Agreement Functions for Distributed Computing Models

Petr Kuznetsov(✉) and Thibault Rieutord

LTCI, Télécom ParisTech, Université Paris Saclay, Paris, France
petr.kuznetsov@telecom-paristech.fr

Abstract. The paper proposes a surprisingly simple characterization of a large class of models of distributed computing, via an *agreement function*: for each set of processes, the function determines the best level of set consensus these processes can reach. We show that the task computability of a large class of *fair* adversaries that includes, in particular *superset-closed* and *symmetric* one, is precisely captured by agreement functions.

1 Introduction

In general, a model of distributed computing is a set of *runs*, i.e., all allowed interleavings of *steps* of concurrent processes. There are multiple ways to define these sets of runs in a tractable way.

A natural one is based on *failure models* that describe the assumptions on where and when failures might occur. By the conventional assumption of *uniform* failures, processes fail with equal and independent probabilities, giving rise to the classical model of *t-resilience*, where at most t processes may fail in a given run. The extreme case of $t = n - 1$, where n is the number of processes in the system, corresponds to the *wait-free* model.

The notion of *adversaries* [6] generalizes uniform failure models by defining a set of process subsets, called *live sets*, and assuming that in every model run, the set of *correct*, i.e., taking infinitely many steps, processes must be a live set. In this paper, we consider adversarial read-write shared memory models, i.e., sets of runs in which processes communicate via reading and writing in the shared memory and live sets define which sets of processes can be correct.

A conventional way to capture the power of a model is to determine its *task computability*, i.e., the set of distributed tasks that can be solved in it. For example, consider the 0-*resilient* adversary $\mathcal{A}_{0\text{-}res}$ defined through a single live set $\{p_1, \ldots, p_n\}$: the adversary says that no process is allowed to fail (by taking only finitely many steps). It is easy to see that the model is strong enough to solve *consensus*, and, thus, any task [14].[1]

T. Rieutord—Supported by ANR project DISCMAT, grant agreement ANR-14-CE35-0010-01.

[1] In the "universal" task of consensus, every process has a private *input* value, and is expected to produce an *output* value, so that (validity) every output is an input of some process, (agreement) no two processes produce different output values, and (termination) every process taking sufficiently many steps returns.

A. El Abbadi and B. Garbinato (Eds.): NETYS 2017, LNCS 10299, pp. 175–190, 2017.
DOI: 10.1007/978-3-319-59647-1_14

In this paper, we propose a surprisingly simple characterization of the task computability of a large class of adversarial models through *agreement functions*.

An agreement function α maps subsets of processes $\{p_1, \ldots, p_n\}$ to positive integers in $\{0, 1, \ldots, n\}$. For each subset P, $\alpha(P)$ determines, intuitively, the level of *set consensus* that processes in P can reach when no other process is active, i.e., the smallest number of distinct input values they can decide on.

For example, the agreement function of the wait-free shared-memory model is $\alpha_{wf} : P \mapsto |P|$ and the t-resilient model, where at most t processes may fail or not participate, has $\alpha_{t,res} :\ P \mapsto \max(0, |P| - n + t + 1)$.

The agreement function of an adversary $\mathcal{A}$ can be computed using the notion of *set consensus power* of an adversary introduced in [13]: $\alpha_{\mathcal{A}}(P) = \mathit{setcon}(\mathcal{A}|_P)$. Here $\mathcal{A}|_P$ is the *restriction of* $\mathcal{A}$ *to* P, i.e., the adversary defined through the live sets of $\mathcal{A}$ that are subsets of P.

To each agreement function α, corresponding to an existing model, we associate a particular model, the *α-model*. The α-model is defined as the set of runs satisfying the following property: the set P of *participating* (taking at least one step) processes in a run is such that $\alpha(P) \geq 1$ and is such that at most $\alpha(P) - 1$ processes take only finitely many steps in it. An algorithm solves a task T in the α-model if processes taking infinitely many steps produces an output in any run.

We show that, for the class of *fair* adversaries, agreement functions "tell it all" about task computability: a task is solvable in a fair adversarial model with agreement function α *if and only if* it is solvable in the α-model. Fair adversaries include notably the class of superset-closed [16,19] and the class of symmetric [23] adversaries. Intuitively, superset-closed adversaries do not anticipate failures of processes: if $S \in \mathcal{A}$ and $S \subseteq S'$, then $S' \in \mathcal{A}$. Symmetric adversaries do not depend on processes identifiers: if $S \in \mathcal{A}$, then for every set of processes S' such that $|S'| = |S|$, we have $S' \in \mathcal{A}$.

A corollary of our result is a characterization of the k-concurrency model [9,10]. Here we use the fact that the k-concurrency model is equivalent, with respect to task solvability, to the *k-obstruction-freedom* [13], a symmetric adversary consisting of live sets of sizes from 1 to k. Thus, the agreement function $\alpha_{k\text{-}conc} :\ P \mapsto \min(|P|, k)$ captures the k-concurrent task computability. An alternative characterization of k-concurrency via a compact *affine* task was recently suggested in [11].

There are, however, models that are not captured by their agreement functions. We give an example of a *non-fair* adversary that solves strictly more tasks than its *α-model*. Characterizing the class of models that can be captured through their agreement function is an intriguing open question.

The rest of the paper is organized as follows. Section 2 gives model definitions. In Sect. 3, we formally define the notion of an agreement function. In Sect. 4, we prove a few useful properties of α-models. In Sect. 5, we present the class of fair adversary, show that superset-closed and symmetric adversaries are fair and that fair adversaries are captured by their agreement functions. In Sect. 6, we give examples of models that are *not* captured by agreement functions. Section 7 reviews related work, and Sect. 8 concludes the paper.

2 Preliminaries

Processes, Runs, Models. Let Π be a system of n asynchronous processes, $p_1, \ldots, p_n$ that communicate via a shared atomic-snapshot memory [1]. The atomic-snapshot (AS) memory is represented as a vector of n shared variables, where each process is associated with a distinct position in this vector, and exports two operations: *update* and *snapshot*. An *update* operation performed by p_i replaces position i with a new value and a *snapshot* operation returns the current state of the vector.

We assume that processes run the *full-information* protocol: the first value each process writes is its *input value*. A process alternates between taking snapshots of the memory and writing back the result of its latest snapshot. A *run* is thus a sequence of process identifiers stipulating the order in which the processes take operations: each odd appearance of i in the sequence corresponds to an *update* and each even appearance corresponds to a *snapshot*. A *model* is a set of runs.

Failures and Participation. A process that takes only finitely many steps of the full-information protocol in a given run is called *faulty*, otherwise it is called *correct*. A process that took at least one step in a given run is called *participating* in it. The set of participating processes in a given run is called its *participating set*. Note that, since every process writes its input value in its first step, the inputs of participating processes are eventually known to every process that takes sufficiently many steps.

Tasks. In this paper, we focus on distributed *tasks* [18]. 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 which invoked the task respect the task specification. Formally, a *task* is defined through a set $\mathcal{I}$ of input vectors (one input value for each process), a set $\mathcal{O}$ of output vectors (one output value for each process), and a total relation $\Delta : \mathcal{I} \mapsto 2^{\mathcal{O}}$ that associates each input vector with a set of possible output vectors. An input $\perp$ denote a *not participating* process and an output value $\perp$ denote an *undecided* process. Check [15] for more details.

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

Solving a Task. We say that an algorithm A solves a task $T = (\mathcal{I}, \mathcal{O}, \Delta)$ in a model M if A ensures that (1) in every run in which processes start with an input vector $I \in \mathcal{I}$, all decided values form a vector $O \in \mathcal{O}$ such that $(I, O) \in \Delta$, and (2) if the run is in M, then every correct process decides.

This gives rise to the notion of task solvability, i.e., a task T is solvable in a model M if and only if there exists an algorithm A which solves T in M.

BGG Simulation. The principal technical tool in this paper is a simulation technique that we call the *BGG simulation*, after Borowski, Gafni, Guerraoui, collecting algorithmic ideas presented in [3,8–10]. The technique allows a system of n processes that communicate via read-write shared memory and k-set consensus objects to *simulate* a k-process system running an arbitrary read-write algorithm. In particular, we can use this technique to run an extended BG simulation [8] on top of these k simulated processes, which gives a simulation of an arbitrary k-concurrent algorithm. An important feature of the simulation is that it adapts to the number of currently active simulated processes a: if it goes below k (after some simulated processes complete their computations), the number of used simulators also becomes a. We refer to [11] for a detailed description of this simulation algorithm.

3 Agreement Functions

Definition 1 (Agreement function). *The* agreement function *of a model* M *is a function* $\alpha : 2^{\Pi} \to \{0, \ldots, n\}$*, such that for each* $P \in 2^{\Pi}$*, in the set of runs of* M *in which no process in* $\Pi \setminus P$ *participates, iterative* $\alpha(P)$*-set consensus can be solved, but* $(\alpha(P) - 1)$*-set consensus cannot. By convention, if* M *contains no (infinite) runs with participating set* P*, then* $\alpha(P) = 0$*.*

Intuitively, for each P, we consider a model consisting of runs of M in which only processes in P participate and determine the best level of set consensus that can be reached in this model, with 0 corresponding to a model that consists of *finite* runs only.

Note the agreement function α of a model M is *monotonic*: $P \subseteq P' \Rightarrow \alpha(P) \leq \alpha(P')$. Indeed, the set of runs of M where the processes in $\Pi \setminus P$ do not take any step is a subset of the set of runs of M where the processes in $\Pi \setminus P'$ do not take any step. In this paper, we only consider monotonic functions α.

Definition 2 (α-model). *Given a monotonic agreement function* α*, the* α-model *is the set of runs in which, the participating set* P *satisfies: (1)* $\alpha(P) \geq 1$*; and, (2) at most* $\alpha(P) - 1$ *participating processes take only finitely many steps.*

We say that a model is *characterized by its agreement function* α if and only if it solves the same set of task as the α-model.

Definition 3 (α-adaptive set consensus). *The* α*-adaptive set consensus task satisfies the* **validity** *and* **termination** *properties of consensus and the* α**-agreement** *property: if at some time* τ*,* k *distinct values have been returned, then the current participating set* P_τ *is such that* $\alpha(P_\tau) \geq k$*.*

We can easily show that any model with agreement function α can solve the α-adaptive set consensus task, i.e., to achieve the best level of set consensus without this an priori knowledge of the set of processes that are allowed to participate [20].

4 Properties of the α-model

We now relate task solvability in the α-model and in M. More precisely, we show that (1) the agreement function of the α-model is α and (2) any task T solvable in the α-model is also solvable in every model with agreement function α.

Theorem 1. *The agreement function of the α-model is α.*

Proof. Take P such that $\alpha(P) > 1$ and consider the set of runs of the α-model in which no process in $\Pi \setminus P$ participates and, thus, according to the monotonicity property, at most $\alpha(P) - 1$ processes are faulty. To solve $\alpha(P)$-set consensus, we use the *safe-agreement* protocol [2], the crucial element of BG simulation. Safe agreement solves consensus if every process that participates in it takes enough steps. The failure of a process then may *block* the safe-agreement protocol. In our case as at most $\alpha(P) - 1$ processes in P can fail, so we can simply run $\alpha(P)$ safe agreement protocols: every process goes through the protocols one by one using its input as a proposed value, if the protocol blocks, it proceeds to the next one in the round-robin manner. The first protocol that returns gives the output value. Since at most $\alpha(P) - 1$ processes are faulty, at least one safe agreement eventually terminates, and there are at most $\alpha(P)$ distinct outputs. To see that $(\alpha(P) - 1)$ cannot be solved in this set of runs, recall that one cannot solve $(\alpha(P) - 1)$-set consensus $(\alpha(P) - 1)$-resiliently [2,18,22].

The following result is instrumental in our characterizations of *fair* adversaries:

Theorem 2. *For any task T solvable in an α-model, T is solvable in any read-write shared memory model which solves the α-adaptive set consensus task.*

Proof. Using α-adaptive set consensus and read-write shared memory, we can run *BGG*-simulation so that, when the participating set is P, at most $\alpha(P)$ *BG simulators* are activated and at least one is live (i.e., takes part in infinitely many simulation steps). Moreover, we make a process provided with a (simulated) task output to stop proposing simulated steps to BGG simulation. Hence, the number of active simulators is also bounded by the number of participating processes without an output, with at least one live BG simulator if there is a correct process without a task output.

These BG simulators are used to simulate an execution of a protocol solving T in the α-model. And so, since any finite run can be extended to a valid run of the α-model, the protocol can only provide valid outputs.

We make BG simulators execute the *breadth-first* simulation: every BG simulator executes an infinite loop consisting of (1) updating the estimated participating set P, then (2) try to execute a simulation step of every process in P, one by one.

Now assume that there exist $k \geq 1$ correct processes that are never provided with a task output. BGG simulation ensure that we eventually have at most $min(k, \alpha(P))$ active simulators, with at least one live among them. Let s be such a live simulator. After every process in P have taken their first steps, s tries to simulate steps for every process of P infinitely often. A process simulation step can be blocked forever only due to an active but not live BG simulator[2], thus there are at most $min(k, \alpha(P)) - 1$ simulated processes in P taking only finitely many steps.

As at most $\alpha(P) - 1$ processes have a finite number of simulated steps, the simulated run is a valid run of the α-model. Moreover, as at most $k-1$ processes have a finite number of simulated steps, there is one process never provided with a task output simulated as a correct process. But, a protocol solving a task eventually provides task outputs to every correct process — a contradiction.

Any model can solve its associated α-adaptive set consensus task [20]. Along with Theorem 2, we derive that:

Corollary 1. *Let M be any model, α_M be its agreement function, and T be any task that is solvable in the α_M-model. Then M solves T.*

5 Characterizing Fair Adversaries

An *adversary* $\mathcal{A}$ is a set of subsets of Π, called *live sets*, $\mathcal{A} \subseteq 2^{\Pi}$. 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 is thus defined as the set of $\mathcal{A}$-compliant runs.

An adversary is *superset-closed* [19] 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* adversary consisting of sets of $n - t$ or more processes.

An adversary $\mathcal{A}$ is a *symmetric* adversary if it does not depend on process identifiers: $\forall S \in \mathcal{A}$, $\forall S' \subseteq \Pi$, $|S'| = |S| \implies S' \in \mathcal{A}$. Symmetric adversaries provides another interesting generalization of the classical t-resilience condition and k-obstruction-free progress condition [9] which was previously formalized by Taubenfeld as its symmetric progress conditions [23].

5.1 Set Consensus Power

The notion of the *set consensus power* [12] was originally proposed to capture the power of adversaries in solving *colorless* tasks [3,4], i.e., tasks that can be defined by relating *sets* of inputs and outputs, independently of process identifiers.

Definition 4. *The* set consensus power *of $\mathcal{A}$, denoted by $setcon(\mathcal{A})$, is defined as follows:*

- *If $\mathcal{A} = \emptyset$, then $setcon(\mathcal{A}) = 0$*
- *Otherwise, $setcon(\mathcal{A}) = \max_{S \in \mathcal{A}} \min_{a \in S} setcon(\mathcal{A}|_{S \setminus \{a\}}) + 1$.*[3]

[2] Note that the extended BG-simulation provides a mechanism which ensures that a simulation step is not blocked forever by a no longer active BG simulator.

[3] $\mathcal{A}|_P$ is the adversary consisting of all live sets of $\mathcal{A}$ that are subsets of P.

Thus, for a non-empty adversary $\mathcal{A}$, $setcon(\mathcal{A})$ is determined as $setcon(\mathcal{A}|_{S\setminus\{a\}})$ + 1 where S is an element of $\mathcal{A}$ and a is a process in S that "max-minimize" $setcon(\mathcal{A}|_{S\setminus\{a\}})$. Note that for $\mathcal{A} \neq \emptyset$, $setcon(\mathcal{A}) \geq 1$.

It is shown in [12] that $setcon(\mathcal{A})$ is the smallest k such that $\mathcal{A}$ can solve k-set consensus.

It was previously shown in [13] that for a superset-closed adversary $\mathcal{A}$, the set consensus power of $\mathcal{A}$ is equal to $csize(\mathcal{A})$, where $csize(\mathcal{A})$ denote the minimal hitting set size of $\mathcal{A}$, i.e., a minimal subset of Π that intersects with each live set of $\mathcal{A}$. Therefore if $\mathcal{A}$ is superset-closed, then $setcon(\mathcal{A}) = csize(\mathcal{A})$. For a symmetric adversary $\mathcal{A}$, it can be easily derived from the definition of $setcon$ that $setcon(\mathcal{A}) = |\{k \in \{1, \ldots, n\} : \exists S \in \mathcal{A}, |S| = k\}|$.

Theorem 3. *The agreement function of adversary $\mathcal{A}$ is $\alpha_{\mathcal{A}}(P) = setcon(\mathcal{A}|_P)$.*

Proof. An algorithm A_P that solves $\alpha_{\mathcal{A}}(P)$-set consensus, assuming that the participating set is a subset of P, is a straightforward generalization of the result of [12]. It is shown in [12] that $setcon(\mathcal{A})$-set consensus can be solved in $\mathcal{A}$. But if we restrict the runs to assume that the processes in $\Pi \setminus P$ do not take a single step, then the set of possible live sets reduces to $\mathcal{A}|_P$. Thus using the agreement algorithm of [12] for the adversary $\mathcal{A}|_P$, we obtain a $setcon(\mathcal{A}|_P)$-set consensus algorithm, or equivalently, an $\alpha_{\mathcal{A}}(P)$-set consensus algorithm.

It is immediate from Theorem 3 that $\mathcal{A} \subseteq \mathcal{A}'$ implies $setcon(\mathcal{A}) \leq setcon(\mathcal{A}')$.

5.2 Fair adversaries

In this paper we propose a class of adversaries which encompasses both classical classes of super-set closed and symmetric adversaries. Informally, an adversary is *fair* if its set consensus power does not change if only a subset of the processes are participating in an agreement protocol.

More precisely, consider $\mathcal{A}$-compliant runs with participating set P and assume that processes in $Q \subseteq P$ want to reach agreement *among themselves*: only these processes propose inputs and are expected to produce outputs. We can only guarantee outputs to processes in Q when the set of correct processes include some process in Q, i.e., when the current live set intersect with Q. Thus, the best level of set consensus reachable by Q is defined the set consensus power of adversary $\mathcal{A}|_{P,Q} = \{S \in \mathcal{A}|_P, S \cap Q \neq \emptyset\}$, unless $|Q| < setcon(\mathcal{A}|_P)$.

Definition 5 (Fair adversary). *An adversary $\mathcal{A}$ is fair if and only if:*

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

Property 1.

$$setcon(\mathcal{A}|_{P,Q}) \leq \min(|Q|, setcon(\mathcal{A}|_P))$$

Proof. For any $P \subseteq \Pi$ and $Q \subseteq P$, $\mathcal{A}|_{P,Q} = \{S \in \mathcal{A}|_P, S \cap Q \neq \emptyset\}$ is a subset of $\mathcal{A}|_P$ and, thus, $setcon(\mathcal{A}|_{P,Q}) \leq setcon(\mathcal{A}|_P)$. Moreover, $setcon(\mathcal{A}|_{P,Q}) \leq |Q|$, as $|Q|$-set consensus can be solved in $\{S \in \mathcal{A}|_P, S \cap Q \neq \emptyset\}$ as follows: every process waits until some process in Q writes its input and decides on it.

Theorem 4. *Any superset-closed adversary is fair.*

Proof. Suppose that there exists a superset-closed adversary $\mathcal{A}$ that is not fair, i.e., by Property 1, $\exists P \subseteq \Pi, \exists Q \subseteq P, setcon(\{S \in \mathcal{A}|_P, S \cap Q \neq \emptyset\}) < min(|Q|, setcon(\mathcal{A}|_P))$. Clearly $\mathcal{A}|_P$ and $\mathcal{A}|_{P,Q}$ are also superset-closed and, thus, $setcon(\mathcal{A}|_P) = csize(\mathcal{A}|_P)$ and $setcon(\mathcal{A}|_{P,Q}) = csize(\mathcal{A}|_{P,Q})$.

Since $setcon(\mathcal{A}|_{P,Q}) < |Q|$, a minimal hitting set H' of $\mathcal{A}|_{P,Q}$ is such that $|H'| < |Q|$, and therefore there exists a process $q \in Q$, $q \notin H'$. Also, since $setcon(\mathcal{A}|_{P,Q}) < setcon(\mathcal{A}|_P)$, H' is not a hitting set of $\mathcal{A}|_P$. Thus, there exists $S \in \mathcal{A}|_P$ such that $S \cap H' = \emptyset$. Hence, $(S \cup \{q\}) \cap H' = \emptyset$. Since $\mathcal{A}|_P$ is superset closed, we have $S \cup \{q\} \in \mathcal{A}|_P$ and, since $q \in Q$, $S \cup \{q\} \in \mathcal{A}|_{P,Q}$. But $(S \cup \{q\}) \cap H' = \emptyset$—a contradiction with H' being a hitting set of $\mathcal{A}|_{P,Q}$.

Theorem 5. *Any symmetric adversary is fair.*

Proof. The set consensus power of a generic adversary $\mathcal{A}$ is defined recursively through finding $S \in \mathcal{A}$ and $p \in S$ which max-minimize the set consensus power of $\mathcal{A}|_{S\setminus\{p\}}$. Let us recall that if $\mathcal{A} \subseteq \mathcal{A}'$ then $setcon(\mathcal{A}) \leq setcon(\mathcal{A}')$. Therefore, S can always be selected to be *locally maximal*, i.e., such that there is no live set in $S' \in \mathcal{A}$ with $S \subsetneq S'$.

Suppose by contradiction that $\mathcal{A}$ is symmetric but not fair, i.e., by Property 1, for some $P \subseteq \Pi$ and $Q \subseteq P$, $setcon(\mathcal{A}|_{P,Q}) < min(|Q|, setcon(\mathcal{A}|_P))$. We show that if the property holds for P and Q such that $\mathcal{A}|_{P,Q} \neq \emptyset$ then it also holds for some $P' \subsetneq P$ and $Q' \subseteq Q$.

First, we observe that $|Q| > 1$, otherwise $setcon(\mathcal{A}|_{P,Q}) = 0$ and, thus, we have $\mathcal{A}|_{P,Q} = \emptyset$.

Since $\mathcal{A}$ is symmetric, $\mathcal{A}|_P$ is also symmetric. Thus, for every $S \in \mathcal{A}|_P$ and $p \in S$ such that $setcon(\mathcal{A}|_P) = 1 + setcon(\mathcal{A}|_{S\setminus\{p\}})$, any S' such that $|S'| = |S|$ and for any $p' \in S'$, we also have $setcon(\mathcal{A}|_P) = 1 + setcon(\mathcal{A}|_{S'\setminus\{p'\}})$. Since we can always choose S to be a maximal set, we derive that the equality holds for every maximal set S in $\mathcal{A}|_P$ and every $p \in S$.

Let us recall that, by the definition of *setcon*, there exists $L \in \mathcal{A}|_{P,Q}$ and $a \in L$ such that $setcon(\mathcal{A}|_{P,Q}) = 1 + setcon((\mathcal{A}|_{P,Q})|_{L\setminus\{a\}}) = setcon(\mathcal{A}|_{L,Q})$. Since $\mathcal{A}|_P$ is symmetric, for all L', $|L'| = |L|$ and $L \cap Q \subseteq L' \cap Q$, we have $setcon(\mathcal{A}|_{L',Q}) \geq setcon(\mathcal{A}|_{L,Q})$. Indeed, modulo a permutation of process identifiers, $\mathcal{A}|_{L',Q}$ contains all the live sets of $\mathcal{A}|_{L,Q}$ plus live sets in $\mathcal{A}|_{L'}$ that overlap with $(L' \cap Q) \setminus (L \cap Q)$. Since $setcon(\mathcal{A}|_{L,Q}) = setcon(\mathcal{A}|_{P,Q})$ and $L' \in \mathcal{A}|_{P,Q}$, we have $setcon(\mathcal{A}|_{L',Q}) = setcon(\mathcal{A}|_{L,Q})$. Therefore, for any $a \in L'$, $setcon(\mathcal{A}|_{L'\setminus\{a\},Q}) < setcon(\mathcal{A}|_{L'\setminus\{a\}})$.

In particular, for L' with $L' \cap Q \in \{L', Q\}$, $setcon(\mathcal{A}|_{L',Q}) = setcon(\mathcal{A}|_{L,Q})$. Note that $L' \nsubseteq Q$, otherwise, $\mathcal{A}|_{L',Q} = \mathcal{A}|_{L'}$ and, thus, $setcon(\mathcal{A}|_{L',Q}) = setcon(\mathcal{A}|_{L'}) = setcon(\mathcal{A}|_P)$, contradicting our assumption.

Thus, let us assume that $Q \subsetneq L'$. Note that $Q' = Q \setminus \{a\} \subsetneq L' \setminus \{a\}$, and since $|Q| \geq 2$, $Q' \neq \emptyset$, we have $setcon(\mathcal{A}|_{P',Q'}) < setcon(\mathcal{A}|_{P'})$ for $P' = L' \setminus \{a\}$ and $Q' \subseteq P'$, $Q' \neq \emptyset$. Furthermore, since $setcon(\mathcal{A}|_{P,Q}) < |Q|$, we have $setcon(\mathcal{A}|_{P',Q'}) < |Q'|$.

By applying this argument inductively, we end up with a live set P and $Q \subseteq P$ such that $setcon(\mathcal{A}|_P) \geq 1$, $Q \neq \emptyset$ and $setcon(\mathcal{A}|_{P,Q}) = 0$. By the definition of $setcon$, $\mathcal{A}|_P \neq \emptyset$ and $\mathcal{A}|_{P,Q} = \emptyset$. But $\mathcal{A}|_P$ is symmetric and $Q \neq \emptyset$, so for every $S \in \mathcal{A}|_P$, there exists $S' \in \mathcal{A}|_P$ such that $|S| = |S'|$ and $S' \cap Q \neq \emptyset$, i.e., $\mathcal{A}|_{P,Q} \neq \emptyset$—a contradiction.

Note that not all adversaries are fair. For example, the adversary $\mathcal{A} = \{\{p_1\}, \{p_2, p_3\}, \{p_1, p_2, p_3\}\}$ is not fair. On the other hand, not all fair adversaries are either super-set closed or symmetric. For example, the adversary $\mathcal{A} = 2^{\{p_1,p_2,p_3\}} \setminus \{p_1, p_2\}$ is fair but is neither symmetric not super-set closed. Understanding what makes an adversary fair is an interesting challenge.

5.3 Task Computability in Fair Adversarial Models

In this section, we show that the task computability of a fair adversarial $\mathcal{A}$-model is fully grasped by its associated agreement function $\alpha_{\mathcal{A}}$.

Algorithm 1. Code for BG simulator s_i to simulate adversary $\mathcal{A}$.

```
Shared variables: R[1,...,α_A(Π)] ← (⊥,∅), P_MEM[p_1,...,p_n] ← ⊥;
Local variables: S_cur, S_tmp, P, A, W ∈ 2^Π, p_cur, p_tmp ∈ ℕ, S_cur ← ∅;

Repeat
    P = {p ∈ Π, P_MEM[p] ≠ ⊥};
    A = {p ∈ P, P_MEM[p] ≠ ⊤};
    if i ≥ min(|A|, α_A(P)) then
        W = P;
        for j = α_A(Π) down to i+1 do
            (p_tmp, S_tmp) ← R[j];
            if (p_tmp ≠ ⊥) ∧ (S_tmp ⊆ W) ∧ ((setcon(A|_{S_tmp,A}) ≥ j)) then
                W ← S_tmp \ {p_tmp};

        if (S_cur ⊈ W) ∨ (setcon(A|_{S_cur,A}) < i) then
            if ∃S ∈ A|_W, setcon(A|_{S,A}) ≥ i then
                S_cur = S ∈ A|_W such that setcon(A|_{S,A}) ≥ i;
            else S_cur = S ∈ A|_P;
            p_cur = S_cur.first();
            R[i] ← (p_cur, S_cur);

        if (SimulateStep(p_cur) = SUCCESS) then
            if Outputed(p_cur) then P_MEM[p_cur] = ⊤;
            p_cur = S_cur.next(p_cur);
        else AbortStep(p_cur) ;
Forever;
```

Using BGG simulation, we show that the $\alpha_{\mathcal{A}}$-model can be used to solve any task T solvable in the $\mathcal{A}$-model. In the simulation, up to $\alpha(P)$ BG simulators execute the given algorithm solving T, where P is the participating set of the current run. We adapt the currently simulated live set to include processes not yet provided with a task output, and ensure that the chosen live set is simulated sufficiently long until some active processes are provided with outputs of T. The simulation terminates as soon as all correct processes are provided with outputs.

The code for BG simulator $b_i \in \{b_1, \ldots, b_{\alpha_{\mathcal{A}}(\Pi)}\}$ is given in Algorithm 1. It consists of two parts: (1) selecting a live set to simulate (lines 7–17), and (2) simulating processes in the selected live set (lines 18–21).

Selecting a Live Set. This is the most involved part. The idea is to select a participating live set $L \subseteq P$ such that: (1) the set consensus power of $\mathcal{A}|_{L,A} = \{S \in \mathcal{A}|_L, S \cap A \neq \emptyset\}$, with A the set of participating processes not yet provided with a task output, is greater than or equal to the BG simulator identifier i; (2) L is a subset of the live sets currently selected by live BG simulators with greater identifiers; (3) L does not contain the processes currently simulated by live BG simulators with greater identifiers.

The live set selection in Algorithm 1 consists in two phases. First, BG simulators determine a *selection window* W, $W \subseteq P$, i.e., the largest set of processes which is a subset of the live sets selected by live BG simulators with greater identifiers, and which excludes the processes currently selected by live BG simulators with greater identifiers (lines 7–11). This is done iteratively on all BG simulators with greater identifiers, from the greatest to the lowest. At each iteration, if the targeted BG simulator b_k *appears live*, the current window is restricted to the live set selected by b_k, but excluding the process selected by b_k. Determining if b_k appears live is simply done by checking whether, with the current simulation status observed, the live set selected by b_k is *valid*, i.e., satisfies conditions (1), (2) and (3) above.

The second phase (lines 12–17), consists in checking if the currently selected live set is valid (line 12). If not, the BG simulator tries to select a live set L which belongs to the selection window W, and hence satisfies (2) and (3), but also such that the set consensus power of $\mathcal{A}_{L,A}$ is greater than i, the BG simulator identifier (line 14). If the simulator does not find such a live set, it simply selects any available live set (line 15).

Simulating a Live Set. The idea is that, if the selected live set does not change, the BG simulator simulates steps of every process in its selected live set infinitely often. Unlike conventional variations of BG simulations, a BG simulator here does not skip a blocked process simulation, instead it aborts and re-tries the same simulation step until it is successful.

Intuitively, this does not obstruct progress because, in case of a conflict, there are two live BG simulators blocked on the same simulation step, but the BG simulator with the smaller identifier will eventually change its selected live set and release the corresponding process.

Pseudocode. The protocol executed by processes in the $\alpha_{\mathcal{A}}$-model is the following: Processes first update their status in P_{MEM} by replacing $\perp$ with their initial state. Then, processes participate in an $\alpha_{\mathcal{A}}$-adaptive BGG simulation (i.e., BGG simulation runs on top of an $\alpha_{\mathcal{A}}$-adaptive set consensus protocol), where BG simulators use Algorithm 1 to simulate an algorithm solving a given task T in the adversarial $\mathcal{A}$-model. When a process p observes that $P_{MEM}[p]$ has been set to $\top$ ("termination state"), it stops to propose simulation steps.

Proof of Correctness. Let P_f be the participating set of the $\alpha_{\mathcal{A}}$-model run, and let A_f be the set of processes $p \in P_f$ such that $P_{MEM}[p]$ is never set to $\top$.

Lemma 1. *There is a time after which variables P and A in Algorithm 1 become constant and equal to A_f and P_f for all live BG simulators.*

Proof. Since Π is finite, the set of processes p such that $P_{MEM}[p] \neq \perp$ eventually corresponds to P_f as the first step of p is to set $P_{MEM}[p]$ to its initial state and $P_{MEM}[p]$ can only be updated to $\top$ afterwards. As after $P_{MEM}[p]$ is set to $\top$, it cannot be set to another value, eventually, the set of processes from P_f such that $P_{MEM}[p] \neq \top$ is equal to A_f. Live BG simulators update P and A infinitely often, so eventually their values of P and A are equal to P_f and A_f respectively.

Lemma 2. *If A_f contains a correct process, then there is a correct BG simulator with an identifier smaller or equal to $min(|A_f|, \alpha_{\mathcal{A}}(P_f))$.*

Proof. In our protocol, eventually only correct processes in A_f are proposing BGG simulation steps. Thus eventually, at most $|A_f|$ distinct simulations steps are proposed. The $\alpha_{\mathcal{A}}$-adaptive set consensus protocol used for BGG simulation ensures that at most $\alpha_{\mathcal{A}}(P_f)$ distinct proposed values are decided. But as there is a time after which only processes in A_f propose values, eventually, $min(|A_f|, \alpha_{\mathcal{A}}(P_f))$-set consensus is solved. Thus BGG simulation ensures that, when this is the case, there is a live BG simulator with an identifier smaller or equal to $min(|A_f|, \alpha_{\mathcal{A}}(P_f))$.

Suppose that A_f contains a correct process, and let b_m be the greatest live BG simulator such that $m \leq min(|A_f|, \alpha_{\mathcal{A}}(P_f))$ (by Lemma 2). Let $S_i(t)$ denote the value of S_{cur} and let $p_i(t)$ denote the value of p_{cur} at simulator b_i at time t. Let also τ_f be the time after which every active but not live BG simulators have taken all their steps, and after which A and P have become constant and equal to A_f and P_f for every live BG simulator (by Lemma 1).

Lemma 3. *For every live BG simulator b_s, with $s \leq min(|A_f|, \alpha_{\mathcal{A}}(P_f))$, eventually, b_s cannot fail the test on line 13.*

Proof. Consider a correct BG simulator b_s starting a round after time τ_f. Let W_s be the value of W at the end of line 11. Two cases may arise:

- If $W_s = P_f$, as $\mathcal{A}$ is fair, then $setcon(\mathcal{A}|_{W_s,A_f}) = \min(|A_f|, setcon(\mathcal{A}|_{P_f}))$. Thus, $setcon(\mathcal{A}|_{W_s,A_f}) \geq s$.

– Otherwise, W_s is set on line 11 to some $S_{target} \setminus \{p_{target}\}$ at some iteration l, with $setcon(\mathcal{A}|_{S_{target},A_f}) \geq l$ for $l > s$. We have $setcon(\mathcal{A}|_{W_s,A_f}) = setcon((\mathcal{A}|_{S_{target},A})|_{S_{target}\setminus\{p_{target}\}})$ which, by the definition of *setcon*, is greater or equal to $setcon(\mathcal{A}|_{S_{target},A}) - 1 \geq l - 1 \geq s$, so we have $setcon(\mathcal{A}|_{W_s,A_f}) \geq s$.

By the definition of *setcon*, as $setcon(\mathcal{A}|_{W_s,A_f}) \geq s$, there exists $S \subseteq W_s$ such that $setcon(\mathcal{A}|_{S,A_f}) \geq s$. So, eventually b_s will always succeed the test on line 13.

Lemma 4. *For every live BG simulator b_s, with $s \leq min(|A_f|, \alpha_{\mathcal{A}}(P_f))$, eventually, the value of W computed at the end of iteration $m+1$ (at lines 8–11) is equal to some constant value $W_{m,f}$.*

Proof. No BG simulator b_l, with $l > m$, executes lines 7–21 after time τ_f. Therefore $R[l]$ is constant after time τ_f, $\forall l > m$. As the computation of W, on lines 7–11, only depends on the value of A, P and $R[l]$, for $\alpha_{\mathcal{A}}(\Pi) \geq l > m$, all constant after time τ_f, then the value of W computed at the end of line 11 for iteration $m+1$ is the same at every round initiated after time τ_f for any live BG simulator b_s, with $s \leq min(|A_f|, \alpha_{\mathcal{A}}(P_f))$.

Lemma 5. *If A_f contains a correct process, then the set of processes with an infinite number of simulated steps is a live set of $\mathcal{A}$ containing a process of A_f.*

Proof. As b_m is live, it proceeds to an infinite number of rounds. By Lemma 4, eventually b_m computes the same window in every round. By Lemma 3, if b_m does not have a valid live set selected, then it eventually selects a valid one for $W_{m,f}$. Thus, eventually b_m never changes its selected live set. Let $S_{m,f}$ be this live set. Afterwards, in each round, b_m tries to complete a simulation step of $p_m(t)$ and, if successfully completed, changes $p_m(t)$ in a round robin manner among $S_{m,f}$. Two cases may arise:

– If $p_m(t)$ never stabilizes, then the set of processes with an infinite number of simulated steps includes $S_{m,f}$. By Lemma 4, every other live BG simulator with a smaller identifier computes the same value of W at the end of round $m+1$ (of the loop at lines 8–11). Thus, after the $S_{m,f}$ is selected by b_m, as $S_{m,f}$ is valid, every BG simulator will select a subset of $S_{m,f}$ for its window value in every round. Moreover, by Lemma 3, these BG simulators will always find valid live sets to select, and so they will eventually simulate only processes in $S_{m,f}$. Thus, the set of processes with infinitely many simulated steps is equal to $S_{m,f}$, a live set intersecting with A_f.
– Otherwise, $p_m(t)$ eventually stabilizes on some $p_{m,f}$. Therefore, b_m attempts to complete a simulation step of $p_{m,f}$ infinitely often. Two sub-cases may arise:
  – Either $|S_{m,f}| = 1$ and, therefore, b_m is the only one live BG simulator performing simulation steps, and thus, the set of processes with an infinite number of simulated steps is equal to $S_{m,f}$, a live set intersecting with A_f.

- Otherwise, by Lemma 4, every live BG simulator with a smaller identifier eventually selects a window, and thus a live set (Lemma 3), which is a subset of $S_{m,f} \setminus \{p_{m,f}\}$. Thus every live BG simulator with a smaller identifier eventually selects processes to simulate distinct from $p_{m,f}$ and, thus, cannot block b_m infinitely often—a contradiction.

Lemma 6. *If $\mathcal{A}$ is fair, then any task T solvable in the $\mathcal{A}$-model is solvable in the $\alpha_{\mathcal{A}}$-model.*

Proof. Let us assume that it is not the case: there exists a task T and a fair adversary $\mathcal{A}$ such that T is solvable in the adversarial $\mathcal{A}$-model but not in the $\alpha_{\mathcal{A}}$-model. As every finite run of the $\mathcal{A}$-model can be extended to and $\mathcal{A}$-compliant run, the simulated algorithm can only provide valid outputs to the simulated processes. Thus, it can only be the case that a correct process is not provided with a task output, i.e., belongs to A_f.

Therefore, by Lemma 5, the simulation provides an $\mathcal{A}$-compliant run, i.e., the set of processes with an infinite number of simulated steps is a live set. As the run is $\mathcal{A}$-compliant then each process p with an infinite number of simulated steps is eventually provided with a task output and thus $p_{MEM}[p]$ is set to $\top$. Thus, they cannot belong to A_f — a contradiction.

Combining Corollary 1 and Lemma 6 we obtain the following result:

Theorem 6. *For any fair adversary $\mathcal{A}$, the adversarial $\mathcal{A}$-model and the $\alpha_{\mathcal{A}}$-model are equivalent regarding task solvability.*

6 Agreement Functions Do not Always Tell it All

We observe that agreement functions are not able to characterize the task computability power of *all* models. In particular there are non-fair adversaries not captured by their agreement functions.

Consider for example the adversary $\mathcal{A} = \{\{p_1\}, \{p_2, p_3\}, \{p_1, p_2, p_3\}\}$. It is easy to see that $setcon(\mathcal{A}) = 2$, but that $setcon(\mathcal{A}|_{\Pi,\{p_2,p_3\}}) = 1$ which is strictly smaller than $\min(|\{p_2, p_3\}|, setcon(\mathcal{A})) = 2$. Therefore, $\mathcal{A}$ is non-fair.

Consider the task $Cons_{2,3}$ consisting in consensus among p_2 and p_3: every process in $\{p_2, p_3\}$ proposes a value and every correct process in $\{p_2, p_3\}$ decides a proposed value, so that p_2 and p_3 cannot decide different values. $Cons_{2,3}$ is solvable in the adversarial $\mathcal{A}$-model: every process in $\{p_2, p_3\}$ simply waits until p_2 writes its proposed value and decides on it. Indeed, this protocol solves $Cons_{2,3}$ in the $\mathcal{A}$-model as if p_3 is correct, p_2 is also correct.

The agreement function of $\mathcal{A}$, $\alpha_{\mathcal{A}}$, is equal to 0 for $\{p_2\}$ or $\{p_3\}$, to 2 for $\{p_1, p_2, p_3\}$, and to 1 for all other values. It is easy to see that $\alpha_{\mathcal{A}}$ only differs from α_{1-res}, the agreement function of the 1-resilient adversary, for $\{p_1\}$ where $\alpha_{\mathcal{A}}(\{p_1\}) = 1 > \alpha_{1-res}(\{p_1\}) = 0$. Therefore, $\forall P \subseteq \Pi, \alpha_{\mathcal{A}}(P) \geq \alpha_{1-res}(P)$, and thus any task solvable in the $\mathcal{A}$-model is solvable in the 1-resilient model.

The impossibility of solving such a task 1-resiliently can be directly derived from the characterization of task solvable t-resiliently from [8]. Indeed, let p_1

wait for some process to output in order to decide the same value. Processes p_2 and p_3 use the ability to solve consensus among themselves to output a unique value. As there are two correct processes in the system, p_2 or p_3 will eventually terminate and thus p_1 will not wait indefinitely. This gives a 3-process 1-resilient consensus algorithm—a contradiction [7,21]. Thus, the $\mathcal{A}$-model is not equivalent with the $\alpha_{\mathcal{A}}$-model, even though they have the same agreement function.

7 Related Work

Adversarial models were introduced by Delporte et al. in [6]. With respect to colorless tasks, Herlihy and Rajsbaum [17] characterized a class *superset-closed* [19] adversaries (closed under the superset operation) via their minimal core sizes. Still with respect to colorless tasks, Gafni and Kuznetsov [12] derived a characterization of general adversary using its *consensus power* function *setcon*. A side result of this present paper is an extension of the characterization in [12] to any (not necessarily colorless) tasks.

Taubenfeld [23] introduced the notion of symmetric progress conditions, equivalent to our symmetric adversaries.

The BG simulation establishes equivalence between t-resilience and wait-freedom with respect to task solvability [3,4,8]. Gafni and Guerraoui [10] showed that if a model allows for solving k-set consensus, then it can be used to simulate a *k-concurrent* system in which at most k processes are concurrently invoking a task. In our simulation, we use the fact that a model M associated to an agreement function α_M allows to solve an α-adaptive set consensus, using the technique proposed in [5], which enables a composition of the ideas in [3,4,8] and [10]. Running BG simulation on top of a k-concurrent system, we are able to derive the equivalence between fair adversaries and their corresponding α-models.

8 Concluding Remarks

By Theorem 6, task computability of a fair adversary $\mathcal{A}$ is *characterized* by its agreement function α: a task is solvable with $\mathcal{A}$ if and only if it is solvable in the α-model. The result implies characterizations of superset-closed [16,19] and symmetric [23] adversaries and, via the equivalence result established in [9], the model of k-concurrency.

As a corollary, for all models M and M' characterized by their agreements functions, such that $\forall P \in \Pi, \alpha_{M'}(P) \geq \alpha_M(P)$, we have that M is *stronger* than M', i.e., the set of tasks solvable in M contains the set of tasks solvable in M'. In particular, if the two agreement functions are equal, then M and M' solve exactly the same sets of tasks. Note that if a model M is characterized by its agreement function α, then it belongs to the weakest equivalence class among the models whose agreement function is α.

An intriguing open question is therefore how to precisely determine the scope of the approach based on agreement functions and if it can be extended to capture larger classes of models.

References

1. Afek, Y., Attiya, H., Dolev, D., Gafni, E., Merritt, M., Shavit, N.: Atomic snapshots of shared memory. J. ACM **40**(4), 873–890 (1993)
2. Borowsky, E., Gafni, E.: Generalized FLP impossibility result for t-resilient asynchronous computations. In: STOC, pp. 91–100. ACM Press, May 1993
3. Borowsky, E., Gafni, E.: Immediate atomic snapshots and fast renaming. In: PODC, pp. 41–51. ACM Press, New York (1993)
4. Borowsky, E., Gafni, E., Lynch, N.A., Rajsbaum, S.: The BG distributed simulation algorithm. Distrib. Comput. **14**(3), 127–146 (2001)
5. Delporte-Gallet, C., Fauconnier, H., Gafni, E., Kuznetsov, P.: Wait-freedom with advice. Distrib. Comput. **28**(1), 3–19 (2015)
6. Delporte-Gallet, C., Fauconnier, H., Guerraoui, R., Tielmann, A.: The disagreement power of an adversary. Distrib. Comput. **24**(3–4), 137–147 (2011)
7. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. J. ACM **32**(2), 374–382 (1985)
8. Gafni, E.: The extended BG-simulation and the characterization of t-resiliency. In: STOC, pp. 85–92 (2009)
9. Gafni, E., Guerraoui, R.: Simulating few by many: limited concurrency = set consensus. Technical report (2009). http://web.cs.ucla.edu/eli/eli/kconc.pdf
10. Gafni, E., Guerraoui, R.: Generalized universality. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 17–27. Springer, Heidelberg (2011). doi:10.1007/978-3-642-23217-6_2
11. Gafni, E., He, Y., Kuznetsov, P., Rieutord, T.: Read-write memory and k-set consensus as an affine task. In: OPODIS (2016). Technical report. https://arxiv.org/abs/1610.01423
12. Gafni, E., Kuznetsov, P.: Turning adversaries into friends: simplified, made constructive, and extended. In: Lu, C., Masuzawa, T., Mosbah, M. (eds.) OPODIS 2010. LNCS, vol. 6490, pp. 380–394. Springer, Heidelberg (2010). doi:10.1007/978-3-642-17653-1_28
13. Gafni, E., Kuznetsov, P.: Relating L-resilience and wait-freedom via hitting sets. In: ICDCN, pp. 191–202 (2011)
14. Herlihy, M.: Wait-free synchronization. ACM Trans. Prog. Lang. Syst. **13**(1), 123–149 (1991)
15. Herlihy, M., Kozlov, D.N., Rajsbaum, S.: Distributed Computing Through Combinatorial Topology. Morgan Kaufmann, Burlington (2014)
16. Herlihy, M., Rajsbaum, S.: The topology of shared-memory adversaries. In: PODC, pp. 105–113 (2010)
17. Herlihy, M., Rajsbaum, S.: Simulations and reductions for colorless tasks. In: PODC, pp. 253–260 (2012)
18. Herlihy, M., Shavit, N.: The topological structure of asynchronous computability. J. ACM **46**(2), 858–923 (1999)
19. Kuznetsov, P.: Understanding non-uniform failure models. Bull. EATCS **106**, 53–77 (2012)

20. Kuznetsov, P., Rieutord, T.: Agreement functions for distributed computing models. CoRR, abs/1004.4701 (2017)
21. Loui, M.C., Abu-Amara, H.H.: Memory requirements for agreement among unreliable asynchronous processes. Adv. Comput. Res. **4**, 163–183 (1987)
22. Saks, M., Zaharoglou, F.: Wait-free k-set agreement is impossible: The topology of public knowledge. SIAM J. Comput. **29**, 1449–1483 (2000)
23. Taubenfeld, G.: The computational structure of progress conditions. In: Lynch, N.A., Shvartsman, A.A. (eds.) DISC 2010. LNCS, vol. 6343, pp. 221–235. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15763-9_23