

Wait-free Solvability of Colorless Tasks in Anonymous Shared-memory Model^{*†}

Nayuta Yanagisawa

Abstract

We investigate the fault-tolerant capability of distributed systems in anonymous and unreliable settings. In the anonymous model, processes have no identifiers and execute an identical code. The present paper assumes that processes are prone to crash failures and communicate via multi-writer/multi-reader shared registers.

We study the wait-free solvability of colorless tasks in the anonymous model. Especially, we propose an anonymous full-information protocol for colorless tasks and give a topological characterization of colorless tasks that are wait-free solvable in the anonymous model. The characterization implies that the anonymity does not reduce the computational power of the asynchronous shared-memory model as long as colorless tasks are concerned. We also show that some of the existing topological arguments on the eponymous model apply to the anonymous one.

1 Introduction

In the majority of the studies on the theory of distributed computing, it is assumed that processes are *eponymous*, i.e., processes have unique identifiers and are capable of using them. However, there are certain distributed systems, in which processes cannot make use of their identifiers. For example, peer-to-peer file sharing systems require anonymization for the reason of privacy [13]. In some sensor networks, there are no physical means to uniquely identify individual sensors [3]. In addition to these practical motives, it is theoretically interesting to investigate whether the existence of unique identifiers is intrinsically needed for the design of distributed algorithms.

The present paper investigates the fault-tolerant capability of distributed systems in the *anonymous* asynchronous shared-memory model, in which processes with no identifiers execute an identical code. An arbitrary number of

^{*}This is a post-peer-review, pre-copyedit version of an article published in Theory of Computing Systems. The final authenticated version is available online at: <https://doi.org/10.1007/s00224-017-9819-0>.

[†]A preliminary version of the present paper has appeared in Proceedings of the 18th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), pp. 415–429 (2016) [37].

processes may fail by *crashing*. Processes communicate via *multi-writer/multi-reader* (MWMR) atomic registers, which are initialized to some default value. The use of single-writer shared registers is out of concern of this paper because they allow processes to identify themselves [20], conflicting with the notion of anonymity.

In the past few decades, the topological nature of distributed computing [26] has been extensively studied mainly in the context of the (eponymous) fault-tolerant shared-memory distributed computing. Most significantly, the asynchronous computability theorem [28], which gives a necessary and sufficient condition for distributed decision tasks to be wait-free solvable, and impossibility results [17, 21] for various decision tasks (e.g., k -set agreement task [12] and renaming task [6]) have been devised. One of the fundamental building blocks of the topological theory of distributed computing is the *full-information protocol* [21], which is the most generic form of protocol that can instantiate any implementable protocol. This universality of the full-information protocol significantly simplifies the arguments on the computability issues.

We extend the topological theory of distributed computing to encompass the anonymous asynchronous shared-memory model. Specifically, we investigate the wait-free solvability of *colorless tasks* in the anonymous model. A colorless task is a decision problem that relates each input assignment to a set of possible output assignments without referring to which process has which value. In other words, a colorless task only concerns the set of values held in the system. Throughout this paper, we assume that the set of possible input values to each colorless task is finite. Colorless tasks cover a significant class of decision tasks such as consensus [18], set agreement [12], and loop agreement [23] tasks and have been widely studied mainly in the context of the eponymous shared-memory distributed computing [24, 25, 27, 33].

The primary obstacle to the topological theory of anonymous shared-memory computing is the lack of a full-information protocol. We propose an *anonymous full-information protocol* for colorless tasks that uses *atomic weak set objects*, which are shared objects implemented from MWMR registers. The atomic weak set object implements a set object of limited capability: each process can add a value v to the object, which is initially an empty set, and get a consistent set of values that have been added before.

The fundamental idea in the implementation of the atomic weak set object is to allocate a register for each different value rather than for each individual process and to let each process announce the addition of a value v by writing v to the register allocated for v . For efficient allocation of registers, we introduce a *homonymous renaming object*, which is a generalization of the renaming object [6]. Whenever the homonymous renaming object receives different input names proposed by different processes, it responds a different output name to each of the distinct processes.

We also give a topological characterization of colorless tasks that are wait-free solvable in the anonymous asynchronous shared-memory model. The characterization implies that, as long as colorless tasks are concerned, the anonymity does not reduce the computational power of the asynchronous shared-memory

model. That is, a colorless task is wait-free solvable in the anonymous model if and only if it is wait-free solvable in the eponymous one. This computational equality indicates that some of the topological arguments [21] for colorless tasks in the eponymous asynchronous shared-memory model are applicable to the anonymous ones. As an example, we show that it is undecidable whether a given decision task, which is not necessarily colorless, is wait-free solvable in the anonymous model. It is also implied that the classification of loop agreement tasks [23] is also possible in the anonymous model.

To summarize, the major contributions of the present paper are:

Atomic weak set objects. We propose an anonymous wait-free implementation of atomic weak set objects, making use of homonymous renaming objects for the sake of space efficiency;

Full-information protocol. We propose an anonymous full-information protocol for colorless tasks using atomic weak set objects;

Characterization of wait-free solvability. We give a topological characterization of colorless tasks that are wait-free solvable in the anonymous model;

Computational equality. We show that the computational power of the anonymous model is essentially the same as that of the eponymous one, as long as colorless tasks are concerned. This leads to the undecidability result of the wait-free solvability of decision tasks in the anonymous model.

The present paper refines the construction of the anonymous full-information protocol, compared to the preliminary version [37] by the author, by the use of atomic weak set objects. Atomic weak set objects are more appropriate as the primitive construct of the protocol, while atomic snapshot objects employed in the preliminary version are unnecessarily powerful for the purpose. The refined protocol optimizes the implementation, thereby improving the space complexity.

This paper is organized as follows. In section 2, we provide some basic definitions and elementary facts in topology. Section 3 describes the anonymous asynchronous shared-memory model and the colorless task and presents some fundamental theorems concerning the eponymous wait-free solvability of colorless tasks. In section 4, we introduce an atomic weak set object and present its anonymous wait-free implementation. Section 5 presents an anonymous full-information protocol and studies its space complexity. In section 6, we present our main results, a characterization of colorless tasks that are wait-free solvable in the anonymous model and the computational equality between the anonymous model and the eponymous one. Section 7 concludes the paper and presents directions for further research.

Related Work

Herlihy and Shavit [28] have studied the shared-memory model that has a certain kind of anonymity, in which processes are allowed to use unique identifiers in

a very restricted way. They have established the anonymous computability theorem, which gives a necessary and sufficient condition for a given decision task to be wait-free solvable in their model. The crucial difference between their work and the present paper is that they have assumed single-writer/multi-reader (SWMR) registers. Thus, their results are not immediately applicable to our setting, where processes are not allowed to use identifiers at all and only make use of MWMR registers.

Colorless tasks have been firstly introduced by Herlihy and Rajsbaum [22] under the name of convergence tasks for the purpose of analyzing decidability of the wait-free solvability of decision tasks. Gafni and Koutsoupias [19] have shown, for the first time, that the wait-free solvability for three or more processes is undecidable in the eponymous asynchronous shared-memory model. Herlihy and Rajsbaum [22] have extended this undecidability result to encompass a shared-memory model with (m, k) -set agreement objects.

The fault-tolerant solvability of colorless tasks has been extensively studied mainly in the context of the eponymous shared-memory distributed computing. Herlihy and Rajsbaum [24] have given a topological characterization of colorless tasks that are solvable against an adversarial scheduler characterized by the core and survivor sets [31] in the presence of crash failures. Herlihy et al. [27] have given a topological characterization of colorless tasks that are solvable in the d -solo model with crash failures. Mendes et al. [33] have given a necessary and sufficient condition for colorless tasks to have a t -resilient protocol in Byzantine message-passing systems.

Herlihy et al. [21] have discussed (eponymous) colorless protocols, in which each process only uses process identifiers for accessing shared objects. In other words, the local computation of each process is ignorant of process identifiers. Our anonymous full-information protocol can be seen as an anonymous variant of the colorless layered immediate snapshot protocol, found in Chapter 4 of [21].

Herlihy and Rajsbaum [25] have studied simulations between models of distributed systems concerning colorless tasks. It would be also possible to establish the equivalence between the anonymous model and the eponymous one by simulating the colorless layered immediate snapshot protocol in the anonymous model. However, we focus here on the topological approach and give the anonymous full-information protocol in a direct way.

Some papers concern wait-free computability in the anonymous shared-memory model in the presence of crash failures. Guerraoui and Ruppert [20] have studied what class of shared object can be implemented from multi-writer atomic registers in the anonymous asynchronous shared-memory model. Ruppert [35] has given a classification of shared objects of various types based on their capability to wait-free solve consensus and naming tasks in the anonymous model.

In several papers, the capability of the failure-free anonymous shared-memory model has been investigated. Attiya et al. [7] have shown that consensus is uniformly solvable in the anonymous model in which shared registers are initialized to some default value. They have also given a characterization of the solvable class of consensus-like decision tasks, called agreement tasks. Besides, Jayanti

and Toueg [30] have shown that consensus is not solvable if shared registers are not initialized to any known state.

Starting from [2], there has been a stream of studies investigating the computability and complexity of the anonymous message-passing model. Aspnes et al. [5] have studied simulations between a distributed system with broadcast and a shared-memory system in the anonymous and reliable setting.

There also have been a series of studies on the homonymous message-passing model [15], in which ℓ distinct identifiers are assigned to n processes ($1 \leq \ell \leq n$) and several processes may be assigned the same identifier. The anonymous model and the eponymous model can be seen as the two extreme cases of the homonymous model.

Delporte-Gallet and Fauconnier [14] have proposed the weak set object supporting a limited class of set operations, add and get, which are not atomic nevertheless. The atomic weak set object proposed in the present paper provides an enhanced atomic implementation of Delporte-Gallet and Fauconnier's weak set object. Baldoni et al. [8, 9] have extended the weak set object to the set object that also supports a remove operation and satisfies a weaker consistency condition, called per-element sequential consistency.

2 Topological Preliminaries

We briefly present topological notions that are commonly used in the topological theory of distributed computing. See [21] or [36] for more detailed treatment.

2.1 Abstract Simplicial Complex

A *finite abstract simplicial complex* K on a *finite* set $V(K)$ of *vertices* is a family of nonempty subsets of $V(K)$, called *abstract simplices*, such that

1. $\{v\} \in K$ for every $v \in V(K)$;
2. $s \in K$ and $s' \subseteq s$ imply $s' \in K$.

If there is no ambiguity, we occasionally write *simplex* and *complex*, dropping the prefix 'abstract.' A subset of a simplex s is called a *face* of s . We denote by s , by an abuse of notation, the complex that consists of s and its faces. A *subcomplex* L of a complex K is a subset of K that is also a complex.

Let us write $\#A$ for the cardinality of a set A . A simplex $s \in K$ is said to be of *dimension* $\#s - 1$, denoted by $\dim s$. The dimension of a complex K , denoted by $\dim K$, is defined to be the maximum dimension of simplices contained in K . The *k-skeleton* of a complex K , denoted by $\text{skel}^k K$ is a subcomplex of K that consists of the simplices of dimension less than or equal to k .

Let K_1 and K_2 be complexes. A *simplicial map* $\phi : K_1 \rightarrow K_2$ is a function from $V(K_1)$ to $V(K_2)$ such that $s \in K_1$ implies $\phi(s) \in K_2$, where $\phi(s)$ is the image of s . It is easy to see that the composition of two simplicial maps is also a simplicial map.

A *carrier map* Φ from K_1 to K_2 is a mapping $\Phi : K_1 \rightarrow 2^{K_2}$ such that

1. $\Phi(s)$ is a subcomplex of K_2 for every $s \in K_1$;
2. $s \in K_1$ and $s' \subseteq s$ imply $\Phi(s') \subseteq \Phi(s)$.

We write $\Phi(L_1) = \cup_{s \in L_1} \Phi(s)$ for a subcomplex L_1 of K_1 . Any pair of carrier maps are composed to a carrier map in a trivial way.

2.2 Geometric Realization

Given an abstract simplicial complex K , we associate a corresponding topological space $|K| \subseteq \mathbb{R}^d$, called the *geometric realization* of K , for a sufficiently large positive integer d . For a simplex $s = \{v_0, \dots, v_k\} \in K$, $|s|$ denotes the convex hull of $v_0, \dots, v_k$ that are placed in $\mathbb{R}^d$ in affinely independent positions. Let $|K| = \cup_{s \in K} |s|$, the union of all convex hulls of simplices in K so that every pair of common faces of simplices are identified. This construction is unique up to homeomorphism.

Let K_1 and K_2 be complexes. Each point $x \in |K_1|$ can be represented by the form of $x = \sum_{i=0}^k \alpha_i v_i$ for some simplex $\{v_0, \dots, v_k\} \in K_1$ and $\alpha_i \geq 0$ satisfying $\sum_{i=0}^k \alpha_i = 1$. A simplicial map $\phi : K_1 \rightarrow K_2$ induces a continuous map $|\phi| : |K_1| \rightarrow |K_2|$ such that $x = \sum_{i=0}^k \alpha_i v_i$ implies $|\phi|(x) = \sum_{i=0}^k \alpha_i \phi(v_i)$.

Let $f : |K_1| \rightarrow |K_2|$ be a continuous map and $\Delta : K_1 \rightarrow K_2$ be a carrier map. The map f is said to be *carried by* Δ if $f(|s|) \subseteq |\Delta(s)|$ for every $s \in K_1$. A simplicial map $\phi : K_1 \rightarrow K_2$ is called a *simplicial approximation* of f if $f(x) \in |s_2|$ implies $|\phi|(x) \in |s_2|$ for every $x \in |K_1|$ and $s_2 \in K_2$.

Lemma 1 ([36, Corollary 3.4.4]). Let $f : |K_1| \rightarrow |K_2|$ be a continuous map such that $f(|L_1|) \subseteq |L_2|$ for $L_1 \subseteq K_1$ and $L_2 \subseteq K_2$ and let $\phi : K_1 \rightarrow K_2$ be a simplicial approximation of f . Then $\phi|_{L_1}$ maps L_1 to L_2 and is a simplicial approximation of $f|_{|L_1|}$.

2.3 Barycentric Subdivision

A complex K_1 is said to be a *subdivision* of a complex K_2 if $|K_1| = |K_2|$ and, for each simplex $s_2 \in K_2$, there is a subcomplex L_1 of K_1 such that $|s_2| = |L_1|$.

Let us write $\text{Bary } K$ for the *barycentric subdivision* of a complex K . The vertices of $\text{Bary } K$ are the simplexes of K and the simplexes of $\text{Bary } K$ are the sets $\{s_0, \dots, s_k\}$ such that $s_0 \subseteq \dots \subseteq s_k$. Geometrically, $\text{Bary } K$ divides every simplex of K at its barycenter as shown in Fig. 1.

The barycentric subdivision induces a carrier map from K to $\text{Bary } K$, by which $s \in K$ is mapped to $\text{Bary } s$. The carrier map is denoted by $\text{Bary} : K \rightarrow \text{Bary } K$ by an abuse of notation. We write $\text{Bary}^b K$ for the *b-iterated barycentric subdivision* of K , that is, $\text{Bary}^b K = \text{Bary}(\text{Bary}^{b-1} K)$ with $\text{Bary}^0 K = K$.

Lemma 2 ([36, Theorem 3.4.8]). Let K_1 and K_2 be finite simplicial complexes and $f : |K_1| \rightarrow |K_2|$ be a continuous map. There exists an integer N such that if $b \geq N$, there are simplicial approximations $\phi : \text{Bary}^b K_1 \rightarrow K_2$ of f .

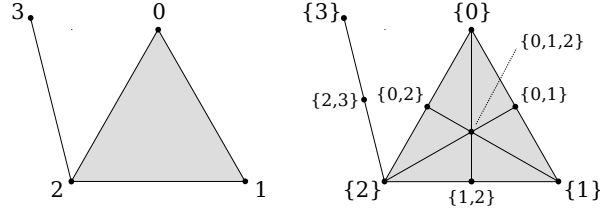


Figure 1: A complex K and its barycentric subdivision $\text{Bary } K$ [37]

3 Anonymous Model and Colorless Tasks

In this section, we describe the anonymous asynchronous shared-memory model and the colorless tasks.

3.1 Anonymous Asynchronous Shared-Memory Model

In the *anonymous asynchronous shared-memory model*, a distributed system consists of $n + 1$ sequential processes that have no identifiers and execute an identical code, where the number of processes is known in advance. Processes are asynchronous, i.e., there is no assumption on the relative speed of the processes. We assume that an arbitrary number of processes may fail by *crashing*, in which case they simply halt and remain silent.

The processes communicate via *multi-writer/multi-reader* (MWMR) registers, on which different processes can read and write. Let $\text{REG}[0, \dots, m - 1]$ denote an array of m registers. The read operation, denoted by $\text{read}(i)$, returns the state of $\text{REG}[i]$. The write operation, denoted by $\text{write}(i, v)$, changes the state of $\text{REG}[i]$ to v and return *ack*. Registers are assumed to be atomic (linearizable) [29]: Each operation is considered to be executed at an instant, which is called a linearization point, between its invocation and response. We assume that every register is initialized to a default value $\perp$. In this paper, we solely concern an unbounded register, which is a register that can hold one out of infinitely many different values.

Note that all the shared objects in this paper are *oblivious*: each of them has an identical interface to all processes. This means that processes cannot identify themselves by invoking operations on shared objects. All the shared objects in this paper are also assumed to be atomic. In the succeeding sections, we present distributed algorithms in the pseudocode style, where a shared object is presented in upper cases (e.g., REG), and a local variable is presented in lower cases (e.g., *view*).

The traditional asynchronous shared-memory model [32], in which processes have unique identifiers and are capable of using them, is referred to as the *eponymous* asynchronous shared-memory model.

3.2 Colorless Tasks

Throughout this paper, we are solely concerned with a class of decision tasks called *colorless tasks* [21].

Definition 3. A colorless task is a triple $T = (I, O, \Delta)$, where I and O are simplicial complexes and $\Delta : I \rightarrow 2^O$ is a carrier map between them. We assume that $\max\{\dim I, \dim O\} < n + 1$.

I and O are called *input complex* and *output complex*, respectively. The set of vertices $V(I)$ (resp., $V(O)$) consists of possible input (resp., output) values for the system. Each simplex in I (resp., O) represents a possible set of input (resp., output) values of the colorless task. Being colorless, processes' identifiers are not contained in I or O .

Each process starts with its own input value $v \in V(I)$, where distinct processes can have the same input value. In each execution path, if the set of all input values s is in I , then the set of all output values must be in $\Delta(s)$. If there is an anonymous (resp., eponymous) wait-free protocol solving the colorless task, we say that the colorless task is *anonymously* (resp., *eponymously*) *wait-free solvable*.

The following observation holds because every anonymous protocol trivially works as an eponymous one.

Lemma 4. If a colorless task T is anonymously wait-free solvable, T is also eponymously wait-free solvable.

Colorless tasks cover a significant class of decision tasks such as consensus [18], k -set agreement [12], and loop agreement [23] tasks. On the other hand, they do not cover decision tasks concerning process identifiers (e.g., renaming tasks [6]) and ones with an unbounded number of input values (e.g., approximate agreement tasks [16]).

Let us see some examples of colorless tasks.

Example. Let b be a positive integer. The b -iterated *barycentric agreement task* with an input complex I is a colorless task $T_b = (I, \text{Bary}^b I, \text{Bary}^b)$. We will see later that T_b is anonymously wait-free solvable for any input complex I .

Example. The k -set *agreement task* with an input complex I is a colorless task $T = (I, \text{skel}^k I, \text{skel}^k)$, where skel^k denotes a carrier map from I to $\text{skel}^k I$ mapping $s \in I$ to $\text{skel}^k s$. It is known that T is not eponymously wait-free solvable for $k < n + 1$ [28].

Example. Let $D^2 = \{\{0\}, \{1\}, \{2\}, \{0, 1\}, \{1, 2\}, \{2, 0\}, \{0, 1, 2\}\}$. Assume that K is a 2-dimensional complex and ℓ is a simple edge loop in K , which is divided into three distinct edge path $\ell_{0,1}$, $\ell_{1,2}$, $\ell_{0,2}$ by points v_0, v_1, v_2 (see Fig. 2). The *loop agreement task* with a triangle loop ℓ , denoted by $T_{K,\ell} = (D^2, K, \Delta)$, is a colorless task such that the carrier map Δ is defined by

$$\Delta(s) = \begin{cases} v_i & \text{if } s = \{i\}, \\ \ell_{i,j} & \text{if } s = \{i, j\}, 0 \leq i < j \leq 2, \\ K & \text{if } s = \{0, 1, 2\}. \end{cases}$$

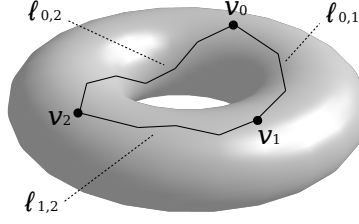


Figure 2: A 2-dimensional complex K and a simple edge loop ℓ [37]

See [23] for more detailed treatments of the loop agreement task.

The followings are the fundamental theorems concerning the eponymous wait-free solvability of colorless tasks.

Theorem 5 ([24, Theorem 4.3]). A colorless task $T = (I, O, \Delta)$ is eponymously wait-free solvable if and only if there is a continuous map $f : |I| \rightarrow |O|$ carried by Δ .

Theorem 6 ([22, Theorem 1 and 2]). It is undecidable whether a given loop agreement task is eponymously wait-free solvable, where $n \geq 2$.

Theorem 5 characterizes the eponymous wait-free solvability of colorless tasks in a purely topological manner and Theorem 6 implies the undecidability of general decision tasks for three or more processes. In Section 6, we establish counterparts to these theorems for the anonymous model.

4 Atomic Weak Set Object

An *atomic weak set object*, denoted by SET, is an atomic object that solely provides two set operations, *add* and *get*. The add operation, denoted by $\text{add}(v)$, takes an argument v and returns ack ; The get operation, denoted by $\text{get}()$, takes no argument and returns the set of values that have appeared as the arguments of all the proceeding add operations. The object is assumed to be initially $\emptyset$. In this paper, we concern a one-shot version of the atomic weak set object. The word “one-shot” means that each process can invoke each operation at most once per object.

We propose an anonymous wait-free implementation of a one-shot atomic weak set object. The fundamental idea of the implementation is to allocate a different register for each different value v rather than for each individual process. A process announces the addition of value v by writing v to the register allocated for v . We make use of the double-collect technique [1] to inspect the set of values that have been added to the set object so far.

From the view point of space complexity, it is not preferable to allocate registers for all (finitely or even infinitely many) possible values in advance,

especially when only a relatively small number of values are to be added. For the space efficiency, we allocate registers *on demand* per each invocation of $\text{add}(v)$ for a new value v .

For on demand allocation, we introduce a *homonymous renaming object*, denoted by HRN. A homonymous renaming object is a one-shot object, which is a generalization of the renaming object [6]. The object HRN provides a single operation propose , denoted by $\text{propose}(v)$, which takes an *input name* v as an argument and returns an *output name* w satisfying the following condition: two propose operations invoked with distinct input names return distinct output names. Note that processes with the same input name need not return the same output name. The difference between the homonymous renaming object and the renaming object is that the former allows two different processes to invoke propose operations with the same input name, whereas the latter does not.

We obtain an anonymous wait-free implementation of a homonymous renaming object by slightly modifying a splitter-based implementation of a renaming object by Aspnes [4]. The space complexity of the homonymous renaming object is $(4\sqrt{2} + o(1))n^2$ and the size of the output namespace is $C(n+1)$, where $C(m) = 4\lceil\sqrt{m}\rceil^3 - \lceil\sqrt{m}\rceil^2$. See Appendix A for the details of the implementation.

Algorithm 1 presents an anonymous wait-free implementation of the atomic weak set object SET. An $\text{add}(v)$ operation allocate a register for the value v by invoking $\text{propose}(v)$ on HRN and writes v to the register. A $\text{get}()$ operation repeatedly reads $\text{REG}[0, \dots, C(n+1) - 1]$ until it sees two identical collects.

Algorithm 1 Anonymous wait-free implementation of SET_V

```

1: shared  $\text{REG}[i]$  initially holds  $\perp$  for all  $i = 0, \dots, C(n+1) - 1$ 
2: shared HRN is a homonymous renaming object

3: operation  $\text{add}(v)$ 
4:    $i := \text{HRN.propose}(v)$                                 // allocate a register for  $v$ 
5:    $\text{REG.write}(i, v)$ 

6: operation  $\text{get}()$ 
7:   repeat
8:     for  $i = 0$  to  $C(n+1) - 1$  do
9:        $\text{collect0}[i] := \text{REG.read}(i)$                       // first collect
10:    for  $i = 0$  to  $C(n+1) - 1$  do
11:       $\text{collect1}[i] := \text{REG.read}(i)$                       // second collect
12:    until  $\text{collect0} = \text{collect1}$                           // identical collects
13:    return the set of non- $\perp$  values in  $\text{collect0}$ 

```

Theorem 7. Algorithm 1 is an anonymous wait-free implementation of an atomic weak set object, where the space complexity is $(4\sqrt{2} + o(1))n^2$.

Proof. For wait-freedom, it is enough to prove that the execution of the repeat-until loop (line 7-12) in the get operation terminates within a finite number of steps. Since the object is one-shot, at most n write operations to REG can occur during the loop. Thus, after at most $n + 1$ iterations of the loop, the double-collect succeeds: a process that invokes the repeat-until loop sees two identical collects and exits the loop.

For atomicity, we have to show how to take linearization points. The linearization point of each add operation should be taken at the linearization point of the write operation. The linearization point of each get operation is taken at an arbitrary point between the last read of the first collect and the first read of the second collect of a successful double-collect. $\text{REG}[0, \dots, C(n+1) - 1]$ is not modified during the above interval, because each $\text{REG}[i]$ is not modified between two $\text{read}(i)$ operations occurring in the duration of the successful double-collect, for every $i = 0, \dots, C(n+1) - 1$.

The implementation makes use of $C(n+1) = (4 + o(1))n^{\frac{3}{2}}$ registers for REG and $(4\sqrt{2} + o(1))n^2$ registers for HRN. Thus, the space complexity is $(4\sqrt{2} + o(1))n^2$. $\square$

The atomic weak set object satisfies the following inclusion property:

Lemma 8. If t_0 and t_1 are the return values of two get operations on the same object SET, either $t_0 \subseteq t_1$ or $t_1 \subseteq t_0$ holds.

5 Anonymous Full-Information Protocol

We propose an *anonymous full-information protocol*, a generic form of protocol that can instantiate any implementable protocol for colorless tasks. Our full-information protocol makes use of multiple atomic weak set objects and is an anonymous variant of the colorless layered immediate snapshot protocol [21]. The colorless layered immediate snapshot protocol makes use of *single-writer* shared objects, called immediate snapshot objects [10], and thus cannot be executed in the anonymous model. The universality of our full-information protocol will be shown in Section 6.

Algorithm 2 presents the anonymous full-information protocol, denoted by $P(b, \delta)$, for a colorless task $T = (I, O, \Delta)$. The protocol $P(b, \delta)$ is parameterized by a positive integer b and a simplicial map $\delta : \text{Bary}^b I \rightarrow O$ and uses b distinct atomic weak set objects SET_ℓ ($\ell = 0, \dots, b-1$). The space complexity of the protocol $P(b, \delta)$ is $b(4\sqrt{2} + o(1))n^2$.

In the protocol $P(b, \delta)$, each process starts with its private input value (line 2) and assigns it to the local variable view (line 3). Thereafter, the process iterates from $\ell = 0$ to $b-1$, the operation of adding its view to SET_ℓ and updating its view by the result of a get operation to SET_ℓ (line 4-6). At last, the process determines its output value by applying the map δ to the value of view (line 7).

Theorem 9. The anonymous full-information protocol $P(b, \text{id})$ solves the b -iterated barycentric subdivision task $T_b = (I, \text{Bary}^b I, \text{Bary}^b)$ for any input complex I , where $\text{id} : \text{Bary}^b I \rightarrow \text{Bary}^b I$ denotes an identity simplicial map.

Algorithm 2 Anonymous full-information protocol $P(b, \delta)$

```

1: shared  $\text{SET}_\ell$  initially holds  $\emptyset$  for  $\ell = 0, \dots, b-1$ 

2: protocol  $P(b, \delta)(v)$ 
3:    $\text{view} := v$                                 // initially view is input
4:   for  $\ell := 0$  to  $b-1$  do
5:      $\text{SET}_\ell.\text{add}(\text{view})$                         // add view to  $\text{SET}_\ell$ 
6:      $\text{view} := \text{SET}_\ell.\text{get}()$                     // update view
7:   return  $\delta(\text{view})$                           // apply  $\delta$  to final view

```

Proof. We prove the theorem by induction on b .

The base case $b = 1$ follows the structure of the proof of Theorem 4.2.8 in [21]. Assume that the set of all inputs to $P(1, \text{id})$ is $s \in I$ and that s_i is the output value of the process p_i . It holds that $s_i \subseteq s$ for every i . Also, either $s_i \subseteq s_j$ or $s_j \subseteq s_i$ holds for every i and j by Lemma 8. Thus, the set of all outputs of $P(1, \text{id})$ must be $t = \{s_{i_0}, \dots, s_{i_k}\}$ for some chain $s_{i_0} \subseteq \dots \subseteq s_{i_k}$. By definition, t is a simplex of $\text{Bary } s$, and this shows that $P(1, \text{id})$ solves T_1 .

For the induction step, we prove that the protocol $P(b, \text{id})$ solves the task T_b . The protocol $P(b, \text{id})$ can be seen as a successive execution of protocols $P(b-1, \text{id})$ and $P(1, \text{id})$, where input complexes for $P(b-1, \text{id})$ and $P(1, \text{id})$ are I and $\text{Bary}^{b-1}I$, respectively. If the set of all inputs to $P(b, \text{id})$ is $s \in I$, the set of all inputs to the subprotocol $P(b-1, \text{id})$ is either s or its face and thus the set t of all outputs of $P(b-1, \text{id})$ is in $\text{Bary}^{b-1}s$ by induction hypothesis. Hence, the set of all inputs to the subprotocol $P(1, \text{id})$ is either t or its face and the set of all outputs of $P(1, \text{id})$ is in $\text{Bary } t$, which is a subcomplex of $\text{Bary}^b s$. Therefore, $P(b, \text{id})$ solves T_b . $\square$

Remark. In the eponymous model, a multi-shot atomic weak set object is implementable on top of $n+1$ registers in a similar way as the implementation of the atomic snapshot object [1]. Thus, the space complexity of the protocol $P(b, \delta)$ can be reduced to $n+1$, which is considerably smaller than $b(4\sqrt{2} + o(1))n^2$.

6 Characterization of Wait-Free Solvability

In this section, we give a topological characterization for colorless tasks to be anonymously wait-free solvable.

Theorem 10. A colorless task $T = (I, O, \Delta)$ is anonymously wait-free solvable if and only if there is a continuous map $f : |I| \rightarrow |O|$ carried by Δ .

Proof. The only if part immediately follows from Lemma 4 and Theorem 5.

For the if part, let $f : |I| \rightarrow |O|$ be a continuous map carried by Δ . There is a simplicial approximation $\delta : \text{Bary}^b I \rightarrow O$ of f by Lemma 2. Lemma 1 implies that δ satisfies $\delta(\text{Bary}^b s) \subseteq \Delta(s)$ for every $s \in I$. We prove that the protocol

$P(b, \delta)$ solves the task T for the above b and δ . $P(b, \delta)$ can be seen as a protocol, in which each process executes $P(b, \text{id})$ as a subprotocol and then determines its output according to the map δ . If the set of all inputs to $P(b, \delta)$ is $s \in I$, the set of all outputs of the subprotocol $P(b, \text{id})$ must be in $t \in \text{Bary}^b s$ by Theorem 9 and is mapped into $\Delta(s)$ by $\delta : \text{Bary}^b I \rightarrow O$. $\square$

The theorem says that a colorless task $T = (I, O, \Delta)$ is anonymously wait-free solvable if and only if there is a continuous map $f : |I| \rightarrow |O|$ that is consistent with the carrier map Δ . The proof of the theorem also implies that the anonymous full-information protocol is universal for colorless tasks, i.e., every anonymously wait-free solvable colorless task is solved by an anonymous full-information protocol. That is because, if a colorless task is anonymously wait-free solvable, there is a continuous map carried by its carrier map, and the continuous map induces an anonymous full-information protocol solving the colorless task.

Interestingly, the solvability condition of Theorem 10 is superficially the same as one for the eponymous model. Thus, Theorem 5 and 10 establish the following computational equality.

Theorem 11. A colorless task is anonymously wait-free solvable if and only if it is eponymously wait-free solvable.

As a consequence of Theorem 11, the following undecidability results hold.

Proposition 12. It is undecidable whether a loop agreement task is anonymously wait-free solvable, where $n \geq 2$.

Proof. The proposition is clear from Theorem 6 and Theorem 11. $\square$

Proposition 13. It is undecidable whether a decision task, which is not necessarily colorless, is wait-free solvable in the anonymous asynchronous shared-memory model, where $n \geq 2$.

Proof. The proposition immediately follows from Proposition 12 because general decision tasks include colorless tasks. $\square$

We can also show that the classification of loop agreement tasks [23] is possible in the anonymous asynchronous shared-memory model. This indicates that there is an infinite hierarchy on the mutual implementability of colorless tasks in the anonymous model parallel to the eponymous case [23].

7 Conclusion and Further Research

We have extended the topological theory of distributed computing to encompass the anonymous asynchronous shared-memory model, in which the number of processes is a priori known. Specifically, we have devised the anonymous full-information protocol by means of atomic weak set objects and given a topological characterization of colorless tasks that are wait-free solvable in the anonymous

model. Our characterization implies that the anonymity does not reduce, as long as colorless tasks are concerned, the computational power of the asynchronous shared-memory model. We have also proved that the wait-free solvability of general decision tasks is undecidable in the anonymous model.

It is easy to see that the very same results would hold for the homonymous asynchronous shared-memory model prone to crash failures, because the homonymous model [15] is at least as powerful as the anonymous model.

A promising future research direction would be to investigate t -resilient solvability of colorless tasks in the anonymous model. This would be harder than the eponymous case [24] because physically different processes with the same local state are indistinguishable due to anonymity, and thus each process cannot simply wait for other processes, even when no process would fail [7].

It would also be challenging to extend our characterization to the case of general decision tasks, giving a *totally anonymous version* of the asynchronous computability theorem [28]. Our anonymous full-information protocol is insufficient for this purpose, because the protocol is ignorant of the initial values given to the processes. We would need to devise a refined version of the full-information protocol, which conveys richer information including initial values of processes.

Acknowledgements

I am grateful to anonymous reviewers for their insightful comments. I would like to express my gratitude to Prof. Susumu Nishimura for enlightening discussion and helpful advice on writing this paper.

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] Angluin, D.: Local and global properties in networks of processors. In: *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 82–93 (1980)
- [3] Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. *Distributed Computing* **18**(4), 235–253 (2006)
- [4] Aspnes, J.: Slightly smaller splitter networks. Tech. Rep. YALEU/DCS/TR-1438, Yale University Department of Computer Science (2010)
- [5] Aspnes, J., Fich, F.E., Ruppert, E.: Relationships between broadcast and shared memory in reliable anonymous distributed systems. *Distributed Computing* **18**(3), 209–219 (2006)

- [6] Attiya, H., Bar-Noy, A., Dolev, D., Peleg, D., Reischuk, R.: Renaming in an asynchronous environment. *J. ACM* **37**(3), 524–548 (1990)
- [7] Attiya, H., Gorbach, A., Moran, S.: Computing in totally anonymous asynchronous shared memory systems. *Information and Computation* **173**(2), 162–183 (2002)
- [8] Baldoni, R., Bonomi, S., Raynal, M.: Value-based sequential consistency for set objects in dynamic distributed systems. In: *Proceedings of the 16th International Euro-Par Conference (Euro-Par)*, pp. 523–534 (2010)
- [9] Baldoni, R., Bonomi, S., Raynal, M.: Implementing set objects in dynamic distributed systems. *Journal of Computer and System Sciences* **82**(5), 654 – 689 (2016)
- [10] Borowsky, E., Gafni, E.: Immediate atomic snapshots and fast renaming. In: *Proceedings of the 12th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 41–51 (1993)
- [11] Capdevielle, C., Johnen, C., Kuznetsov, P., Milani, A.: On the Uncontended Complexity of Anonymous Consensus. In: E. Anceaume, C. Cachin, M. Potop-Butucaru (eds.) *19th International Conference on Principles of Distributed Systems (OPODIS 2015), Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 46, pp. 1–16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2016)
- [12] Chaudhuri, S.: More choices allow more faults: Set consensus problems in totally asynchronous systems. *Information and Computation* **105**(1), 132 – 158 (1993)
- [13] Chothia, T., Chatzikokolakis, K.: A survey of anonymous peer-to-peer file-sharing. In: *Proceedings of Satellite Workshop of the International Conference on Embedded and Ubiquitous Systems (EUS)*, pp. 744–755 (2005)
- [14] Delporte-Gallet, C., Fauconnier, H.: Two consensus algorithms with atomic registers and failure detector ω . In: *Proceedings of the 10th International Conference on Distributed Computing and Networking (ICDCN)*, pp. 251–262 (2009)
- [15] Delporte-Gallet, C., Fauconnier, H., Guerraoui, R., Kermarrec, A.M., Ruppert, E., Tran-The, H.: Byzantine agreement with homonyms. In: *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pp. 21–30 (2011)
- [16] Dolev, D., Lynch, N.A., Pinter, S.S., Stark, E.W., Weihl, W.E.: Reaching approximate agreement in the presence of faults. *J. ACM* **33**(3), 499–516 (1986)
- [17] Fich, F., Ruppert, E.: Hundreds of impossibility results for distributed computing. *Distributed Computing* **16**(2), 121–163 (2003)

- [18] Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. *J. ACM* **32**(2), 374–382 (1985)
- [19] Gafni, E., Koutsoupias, E.: Three-processor tasks are undecidable. *SIAM Journal on Computing* **28**(3), 970–983 (1999)
- [20] Guerraoui, R., Ruppert, E.: Anonymous and fault-tolerant shared-memory computing. *Distributed Computing* **20**(3), 165–177 (2007)
- [21] Herlihy, M., Kozlov, D., Rajsbaum, S.: Distributed computing through combinatorial topology. Morgan Kaufmann (2013)
- [22] Herlihy, M., Rajsbaum, S.: The decidability of distributed decision tasks (extended abstract). In: *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 589–598 (1997)
- [23] Herlihy, M., Rajsbaum, S.: A classification of wait-free loop agreement tasks. *Theoretical Computer Science* **291**(1), 55 – 77 (2003)
- [24] Herlihy, M., Rajsbaum, S.: The topology of shared-memory adversaries. In: *Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pp. 105–113 (2010)
- [25] Herlihy, M., Rajsbaum, S.: Simulations and reductions for colorless tasks. In: *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 253–260 (2012)
- [26] Herlihy, M., Rajsbaum, S., Raynal, M.: Power and limits of distributed computing shared memory models. *Theoretical Computer Science* **509**, 3 – 24 (2013)
- [27] Herlihy, M., Rajsbaum, S., Raynal, M., Stainer, J.: Computing in the presence of concurrent solo executions. In: *Proceedings of the 11th Latin American Theoretical Informatics Symposium (LATIN)*, pp. 214–225 (2014)
- [28] Herlihy, M., Shavit, N.: The topological structure of asynchronous computability. *J. ACM* **46**(6), 858–923 (1999)
- [29] Herlihy, M.P., Wing, J.M.: Linearizability: A correctness condition for concurrent objects. *ACM Trans. Program. Lang. Syst.* **12**(3), 463–492 (1990)
- [30] Jayanti, P., Toueg, S.: Wakeup under read/write atomicity. In: *Proceedings of the 4th International Workshop on Distributed Algorithms*, pp. 277–288 (1991)
- [31] Junqueira, F.P., Marzullo, K.: Synchronous consensus for dependent process failures. In: *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS)*, pp. 274–283 (2003)
- [32] Lynch, N.A.: Distributed algorithms. Morgan Kaufmann (1996)

- [33] Mendes, H., Tasson, C., Herlihy, M.: Distributed computability in byzantine asynchronous systems. In: Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC), pp. 704–713 (2014)
- [34] Moir, M., Anderson, J.H.: Wait-free algorithms for fast, long-lived renaming. *Science of Computer Programming* **25**(1), 1 – 39 (1995)
- [35] Ruppert, E.: The anonymous consensus hierarchy and naming problems. In: Proceedings of the 11th International Conference on Principles of Distributed Systems (OPODIS), pp. 386–400 (2007)
- [36] Spanier, E.: Algebraic topology, vol. 55. McGraw-Hill (1966). (reprinted by Springer-Verlag)
- [37] Yanagisawa, N.: Wait-free solvability of colorless tasks in anonymous shared-memory model. In: Proceedings of the 18th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), pp. 415–429 (2016)

Appendix

A Homonymous Renaming Object

We propose an anonymous wait-free implementation of a homonymous renaming object, where the space complexity is $(4\sqrt{2} + o(1))n^2$. Our implementation is similar to the splitter-based implementation of the renaming object [34, 4], except that we use *value-splitters* [11] instead of splitters. The value-splitter is a one-shot shared object that provides a single operation `split`, denoted by `split( $v$ )`, that takes an argument v and returns either `stop`, `down` or `right`. The value-splitter satisfies the following three conditions:

VS-Agreement. If two invocations `split( $v$ )` and `split( $v'$ )` by different processes both return `stop`, then $v = v'$.

VS-Solo execution. If only one `split` operation is invoked, it returns `stop`.

VS-Nontriviality. If m `split` operations are invoked, at most $m - 1$ of them return `right` and at most $m - 1$ of them return `down`.

Algorithm 3 presents an anonymous wait-free implementation of the value-splitter object. This is a slight modification of the implementation of the 2-valued value-splitter [11], which returns either `stop` or `down` and only satisfies VS-agreement and VS-solo execution conditions. We assume that every value is totally ordered and expressions $v > \perp$, $v < \perp$, and $v = \perp$ are evaluated to false for any $v \neq \perp$. When invoked, `split( $v$ )` writes v to `REG[ $i$ ]` for $i = 0, \dots, k - 1$ unless otherwise it sees at least one non- $\perp$ value other than v in `REG[0], \dots, REG[ $i - 1$ ]` or non- $\perp$ value in `REG[ $i$ ]` for some k satisfying $k^2 - 3k + 6 > 2(n + 1)$. If `split( $v$ )` successfully writes into `REG[ $k - 1$ ]`, it returns `stop`, otherwise, it returns either `right` or `down`, depending on the values it has scanned.

Algorithm 3 Anonymous wait-free implementation of value-splitter

```
1: shared REG[ $i$ ] is initially holds  $\perp$  for  $i = 0, \dots, k-1$  with  $k^2 - 3k + 6 > 2(n+1)$ 

2: operation split( $v$ )
3:    $last\_written := -1$ 
4:   while  $last\_written \leq k-1$  do
5:      $i := 0$ 
6:     while  $i \leq last\_written$  do
7:        $content := \text{REG.read}(i)$ 
8:       if  $content < v$  then
9:         return right
10:      else if  $content > v$  then
11:        return down
12:       $i := i + 1$ 
13:     $content := \text{REG.read}(i)$ 
14:    if  $content \leq v$  then
15:      return right
16:    else if  $content > v$  then
17:      return down
18:    else if  $content = \perp$  then
19:       $\text{REG.write}(last\_written + 1, v)$ 
20:     $last\_written := last\_written + 1$ 
21:  return stop
```

Lemma 14. Algorithm 3 is an anonymous wait-free implementation of the value-splitter, where the space complexity is $(\sqrt{2} + o(1))\sqrt{n}$.

Proof. It is immediate that Algorithm 3 satisfies VS-agreement and VS-solo execution conditions by Lemma 10 and Theorem 11 in [11].

We prove, by contradiction, that the implementation satisfies the VS-nontriviality condition. We may assume that executions are non-faulty, because, if some process fails, the condition is automatically satisfied. Assume that all the m invocations of the split operation have returned **down**. Then, each of the m operations should have seen a value greater than its value at lines 7 or 13. This is a contradiction, because there must be the maximum value among the m values. Likewise, assume that all the m invocations of the split operation have returned **right** and that there have been exactly m_0 split operations, out of m , whose values are the minimum among the m values. All the m_0 operations should have seen the minimum value at Line 13 in some iteration of the while loop. On the other hand, if a split operation sees the minimum value in $\text{REG}[\ell]$ at line 13, another split operation should have written the minimum value to $\text{REG}[\ell]$ in advance. Thus, there must be at least one split operation with the minimum value other than the above m_0 operations. This is a contradiction. $\square$

We introduce the *value-splitter network* and construct a homonymous renaming object on it. A value-splitter network is a network that consists of

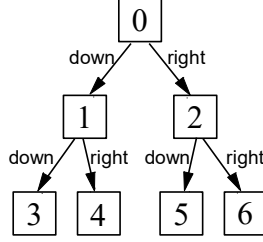


Figure 3: An example of the value-splitter network

value-splitters and virtual wires connecting them. In the network, a value-splitter has two outgoing wires labeled by **right** and **down**, respectively. The value-splitter network is a variant of the splitter-network [4]: The former has the same network topology as the latter and is obtained just by replacing each splitter with the value-splitter.

We implement a homonymous renaming object by the value-splitter network as follows. To give the readers intuition, we first present a simple but space inefficient implementation and then present a more space efficient one.

For the simple implementation, we consider the value-splitter network, which consists of $2^{n+1} - 1$ value-splitters, in the form of the perfect binary tree of height $n + 1$. We assume that value-splitters are numbered by the breadth-first visit order. Fig. 3 schematically presents the network for the case of $n = 2$. The operation **propose**(v), starting from the root of the network, traverses down the network as follows. Each time it visits a node, it invokes **split**(v) on the value-splitter object of the node. If **right** or **down** is returned, it visits the descendant node by traversing the wire labeled by the returned value; If **stop** is returned, it terminates traversal and returns the node number as the output name.

Lemma 15. There exists an anonymous wait-free implementation of the homonymous renaming object for $n + 1$ processes, where the space complexity is $(\sqrt{2} + o(1))\sqrt{n}(2^{n+1} - 1)$ and the size of the output namespace is $2^{n+1} - 1$.

Proof. It is easy to prove that an invocation of **propose**(v) operation is eventually terminates by induction on the height of the network.

By VS-agreement, it is impossible for a pair of two processes with distinct input names to have **stop** as a response from the same value-splitter object. This means that no two processes with distinct input names agree on the same output name. $\square$

To improve the space efficiency, we make use of the network topology as of Aspnes' splitter network (Theorem 4 in [4]), which consists of $(4 + o(1))n^{\frac{3}{2}}$ splitters. By letting processes traverse in the same way as Lemma 15, we obtain the following theorem.

Theorem 16. There exists an anonymous wait-free implementation of a homonymous renaming object for $n + 1$ processes, where the space complexity $(4\sqrt{2} +$

$o(1))n^2$. The size of the output namespace is $C(n+1)$, where $C(m) = 4\lceil\sqrt{m}\rceil^3 - \lceil\sqrt{m}\rceil^2$.

As the splitter object also satisfies VS-solo execution and VS-nontriviality conditions, the arguments concerning termination property in [4] are also valid for the value-splitter network. We can prove, in the same way as Lemma 15, that two processes with distinct input names never agree on the same output name.