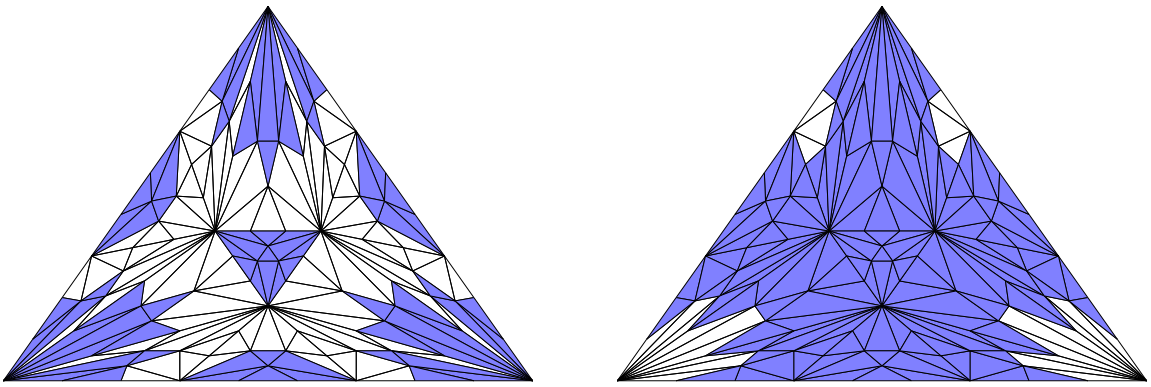
Definition 7. $\forall \sigma \in \text{Chr}\,\mathbf{s}$, $Critical_\alpha(\sigma) \equiv$

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

Examples of critical simplices for two 3-processes fair models are depicted in Figure 5. The critical simplices are displayed in orange. As it can be observed, the set of critical simplices is not inclusion-closed, hence it does not define a simplicial complex.

Given a simplex $\sigma \in \text{Chr}\,\mathbf{s}$, we denote as $\mathcal{CS}_\alpha(\sigma)$ the set of critical simplices in σ, that is $\mathcal{CS}_\alpha(\sigma) = \{\sigma' \subseteq \sigma : Critical_\alpha(\sigma')\}$. Moreover, identifying the set of processes which compose some critical simplex will be useful. Thus, let $\mathcal{CSM}_\alpha(\sigma)$ (critical simplices members) be the set of vertices of some $\sigma \in \text{Chr}\,\mathbf{s}$ which belongs to some critical simplex in σ, formally $\mathcal{CSM}_\alpha(\sigma) = \{\sigma' \in Cl(\mathcal{CS}_\alpha(\sigma)) : \dim(\sigma') = 0\}$. Note that critical simplices members can be seen also as a sub-complex of $\text{Chr}\,\mathbf{s}$. Intuitively, processes with the smallest $View^2$ should belong to this set. Similarly we also define the notion of the critical simplex view, $\mathcal{CSV}_\alpha(\sigma)$, which corresponds to the set of processes observed by a critical simplex in its $View^1$. It can be simply obtained by taking the carrier in $\mathbf{s}$ of a critical simplex, that is $\mathcal{CSV}_\alpha(\sigma) = carrier(\mathcal{CSM}_\alpha(\sigma), \mathbf{s})$.

Concurrency level. Critical simplices provide a mechanism to select particular $View^1$. This can be used to solve agreement protocols with the desired k-set consensus for an observed participation. But



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

(b) Affine task for the adversary defined as $\{p_2\}$, $\{p_1, p_3\}$ plus all supersets.

Figure 7: Some examples of affine tasks $\mathcal{R}_\mathcal{A}$ in blue (with p_2 the top vertex, p_1 the bottom left vertex and p_3 the bottom right vertex).

unfortunately this works only when the set of processes trying to solve a set-consensus operation was observed by the critical simplices, i.e., when processes belong to $\mathcal{CSV}_\alpha(\sigma)$. When this is not the case, processes should be able to solve set-consensus operations by themselves. This is where the limition on the size of contention simplices will come in. But this limitation should still be made according to the observed participation. This is done according to the agreement power associated with the observed critical simplices. We define this restriction using the following notion of *concurrency level*:

Definition 8. *[Concurrency map]* $\forall \sigma \in \mathrm{Chr}\,\mathbf{s}$:

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

Note that we add 0 to the set of agreement powers in case this set is empty. The concurrency map is displayed in Figure 6 for examples of 3-processes models. Each simplex of $\mathrm{Chr}\,\mathbf{s}$ is associated with a concurrency level. One can observe that the set of simplices with a concurrency level equal to k corresponds to the simplices in the *star* of the critical simplices associated with an agreement power equal to k and which are not in the *star* of a critical simplex associated with a greater agreement power.

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

Definition 9. *[$\mathcal{R}_\mathcal{A}$]* $\mathcal{R}_\mathcal{A} = Cl(\{\sigma \in facets(\mathrm{Chr}^2\,\mathbf{s}) : \forall \theta \subseteq (\sigma), P(\theta, \sigma)\}$ *with P such that (with* $\tau = carrier(\theta, \mathrm{Chr}\,\mathbf{s})$ *and* $\rho = carrier(\sigma, \mathrm{Chr}\,\mathbf{s})$*):*

$$P(\theta, \sigma) \equiv \ \theta \in Cont_2 \wedge (\chi(\theta) \cap \chi(\mathcal{CSM}_\alpha(\rho)) \cap \chi(\mathcal{CSV}_\alpha(\tau))) = \emptyset \implies dim(\theta) < Conc_\alpha(\tau).$$

Intuitively, a simplex $\sigma \in \mathrm{Chr}^2\,\mathbf{s}$ is in $\mathcal{R}_\mathcal{A}$ if and only 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 its own.

Examples of affine tasks for 3-processes α-models are depicted in Figure 7.

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

To show that any task T solvable in $\mathcal{R}^*_\mathcal{A}$ is solvable in a fair $\mathcal{A}$-model, we present an algorithm solving $\mathcal{R}_\mathcal{A}$ in the α-model. By iterating this task, we obtain $\mathcal{R}^*_\mathcal{A}$ and can solve T.

5.1 Algorithm Description

In our solution of $\mathcal{R}_\mathcal{A}$, presented in Algorithm 1, every process accesses two immediate snapshot objects: *FirstIS* to which it proposes its initial state, and *SecondIS* to which it proposes the outcome of *FirstIS*. Recall that outcomes of *SecondIS* form a simplex in $\mathrm{Chr}^2\,\mathbf{s}$ [22]. To ensure that simplices are in $\mathcal{R}_\mathcal{A}$, after finishing *FirstIS*, processes wait for their turns to proceed to *SecondIS*.

In this *waiting phase* (Lines 5–9), processes check a specific condition on the *IS* outcomes that they share with each others in registers $IS1[1,\ldots,n]$ and $IS2[1,\ldots,n]$. Each process p_i periodically checks whether either (1) it belongs to a critical simplex by using the formula at Line 7, or (2) if the number, computed at Line 8, of non-terminated processes ($IS2[j] = \emptyset$) which may have a smaller *FirstIS* output ($j \in IS1[i]$ and $IS1[j] \neq IS1[i]$) is smaller that some "level of concurrency". This level of concurrency is computed at Line 9 as the maximum between (1) the agreement power associated with the $View^1$ of the process itself ($\alpha(IS1[i])$) or (2) with the concurrency levels shared using the *Conc* registers by "terminated" critical simplices, i.e., a critical simplex with all its processes provided with *secondIS* outputs (Line 12).

Algorithm 1: Resolution of $R_{\mathcal{A}}$ in the α-model for process p_i.

```
1  Immediate Snapshot Objects: FirstIS, SecondIS;
2  Shared Registers: Conc[1],...,Conc[n] ∈ {0,...,n}, initially 0;
3  IS1[1],...,IS1[n] ∈ 2^Π, initially ∅ and IS2[1],...,IS2[n] ∈ 2^(2^Π), initially ∅;

4  R_A(input_i):
5  |  IS1[i] ← FirstIS(input_i);
6  |  wait until crit ∨ (rank < conc) with
7  |  crit = (α(IS1[i]) > α(IS1[i] \ {p_j ∈ Π : IS1[j] = IS1[i]}))
8  |  and rank = |{p_j ∈ IS1[i] : IS2[j] = ∅ ∧ IS1[j] ≠ IS1[i]}|
9  |  and conc = max(α(IS1[i]), max_{j∈{1,...,n}}(Conc[j]));
   |
10 |  IS2[i] ← SecondIS(IS1[i]);
11 |  if (α(IS1[i]) > α(IS1[i] \ {p_j ∈ Π : (IS1[j] = IS1[i]) ∧ (IS2[j] ≠ ∅)})) then
12 |  |  Conc[i] ← α(IS1[i]);
13 |  return(IS2[i]);
14 End R_A;
```

Intuitively, the waiting phase is used to ensure that *critical processes*, i.e., members of critical simplices, are prioritized to proceed with *SecondIS* over non-critical ones. A process may proceed to its *SecondIS* as soon as it knows that it belongs to some *critical* simplex ($crit = true$). A non-critical process is allowed to exit its waiting phase only when the number of potentially contending processes is smaller than the computed concurrency level ($rank < conc$). The proof relies mostly on showing that there are enough critical simplices to prevent non-critical processes from being blocked in the waiting phase.

5.2 Proof Sketch

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 participation 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.

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

5.3 Distribution of Critical Simplices

Let us first look at the case in which no participating process fails before updating its *IS1* output to the memory. In this case, the set of *IS1* views forms a simplex $\sigma \in \mathrm{Chr}\,\mathbf{s}$ such that $\chi(\sigma) = \chi(carrier(\sigma, \mathbf{s}))$: 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]:* $\forall \sigma \in \mathrm{Chr}\,\mathbf{s}, \forall l \in \mathbb{N}$:

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

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 $csize(\mathcal{Q}) \geq 0$. Now consider a simplex $\sigma \in \mathrm{Chr}\,\mathbf{s}$ such that $\chi(\sigma) = \chi(carrier(\sigma))$ 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:

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

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

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

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:

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

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 $carrier(\tau', \mathbf{s}) = carrier(\sigma, \mathbf{s})$ 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 $\alpha(P) \geq \alpha(P \setminus Q) \geq \alpha(P) - |Q|$. Note that this property was shown to be true for any fair model in [24] (see Section 3). 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 obtain that $\alpha(\chi(\sigma) \setminus \chi(\tau')) < \alpha(\chi(\sigma))$, and hence, that $Critical_\alpha(\tau')$. Since by construction $\beta = \sigma \setminus \tau$, we have the following inequality: $csize(\mathcal{CS}_\alpha(\sigma)) \geq csize(\mathcal{CS}_\alpha(\tau)) + csize(\mathcal{CS}_\alpha(\beta))$. Moreover, as $\alpha(\chi(\sigma)) \geq l$, we obtain:

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

   But as any subset τ' of τ, such that $|\tau'| > |\tau| - m$, is critical, 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 proportionally 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:

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

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. □

5.4 Algorithm Liveness

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 in any α-model.*

Proof. Let P be the participating set and let us assume that there is a correct process which never terminates. Let p be the correct processes which does not terminate with the smallest *IS1* view, let $v \in \mathrm{Chr}\,\mathbf{s}$ be the vertex corresponding to its *IS1* view, and let $\sigma \in \mathrm{Chr}\,\mathbf{s}$ be the simplex corresponding to the set of IS^1outputs when *IS1* has been updated for the last time.

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

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

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

Moreover, any process in $P \setminus \chi(carrier(\sigma, s))$ must have failed. Thus, in $\chi(carrier(\sigma, s))$ at most $\alpha(P) - 1 - (|P \setminus \chi(carrier(\sigma, s))|)$ 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 $Q = P \setminus \chi(carrier(\sigma, s))$, that at most $\alpha(\chi(carrier(\sigma, s))) - 1$ processes in $\chi(carrier(\sigma, s))$ may fail.

Let $m_1 = |\chi(carrier(\sigma, \mathbf{s})) \setminus (\chi(\sigma) \cup \chi(carrier(v, \mathbf{s})))|$, be the number of processes from $\chi(carrier(\sigma, \mathbf{s}))$ 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 $\alpha(\chi(carrier(v, \mathbf{s})))$, which fail after updating their *IS1* but before updating their *IS2*.

Let us now assume that $\alpha(\chi(carrier(\sigma,s))) - \alpha(\chi(carrier(v,s))) > m_1 + m_2$, then by selecting $l = \alpha(\chi(carrier(\sigma,s))) - m_2 - m_1$, we have $l > \alpha(\chi(carrier(v,s))))$, and hence, we obtain that:

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

If no critical simplex in $\{\theta \in \mathcal{CS}_\alpha(\sigma), \alpha(\chi(carrier(\theta,\mathbf{s}))) \geq \alpha(\chi(carrier(\sigma,s))) - m_2 - m_1\}$ 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 $\alpha(\chi(carrier(\sigma,s))) - m_2 - m_1$ terminates its *IS2*. Therefore eventually some process updates its *Conc* register (on line 12) to at least $\alpha(\chi(carrier(\sigma,s))) - m_2 - m_1$.

Now let us look back at p. It fails to terminate and thus never succeeds to pass the test on line 6. Therefore we have that the number of processes seen by p which do not terminate and do not have the same *IS1* view as p are strictly more than the value of $\max(\alpha(IS1[i]), \max_{j\in\{1,\ldots,n\}}(Conc[j]))$, with $IS1[i]$ equal to $\chi(carrier(v,s))$. As p is the correct process with the smallest *IS1* view which does not terminate, it implies that there are strictly more than $\max(\alpha(\chi(carrier(v,s))), \max_{j\in\{1,\ldots,n\}}(Conc[j]))$ failed processes with an *IS1* view strictly smaller than p. These failed processes are neither accounted in m_1 nor in m_2. Therefore, as at most $\alpha(\chi(carrier(\sigma,s))) - 1$ processes in $\chi(carrier(\sigma,s))$ may fail, there are at most $\alpha(\chi(carrier(\sigma,s))) - 1 - m_1 - m_2$ such processes which may fail. Thus $\alpha(\chi(carrier(\sigma,s))) - m_1 - m_2 - 1 \geq \max(\alpha(\chi(carrier(v,s))), \max_{j\in\{1,\ldots,n\}}(Conc[j]))$.

Two cases may arise:

- If $\alpha(\chi(carrier(\sigma,s))) - \alpha(\chi(carrier(v,s))) > m_1 + m_2$, then some process sets its *Conc* register to a value greater than or equal to $\alpha(\chi(carrier(\sigma,s))) - m_2 - m_1$ — A contradiction.
- Otherwise, $\alpha(\chi(carrier(\sigma,s))) - m_1 - m_2 - 1 \geq \alpha(\chi(carrier(v,s)))$ and so, we obtain a contradiction with the fact that $\alpha(\chi(carrier(\sigma,s))) - \alpha(\chi(carrier(v,s))) \leq m_1 + m_2$.

□

5.5 Algorithm Safety

Showing the safety of Algorithm 1 bears some similarities with the liveness proof. In particular, it relies on the same Lemma 3 on the distribution of critical simplices.

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 execute two rounds of an immediate snapshot protocol. Therefore the set of outputs forms a simplex in $\sigma \in \mathrm{Chr}^2\,\mathbf{s}$. Without loss of generality, we can assume that no process fails and thus that $dim(\sigma) = n - 1$. Indeed, if $\sigma \notin \mathcal{R}_\mathcal{A}$, then if failed processes were just slow and resumed their execution and terminate, it would produce $\sigma' \notin \mathcal{R}_\mathcal{A}$. Let us assume by contradiction that $\sigma \notin \mathcal{R}_\mathcal{A}$, this implies that there exists $\theta \subseteq \sigma$ such that (for $\tau = carrier(\theta, \mathrm{Chr}\,\mathbf{s})$ and $\rho = carrier(\sigma, \mathrm{Chr}\,\mathbf{s})$):

$$(\theta \in Cont_2) \wedge ((\chi(\theta) \cap (\chi(\mathcal{CSM}_\alpha(\rho)) \cup \chi(\mathcal{CSV}_\alpha(\tau))) = \emptyset) \wedge (dim(\theta) \geq Conc_\alpha(\tau)).$$

As $\theta \in Cont_2$, we can order the processes associated with vertices from θ according to their IS^2 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 IS^2 view, and as $\theta \in Cont_2$, q_1 also has the largest IS^1 view.

Consider the state of the execution at the time where q_1 successfully passes the test on Line 6. To pass this test, q_1 witnessed *IS1*, *Conc* and *IS2* states such that (with $q_1 = p_i$):

$$(\alpha(IS1[i]) > \alpha(IS1[i] \setminus \{p_j \in \Pi : IS1[j] = IS1[i]\})) \vee$$

$$\left(|\{p_j \in IS1[i] : IS2[j] = \emptyset \wedge IS1[j] \neq IS1[i]\}| < max\left(\alpha(IS1[i]), max_{j\in\{1,\ldots,n\}}(Conc[j])\right)\right)$$

If $(\alpha(IS1[i]) > \alpha(IS1[i] \setminus \{p_j \in \Pi : IS1[j] = IS1[i]\}))$, then it implies that q_1 belongs to a critical simplex. Indeed, it would belong to a set of processes sharing the same IS^1 view and such that, removing this set of processes from their IS^1 view would result in a set with a strictly smaller agreement power. But this would contradict $\chi(\theta) \cap \chi(\mathcal{CSM}_\alpha(carrier(\sigma, \mathrm{Chr}\,\mathbf{s}))) \neq \emptyset$ as it would include q_1. Therefore we have:

$$|\{p_j \in IS1[i] : IS2[j] = \emptyset \wedge IS1[j] \neq IS1[i]\}| < max\left(\alpha(IS1[i]), max_{j\in\{1,\ldots,n\}}(Conc[j])\right)$$

Two cases may arise:

- $max\left(\alpha(IS1[i]), max_{j\in\{1,\dots,n\}}(Conc[j])\right) \neq \alpha(IS1[i])$: In this case, a register in *Conc* was set on Line 12 to a value greater than $\alpha(IS1[i])$. It implies that a critical simplex associated with an agreement level strictly greater than $|\{p_j \in IS1[i] : IS2[j] = \emptyset \wedge IS1[j] \neq IS1[i]\}|$ terminated its computation and thus is included in $carrier(\theta, \operatorname{Chr}\mathbf{s})$. But we can observe that $(\chi(\theta) \setminus \{q_1\}) \subseteq \{p_j \in IS1[i] : IS2[j] = \emptyset \wedge IS1[j] \neq IS1[i]\}$, and hence, that $\dim(\theta) < Conc_\alpha(\tau)$ — a contradiction with $\sigma \notin \mathcal{R}_\mathcal{A}$.

- $max\left(\alpha(IS1[i]), max_{j\in\{1,\dots,n\}}(Conc[j])\right) = \alpha(IS1[i])$: Let c be the highest agreement power associated with a terminated critical simplex (with $c = 0$ if there is no terminated critical simplex is terminated). Therefore we have $Conc_\alpha(\tau) \geq c$. Let $\lambda \in \operatorname{Chr}\mathbf{s}$ be the simplex corresponding to the set of IS^1 views of processes in $IS1[i]$ which shared their IS^1 view at the time q_1 passed the test on Line 6. Consider the simplex $\lambda' \in \operatorname{Chr}\mathbf{s}$ corresponding to the completion of λ with the vertices corresponding to IS^1 view of the processes in $\chi(\theta)$ which may be missing from λ. Note that, since q_1 has the largest IS^1 view among processes from $\chi(\theta)$, $\chi(carrier(\lambda', \mathbf{s})) = \chi(carrier(\lambda, \mathbf{s})) = IS1[i]$. Moreover, since $\chi(\lambda) = \{p_j \in IS1[i] : IS1[j] \neq \perp\}$, we obtain that $\chi(\lambda') = \{p_j \in IS1[i] : IS1[j] \neq \perp\} \cup \chi(\theta)$. According to Corollary 4 applied to λ' with $l = c + 1$, we obtain that:

  $$\alpha(IS1[i]) - c - |(\chi(carrier(\lambda', c)) \setminus \chi(\lambda'))| \leq csize(\{\phi \in \mathcal{CS}_\alpha(\lambda') : \alpha(\chi(\phi)) \geq c + 1\}).$$

  Note that, since there is no terminated critical simplex with an agreement power greater than or equal to $c + 1$, it implies that one process of each critical simplex identified in λ' did not terminate its IS^2, hence a minimal hitting set. Let S_c be this minimal hitting of size equal to $csize(\{\phi \in \mathcal{CS}_\alpha(\lambda') : \alpha(\chi(\phi)) \geq c + 1\}$. Note that S_c does not include any process in $\chi(\theta)$. Indeed, given a critical simplex with the same IS^1 view as q_i, adding q_i to the critical simplex would produce a critical simplex, but by assumption processes in $\chi(\theta)$ do not belong to any critical simplex. We also have that S_c does not intersect $S_\emptyset = \{p_j \in IS1[i] : IS1[j] = \perp\}$. Hence, $|S_c| + |S_\emptyset \setminus \chi(\theta)| + |\chi(\theta)| = |S_c \cup S_\emptyset \cup \chi(\theta)|$. Therefore, as $|(\chi(carrier(\lambda', c)) \setminus \chi(\lambda'))| = |S_\emptyset \setminus \chi(\theta)|$ we obtain that $\alpha(IS1[i]) - c \leq |S_c \cup S_\emptyset \cup \chi(\theta)| - |\chi(\theta)|$.

  Let us now check that $S_c \cup S_\emptyset \cup \chi(\theta) \subseteq \{q_1\} \cup S_T$, with $S_T = \{p_j \in IS1[i] : IS2[j] = \emptyset \wedge IS1[j] \neq IS1[i]\}$. All are clearly included in $IS1[i]$ by construction. For processes in $S_\emptyset$, since they have their register in *IS1* equal to $\perp$, it is also the case for their register in *IS2*. For processes in $\chi(\theta)$, they have a strictly smaller IS^1 view by assumption. For the IS^2 view, they will have a strictly larger view than q_1. But since q_1 did not start its second immediate snapshot protocol, processes in $\chi(\theta)$ could not have terminated it. For processes in S_c, they do not share the same IS^1 view as any process in $\chi(\theta)$ since they are members of critical simplices, in particular, they thus have a distinct IS^1 view from q_1. By assumption, they did not terminate their second immediate snapshot protocol, and hence also have their *IS2* register still equal to $\perp$.

  Therefore, we have $S_c \cup S_\emptyset \cup \chi(\theta) \subseteq \{q_1\} \cup S_T$, and hence, $|S_c \cup S_\emptyset \cup \chi(\theta)| \leq 1 + |S_T|$. But since we also have $|S_T| < \alpha(IS1[i])$ and $\alpha(IS1[i]) - c \leq |S_c \cup S_\emptyset \cup \chi(\theta)| - |\chi(\theta)|$, we obtain that:

  $$|S_T| < |S_c \cup S_\emptyset \cup \chi(\theta)| - |\chi(\theta)| + c \leq |S_T| + 1 - |\chi(\theta)| + c.$$

  Thus $|\chi(\theta)| \leq c$. But recall that $Conc_\alpha(\tau) \geq c$, and so, $|\chi(\theta)| \leq Conc_\alpha(\tau)$ — a contradiction with $\sigma \notin \mathcal{R}_\mathcal{A}$.

□

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

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

As for other solutions of affine task, we can iterate this solution in order to simulate a run of $\mathcal{R}^*_\mathcal{A}$. Using this simulation we can therefore solve any task which is solvable in $\mathcal{R}^*_\mathcal{A}$:

Theorem 8. *Any task solvable in $\mathcal{R}^*_\mathcal{A}$ is solvable in the $\mathcal{A}$-model.*

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 1), 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 in 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 $\min_Q(V) = \min\{p \in V \cap Q\}$ be this map. The map μ_Q is therefore defined as follows:

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

Let us first show that, for any vertex $v \in \mathcal{R}_\mathcal{A}$ 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 9. *[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 $CS_\alpha(\mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s})) \subseteq \{\sigma \in \mathrm{Chr}\,\mathbf{s}; \sigma \subseteq \mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s})\}$, and as $\mu_Q(v) = \min_Q \circ \delta_Q(v)$, we obtain that:

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

As for any simplex $\sigma \in \mathrm{Chr}^2\,\mathbf{s}$, we have $\mathit{carrier}(\mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s}), \mathbf{s}) = \mathit{carrier}(v, \mathbf{s})$, thus Property 9 is verified if $\chi(\mathcal{CSV}_\alpha(\mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s}))) \cap Q \neq \emptyset$.

Now let us assume that $\chi(\mathcal{CSV}_\alpha(\mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s}))) \cap Q = \emptyset$. Due to the self-inclusion property, $\exists v' \in \mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s}))$ such that $\chi(v') = \chi(v)$. The self-inclusion property again implies that $\exists v'' \in \mathit{carrier}(v', \mathbf{s})$ such that $\chi(v'') = \chi(v') = \chi(v)$. Hence, as $\chi(v) \in Q$, $\exists v' \in \mathit{carrier}(v, \mathrm{Chr}\,\mathbf{s})$ such that $\chi(\mathit{carrier}(v', \mathbf{s})) \cap Q \neq \emptyset$. 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 $\chi(\mathit{carrier}(v, \mathbf{s}))$ 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 10. *[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 11. $\forall \sigma \in \mathrm{Chr}\,\mathbf{s}$, $\forall \theta_1, \theta_2 \in \mathcal{CS}_\alpha(\sigma)$*:*

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

Proof. Let us consider some simplex $\sigma \in \mathrm{Chr}\,\mathbf{s}$ and some critical simplices $\theta_1, \theta_2 \in \mathcal{CS}_\alpha(\sigma)$ such that $\alpha(\chi(carrier(\theta_1, \mathbf{s}))) = \alpha(\chi(carrier(\theta_2, \mathbf{s})))$. The inclusion property implies, 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 11 is verified) or else that $\chi(\theta_2) \cap \chi(carrier(\theta_1, \mathbf{s})) = \emptyset$.

Let us now 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 $\alpha(\chi(carrier(\theta_2, \mathbf{s}) \setminus \theta_2)) < \alpha(\chi(carrier(\theta_2, \mathbf{s})))$, 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 10:

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 smaller *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 since $\mathcal{CSV}_\alpha(carrier(\beta, \mathrm{Chr}\,\mathbf{s})) \cap Q = \emptyset$ is satisfied, 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, we obtain that $Conc_\alpha(carrier(\beta, \mathrm{Chr}\,\mathbf{s})) \geq k$.

Having $Conc_\alpha(carrier(\beta, \mathrm{Chr}\,\mathbf{s})) \geq k$, it implies that we have $\exists \sigma_c \in \mathcal{CS}_\alpha(carrier(\beta, \mathrm{Chr}\,\mathbf{s}))$ such that $\alpha(\chi(carrier(\sigma_c, \mathbf{s}))) \geq k$. But, since $\chi(carrier(\sigma_c, \mathbf{s})) \subseteq \mathcal{CSV}_\alpha(carrier(\beta, \mathrm{Chr}\,\mathbf{s}))$, we obtain that $\chi(carrier(\sigma_c, \mathbf{s})) \cap Q = \emptyset$. As the inclusion property implies that any *IS1* view must be strictly larger to intersect with Q, and as there are at most one *IS1* view associated with a critical simplex by agreement level (Lemma 11), all *IS1* views corresponding to some critical simplex in $carrier(\sigma, \mathrm{Chr}\,\mathbf{s})$ are 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 11 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 12. *[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 $\mathcal{CSV}_\alpha(\chi(carrier(v, \operatorname{Chr}\mathbf{s})))$, a subset of $carrier(v, \mathbf{s})$, or with $carrier(v', \mathbf{s})$ for a vertex $v' \in carrier(v, \operatorname{Chr}\mathbf{s})$, also a subset of $carrier(v, \mathbf{s})$. □

6.3 Correctness of the simulation

Let us first show that all simulated operations are safe. Since the composable shared memory simulation is safe, we only need to show that simulated α-adaptive set consensus operations 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 13. *The shared memory and α-adaptive set consensus simulation in $\mathcal{R}^*_\mathcal{A}$ is safe.*

Proof. For α-adaptive set consensus operations, Property 12 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 9 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 10 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.

Before completing an α-adaptive set consensus operation, processes make sure that all processes they observed are participating in the simulated run (by 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. For this, we only need to show that a process obtaining the smallest IS view among non-terminated processes eventually completes its α-adaptive set consensus operation.

Lemma 14. *In the shared memory and α-adaptive set consensus simulation in $\mathcal{R}^*_\mathcal{A}$, all processes eventually terminate.*

Proof. as soon as they observe a process with a decision estimate, processes adopt a decision estimate for any α-adaptive set consensus operation they may participate to. Hence, if a process has a pending α-adaptive set consensus operation and has the smallest IS view among non-terminated processes, all competitors will have a decision in the next iteration of the affine task. But since a process commits a decision estimate when all non-terminated processes which participate in an operation share a decision estimate during some iteration, a process obtaining the smallest IS view among non-terminated processes eventually commits its operation in the following affine task iteration and hence resume its shared-memory simulation. □

By Lemmata 13 and 14, 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. But it was shown in Section 3 that the α-adaptive set consensus model, the α-model and a corresponging fair adversarial model are all equivalent in term of task solvability. Hence, since we have shown in Theorem 8 that all task solvable in $\mathcal{R}^*_\mathcal{A}$ are solvable in the $\mathcal{A}$-model, we obtain the following equivalence result:

Theorem 15. *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 16. *[Fair Asynchronous Computability Theorem [FACT]] A task $T = (\mathcal{I}, \mathcal{O}, \Delta)$ is solvable in the 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.*

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-c7-TRlosed* adversaries. 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. The characterization only applies to colorless tasks, and it does not allow us to express the adversary in an 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 [6,11] (i.e., "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 models.

Saraph et al. [30] gave a compact combinatorial characterization of t-resilient task computability. Note that $\mathcal{A}_{t-res}$ 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 $\mathcal{R}_{t-res}$ in [30]. Gafni et al. [12] 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 agreement functions and a fair adversaries 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 [13]. It has been shown in [24] that agreement functions encode enough information to characterize the task computability of any fair adversary.

A short version of this paper appeared as a conference brief announcement [26], and as an extended version without formal proofs and shorten explanation [27].

8 Concluding remarks

This paper generalizes all existing topological characterizations of distributed computing models [12, 15, 19, 20, 30]. It applies to all tasks (not necessarily colorless) and all *fair* adversarial models (not necessarily t-resilience or k-obstruction-freedom). 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 second degree of the standard chromatic subdivision.

Interestingly, unlike [30], we cannot rely on the *shellability* [18] (and, thus, link-connectivity) of the affine task. 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 the simplicial structure). For example, the existence of a continuous map from $|\mathcal{R}_{\mathcal{A}_{t-res}}|$ to any $|\mathcal{R}^k_{\mathcal{A}_{t-res}}|$ implies that $\mathcal{R}_{\mathcal{A}_{t-res}}$ indeed captures the general task computability of $\mathcal{A}_{t-res}$ [30]. In general, however, the existence of a continuous map onto $\mathcal{C}$ only allows us to converge on a *single* vertex [18]. If $\mathcal{C}$ is not link-connected, converging on one vertex allows us to compute an output only for a single process, and not more. Unfortunately, only very special adversaries, such as $\mathcal{A}_{t-res}$, have link-connected counterparts (see, e.g., the affine task corresponding to 1-obstruction-freedom in Figure 7a). Instead of relying on link-connectivity, this paper takes an explicit algorithmic way of showing that iterations of $\mathcal{R}_{\mathcal{A}}$ simulate $\mathcal{A}$. An interesting question is to which extent point-set topology and continuous maps can be applied in affine characterizations.

Given that some models out of this class cannot be grasped by agreement functions (see [24] for examples), going beyond fair adversarial models is an important challenge. 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 [12], such as, e.g., generic adversarial models or the *set consensus collections* models [8] for which only special cases of k-set consensus [12] and k-test-and-set [25] 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. Round-by-round fault detectors (extended abstract): Unifying synchrony and asynchrony. In *Proceedings of the 17th Symposium on Principles of Distributed Computing*, 1998.

[12] 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.

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

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

[15] E. Gafni, P. Kuznetsov, and C. Manolescu. A generalized asynchronous computability theorem. In *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 222–231, 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 and T. Rieutord. Affine Tasks for k-Test-and-Set. working paper or preprint, June 2018.

[26] 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.

[27] P. Kuznetsov, T. Rieutord, and Y. He. An asynchronous computability theorem for fair adversaries. In C. Newport and I. Keidar, editors, *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 387–396. ACM, 2018.

[28] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[29] 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.

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

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

[32] 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, 31].

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* $\mathrm{Chr}\, C$. Let us first define $\mathrm{Chr}\, \mathbf{s}$ for the standard simplex $\mathbf{s}$. The vertices of $\mathrm{Chr}\, \mathbf{s}$ 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, $\mathrm{Chr}\, s$ 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 1a(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.