# Turning Adversaries into Friends: Simplified, Made Constructive, and Extended

Eli Gafni[1] and Petr Kuznetsov[2]

[1] Computer Science Department, UCLA
[2] Deutsche Telekom Laboratories/TU Berlin

**Abstract.** A *liveness contract* is an agreement between the specifier of a system and a task to solve, and the programmer who makes her living by delivering protocols. In a shared-memory system, a liveness contract specifies infinite suffixes of executions in which the programmer is required to solve a distributed task. If the behavior of the system does not comply with the specification, no output is required. A convenient way to describe a large class of liveness contracts was recently proposed by Delporte et al. For a system $\Pi$ of $n$ processes, an *adversary* is a set $\mathcal{A}$ of subsets of $\Pi$. The system is required to make progress only in executions in which the set of correct processes is in $\mathcal{A}$.

Given an adversary $\mathcal{A}$ and a task $T$, should the programmer sign the contract? Can she deliver?

In this paper, we give a very simple resolution of this question for colorless tasks that contrasts with more involved arguments of the original paper of Delpote et al. More importantly, our resolution is constructive — it tells the programmer how to use $\mathcal{A}$ to solve $T$, when it is solvable.

Our framework naturally generalizes to systems enriched with more powerful objects than read-write registers. We determine necessary and sufficient conditions for an adversary $\mathcal{A}$ to solve consensus using $j$-process consensus objects and read-write registers, which resolves an open question raised recently by Taubenfeld.

## 1 Introduction

Distributed computing is about overcoming asynchrony and failures. Wait-free system, a system where we make no assumptions about some synchrony or correctness of some processes, can solve only few interesting tasks. To solve more interesting tasks, we should make more assumptions about the system behavior.

Recently [7], Delporte et al. proposed a class of assumptions that they called *adversaries.* In their view, an *adversary* controls *sets* of processes that may fail in a given execution, regardless of the time when they fail. Put differently, an adversary is defined as a collection $\mathcal{A}$ of sets of processes, and they only consider executions where some element in $\mathcal{A}$ is exactly the set of correct processes. Following [7], in this paper, we explore the ability of such adversaries to enhance solvability of *distributed tasks*, defined in terms of inputs the processes receive, outputs the processes produce, and a binary relation that maps inputs to the sets of possible outputs.

Why are adversaries interesting to look at? In a shared-memory system, it is straightforward to ensure that the outputs a protocol solving a task provides are always correct [10,22]. However, ensuring that the outputs can indeed be eventually produced is sometimes tricky. Therefore, an adversary can be viewed as a *liveness* property, that specifies under which condition the correct processes are expected to produce outputs.

Given a task $T$ and an adversary $\mathcal{A}$, can the task be solved? It is known that this question is in general undecidable [12,16], and Delporte et al. [7] reduced it to the question of *$k$-resilient* solvability, i.e., assuming an adversary that consists of all sets of $n-k$ or more processes, restricted to *colorless* tasks (also called convergence tasks [4]). The resolution proposed in [7] is not easy to follow, and moreover, it is not constructive — it does not tell the programmer of the protocol how to use the adversary $\mathcal{A}$ to solve a colorless task $T$, when $T$ is solvable.

In this paper, we give a simpler constructive resolution of the question.

How to use such a condition $\mathcal{A}$ is shown below on a "back of an envelope" example. The paper is just a detailed elaboration of the envelope.

Consider a system of four processes, $p$, $q$, $r$, and $s$, and consider the *obstruction-free* adversary $\mathcal{A}_{OF}$ defined as the set of all singletons $\{\{p\},\{q\},\{r\},\{s\}\}$.[1] Thus, $\mathcal{A}_{OF}$ stipulates that an algorithm solving a task is only required to make progress if some process is eventually forever running solo. It is immediate that $\mathcal{A}_{OF}$ allows for solving consensus [8]: a sequence of commit-adopt [10] instances, where the first instance is called with the input value, every next instance is called with the value returned by the previous instance, and the first committed value is returned. Thus, the *set consensus power* [13] of $\mathcal{A}_{OF}$, i.e., the smallest $k$ such that $k$-set agreement can be solved in the presence of $\mathcal{A}_{OF}$, is 1.

In general, under which condition an adversary $\mathcal{A}$ allows for solving consensus? In this paper, we show that $\mathcal{A}$ provides consensus if for all $S \in \mathcal{A}$ all subsets of $S$ that belong to $\mathcal{A}$ have a non-empty intersection. Intuitively, a correct process in the intersection acts as a leader in a classical eventual leader-based consensus protocol [6].

What if we weaken $\mathcal{A}_{OF}$ by adding one more allowed set of correct processes: $\mathcal{A}'_{OF} = \{p, q, r, s, pqrs\}$: either some process eventually runs solo, or no process fails? What is the set consensus power of $\mathcal{A}'_{OF}$? It is easy to observe that $\mathcal{A}'_{OF}$ allows for solving 2-set agreement: As $\{p,q,r,s\}$ can do consensus and $\{pqrs\}$ can do consensus, run both in parallel.

But can we solve consensus with $\mathcal{A}'_{OF}$? The answer is "no". Indeed, by assuming the converse, that there exists a read-write protocol $P$ that, under $\mathcal{A}'_{OF}$, solves consensus, we can derive a read-write consensus protocol for 2 processes violating [9,20], as follows.

We take 2 simulators $s_0$ and $s_1$ that mimic a run of $P$ in $\mathcal{A}'_{OF}$ using *BG-agreement* [3,4] to make sure that every step in $P$ is simulated consistently across the simulators. Initially, $s_0$ tries to start $P$ with all 0s and $s_1$ with all 1s as input values of $p$, $q$, $r$, and $s$. Recall that BG-agreement is allowed to block forever if one of the simulators fails in the middle of it. Steps of $P$ are simulated

[1] For brevity, we simply write $\{p,q,r,s\}$ in the following.

in a round-robin fashion on the codes of $p, q, r, s$ until a decision value is output in the simulated run or one code blocks because of an unresolved BG-agreement. This unresolved agreement may block the code of either $p$ or $q$ but not both (we do not need other singleton sets in $\mathcal{A}'_{OF}$ for the simulation). Say the code of $p$ is blocked. Thus, a live simulator, say $s_0$, picks $q$ and simulates just it, as long as the unresolved BG-agreement on the code of $p$ stays unresolved. In case it does resolve, $s_0$ resumes again to continue round-robin on the codes of $p$, $q$, $r$ and $s$. Thus, if no BG-agreement remains unresolved forever, the codes of all processes $p, q, r, s$ accept infinitely many simulated steps. Otherwise, an eventually solo execution of $p$ or $q$ is simulated. Thus, the correct processes in the simulated execution of $P$ are $\{pqrs\}$ and $P$ should output, else, $p$ or $q$ continue forever solo, and $P$ again should output. Thus, set consensus number of $\mathcal{A}'_{OF}$ is 2.

Further, imagine that we want to boost the power of $\mathcal{A}'_{OF} = \{pqrs, p, q, r, s\}$ using objects that solve consensus among two or more processes [18]. A simple extension of the argumentation above shows that $j$-process consensus objects are necessary and sufficient for solving consensus with $\mathcal{A}'_{OF}$, where $j$ is the maximum of the hitting set size of $S$ in $\mathcal{A}'_{OF}$, over all $S \in \mathcal{A}'_{OF}$.[2] In our case, the hitting set size of $\{pqrs\}$ in $\mathcal{A}'_{OF}$ is 4 and, thus, we need 4-process consensus. But if we restrict ourselves to the adversary $\mathcal{A}''_{OF} = \{pqrs, p, q\}$, then we would need only 2-process consensus.

In this paper, we generalize the observations made above for the special case of $\mathcal{A}'_{OF}$, to any adversary of [7]. We introduce an alternative definition of the set consensus power of an adversary $\mathcal{A}$, a positive one as we view adversaries as helpful entities: The smallest $k$ such that $k$-set agreement can be solved in the presence of $\mathcal{A}$.[3] Then we provide a simple characterization of the set consensus power of an adversary. Our characterization is self-consistent and, unlike the definition given in [7] does not involve reductions to $k$-resilience.

Our simulations allow us to derive a more general result: every two adversaries that have the same set consensus power $k$ agree on the set of colorless tasks they are able to solve. Informally, colorless tasks allow every process to adopt an input or output value from any other participating process. Thus, every colorless task is equivalent to some level of set agreement. Our technique is based on simple direct simulations and it does not employ failure detector-based reductions of [7].

Recently, following [7], Herlihy and Rajsbaum [17], and a concurrent paper [14], considered a restricted set of adversaries that are closed under superset: for every $S \in \mathcal{A}$, every its superset $S' \subseteq \Pi$ is also in $\mathcal{A}$. Informally, such adversaries say what sets of processes are expected to be live, but do not say which sets of processes are supposed to fail. By employing elements of modern combinatorial topology, [17] derives the characterization of colorless tasks with respect to superset-closed adversaries. In [14], we derive this result employing a very simple

[2] The hitting set size of $S$ in $\mathcal{A}$ is the size of the minimum-cardinality subset of $S$ that meets every element of $\mathcal{A}$ subset of $S$.

[3] More precisely, [7] talks about the *disagreement power* of $\mathcal{A}$ which is the largest $d$ such that $d$-set agreement cannot be solved in the presence of $\mathcal{A}$. The disagreement power of $\mathcal{A}$ is the set consensus power of $\mathcal{A}$ minus one.

simulation algorithm, a precursor to the one in this paper. Indeed, our paper [14] generalizes naturally to unrestricted adversaries. We suspect that doing the same with [17] is a major undertaking.

Imbs et al. [18] and Taubenfeld [21] considered special classes of *progress conditions* in the context of shared-memory systems enriched with consensus objects shared by subsets of $j < n$ processes. We observe that, with respect to colorless tasks, progress conditions of [18,21] are in fact special cases of adversaries [7]. Then we reconstruct the characterization of the power of *leveled* adversaries [21] to solve consensus using $j$-process consensus objects and extend the result to general adversaries, closing a question left open in [21].

This paper provides therefore a purely algorithmic characterization of adversaries that neither involves "esoteric" (for the distributed community) topological arguments, as [17], nor does it rely upon weakest failure detector results, as [7]. Neither it is stuck in the 80's resorting at these days and age to bivalency arguments [18,21]. Overall, this supports the contention that beyond dealing with sub-consensus tasks [15], topology is the analogue of plowing your field with an F16 fighter rather than a simple tractor — the F16 may do the job faster, but it takes years to master and you are liable to crash because of the low altitude flying and sharp turns plowing requires. Bivalency is the Ox. You can go with it so far but no more. The golden path, between bivalency and topology, is the BG simulation [3,4]. A tractor - simple, yet powerful and exactly suitable for the job.

The rest of the paper is organized as follows. Section 2 briefly describes our system model. Section 3 defines the notion of the power of a general adversary. Section 4 presents our characterization of adversaries with respect to colorless tasks. Section 5 extends our characterization to other computing models. Section 6 overviews the related work and concludes the paper.

## 2 Model

We adopt the conventional read-write shared memory model and only describe necessary details.

*Processes and objects.* We consider a distributed system composed of a set $\Pi$ of $n$ processes $\{p_1, \ldots, p_n\}$ $(n \geq 2)$. Processes communicate by applying atomic operations on a collection of *shared objects*. In the most of this paper, we assume that the shared objects are registers that export only atomic read-write operations. The shared memory can be accessed using atomic snapshot operations [1]. An *execution* is a pair $(I, \sigma)$ where $I$ is an initial state and $\sigma$ is a sequence of process ids. A process that takes at least one step in an execution is called *participating.* A process that takes infinitely many steps in an execution is said to be *correct*, otherwise, the process is *faulty.*

*Distributed tasks.* A *task* is defined through a set $\mathcal{I}$ of input $n$-vectors (one input value for each process, where the value is $\perp$ for a non-participating process), a set $\mathcal{O}$ of output $n$-vectors (one output value for each process, $\perp$ for non-terminated processes) and a total relation $\varDelta$ that associates each input vector with a set of possible output vectors. A protocol *wait-free* solves a task $T$ if in every execution, every correct process eventually outputs, and all outputs respect the specification of $T$.

*Correct sets and adversaries.* The *correct set* of an execution $e$, denoted $correct(e)$ is the set of processes that appear infinitely often in $e$. An *adversary* [7] is a collection of subsets of $\varPi$. We say that an execution $e$ is $\mathcal{A}$-*compliant* if $correct(e) \in \mathcal{A}$.

*Hitting sets.* Given a set system $(\varPi, \mathcal{A})$ where $\mathcal{A}$ is a set of subsets of $\varPi$, a set $H \subseteq \varPi$ is a *minimum cardinality hitting set of* $(\varPi, \mathcal{A})$ if it is a minimum cardinality subset of $\varPi$ that meets every set in $\mathcal{A}$. The *hitting set size* of $(\varPi, \mathcal{A})$, i.e., the size of a minimum cardinality hitting set of $(\varPi, \mathcal{A})$, is denoted by $h(\mathcal{A})$. Obviously, if $h(\mathcal{A}) = 1$, then $\forall \mathcal{A}' \subseteq \mathcal{A}$, $\mathcal{A} \neq \emptyset$, $h(\mathcal{A}') = 1$. Finding the hitting set size is NP-complete [19].

*Colorless tasks.* In *colorless* task (also called *convergence* tasks [4]) processes are free to use each others' input and output values, so the task can be defined in terms of input and output *sets* instead of vectors.

Formally, let $val(U)$ denote the set of non-$\bot$ values in a vector $U$. In a colorless task, for all input vectors $I$ and $I'$ and all output vectors $O$ and $O'$, such that $(I, O) \in \varDelta$, $val(I') \subseteq val(I)$, $val(O') \subseteq val(O)$, we have $(I', O) \in \varDelta$ and $(I, O') \in \varDelta$.

*The Commit-Adopt protocol.* The *commit-adopt* abstraction (CA) [10] exports one operation *propose*$(v)$ that returns $(commit, v')$ or $(adopt, v')$, for $v', v \in V$, and guarantees that (a) every returend value is a proposed value, (b) if only one value is proposed then this value must be committed, (c) if a process commits on a value $v$, then every process that returns adopts $v$ or commits $v$, and (d) every correct process returns. The commit-adopt abstraction can be implemented wait-free.

*The BG-simulation technique.* BG-simulation is a technique by which $k + 1$ processes $s_1, \ldots, s_{k+1}$, called *simulators*, can wait-free simulate a $k$-resilient execution of any asynchronous $n$-process protocol $A$ [3,4]. The simulation guarantees that each simulated step of every process $p_j$ is either agreed on by all simulators, or one less simulator participates further in the simulation for each step which is not agreed on.

The central building block of the simulation is the *BG-agreement* protocol. The protocol is safe—every decided value was previously proposed, and no two different values are decided— but not necessarily live. If a simulator slows down in the middle of BG-agreement, the protocol's execution at other correct simulators may "block" until the slow simulator finishes the protocol. If the simulator is faulty, no simulator is guaranteed to decide.

Suppose the simulation tries to promote $m > k$ codes in a fair (e.g., round-robin) way. As long there is a live simulator, at least $m - k$ simulated processes accept infinitely many steps of $A$ in the simulated execution.

## 3 Set Consensus Power of $\mathcal{A}$

Let $\mathcal{A}$ be an adversary and take any set $S \subseteq P$. $\mathcal{A}_S$ denotes the adversary that consists of $S$ and all elements of $\mathcal{A}$ that are subsets of $S$. E.g., for $\mathcal{A} = \{pq, qr, q, r\}$ and $S = qr$, $\mathcal{A}_S = \{qr, q, r\}$.

Let $S \in \mathcal{A}$ and take $a \in S$. Then $A_{S,a}$ denotes the adversary that consists of all elements of $\mathcal{A}_S$ that *do not* include $a$. E.g., for $\mathcal{A} = \{pq, qr, q, r\}$, $S = qr$, and $a = q$, $\mathcal{A}_{S,a} = \{r\}$. Note that if the hitting set size of $(\Pi, \mathcal{A}_S)$ is 1, then for every $a \in S$ that meets every set in $\mathcal{A}_S$, we have $\mathcal{A}_{S,a} = \emptyset$. Thus:

**Lemma 1.** *$h(\mathcal{A}_S) > 1$ if and only if $\forall a \in S :\ \mathcal{A}_{S,a} \neq \emptyset$.*

**Definition 1.** *The quantity denoted $setcon(\mathcal{A})$, which will later be shown to be the set consensus power of $\mathcal{A}$, is defined as follows:*

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

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

We say that $S \in \mathcal{A}$ is *proper* if it is not a subset of any other element in $\mathcal{A}$. Let *proper*$(\mathcal{A})$ denote the set of proper elements in $\mathcal{A}$. Note that since for all $S' \subset S$, $\min_{a\in S'} setcon(\mathcal{A}_{S',a}) \leq \min_{a\in S} setcon(\mathcal{A}_{S,a})$, we can replace $S \in \mathcal{A}$ with $S \in proper(\mathcal{A})$ in Definition 1.

For example, for $\mathcal{A} = \{pqr, pq, pr, p, q, r\}$, we have $setcon(\mathcal{A}) = 2$: for $S = pqr$ and $a = p$, we have $\mathcal{A}_{S,a} = \{q, r\}$ and $setcon(\mathcal{A}_{S,a}) = 1$. Intuitively, in an execution where the correct set belongs to $\mathcal{A} - \mathcal{A}_{S,a} = \{pqr, pq, pr, p\}$, process $p$ can act as a leader for solving consensus. If the execution's correct set belongs to $\mathcal{A}_{S,a} = \{q, r\}$ (either $q$ or $r$ eventually runs solo) then $q$ and $r$ can solve consensus using an obstruction-free algorithm. Running the two algorithms in parallel, we obtain a solution to 2-set agreement. The reader can easily verify that any other choice of $a \in pqr$ results in larger values of $setcon(\mathcal{A}_{S,a})$.

As another example, consider the $t$-resilient adversary $\mathcal{A}_{t\text{-}res} = \{S \subseteq \Pi, |S| \geq n - t\}$. It is easy to verify recursively that $setcon(\mathcal{A}_{t\text{-}res}) = t + 1$: at each level $1 \leq j \leq t + 1$ of recursion we consider a set $S$ of $n - j + 1$ elements, pick up a process $p \in S$ and delegate the set of $n-j$ processes that do not include $p$ to level $j+1$. At level $t+1$ we get a set of size $n-t$ and stop. Thus, $setcon(\mathcal{A}_{t\text{-}res}) = t+1$.

More generally, consider *superset-closed* adversaries $\mathcal{A}$ [14]: for every $S \in \mathcal{A}$, every its set $S'$ such that $S \subseteq S' \subseteq \Pi$ is also in $\mathcal{A}$.

**Theorem 1.** *For all superset-closed adversaries $\mathcal{A}$, $setcon(\mathcal{A}) = h(\mathcal{A})$.*

*Proof.* By definition, for $\mathcal{A} = \emptyset$, $setcon(\mathcal{A}) = h(\mathcal{A}) = 0$. By induction, suppose that for all $0 \leq j < k$ and all superset-closed adversaries $\mathcal{A}'$ with $h(\mathcal{A}') = j$, we have $setcon(\mathcal{A}') = j$.

Consider a superset-closed adversary $\mathcal{A}$ such that $h(\mathcal{A}) = k$. The only proper element of $\mathcal{A}$ is the whole set of processes $\Pi$. Thus, $setcon(\mathcal{A}) = \min_{a\in\Pi}$

```
Initially:
    $\forall j$, $\mathcal{A}^j = \emptyset$

$PartitionAdv(\mathcal{A})$
1   $partition(\mathcal{A}, 1)$

$partition(\mathcal{B}, j)$
2   while $\mathcal{B} \neq \emptyset$ do
3       $(B, b) :=$ args $\max_{S \in proper(\mathcal{B})} \min_{a \in S} setcon(\mathcal{B}_{S,a})$
4       $\mathcal{A}^j := \mathcal{A}^j \cup (\mathcal{B}_B - \mathcal{B}_{B,b})$
5       $partition(\mathcal{B}_{B,b}, j+1)$
6       $\mathcal{B} := \mathcal{B} - \mathcal{B}_B$
```

**Fig. 1.** Partitioning an adversary with $setcon = k$

$setcon(\mathcal{A}_{\Pi,a}) + 1$. Since $h(\mathcal{A}) = k$, by removing all elements that include $a$ we obtain an adversary $\mathcal{A}_{\Pi,a}$ such that $h(A_{\Pi,a}) \geq k-1$. (Otherwise, there is a hitting set of $\mathcal{A}$ of size less than $k$.) By picking up $a$ in a hitting set of $\mathcal{A}$ of size $k$ we obtain, by the induction hypothesis, $h(\mathcal{A}_{\Pi,a}) = setcon(\mathcal{A}_{\Pi,a}) = k-1$ and, thus, $setcon(\mathcal{A}) = k$.

For general adversaries, for convenience, we first consider the special case of set consensus power 1. Definition 1 and Lemma 1 imply:

**Lemma 2.** *$setcon(\mathcal{A}) = 1$ if and only if $\forall S \in \mathcal{A}$, $h(\mathcal{A}_S) = 1$*

We show below that the elements of every adversary $\mathcal{A}$ with $setcon(\mathcal{A}) = k$ can be split into $k$ sub-adversaries such that *setcon* of every sub-adversary is 1.

**Theorem 2.** *Let $\mathcal{A}$ be an adversary, and let $setcon(\mathcal{A}) = k$. Then there exists $\mathcal{A}^1, \ldots, \mathcal{A}^k$, a partitioning of $\mathcal{A}$, such that, for all $1 \leq j \leq k$, $setcon(\mathcal{A}^j) = 1$.*

*Proof.* Let $\mathcal{A}$ be an adversary such that $setcon(k)$. Our goal is to partition $\mathcal{A}$ into $k$ sub-adversaries $\mathcal{A}^1, \ldots, \mathcal{A}^k$ such that $\forall j = 1, \ldots, k$, $\forall S \in \mathcal{A}^j$, $h(\mathcal{A}^j_S) = 1$. We construct the desired partitioning of $\mathcal{A}$ using procedure $PartitionAdv(\mathcal{A})$ described in Figure 1.

Suppose that at a level $j \in \{1 \ldots, k\}$, we have $\mathcal{B} \subseteq \mathcal{A}$, a set of elements of $\mathcal{A}$ which were not yet assigned a level. We recursively assign elements of $\mathcal{B}$ to levels $j$ or more using procedure $partition(\mathcal{B}, j)$.

Let $B$ and $b \in B$ max-minimize $setcon(\mathcal{B}_{S,a})$ over all $S \in \mathcal{B}$ and $a \in S$ (ties broken deterministically). Then we assign $\mathcal{B}_B - \mathcal{B}_{B,b}$ to level $j$ and recursively partition $\mathcal{B}_{B,b}$ on level $j+1$ by calling $partition(\mathcal{B}, j+1)$. When we are done, i.e., all elements of $\mathcal{B}_B$ are assigned to levels $j$ or more, we proceed to assigning the remaining elements of $\mathcal{B} - \mathcal{B}_B$ to level $j$ or more, and we repeat this until we exhaust $\mathcal{B}$.

We observe first that this procedure recursively explores all elements in $\mathcal{A}$, i.e., every element $S \in \mathcal{A}$ is assigned to some level $j \geq 1$. By construction, each

$A^j$ only contains sets $S$ with the hitting set size 1, namely, all $S' \in \mathcal{A}_B$ that contain $b$ (chosen in line 3). All other elements of $\mathcal{A}_S$ are delegated to levels $j+1$ or more.

By Definition 1 and Lemma 1, if we start from the whole set $\mathcal{A}$ at level 1 (line 1), and $\mathit{setcon}(\mathcal{A}) = k$, exactly levels $1, \ldots, k$ are populated.

Finally, by construction, for all $j$ and all $S \in \mathit{proper}(\mathcal{A}^j)$, $h(\mathcal{A}^j_S) = 1$. By Lemma 2, for all $j$, $\mathit{setcon}(\mathcal{A}^j) = 1$.

Before we characterize the ability of adversaries to solve generic colorless tasks, we consider the special case of adversaries of $\mathit{setcon} = 1$.

**Theorem 3.** *If $\mathit{setcon}(\mathcal{A}) = 1$, then $\mathcal{A}$ solves consensus.*

*Proof.* Recall that if $\mathit{setcon}(\mathcal{A}) = 1$, then, by Lemma 2, $\forall S \in \mathcal{A}$, $h(\mathcal{A}_S) = 1$. The consensus algorithm is presented in Figure 2. This is a rotating coordinator-based algorithm inspired by the consensus algorithm by Chandra and Toueg [6].

The algorithm proceeds in rounds. In each round $r$, every process $p_i$ first tries to commit its proposal in a new instance of commit-adopt. If $p_i$ succeeds, then the committed value is written in the “decision” register $D$ and returned. Otherwise, $p_i$ adopts the returned value as its current estimate and writes it in $R_i$ equipped with the current round number $r$. Then $p_i$ takes snapshots of $\{R_1, \ldots, R_n\}$ until either a set $S \in \mathcal{A}$ reaches round $r$ or a decision value is written in $D$ (in which case the process returns the value from $D$). If no decision is taken yet, then $p_i$ checks if the coordinator of this round, $p_{r \bmod n}$, is in $S$. If so, $p_i$ adopts the value written in $R_{r \bmod n}$ and proceeds to the next round.

Safety of the algorithm follows from the properties of commit-adopt. Indeed, the first round in which some process commits on some value $v$ in line 14 locks the value for all subsequent rounds.

For liveness, suppose, by contradiction, that the algorithm never terminates in some $\mathcal{A}$-compliant execution $e$. Recall that we only consider executions in which some set in $\mathcal{A}$ is exactly the set of correct processes. Therefore, every correct process goes through infinitely many rounds.

Let $\bar{S} \in \mathcal{A}$ be the set of correct processes in $e$. After a round $r'$ when all processes outside $\bar{S}$ have failed, every element of $\mathcal{A}$ evaluated by a correct process in line 16 is a subset of $\bar{S}$. Finally, since the hitting set size of $\mathcal{A}_{\bar{S}}$ is 1, all these elements of $\mathcal{A}$ overlap on some correct process $p_j$.

Consider round $r = mn + j \geq r'$. In this round, $p_j$ not only belongs to all sets evaluated by the correct processes, but it is also the coordinator ($j = r \bmod n$). Thus, the only value that a process can propose to commit-adopt in round $r+1$ is the value previously written by $p_j$ in $R_j$. Thus, every process that returns from commit-adopt in round $r+1$ commits—a contradiction. Hence, no read-write protocol can solve $T'$ in the presence of $\mathcal{A}$.

Theorems 2 and 3 imply the following:

**Corollary 1.** *Let $\mathcal{A}$ be an adversary such that $\mathit{setcon}(\mathcal{A}) = k$. Then the adversary can solve $k$-set agreement.*

```
Shared variables:
    $D$, initially $\perp$
    $R_1, \ldots, R_n$, initially $\perp$

propose(v)
7   $est := v$
8   $r := 0$
9   $S := P$
10  repeat
11      $r := r + 1$
12      $(flag, est) := CA_r.propose(v)$
13      if $flag = commit$ then
14          $D := est$; $return(est)$        {Return the committed value}
15      $R_i := (est, r)$
16      wait until $\exists S \in \mathcal{A}$, $\forall p_j \in S$: $R_j = (v_j, r_j)$ where $r_j \geq r$ or $D \neq \perp$
                                              {Wait until a set in $\mathcal{A}$ moves}
17      if $p_{r \bmod n+1} \in S$ then
18          $est := v_{r \bmod n+1}$        {Adopt the estimate of the current leader}
19  until $D \neq \perp$
20  $return(D)$
```

**Fig. 2.** Consensus with a "one-level" adversary $\mathcal{A}$, $setcon(\mathcal{A}) = 1$

*Proof.* First we apply Theorem 2 to partition $\mathcal{A}$ into $k$ classes $\mathcal{A}^1, \ldots, \mathcal{A}^k$ such that, for all $j = 1, \ldots, k$, $setcon(\mathcal{A}^j) = 1$. Then every process runs $k$ parallel consensus algorithms established by Theorem 3, one for each $\mathcal{A}^j$, proposing its input value in each of these consensus instances (the idea originally appeared in [2]). Since the set of correct processes in every $\mathcal{A}$-compliant execution belongs to some $\mathcal{A}^j$, at least one consensus instance returns. The process decides on the first such returned value. Moreover, at most $k$ different values are decided and each returned value was previously proposed.

The next section shows that no read-write protocol can solve $(k-1)$-set agreement under an adversary $\mathcal{A}$ such that $setcon(\mathcal{A}) = k$.

## 4 Characterizing Colorless Tasks

In this section, we show a more general result: the set of colorless tasks that can be solved with an adversary $\mathcal{A}$ with $setcon(\mathcal{A}) = k$ is exactly the set of colorless tasks that can be solved $(k-1)$-resiliently, but not $k$-resiliently. The proof is based on two simple applications of BG simulation [3,4].

First, we show that $\mathcal{A}$ solves every $(k-1)$-resiliently solvable colorless task $T$ by presenting an algorithm that, in every $\mathcal{A}$-compliant execution, simulates a $(k-1)$-resilient execution of a protocol solving $T$.

Second, we show that $\mathcal{A}$ cannot solve a colorless task $T'$ that is not $(k-1)$-resiliently solvable by presenting an algorithm that $(k-1)$-resiliently simulates any protocol that solves $T'$ in every $\mathcal{A}$-compliant execution.

```
Local variables:
    $B_1, \ldots, B_k$, initially $\perp$          {Set of currently simulated elements of $\mathcal{A}$}
    $b_1, \ldots, b_k$, initially $\perp$          {Set of currently blocked processes}
    $L$, initially 1          {The current level of simulation}

Code for every simulator $s_i$, $i = 1, \ldots, n$
    $B_1 :=$ the first element $S \in \mathit{proper}(\mathcal{A})$ such that $\mathit{setcon}(\mathcal{A}_S) = k$
            {In some deterministic order}
    $L := 1$
    repeat forever
        $\ell := 1$
        while $\ell < L$ and the current step of $b_\ell$ is still blocked do $\ell = \ell + 1$
        if $\ell < L$ then $L := \ell$          {Return to level $\ell$ if the step of $b_\ell$ is resolved}
        let $p_j \in B_L$ be the process with the least number of simulated steps
        run BG-agreement for the next step of $p_j$
        if the step of $p_j$ is blocked and $L < k$ then
            $b_L := p_j$
            $B_{L+1} :=$ the first set in $\mathit{proper}(\mathcal{A}_{B_L, b_L})$ with power $\geq k - L$
                        {Such a set exists, since $\mathit{setcon}(\mathcal{A}_{B_L}) \geq k - L + 1$}
            $L := L + 1$
```

**Fig. 3.** Simulating an $\mathcal{A}$-compliant execution

**Theorem 4.** *Let $\mathcal{A}$ be an adversary such that $\mathit{setcon}(\mathcal{A}) = k$ and $T$ be a colorless task. $\mathcal{A}$ solves $T$ if and only if $T$ is $(k-1)$-resiliently solvable.*

*Proof.* Let $\mathcal{A}$ be an adversary such that $\mathit{setcon}(\mathcal{A}) = k$.

Let $T$ be a colorless $(k-1)$-resiliently solvable task. By Corollary 1, $\mathcal{A}$ can implement $k$-set agreement. Then we apply the generic algorithm of [11] that solves every $(k-1)$-resilient colorless task using a solution to $k$-set agreement as a black box. Thus, $\mathcal{A}$ solves $T$.

For the other direction, suppose that $\mathcal{A}$ solves a colorless task $T'$ that is *not* solvable $(k-1)$-resiliently, and let *Alg* be the corresponding algorithm.

We describe below a simulation protocol (summarized in Figure 3) that allows $n$ simulators, $s_1, \ldots, s_n$, to $(k-1)$-resiliently simulate an $\mathcal{A}$-compliant execution of *Alg*.

Essentially, the protocol builds upon BG-simulation, except that the *order* in which steps of *Alg* is not fixed in advance. Instead, the order is determined online, based on the currently observed set of participating processes.

Let $B_1$ be an element of $\mathit{proper}(\mathcal{A})$ such that $\mathit{setcon}(\mathcal{A}_{B_1}) = k$ (by the definition of *setcon* such a set exists). Initially, every simulator proceeds by simulating steps of processes in $B_1$ in a round-robin fashion. If simulating a step blocks—some other simulator stopped in the middle of the BG agreement protocol of some process $b_1 \in B_1$—the simulator proceeds to simulating steps of the processes in $B_2$, the "next" not yet blocked element of $\mathcal{A}_{B_1}$ such that $\mathit{setcon}(\mathcal{A}_{B_2}) \geq k - 1$. Indeed, by Definition 1, for all $b \in B$, $\mathit{setcon}(\mathcal{A}_{B_1,b}) \geq$

$setcon(\mathcal{A}_{B_1}) - 1 = k - 1$. Thus, such a set $B_2 \in \mathcal{A}_{B_1,b_1}$ exists. The procedure is then repeated for $B_2$: steps of processes in $B_2$ are simulated as long as no process in $B_2$ is blocked. As soon as a blocked process $b_2$ is observed, the simulator proceeds to simulating $B_3$, an element of $\mathcal{A}_{B_2,b_2}$ that has consensus power at least $k - 2$, etc. Inductively, since $setcon(\mathcal{A}) = k$, if the simulation reaches level $k$, then $B_k \neq \emptyset$.

Every simulator periodically checks if some of the previously blocked agreements are resolved (line 25). If so, the simulator jumps back to the smallest level with a resolved agreement (line 26).

Note that, since every step of *Alg* is agreed upon using the BG-agreement protocol, the simulation constructs a correct execution of *Alg* [3,4]. Now we show that the produced execution is indeed $\mathcal{A}$-compliant, and thus *Alg* must terminate.

First, we observe that no line in the pseudo-code presented in Figure 3 is blocking. Thus, every correct simulator proceeds through infinitely many rounds in lines 24-29. Consider level $\ell$ and suppose that some correct process never observed the currently simulated step of $b_\ell$ being resolved (it is blocked forever by a faulty simulator). Since simulators explore the simulated sets in a deterministic order starting from level 1, every correct process eventually blocks on the same step of $b_\ell$.

Now let $\ell$ be the lowest level in which no step in $B_\ell$ is observed blocked forever. Since there are at most $k - 1$ faulty simulators, and a faulty simulator cannot block more than one simulated process, $\ell \leq k$. Thus, every correct process simulates infinitely many steps of $B_\ell$, and, eventually, every simulated step belongs to a process in $B_\ell$. By construction, $B_\ell \in \mathcal{A}$ and, thus, the simulated run of *Alg* is $\mathcal{A}$-compliant. Therefore, *Alg* must terminate in the simulated execution and we obtain a $(k - 1)$-resilient solution to $T'$ — a contradiction.

The *set consensus power* of an adversary $\mathcal{A}$ is the smallest $k$ such that $\mathcal{A}$ can solve $k$-set agreement. Theorem 4 implies:

**Corollary 2.** *The set consensus power of $\mathcal{A}$ is $setcon(\mathcal{A})$.*

By Theorem 1, determining $setcon(\mathcal{A})$ may boil down to determining the hitting set size of $(\Pi, \mathcal{A})$, and thus, by [19]:

**Corollary 3.** *Determining the set consensus power of an adversary is NP-complete.*

The *disagreement power* of an adversary $\mathcal{A}$ [7], denoted $d(\mathcal{A})$, is the largest $d$ such that $d$-set agreement *cannot* be solved in the presence of $\mathcal{A}$. By Corollary 2, $d(\mathcal{A}) = setcon(\mathcal{A}) - 1$.

## 5 Extension to Other Models

In a recent paper [18], Imbs et al. considered *asymmetric* progress conditions that allow for modeling different progress guarantees for different processes. An asymmetric progress condition associates each process $p_i$ with a set $\mathcal{P}_i$ of process subsets that contain $p_i$. Process $p_i$ is expected to make progress (e.g., output a value in a task solution) only if the current set of correct processes is in $\mathcal{P}_i$.

It is easy to see that with respect to the solvability of colorless tasks, the asymmetric progress conditions of [18] can be modeled as adversaries of [7]. Indeed, for each progress condition $\mathcal{P} = \{\mathcal{P}_1, \dots, \mathcal{P}_n\}$, we can construct an adversary $\mathcal{A}^{\mathcal{P}} = \cup_i \mathcal{P}_i$. Since to solve a colorless task, it is sufficient to make sure that at least one process decides, every $\mathcal{P}$-resilient solution to a colorless task implies an $\mathcal{A}^{\mathcal{P}}$-resilient solution, and vice versa.

**Observation 5.** *A colorless task $T$ is solvable with a progress condition $\mathcal{P}$ if and only if it is solvable with the adversary $\mathcal{A}^{\mathcal{P}}$.*

In an even more recent paper [21], Taubenfeld focused on a special case of *leveled* adversaries that only specify the *sizes* of correct sets. Such an adversary $\mathcal{L}$ can be specified as a sequence of number in $\{1, \dots, n\}$: for each $j \in \mathcal{L}$, the adversary contains all process sets of size $j$. The paper shows, among other things, that consensus can be solved with $\mathcal{L}$ using $j$-process consensus objects (i.e., objects that can solve consensus among up to $j$ processes) if and only if $j \geq \mathit{width}(\mathcal{L})$, where $\mathit{width}(\mathcal{L}) = \max(\mathcal{L}) - \min(\mathcal{L}) + 1$.

Note that $\mathit{width}(\mathcal{L})$ is exactly $h(\mathcal{L}_S)$ for any $S \in \mathit{proper}(\mathcal{L})$. Indeed, we need exactly $\mathit{width}(\mathcal{L})$ processes to meet every set of $\min(\mathcal{L})$ processes that is subset of an element of $\mathit{proper}(\mathcal{L})$ (a set of $\max(\mathcal{L})$ processes).

**Theorem 6.** *A leveled adversary $\mathcal{L}$ such that $\mathit{width}(\mathcal{L}) = k$ can wait-free solve consensus using $j$-process consensus objects if and only if $j \geq k$.*

*Proof.* (Sketch) $\mathcal{L}$ can solve consensus using $k$-process consensus and read-write registers as follows. As in the consensus algorithm in Figure 2, every process alternates between instances of commit-adopt and a leader-based reconciliation protocol. The first committed value is written in a decision register and returned. Instead of a single coordinator in a hitting set of size 1, we now select a "coordinator group" of size $k$. Thus, there are $n$ choose $k$ coordinator groups, and we place them in a deterministic order: $C_0, \dots, C_{\binom{n}{k}-1}$. Now process $p_i$ considers itself a coordinator of a round $r$ if $p_i \in C_{r \bmod \binom{n}{k}}$. Furthermore, every round $r$ is associated with a $k$-process consensus object $\mathit{cons}_r$ that can only be accessed by processes in $C_{r \bmod \binom{n}{k}}$.

In Figure 4, we give an update of lines 15-18 of the consensus algorithm in Figure 2, the rest of the algorithm remains unchanged. As in the proof of Theorem 3, eventually, there will be a round $r'$ when only a subset of processes of some $S \in \mathcal{L}$ of size $\max(\mathcal{L})$ take steps and $C_{r' \bmod \binom{n}{k}}$ is a hitting set of $\mathcal{L}_S$ (the adversary that consists of $S$ and all its subsets in $\mathcal{L}$). Thus, in round $r'$, every correct process $p_i$ will adopt the estimate value agreed upon by the processes in $C_{r' \bmod \binom{n}{k}}$: every element of $\mathcal{L}$ evaluated by $p_i$ in line 35 should include at least one process in $C_{r' \bmod \binom{n}{k}}$. Thus, every correct process accesses the instance of commit-adopt in round $r' + 1$ with the same value and decides.

```
if $p_i \in C_{r \bmod \binom{n}{k}}$ then $est := cons_r.propose(est)$
$R_i := (est, r)$
wait until $\exists S \in \mathcal{L}, \forall p_j \in S$: $R_j = (v_j, r_j)$ where $r_j \geq r$ or $D \neq \perp$
                                        {Wait until a set in $\mathcal{L}$ moves}
if $\exists p_j \in C_{r \bmod \binom{n}{k}} \cap S$ then
    $est := v_j$                  {Adopt the estimate of the current coordinator group}
```

**Fig. 4.** Solving consensus with $\mathcal{L}$ and $k$-process consensus objects: replacing lines 15–18 in Figure 2

Now, by contradiction, suppose that we can solve consensus with $\mathcal{L}$ using $(k-1)$-process consensus objects, and let *Alg* be the corresponding algorithm. We establish a contradiction by presenting a wait-free 2-process consensus algorithm.

It is straightforward to extend our simulation in Figure 3 to simulate a protocol that, in addition to read-write registers, uses $(k-1)$-process consensus objects. Indeed, let two simulators simulate steps of *Alg* of a set $B_1 \in \mathcal{L}$ of $\max(\mathcal{L})$ processes in a round-robin fashion. A simulator that fails while simulating a step that accesses a $(k-1)$-process consensus object can block a set $S$ of up to $k-1$ simulated codes that are in process of accessing this object. But since $h(\mathcal{L}_{B_1}) = k$, we still have at least one set $B_2$ of $\min(\mathcal{L})$ processes that are not blocked. By applying the logic used in the proof of Theorem 4, we obtain an $\mathcal{L}$-compliant execution of *Alg*. The simulated execution of *Alg* must terminate — a contradiction.

Theorem 6 is mildly surprising in the sense that the ability of $j$-consensus objects to boost the power of $\mathcal{L}$ to solve consensus has nothing to do with the exact structure of $\mathcal{L}$, but depends only on the size of the hitting set of $\mathcal{L}_S$ for some $S \in \mathcal{L}$ of the maximal size. Indeed, notice that our argumentation has nothing to do with "sequences" or "width," it only uses the hitting set size of $\mathcal{L}_S$ for $S \in proper(\mathcal{L})$. A straightforward extension of Theorem 6 resolves an open question raised in [21].

**Theorem 7.** *An adversary $\mathcal{A}$ can wait-free solve consensus using $j$-process consensus objects if and only if $j \geq \max_{S \in \mathcal{A}}(h(\mathcal{A}_S))$.*

## 6 Concluding Remarks

An adversary, as defined by Delporte et al. [7], is in fact a special case of an environment of [5] that determines which sets of processes are allowed to fail without specifying the timing of failures. Thus, we can rephrase the statement "task $T$ can be solved with adversary $\mathcal{A}$", as "task $T$ can be solved in environment $\mathcal{A}$ using the dummy failure detector". (The output of the dummy failure detector does not depend on the failure pattern.) It is shown in [13] that, with respect to colorless tasks, failure detectors can be split into $n$ equivalence classes, and each class $j$ agrees on the set of tasks it can solve: namely, tasks that can be solved $(j-1)$-resiliently and not $j$-resiliently. Therefore, by applying [13], we conclude that each adversary belongs to one of such equivalence class. This characterization is however a brute-force solution and it does not give us an explicit algorithm to compute the class to which a given adversary belongs.

The approach taken in [7] is based on a three-stage simulation. First, it is shown how an adversary can simulate any *dominating* adversary, where the domination is defined through involved recursive inclusion properties. Second, it is shown that every adversary that does not dominate the $k$-resilient adversary is strong enough to implement the anti-$\Omega_k$ failure detector that, in turn, can be used to solve $k$-set agreement [23]. Finally, it is shown that anti-$\Omega_k$ can be used to solve any colorless task that can be solved $k$-resiliently.

Instead, this paper proposes a self-consistent, constructive and simple characterization of general adversaries of [7], and sketches an extension of the characterization to models that use $j$-process consensus objects [18,21].

**Acknowledgment.** The first author is grateful to Gadi Taubenfeld for a few nights of pillow talk about [18,21].

## References

1. Afek, Y., Attiya, H., Dolev, D., Gafni, E., Merritt, M., Shavit, N.: Atomic snapshots of shared memory. Journal of the ACM 40(4), 873–890 (1993)
2. Afek, Y., Gafni, E., Rajsbaum, S., Raynal, M., Travers, C.: Simultaneous consensus tasks: A tighter characterization of set-consensus. In: Chaudhuri, S., Das, S.R., Paul, H.S., Tirthapura, S. (eds.) ICDCN 2006. LNCS, vol. 4308, pp. 331–341. Springer, Heidelberg (2006)
3. Borowsky, E., Gafni, E.: Generalized FLP impossibility result for t-resilient asynchronous computations. In: STOC, pp. 91–100. ACM Press, New York (May 1993)
4. Borowsky, E., Gafni, E., Lynch, N.A., Rajsbaum, S.: The BG distributed simulation algorithm. Distributed Computing 14(3), 127–146 (2001)
5. Chandra, T.D., Hadzilacos, V., Toueg, S.: The weakest failure detector for solving consensus. Journal of the ACM 43(4), 685–722 (1996)
6. Chandra, T.D., Toueg, S.: Unreliable failure detectors for reliable distributed systems. Journal of the ACM 43(2), 225–267 (1996)
7. Delporte-Gallet, C., Fauconnier, H., Guerraoui, R., Tielmann, A.: The disagreement power of an adversary. In: Keidar, I. (ed.) DISC 2009. LNCS, vol. 5805, pp. 8–21. Springer, Heidelberg (2009)
8. Fich, F.E., Luchangco, V., Moir, M., Shavit, N.: Obstruction-free algorithms can be practically wait-free. In: Fraigniaud, P. (ed.) DISC 2005. LNCS, vol. 3724, pp. 493–494. Springer, Heidelberg (2005)
9. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. Journal of the ACM 32(2), 374–382 (1985)
10. Gafni, E.: Round-by-round fault detectors (extended abstract): Unifying synchrony and asynchrony. In: Proceedings of the 17th Symposium on Principles of Distributed Computing (1998)
11. Gafni, E., Guerraoui, R.: Generalizing state machine replication. Technical report, EPFL (2010), `http://infoscience.epfl.ch/record/150307`

12. Gafni, E., Koutsoupias, E.: Three-processor tasks are undecidable. SIAM J. Comput. 28(3), 970–983 (1999)
13. Gafni, E., Kuznetsov, P.: On set consensus numbers. In: Keidar, I. (ed.) DISC 2009. LNCS, vol. 5805, pp. 35–47. Springer, Heidelberg (2009)
14. Gafni, E., Kuznetsov, P.: L-resilient adversaries and hitting sets. CoRR, abs/1004.4701 (2010) (to appear in ICDCN 2011), `http://arxiv.org/abs/1004.4701`
15. Gafni, E., Rajsbaum, S., Herlihy, M.: Subconsensus tasks: Renaming is weaker than set agreement. In: Dolev, S. (ed.) DISC 2006. LNCS, vol. 4167, pp. 329–338. Springer, Heidelberg (2006)
16. Herlihy, M., Rajsbaum, S.: The decidability of distributed decision tasks (extended abstract). In: STOC, pp. 589–598 (1997)
17. Herlihy, M., Rajsbaum, S.: The topology of shared-memory adversaries. In: PODC (2010)
18. Imbs, D., Raynal, M., Taubenfeld, G.: On asymmetric progress conditions. In: PODC (2010)
19. Karp, R.M.: Reducibility among combinatorial problems. Complexity of Computer Computations, 85–103 (1972)
20. Loui, M.C., Abu-Amara, H.H.: Memory requirements for agreement among unreliable asynchronous processes. Advances in Computing Research 4, 163–183 (1987)
21. 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)
22. Yang, J., Neiger, G., Gafni, E.: Structured derivations of consensus algorithms for failure detectors. In: Proceedings of the 17th ACM Symposium on Principles of Distributed Computing, pp. 297–306 (1998)
23. Zieliński, P.: Anti-omega: the weakest failure detector for set agreement. In: PODC (August 2008)