UNIVERSITY OF CALIFORNIA

Los Angeles

Topological Characterization of Distributed Computability

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Yuan He

2021

ABSTRACT OF THE DISSERTATION

Topological Characterization of Distributed Computability

by

Yuan He
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2021
Professor Eliezer M. Gafni, Chair

The field of distributed computability studies whether a task is solvable in a distributed system, as specified by communication channels, failure patterns, synchronization instructions, etc. Tasks are analogs of functions in distributed computing, which characterizes the coordination problems in a system with multiple processes. Processes start with an input value in a task and then communicate with others to decide an output value. Unlike classical computability theory, task solvability depends on the model of the system. Determining whether a task is solvable in a given model is usually argued on a case-by-case basis.

The celebrated asynchronous computability theorem (ACT) provides a topological characterization of task solvability in the wait-free model. Since the development of ACT, the topological method has achieved significant successes: ACT is generalized to models with additional synchronization objects and models of resiliency. The general

theory of distributed computability, however, has still not been established. It is not clear whether ACT can be generalized for more general models.

In this dissertation, we study task computability in more general models. We show that every set-consensus collection model is equivalent to a vector-set-consensus model. For colorless task solvability, we derive ACT of the vector-set-consensus model. For colored tasks in the k-set-consensus model, we present an alternative formulation of ACT by introducing the notion of color-projection closed complex. For models specified by resiliency, we study the general adversary model and derive the "wait-at-beginning" ACT for colorless tasks. More specifically, we show that processes only need the full power of the adversary at the beginning of a colorless task protocol. We also present a sufficient topological condition to ensure that a task has a solution in the vector-set-consensus model. Specifically, we show that if the task output complex satisfies certain topological properties, it can be solved by the generalized convergence algorithm as a universal protocol.

Our generalizations of ACT show that many models are equivalent to "restricted" iterated immediate snapshot (IIS) models. Thus, this dissertation implies a potential general framework for distributed computing analogous to the "Church-Turing" thesis.

The dissertation of Yuan He is approved.

Sheila A. Greibach

Amit Sahai

Alexander Sherstov

Eliezer M. Gafni, Committee Chair

University of California, Los Angeles

2021

Contents

List of Figures

List of Algorithms

Acknowledgments

First and most important, I would like to thank Prof. Eli Gafni for being my advisor. Eli shows me the beauty of theoretical computer science and how to think hard as a researcher. It wouldn't be possible for me to finish this dissertation without his help. I want to express my sincerest appreciation to my committee members, Prof. Sheila Greibach, Prof. Amit Sahai, and Prof. Alexander Sherstov, for the time and energy on my dissertation and defense. I am also grateful to Prof. Ciprian Manolescu for serving on my Advancement to Candidacy committee. I would also like to express my thanks and appreciation to my friends in the Computer Science department, Tianyi Zhang, Diyu Zhou, Ang Li, and people from the theory lab. Many thanks to my labmate Pei Wu for helping me a lot in the past years. Finally, I would like to thank my parents for their unwavering support and encouragement. I also want to thank my wife Lei Wang and my son Luke, for all the love and support, for their patience. I could not have finished this long journey without you.

VITA

2014–2021	University of California, Los Angeles
	Research Assistant, Computer Science Department
2012–2014	East Carolina University, North Carolina
	M.S., Computer Science
2006–2010	Xidian University, China
	B.S., Computer Science and Technology

Chapter 1

Introduction

Classical computability theory studies whether a problem is decidable on the Turing machine. Whether a problem is computable does not depend on what type of Turing machine is being used. Moreover, according to Church-Turing thesis, it is independent of the model used for computation.

In distributed computing, however, the computability of a problem differs from one system to another. For example, in a synchronous model where processes communicate with others in a round-by-round manner without failure, the system can compute any problem as solvable in a Turing machine. However, if the communication channel is unreliable, the coordinated attack problem illustrates that two parties can never reach common knowledge.

The specification of a distributed model includes many aspects: the communication channels among processes, failure patterns, synchronization primitives, etc. The challenge of solving problems in distributed computing is that a process may only have partial knowledge of others, limited by the model's specification. For example, in an asynchronous system, the relative speed among different processes could be arbitrary. In this case, one process may not rely on the communication or the computation of

another process. Therefore, the more uncertainty a system has, the fewer problems the system can solve. The theory of distributed computability quantifies this uncertainty and determines problem solvability for different models. This dissertation focuses on the distributed computability for models with additional synchronization primitives and models with resiliency.

1.1 Task Solvability in Distributed Computing

The problems studied in distributed computing are called tasks. Tasks are coordination problems in distributed systems and analogous to functions in classical computability theory. In a task, each process starts with an input value and then returns an output value under the constraint of the task specification. The most famous and fundamental example of tasks is consensus. This task requires that processes return an input value such that every process chooses the same value. The seminal result of the impossibility of consensus, proved by Fischer, Lynch, and Paterson [13], shows that consensus is not solvable in the 1-resilient model, i.e., an asynchronous model in which one process is allowed to fail. On the other side of task solvability, the universality of consensus [23] shows that every task has a protocol in the wait-free model if consensus can be solved.

Identifying task solvability helps system designers to program under the correct system specification. However, it is usually difficult to check whether a task is solvable in a distributed model. To show that a task is solvable, one needs to design an algorithm to produce correct outputs in every admissible execution of the model, which is usually a challenge because a model contains an infinite number of executions. Moreover, formal verification methods only work for elementary models and require

limiting the number of processes in practice. By contrast, to show that a task is not solvable, one needs to demonstrate that for any protocol, there is an execution such that the protocol fails to produce correct outputs. For example, the proof of the impossibility of consensus [13] uses the bi-valency argument to show that there always exists an infinite length execution for any consensus protocol, in which processes never finish their computation and agree on the same output. Note that these types of arguments are problem- and model-specific. Thus, developing a general framework to characterize task solvability is appealing.

In 1993, three groups [4, 26, 37] independently used topological arguments to prove that no protocol solves the k-set-agreement task in the t-resilient model. Moreover, Borowsky and Gafni [4, 5] proposed the iterated immediate snapshot (IIS) model. Since then, the IIS model has become the fundamental tool in distributed computability. Studying task solvability in the IIS model provides many advantages. First, IIS is equivalent to the wait-free model, the weakest distributed system in which it allows all but one process to fail. Second, it has a round-by-round computation structure, which significantly simplifies the analysis for computability. More importantly, the IIS is closely related to algebraic topology: the executions of IIS can be represented as the standard chromatic subdivision, a particular type of simplicial complex that is extensively studied in algebraic topology. As a task can also be defined as simplicial complexes, solving a task can be characterized topologically.

With this observation in mind, Herlihy and Shavit [27] prove the celebrated asynchronous computability theorem (ACT) for wait-free task solvability. They showed that the wait-free model can be characterized by the topological representation of the IIS model: a task has a wait-free protocol if and only if there exists a specific simplicial map from a subdivision of the input complex to output complex. Subsequently,

Gafni and Borowsky [6] used an algorithmic approach to present an alternative proof of ACT. Since then, topological methods have achieved remarkable success in distributed computability. Gafni et al. [20] proved the generalized ACT for models that can be defined as a subset of infinite executions in IIS. Their characterization involves the construction of an infinite sequence of complexes and simplicial maps. Saraph et al. [38] proved the ACT for the t-resilient model where at most t processes can fail in the system. Gafni et al. [17] showed the ACT for the k-set-consensus model. By generalizing the ACT for the t-resilient model, Kuznetsov et al. [32] presented the ACT for the fair-adversary model. These results [17, 32, 38] show that restricted IIS models can characterize task solvability for many model. It is not clear, however, whether this type of ACT can be extended in general.

In this dissertation, we make progress in this direction. We suspect that IIS may be analogous to Turing machine for distributed computing. We show that many general models can be considered as "restricted" IIS models for task solvability. More specifically, we prove the ACT for models with additional access to a collection of different types of set-consensus objects and models with general failure patterns. Thus, this dissertation provides insight into a potential unified distributed computability theory: the problem of solving a task in any reasonable distributed model can be analyzed in a restricted IIS model.

1.2 Our contribution

In this section, we briefly discuss our results and some related works.

ACT for Set-Consensus Collection/Vector-set-consensus

The k-set-agreement task relaxes the consensus constrain by allowing k different output values. The k-set-consensus model is a shared memory model that enables every process to solve k-set-agreement by accessing the k-set-consensus object. More generally, the (m, k)-set-consensus object allows up to m processes to solve the k-set-agreement task. The set-consensus collection model [11] is a shared memory model in that processes can access multiple types of set-consensus objects.

We show that, for tasks solvability, every set-consensus collection model is equivalent to a vector-set-consensus model, in which for $m = 1, 2, ..., n + 1$, the model consists of unlimited copies of (m, k_m)-set-consensus object. We prove the ACT of the vector-set-consensus model solvability for colorless tasks, which is a broad class of tasks that characterizes the input and output relation that does not depend on processes ids. Our generalization of ACT shows that the vector-set-consensus model is equivalent to an iterated model in which the protocol complex is a subcomplex in the second chromatic subdivision. We also provide an alternative formulation of ACT for k-set-consensus model [17]. Our construction introduces the notion of color-projection and color-projection closed complex, which provides insight into the connection between tasks and objects.

Colorless ACT for General Adversary

The t-resiliency is the model in which at most t processes can fail. However, in practice, the failure pattern in distributed systems is usually non-uniform, i.e., processes may fail in a correlated way.

Delporte et al. [12] introduced the notion of general adversary to model the

scenario that the failure of one process depends on other processes. An adversary $\mathcal{A}$ is defined as a set of subsets of processes in the system, called lives sets. For every execution in $\mathcal{A}$, the set of correct processes in the execution is a live set. The fair adversary is a restricted adversary that generalizes the t-resilient model and k-obstruction freedom model. Kuznetsov et al. [32] proposed ACT for the fair adversary model. However, the problem of characterizing the general adversary is not solved.

In this dissertation, we partially answer the question. We show that the colorless task solvability in general adversary is related to its set-consensus power. By demonstrating a protocol complex with the same set-consensus power, we prove the "wait-at-beginning" colorless ACT for general adversary. The "wait-at-beginning" property justifies a common intuition of a resiliency model: for colorless tasks, it is enough to only use the set-consensus power at the beginning of a protocol and then proceed in a wait-free manner.

A sufficient Topological Condition For Task solvability

ACTs provides a sufficient and necessary condition to determine whether a task is solvable in a model. However, one must explicitly construct a simplicial map from the iterated protocol complex to the task output complex to apply ACT, which is not convenient from the viewpoint of verification.

Herlihy et al. [24] proved a sufficient topological condition that ensures a task has a solution in the wait-free model. Their result shows that if the task output complex has a certain topological connectivity, it has a wait-free solution. Their proof, however, only indicates the existence of the protocol. Moreover, whether the argument can be generalized to more general models is an open problem.

In this dissertation, we observed that the result by Herlihy et al. can be proved

algorithmically using the convergence algorithm, which was proposed by Borowsky and Gafni [6] to give an alternative proof of the wait-free ACT.

Borowsky [3] also sketched the generalized convergence algorithm for the active-resiliency model and the set-consensus collection model. We extend the generalized convergence algorithm for the vector-set-consensus model with a full proof. Using the generalized convergence algorithm, we prove a sufficient topological condition for task solvability in the vector-set-consensus model: a task is solvable if its output complex has certain global and local connectivity.

1.3 Organization

The rest of this manuscript is organized as follows. Chapter 2 discusses the specification of distributed models and defines the notion of task solvability. We also review some useful simulation techniques. Chapter 3 presents a summary of simplicial complex and some basic results in algebraic topology. Chapter 4 introduces the adaptive-set-agreement task and discusses the relation between the set-consensus collection model and the vector-set-consensus model. In Chapter 5, we prove the colorless ACT for vector-set-consensus model. We also provide an alternative formulation of ACT for the k-set-consensus model. Chapter 6 presents the "wait-at-beginning" colorless ACT for general adversary. Chapter 7 provides a sufficient topological condition to ensure that a colored task has a solution in the vector-set-consensus model. We conclude our results in Chapter 8 and propose some open problems in distributed computability.

Chapter 2

Distributed Computing Model

In this chapter, we present an overview of distributed computing models considered in this dissertation. We introduce the notion of tasks and shared memory objects. We also provide a summary of the IIS model and useful simulation techniques.

2.1 Processes and Communication Model

We consider a system of $n+1$ processes, $p_0, p_1, ..., p_n$, which processes communicate with each other using shared memory. Processes take steps and communicate with others asynchronously. There is no time bound for a process to take a step.

The shared memory model considered in this dissertation is the atomic-snapshot memory [1], which consists of an array of $n+1$ shared registers. Each process p_i writes exclusively to one register and atomically takes a snapshot of the array . It is well known that the atomic-snapshot model can be wait-free implemented [1] from standard single-writer multi-reader (SWMR) shared memory.

We also consider the *Iterated Immediate Snapshot* (IIS) model [4, 5], which has

been extensively studied in theoretical distributed computing. In IIS, processes proceed through a sequence of independent shared memories $\mathrm{IS}_1, \mathrm{IS}_2, ...$ In each round r, the memory IS_r is accessed by processes with a single Immediate Snapshot(IS) operation: each process p_i submits a value v_{ir} and returns a set V_{ir} of submitted values, satisfying the following properties:

1. Self-containment: $v_{ir} \in V_{ir}$
2. Atomicity: for all i, j, $V_{ir} \subseteq V_{jr}$ or $V_{jr} \subseteq V_{ir}$
3. Immediacy: for all i, j, if $v_{ir} \in V_{jr}$, then $V_{ir} \subseteq V_{jr}$.

In IIS model, processes execute the *full-information protocol.* Every process writes its initial value and uses the return value of IS_r as the input value for IS_{r+1}. In each round, every process has the information of its local state and some other processes' local state in previous rounds. After executing enough rounds, a process may apply a decision function from its state to compute an output value.

It is shown in [6, 8] that the wait-free atomic-snapshot model is equivalent to the IIS model. A *run* or an *execution* in IIS [20] can be considered a sequence of non-empty sets of processes: $S_1, S_2, ...,$ where each S_i consists of processes that invoke the ith iteration of immediate snapshot. Furthermore, each S_r is associated with an ordered partition: $S_r = S_r^1 \cup ... \cup S_r^{n_r}$ where $1 \leq n_r \leq n + 1$. Processes in each S_r^ℓ invoke IS_r concurrently. Thus, the order of this partition corresponds to the relative ranking among processes that executes the rth immediate snapshot.

2.2 Failure and Participation

A process is called *faulty* if it crashes and takes only a finite number of steps. Otherwise, it is called *correct*. A process is *participating* if it takes at least one step. The set of participating processes in an execution is called *participating set*.

The *t-resilient model* is the atomic snapshot model where at most t processes can fail. When $t = n$, it is called the *wait-free model*, in which at least one process does not fail.

The notion of *adversary* was first introduced by Delporte et al. [12] to capture the non-uniform failure patterns in a distributed system. Formally, a general adversary $\mathcal{A}$ is a set of live sets such that each live set $S \in \mathcal{A}$ is a non-empty subset of $\Pi = \{0, 1, ..., n\}$. An infinite execution is called $\mathcal{A}$-compliant if the set of correct processes is a live set in $\mathcal{A}$. Thus, an adversary $\mathcal{A}$ can be considered as a model that contains all $\mathcal{A}$-compliant executions.

Many distributed models can be characterized as an adversary. An adversary is called *superset-closed* if for every $S \in \mathcal{A}$ and $S \subseteq S' \subseteq \Pi$, $S' \in \mathcal{A}$. The *symmetric* adversary contains every live set with the same size, i.e., if $S \in \mathcal{A}$, then for any $S' \subseteq \Pi$ such that $|S'| = |S|$, $S' \in \mathcal{A}$. The *k-obstruction-freedom* is a symmetric adversary such that every live set has a size less or equal to k. The t-resilient model is an adversary that contains every set with size at least $(n + 1 - t)$. Thus, t-resilience is both superset-closed and symmetric.

2.3 Task and Object

Task generalizes many coordination problems in distributed computing. It characterizes the input-output relations among processes. A process invokes a task with an input value and returns an output value so that the inputs and the outputs across processes satisfy the task specification.

Formally, a task $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ is a triple, where $\mathcal{I}$ contains the input vectors, $\mathcal{O}$ is the set of output vectors, and a total relation $\Gamma : \mathcal{I} \to \mathcal{O}$ associates each input vector σ, with a set of valid output vectors $\Gamma(\sigma) \subseteq \mathcal{O}$. Each input vector or output vector γ is specified by a sequence of pairs $\gamma = (0, \text{val}_0), (1, \text{val}_1), ..., (n, \text{val}_n)$ where each (i, val_i) is associated to process p_i and val_i is some input or output value. Usually, we require Γ to be monotonic: for all $\sigma, \tau \in \mathcal{I}$, if $\tau \subseteq \sigma$, then $\Gamma(\tau) \subseteq \Gamma(\sigma)$. The requirement can be interpreted as the nature of asynchronous models: if some participating processes have decided their outputs, a process that starts later must choose an output consistent with the decided ones.

The *colorless* task [24] is a large class of tasks in which a process is allowed to adopt the input or output from another participating process. In other words, the set of input values determines the set of output values. Formally, a colorless task is defined as a task that is invariant under the colorless task transformation C, defined as follows.

Let $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ be a task. Without loss of generality, we assume that every output vector is associated to some input vector, i.e., $\mathcal{O} = \bigcup_{\sigma \in \mathcal{I}} \Gamma(\sigma)$. Let γ be an input or output vector in T. The set $\text{Val}(\gamma) = \{v_i \mid (i, \text{val}_i) \in \gamma\}$ consists of values that appear in γ. We define a transformation C that turns a task T into another task $C(T) = (\mathcal{I}', \mathcal{O}', \Gamma')$, defined as follows:

1. $\mathcal{I}' = \mathcal{I}$
2. $\forall \sigma \in \mathcal{I}$, $\Gamma'(\sigma) = \{\tau' \mid \exists \tau \in \Gamma(\sigma), \mathrm{Val}(\tau') \subseteq \mathrm{Val}(\tau)\}$
3. $\mathcal{O}' = \bigcup_{\sigma' \in \mathcal{I}'} \Gamma'(\sigma)$

A colorless task T is a task such that $T = C(T)$. It is not hard to see that $C(T) = C(C(T))$ for any task T.

A task is called *solvable* in a model M, if there exists a protocol such that for any input vector σ and any execution of the protocol in M, every correct process calculates an output with respect to the task specification, i.e., their output values forms an output vector in $\Gamma(\sigma)$.

The *k-set-agreement task* [9] is one of the most important colorless tasks. The input values are from a set of values V, and each process returns an input value from V such that at most k different values are returned in total. The case of 1-set consensus is called *consensus* [13]. Without loss of generality, we consider the *input-less* k-set-agreement, in which processes use their process id as the inputs.

The (m, k)-*set-consensus* object is a one-shot shared memory object that allows m processes to solve k-set-agreement. Every process access the object at most once, and at most m process can access in total. Each accessing process uses a single propose operation: it submits a value to the object and gets a value such that the following properties are satisfied:

1. Validity: every returned value is a value proposed by some process;
2. Agreement: at most k different values are returned
3. Termination: every correct process that proposes an input value gets a returned value.

We sometimes use the k-set-consensus object to denote the $(n+1, k)$-set-consensus object in a system of $n+1$ processes. The k-set-consensus model, is the shared memory model that processes can additionally access unlimited copies of k-set-consensus objects.

2.4 Simulation

In distributed computability, it is not uncommon to show that one problem can be reduced to another problem in a different model. In other words, one model can simulate the protocol of another model. If two models can simulate from each other, they are considered to be equivalent.

The BG simulation [4, 7], introduced by Borowsky and Gafni, allows a wait-free model of $(t+1)$ processes to simulate the t-resilient model of $n+1$ processes. More precisely, consider a system that consists of simulators $s_0, ..., s_t$ and a t-resilient model of processes $p_0, ..., p_n$. The BG simulation simulates a virtual execution of the t-resilient protocol wait-free among $t+1$ simulators.

The center building block of the BG simulation is the *safe-agreement* protocol. It allows processes to reach consensus at the cost of being possibly blocked by a faulty process. A safe-agreement protocol consists of two parts: an unsafe section and a safe section. Every process submits an input value and gets a return value such that the following properties are ensured:

1. Validity: every decided value is a proposed value.

2. Agreement: every process decides on the same value.

3. Weak termination: if no process fails in the unsafe section, every process eventually decides.

The original BG simulation only works for solving colorless tasks. It was later extended by Gafni [14] for colored tasks. The *GG simulation,* proposed by Gafni and Guerraoui [16], allows a shared-memory model that solves k-set-consensus to simulate a wait-free model of k processes consistently. The technique was later extended in [15, 17] to further make k simulated processes run the BG simulation. The combination of two simulations is called *BGG simulation*. It allows a wait-free model equipped with k-set-consensus objects to simulate the *k-concurrency model*, i.e., a shared memory model of $n+1$ processes such that at most k processes are active at any point of time. The BGG simulation can be used in *an active-resilient* model to simulate the *fair adversary* model [30, 32]. A detailed description of BGG simulation can be found in [17, 36].

Chapter 3

Topological Preliminaries

Topology was first applied in distributed computing to prove the impossibility of k-set-agreement [4, 26, 37] in 1993. Subsequently, it has achieved significant successes in distributed computability. In this chapter, we quickly review some basic notions in algebraic topology. We start with a short discussion of simplicial complex. Then, we provide some useful facts in algebraic topology and discuss the framework of topological methods in distributed computing. A complete treatment of this topic can be found in [24, 34, 40].

3.1 Simplicial Complex

Let V be a set, and let $\mathcal{K}$ be a finite collection of non-empty subsets of V. $\mathcal{K}$ is an *abstract simplicial complex* if it satisfies:

1. for any $v \in V$, $\{v\}$ is in $\mathcal{K}$;
2. if $\sigma \in \mathcal{K}$, then $\sigma' \in \mathcal{K}$ if $\sigma' \subseteq \sigma$.

Elements in V are called vertices, and every set in $\mathcal{K}$ is a *simplex.* A simplex σ' is called a *face* or a *subsimplex* if $\sigma' \subseteq \sigma$. A subset of $\mathcal{K}'$ is a *subcomplex* of $\mathcal{K}$ if $\mathcal{K}'$ is also a simplicial complex. The dimension of a simplex $\sigma \in \mathcal{K}$ is the size of σ minus 1. The boundary of a simplex σ, denoted by $\partial\sigma$, is the faces in σ with the dimension of $\dim(\sigma) - 1$. We use n-simplex to denote a simplex with dimension n. A simplex is called a facet in $\mathcal{K}$ if it is not contained in any other simplex in $\mathcal{K}$. A simplicial complex $\mathcal{K}$ is *pure* of dimension n if all its facets have dimension n. The n-skeleton of a complex $\mathcal{K}$, denoted $\text{skel}^n \mathcal{K}$, is the subcomplex formed by all simplices in $\mathcal{K}$ of dimension n or less. We usually use Δ^n to represent the simplicial complex that consists of the standard n-simplices and its faces.

Let $\mathcal{K}$ and $\mathcal{L}$ be simplicial complexes. A *vertex map* $\varphi : \mathcal{K} \to \mathcal{L}$ maps vertices in $\mathcal{K}$ to $\mathcal{L}$. We say φ is a *simplicial map* if it additionally carries simplices to simplices, that is, for $\sigma \in \mathcal{K}$, $\varphi(\sigma)$ is simplex in $\mathcal{L}$, where

$$\varphi(\sigma) = \bigcup \varphi(\{v\}).$$

A simplicial map is *non-collapsing* or *dimension-preserving* if $\dim(\sigma) = \dim(\varphi(\sigma))$.

Let Δ^n be a standard n-simplex with labeled vertices $\{0, 1, ..., n\}$. A chromatic simplicial complex $\mathcal{K}$ of dimension n is a pure n-dimensional simplicial complex equipped with a non-collapsing simplicial map $\chi : \mathcal{K} \to \Delta^n$. Let $\mathcal{K}$ and $\mathcal{L}$ be chromatic simplicial complexes. A simplicial map $\varphi : \mathcal{K} \to \mathcal{L}$ is called *color-preserving* or *chromatic* if for every $v \in \mathcal{K}$, $\chi(v) = \chi(\varphi(v))$.

A *carrier* map Γ between $\mathcal{K}$ and $\mathcal{L}$ is a map

$$\Gamma : \mathcal{K} \to 2^{\mathcal{L}},$$

that maps each simplex σ in $\mathcal{K}$ to a subcomplex $\Gamma(\sigma)$ in $\mathcal{L}$ monotonically. In other words, if $\tau \subseteq \sigma$, then $\Gamma(\tau) \subseteq \Gamma(\sigma)$. A carrier map Γ is *rigid* if $\dim(\sigma) = \dim(\Gamma(\sigma))$. A carrier map Γ is *chromatic* if Γ is rigid and $\chi(\sigma) = \chi(\Gamma(\sigma))$. A simplicial map $\varphi : \mathcal{K} \to \mathcal{L}$ is *carried* by a carrier map Γ or *carrier-preserving* if for every $\sigma \in \mathcal{K}$, $\varphi(\sigma) \in \Gamma(\sigma)$. Fix a carrier map $\Gamma : \mathcal{K} \to 2^{\mathcal{L}}$, and let τ be a simplex in $\mathcal{L}$, the *carrier* of τ in $\mathcal{K}$, denoted $\mathrm{Car}(\tau, \mathcal{K})$ is a simplex $\sigma \in \mathcal{K}$ with the smallest dimension such that $\tau \in \Gamma(\sigma)$.

Let $X = \{\mathbf{x}_1, ..., \mathbf{x}_{n+1}\}$ be a set of $n+1$ *affine independent points* in an Euclidean space. The *convex hull* of X, denoted by $\mathrm{conv}(X)$, is a point set such that every point can be expressed by

$$v = \sum_{i=1}^{n} a_i x_i$$

where $0 \leq a_i \leq 1$, $\sum_{i=0}^{n} a_i = 1$. The standard geometric n-simplex $|\Delta^n|$ is the convex of hull of $n+1$ points $\mathbf{x}_1, ..., \mathbf{x}_{n+1}$ in $\mathbb{R}^{n+1}$ such that each $\mathbf{x}^i$ has the coordinate

$$\mathbf{x}^i_j = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{if } j \neq i \end{cases}$$

for $i \in \{1, ..., n+1\}$. The interior of a simplex σ is defined by $\operatorname{int}\sigma = |\sigma| - |\partial\sigma|$. A geometric simplicial complex $|\mathcal{K}|$ is a collection of geometric simplices such that

1. For every face σ of a simplex in $\mathcal{K}$, $\sigma \in \mathcal{K}$
2. For every two simplices $\sigma, \tau \in \mathcal{K}$, the intersection $\sigma \cap \tau$ is a face of each of them.

Given a geometric simplicial complex $|\mathcal{K}|$, we can define its underlying abstract simplicial complex $\mathcal{K}$ as follows: the vertices in $\mathcal{K}$ is the union of vertices for every simplex in $|\mathcal{K}|$. The simplices in $\mathcal{K}$ are the sets of vertices in X for every simplex

$\sigma = \text{conv}(X)$ in $|\mathcal{K}|$. Given an abstract simplicial complex $\mathcal{K}$ with finite vertices, there exists many geometric simplicial complex $|\mathcal{K}|$ such that its underlying abstract complex is $\mathcal{K}$, called the *geometric realizations* of $\mathcal{K}$.

The *star* of a simplex σ in $\mathcal{K}$, denoted $\text{St}(\sigma, \mathcal{K})$, is the simplicial complex that consists of every simplex τ that contains σ and any face of τ. When $\mathcal{K}$ is clear from the context, we use $\text{St}(\sigma)$ to denote the star of σ in $\mathcal{K}$. The *open star* $\text{St}^\circ(\sigma)$ is the interior of the geometric realization of $\text{St}(\sigma)$.

The link of σ, denoted $\text{Lk}(\sigma, \mathcal{K})$, is the simplicial complex that consists of simplices in $\text{St}(\sigma, \mathcal{K})$ that does not intersect with σ. We use $\text{Lk}(\sigma)$ to represent the link of σ when $\mathcal{K}$ is evident in the background.

Let $\mathcal{K}$ be a simplicial complex. A complex $\mathcal{K}'$ is called a subdivision of $\mathcal{K}$ if

1. For each simplex $\tau \in \mathcal{K}'$, there is a simplex $\sigma \in \mathcal{K}$ such that $\tau \subseteq |\sigma|$.
2. Every simplex $\sigma \in \mathcal{K}$ is the union of finitely many simplices of $\mathcal{K}'$.
3. $|\mathcal{K}| = |\mathcal{K}'|$.

Let $\mathcal{K}$ be a geometric simplicial complex, its *barycentric subdivision* $\text{Bary}(\mathcal{K})$, is inductively constructed over the skeletons of $\mathcal{K}$ as follows: for the i-skeleton of $\mathcal{K}$, insert the barycenter b_σ of each i-dimensional simplex σ, and connect b_σ with the vertices in $\text{Bary}(\partial\sigma)$. The barycentric subdivision can also be defined combinatorially. Let $\mathcal{K}$ be an abstract simplicial complex, $\text{Bary}(\mathcal{K})$ is an abstract simplicial complex whose vertices are non-empty simplices of $\mathcal{K}$. Every simplex in $\text{Bary}(\mathcal{K})$ is a set $\{\sigma_0, ..., \sigma_k\}$ that can be linearly ordered as $\sigma_0 \subset ... \subset \sigma_k$. We use $\text{Bary}^N(\mathcal{K})$ to denote the Nth iteration of the subdivision of $\mathcal{K}$.

Let Δ^n be a standard n-simplex equipped with a coloring χ. The *standard chromatic subdivision* $\text{Ch}(\Delta^n)$ is a subdivision of Δ^n with a canonical coloring. The

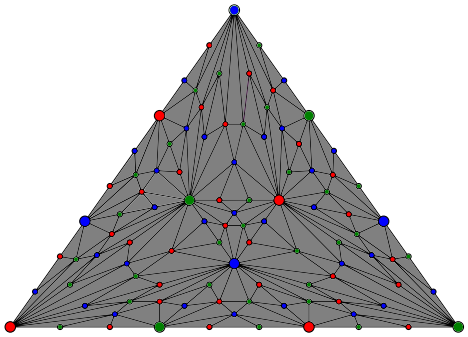


Figure 3.1: The second iteration of standard chromatic subdivision of Δ^2

vertices in $\mathrm{Ch}(\Delta^n)$ can be represented as a pair (i, σ), where σ is a face of Δ^n and $i \in \chi(\sigma)$. The simplices are the sets $\{(i_0, \sigma_0), ..., (i_k, \sigma_k)\}$ such that

1. elements can be linearly ordered as $\sigma_0 \subseteq ... \subseteq \sigma_k$
2. if $i \in \chi(\sigma_j)$, then $\sigma_i \subseteq \sigma_j$.

The coloring of $\mathrm{Ch}(\Delta^n)$ is defined by $\chi(i, \sigma) = i$. Let $|\Delta^n| = \mathrm{conv}(\mathbf{x}_0, ..., \mathbf{x}_n)$ be the standard geometric n-simplex. The geometric realization of $\mathrm{Ch}(\Delta^n)$ can be constructed by identifying each vertex (i, σ) as a point

$$\frac{1}{2k-1}\mathbf{x}_i + \frac{2}{2k-1}\left(\sum_{\substack{j \in \sigma: \\ j \neq i}} \mathbf{x}_j\right)$$

in $\mathbb{R}^{n+1}$, where $k = \dim(\sigma) + 1$. Given a chromatic complex $\mathcal{K}$, the chromatic subdivision $\mathrm{Ch}(\mathcal{K})$ is constructed by applying Ch on each simplex σ in $\mathcal{K}$. We use $\mathrm{Ch}^N(\mathcal{K})$ to denote the Nth iteration of the chromatic subdivision of $\mathcal{K}$. Note that subdivision operators defines the carrier map $\mathrm{Ch} : \mathcal{K} \to \mathrm{Ch}(\mathcal{K})$ and $\mathrm{Bary} : \mathcal{K} \to \mathrm{Bary}(\mathcal{K})$.

3.2 Results in Topology

In this section, we review some useful facts in point-set topology and algebraic topology. The classical reference for the material can be found in [22, 33].

The pasting lemma says that continuous functions on two subspaces can be combined into single one if they coincide on the intersection.

Theorem 3.1 (Pasting lemma)**.** *Let $X = A \cup B$ be a topological space, where A and B are closed in X. Let $f : A \to Y$ and $g : B \to Y$ be continuous functions, and $f(x) = g(x)$ for $x \in A \cap B$. Then the function $h : X \to Y$ defined by*

$$h(x) = \begin{cases} f(x) & \text{if } x \in A \\ g(x) & \text{if } x \in B \end{cases}$$

is continuous.

A simplicial map can be considered as the discrete version of a continuous map.

Definition 3.2. Let $\mathcal{A}$ and $\mathcal{B}$ be simplicial complexes, and let $f : |\mathcal{A}| \to |\mathcal{B}|$ be a continuous map between their geometric realizations. A simplicial map $\phi : \mathcal{A} \to \mathcal{B}$ is a *simplicial approximation* of f if and only if for every vertex v of $\mathcal{A}$, $f(\mathrm{St}^\circ(v)) \subseteq \mathrm{St}^\circ(\phi(v))$.

In general, an arbitrary continuous map $f : |\mathcal{A}| \to |\mathcal{B}|$ may not admit a simplicial approximation. However, if we replace $\mathcal{A}$ with a fine enough subdivision, we can construct a simplicial approximation of f.

Theorem 3.3. *Let $\mathcal{A}$ and $\mathcal{B}$ be simplicial complexes such that $\mathcal{A}$ is finite. Let $f : |\mathcal{A}| \to |\mathcal{B}|$ be a continuous map. There exists an integer N such that for all $n > N$, f has a simplicial approximation $\phi : \mathrm{Bary}^N(\mathcal{A}) \to \mathcal{B}$.*

Note that the barycentric subdivision in Theorem 3.3 can be replaced with standard chromatic subdivision $\mathrm{Ch}^N(\mathcal{A})$.

In topology, two spaces are considered to be equivalent if there is a homeomorphism between them.

Definition 3.4. Let X and Y be topological spaces. X is *homeomorphic* to Y if there is a continuous map $f : X \to Y$ such that f has a continuous inverse.

In algebraic topology, one of the primary studies is a weaker equivalence relation, called homotopy.

Definition 3.5. Let X and Y be topological spaces, and let f and g be two continuous maps from X to Y. f is *homotopic* to g if and only if there is a continuous function $H : X \times [0, 1] \to Y$ such that for all $x \in X$, $H[x, 0] = f(x)$ and $H[x, 1] = g(x)$.

Roughly speaking, being homotopic means one map can be continuously deformed into another. Homotopy defines an equivalence relation among continuous maps from X to Y. The notion of homotopy equivalence can be defined between two topological spaces, a weaker but much broader relation than homeomorphism.

Definition 3.6. Let X and Y be topological spaces. X is homotopy equivalent to Y, denoted by $X \simeq Y$, if there are continuous maps $f : X \to Y$ and $g : Y \to X$, such that $f \circ g \simeq \mathbf{id}_X$ and $g \circ f \simeq \mathbf{id}_Y$ where $\mathbf{id}_X$ and $\mathbf{id}_Y$ are identify maps on X and Y respectively.

Another important topological invariant is n-connectedness. Let S^ℓ denote the n-sphere and D^n denote the n-disk.

Definition 3.7. Let $\mathcal{K}$ be a simplicial complex. $|\mathcal{K}|$ is *n-connected* if for all $0 \leq \ell \leq n$, any continuous map $f : S^\ell \to |\mathcal{K}|$ can be extended to a continuous map $F : D^{\ell+1} \to |\mathcal{K}|$.

We say $\mathcal{K}$ is (-1)-*connected* if it is nonempty. $\mathcal{K}$ is *path-connected* or 0-*connected* if there is a path between any two points. An alternative definition of connectivity uses *homotopy groups*. The *nth homotopy group* of a general topological space X [22], roughly speaking, is the group that elements are homotopy equivalence classes of maps from S^n to X. X is n-connected if and only if its first nth homotopy groups are trivial. If X and Y are homotopy equivalent, they have isomorphic homotopy groups. Consequently, they have the same connectivity.

Fact 3.8. *Let X and Y be topological spaces and homotopy equivalent. X is n-connected if and only if Y is n-connected.*

Here are some useful facts about connectivity in simplicial complex [24].

Fact 3.9. *The standard simplex Δ^n is n-connected.*

Fact 3.10. *The k-skeleton of a n-simplex* $\mathrm{skel}^k(\Delta^n)$ *is $(k-1)$-connected.*

Many simplicial complexes have a local connectivity property, called link-connectivity.

Definition 3.11. Let $\mathcal{K}$ be a pure n-dimensional simplicial complex. $\mathcal{K}$ is *link-connected*, if for all $\sigma \in \mathcal{K}$, $\mathrm{Lk}(\sigma, \mathcal{K})$ is $(n - \dim(\sigma) - 2)$-connected.

The following maps are important in topology.

Definition 3.12. Let A be a subspace of X. A continuous map $f : X \to A$ is a *retraction* if f restrict to A is the identity map on A.

The notion of deformation retraction captures the process of continuously shrinking a space into a subspace.

Definition 3.13. A *deformation retraction* H of space X onto A is a homotopy between the identity map and a retraction. Formally, H is a continuous map $H : X \times [0,1] \to X$ such that $H[-,0] = \mathbf{id}_X$ and $H[-,1] = f$ where $\mathbf{id}_X$ is the identity map on X and f is a retraction onto A. We say A is a deformation retract of X if such H exists.

Deformation retraction is a powerful tool to show homotopy equivalence between two topological spaces.

Theorem 3.14. *Let $A \subseteq X$ be topological spaces. If there is a deformation retraction of X onto A, then X and A are homotopy equivalent.*

3.3 Topological Characterization of Task Solvability

Using the language of simplicial complex, we can define tasks topologically. A task $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ is a triple such that the input complex $\mathcal{I}$ embeds the input vectors. For every simplex $\sigma = \{v_0, v_1, ..., v_n\}$ in $\mathcal{I}$, each vertex v_i is identified as a pair (i, val_i) in the input vector. Similarly, the output complex $\mathcal{O}$ consists of all output vectors. It is easy to verify that $\mathcal{I}$ and $\mathcal{O}$ are indeed simplicial complexes: for every simplex σ in $\mathcal{I}$ and $\mathcal{O}$, any face τ of σ is a simplex since τ corresponds to the configuration where only a subset of processes in σ participates. The task specification $\Gamma : \mathcal{I} \to 2^{\mathcal{O}}$ is a carrier map that maps each simplex $\sigma \in \mathcal{I}$ to a subcomplex in $\mathcal{O}$.

Given a model and an execution of a protocol in this model, we can embed the local state of a process as a vertex. The set of compatible local states among different processes is represented as a simplex. If we embed every possible execution of the protocol as a simplex, the result representation is a simplicial complex, called the

protocol complex. Some processes may share the same local states in two different executions, which corresponds to a shared face between two simplices in the protocol complex. The protocol complex gives a concise and topological representation of the computation in a distributed model. To investigate task solvability, one can study simplicial maps between the protocol complex and task output complex.

In the wait-free model, it is natural to consider the protocol complex of the atomic-snapshot model. The drawback of this approach is that its protocol complex has a complicated structure [27, 29] and may not be embedded in $\mathbb{R}^{n+1}$ for a system of $(n+1)$ processes. However, the iterated immediate snapshot (IIS) model is easier to analyze since it has a round-by-round structure. Moreover, the protocol complex of IIS can be represented as iterated standard chromatic subdivision [24, 28]. If processes execute N rounds immediate snapshot, then the protocol complex is $\mathrm{Ch}^N(\mathcal{I})$. The asynchronous computability theorem [6, 27] reduces the wait-free task solvability to the existence of a certain simplicial map from iterated chromatic subdivision to the task output complex. This approach is later generalized to many other models [17, 31, 32, 38]. In Chapters 5 and Chapter 6, we discuss how to apply this approach to extend ACT on more general model.

Chapter 4

Adaptive Set-Agreement Task and Vector-Set-Consensus

The k-set-agreement is a task that processes synchronize to return at most k different values from their inputs. Delporte et al. [11] and Kuznetsov et al. [30] considered solving set-agreement adaptively according to the participating set in the set-consensus collection model and the fair adversary model.

In this chapter, we first discuss the adaptive set-agreement task, in which the number of different outputs depends on the size of the participating set. Then, we discuss the set-consensus collection model and introduce the vector-set-consensus model. We prove that every set-consensus collection model is equivalent to a vector-set-consensus model for task solvability. We also discuss the iterated adaptive set-agreement task and present a protocol in the vector-set-consensus model.

4.1 Adaptive set-agreement task

In the k-set-agreement task, every process chooses an output from the inputs of participating processes such that the number of different output values is less or equal to k. The adaptive set-agreement task generalizes the k-set-agreement, in which the number of different outputs depends on the size of the participating set.

Definition 4.1. Let $A = k_1, k_2, ..., k_{n+1}$ be a non-decreasing sequence, where for each $i = 1, 2, ..., n+1$, $k_i \in \mathbb{N}$ and $1 \leq k_i \leq i$. The adaptive set-agreement T_A is the task such that every process has an input value and decides an output satisfying the following properties:

1. **Termination**: Every correct process eventually decides an output value.
2. **Validity**: Every decided value is an input value of a participating process.
3. **Adaptive-agreement**: If k_m different outputs are decided at some point of time t, the size of the participating set at t is at least m.

The motivation of the constrain $1 \leq k_i \leq i$ is clear: if at most i processes participate, they can trivially solve i-set-agreement by letting every process choose its input value. The strongest set-agreement that a model can solve is the 1-set-agreement, also known as consensus. The standard k-set-agreement corresponds to the adaptive set-agreement where each k_i is the same constant number.

We also considered the iterated adaptive set-agreement task. An *active* process is a participating process but not with an output. In iterated adaptive-set-agreement, processes solve adaptive set-agreement round by round among active processes. Every process p_i submits a new input value in each round and chooses an output value.

After one round, p_i determines whether it halts or continues to the next adaptive set-agreement. The number of different values chosen in each round depends on how many active processes participate.

Definition 4.2. Let $A = k_1, k_2, ..., k_{n+1}$ be an non-decreasing sequence such that $1 \leq k_i \leq i$ for $i = 1, 2, ..., n+1$. The iterated adaptive set-agreement T_A is an infinite sequence of adaptive set-agreement tasks $T_{A_1}, T_{A_2}, ...$ such that for $r = 1, 2, 3...$, the following properties are satisfied:

1. **Termination**: Every correct process participates T_{A_r} eventually decides an output value.
2. **Validity**: Every decided value in T_{A_r} is an input value of a process that participates T_{A_r}.
3. **Adaptive-agreement**: If k_m different outputs for T_{A_r} are decided at some point of time t, the number of processes that participate T_{A_r} at t is at least m.

4.2 Set-Consensus Collection and Vector-Set-Consensus

Set-Consensus Collection

Many distributed models use extra synchronization primitives. For example, in k-set-consensus model, processes can use the k-set-consensus object in addition to the shared memory. The set-consensus collection [11] generalizes k-set-consensus model by including multiple types of set-consensus objects. Formally, a set-consensus collection model is defined as a set of pairs $\mathcal{SC} = \{(\ell_1, j_1), (\ell_2, j_2), ..., (\ell_t, j_t)\}$ such

that the system has unlimited copies of (ℓ_i, j_i)-set-consensus object where each allows ℓ_i processes to solve j_i-set-agreement. Without loss of generality, we assume $\ell_i < \ell_{i+1}, j_i < j_{i+1}$ and $j_i < \ell_i$

The agreement level $AL_m^{\mathcal{SC}}$, introduced in [11], is the smallest number k such that k-set-agreement can be solved among m processes in the set-consensus collection model, while $(k-1)$-set-agreement can not. It is shown in [11] that $AL_m^{\mathcal{SC}}$ is the minimum number of $\sum_i j_i x_i$ under the constrain $\sum_i \ell_i x_i \geq m$ for $m = 1, 2, ..., n+1$.

Vector-set-consensus

To characterize task solvability in the set-consensus collection model, we introduce the vector-set-consensus model.

Definition 4.3. The vector-set-consensus model $\mathcal{VSC} = (k_1, k_2, ..., k_{n+1})$ is a set-consensus collection model such that the collection consists of (m, k_m)-set-consensus objects for $m = 1, 2, ...n+1$, and the following property is satisfied:

- **Optimality**: For each $m = 1, 2, ..., n+1$, for any multiset

$$S = (a_1, k_{a_1}), (a_2, k_{a_2}), ..., (a_s, k_{a_s})$$

where each $a_i \in \{1, 2, ...n+1\}$, we have $\sum_i k_{a_i} < k_m$ and $\sum_i a_i \geq m$.

Next, we prove some properties of the vector-set-consensus model.

Proposition 4.4. *Let $\mathcal{VSC} = (k_1, k_2, ..., k_{n+1})$ be a vector-set-consensus model. For each k_i and k_j where $i < j$, $k_i \leq k_j \leq k_i + (j - i)$.*

Proof. We first show $k_i \leq k_j$. By way of contradiction, assume that $k_i > k_j$. The optimality property is violated since processes can use (j, k_j)-set-consensus objects to solve $(i, k_i - 1)$-set-agreement. To see $k_j \leq k_i + (j - i)$, notice that j processes can solve $k_i + (j - i)$-set-agreement using (i, k_i)-set-consensus object: assign i processes to access a (i, k_i)-set-consensus object to solve the k_i-set-agreement and let the remaining $(j - i)$ processes return its input values. By optimality property of $\mathcal{VSC}$, we have $k_j \leq k_i + (j - i)$. □

ℓ-safe-agreement

We now discuss the ℓ-safe-agreement [10, 35], which generalizes the safe-agreement [7]. A ℓ-safe-agreement protocol solves ℓ-set-agreement but requires a weaker termination property: a process may never return an output if ℓ processes fail.

Algorithm 1: ℓ-safe-agreement for process p_i

Shared Variables: SM[0..n], init $\perp$: single-writer multi-reader memory

```
1  SM[i] ← (v_i, 1)
2  snap ← snapshot(SM)
3  if ∃ snap[j] ∈ snap where snap[j] = (−, 2) then
4  |  snap[i] ← (v_i, 0)
5  else
6  |  snap[i] ← (v_i, 2)
7  repeat
8  |  snap ← snapshot(SM)
9  |  t ← |{j | snap[j] = (−, 1)}|
10 until t ≤ ℓ − 1
11 return snap[j].v_j where j is the smallest index such that
     snap[j].level = 2
```

The ℓ-safe-agreement protocol is presented in Algorithm 1. The algorithm is a generalization of the safe-agreement protocol in [7]. It uses a shared memory array SM[0..n] where each SM[i] contains a pair (v_i, level_i) that is uniquely written by process p_i. The unsafe region consists of lines 1-6. In the beginning, p_i updates its proposal and reaches level 1 at line 1. Then, p_i takes a snapshot of the memory to check whether there is a process at level 2 . If so, p_i retreats to level 0; otherwise, it advances to level 2 (lines 3-6). Next, at lines 7-10, p_i repeatedly read the memory and count the number of processes at level 1. If p_i observes less than ℓ processes stay at level 1, it returns a level-2 value with the smallest id in its snapshot.

Once a process exits the unsafe section, no other process can reach level 2. Therefore, the correctness follows from the fact that each process only chooses a level-2 value as its output, and every process misses at most $\ell - 1$ level-2 values when it decides. To see the liveness property, notice that processes can only be blocked at lines 7-10 when at least ℓ processes stay in the unsafe region. Hence, if at most $\ell - 1$ process fail, every process eventually terminates.

We now show that the set-consensus power of $\mathcal{VSC}$ is fully grasped by the sequence $(k_1, k_2, ..., k_{n+1})$.

Theorem 4.5. *Let $\mathcal{VSC} = (k_1, k_2, ..., k_{n+1})$ be a vector-set-consensus model. For $m = 1, 2, ..., n + 1$, m processes can not solve $(k_m - 1)$-set-agreement task in $\mathcal{VSC}$.*

Proof. (Adapted from [11]) By way of contradiction, suppose there is a protocol for m processes to solve $(k_m - 1)$-set-agreement. We use the BG simulation to construct a wait-free $(k_m - 1)$-set-agreement protocol for k_m simulators, which contradicts the impossibility of k-set-agreement [4, 26, 37]. Without loss of generality, we assume that every process's write in the $\mathcal{VSC}$ protocol is uniquely determined by its snapshots and values returned from accessing set-consensus-objects. In BG simulation, simulators

$s_1, s_2, ..., s_{k_m}$ repeatedly simulate steps of m processes $p_1, p_2, ..., p_m$ in a round-robin manner.

1. The snapshot operation of p_j is simulated by invoking the safe-agreement protocol.

2. Each simulator uses the ℓ-safe-agreement to simulate the set-consensus objects in $\mathcal{VSC}$. It submits an estimate to the t-safe-agreement for each (t, k_t)-set-consensus object access. The t-safe-agreement guarantees at most t different proposals are returned. Since simulators may get different values for simulating the same access, they further use safe-agreement to agree on a returned value for each access.

3. If the safe-agreement or the ℓ-safe-agreement is not resolved for the simulation of process p_j, the simulator selects the next process in a round-robin way.

It is not hard to see the correctness of the simulation since the safe-agreement produces a consistent snapshot simulation, and ℓ-safe-agreement simulates a set-consensus-object.

To show the liveness property, notice that the simulation may be blocked if simulators fail in the execution of safe-agreement or ℓ-safe-agreement. At most one simulated process will be blocked if one simulator fails in the safe-agreement. At most t processes may be blocked for accessing a (t, k_t)-set-consensus object if k_t simulators fail in the corresponding k_t-safe-agreement. Hence, it suffices to prove that at least one simulated process is not blocked. By way of contradiction, assume m processes are blocked. Then, there exists a multiset

$$\{(t_1, \ell_1), (t_2, \ell_2)...(t_q, \ell_q)\}$$

such that each $t_i \in \{1, 2, ...n+1\}$, $\ell_i = k_{t_i}$. Each (t_i, ℓ_i) represents a non-resolved ℓ_i-safe-agreement and blocks t_i simulated processes. Moreover, we have

$$\sum_i \ell_i \leq k_m - 1, \ \sum_i t_i \geq m$$

since at most $k_m - 1$ simulators fail in the wait-free model. However, this contradicts the optimality property of vector-set-consensus model. Therefore, at most $(m-1)$ processes are blocked, and the simulation produces a wait-free set-agreement protocol. □

In distributed computing, it is not uncommon to compare the power of solving tasks between two models. We say a model M is as powerful as M', denoted by $M \preceq M'$, if for every task T solvable in M, there exists a protocol solves T in model M'. If $M \preceq M'$ and $M' \preceq M$, then M and M' are equivalent regarding task solvability. Let $\mathcal{VSC} = (k_1, k_2, ..., k_{n+1}), \mathcal{VSC}' = (k'_1, k'_2, ..., k'_{n+1})$ be two vector-set-consensus models. It is not hard to see that $\mathcal{VSC} \preceq \mathcal{VSC}'$ if $k_i \geq k'_i$ for $i = 1, 2, ..., n+1$. Hence, $\preceq$ induces a partial order in vector-set-consensus models.

Given a set-consensus collection model $\mathcal{SC}$, we can construct an equivalent vector-set-consensus mode

Definition 4.6. Let $\mathcal{SC}$ be a set-consensus collection model. The corresponding vector-set-consensus model $\mathcal{VSC} = (k_1, k_2, ..., k_{n+1})$ of $\mathcal{SC}$ satisfies that for each $m = 1, 2, ...n+1$, $k_m = AL_m^{\mathcal{SC}}$ where $AL_m^{\mathcal{SC}}$ denotes the agreement level of $\mathcal{SC}$ for m processes.

It is not hard to see that $\mathcal{SC}$ and its corresponding $\mathcal{VSC}$ model have the same power for solving tasks.

Theorem 4.7. *A task T is solvable in set-consensus collection model $\mathcal{SC}$ if and only if T is solvable in the vector-set-consensus model $\mathcal{VSC}$.*

Proof. We show that the access of the set-consensus objects can be simulated from each other. If a (m, k_m)-set-consensus object in $\mathcal{VSC}$ is included in $\mathcal{SC}$, processes access it verbatim. If it is not included in the $\mathcal{SC}$, by definition of $\mathcal{VSC}$, there exists a multiset $S = (a_1, b_1), (a_2, b_2), ..., (a_t, b_t)$ where each (a_i, b_i) is a set-consensus object in $\mathcal{SC}$ such that $\sum_i b_i = k_m$ and $\sum_i a_i \geq m$. Let each process p_i access the (a_j, b_j)-set-consensus object in $\mathcal{SC}$ with the smallest $j \in \{1, 2, ..., t\}$ such that $\sum_{t=1}^{t=j} a_j \geq i$ to simulate the (m, k_m)-set-consensus object. The number of different outputs is bounded by k_m since $\sum_i b_i = k_m$.

The other direction is straightforward. For each (ℓ, j)-set-consensus object in $\mathcal{SC}$, by optimality property, there is a (ℓ, j')-set-consensus object in $\mathcal{VSC}$ where $j' \leq j$. Processes simply use (ℓ, j)-set-consensus object instead of (ℓ, j)-set-consensus objects in $\mathcal{SC}$. Since $\mathcal{SC}$ and $\mathcal{VSC}$ can simulate protocols from each other, two models solve the same set of tasks. □

Note that it is not always the case that two vector-set-consensus models are comparable. For example, consider a system with 6 processes. Let $\mathcal{VSC}$ and $\mathcal{VSC}'$ be the corresponding vector-set-consensus models for $\mathcal{SC} = \{(3, 2)\}$ and $\mathcal{SC}' = \{(5, 3)\}$. A straightforward calculation shows that $\mathcal{VSC} = (1, 2, 2, 4, 4, 4)$ and $\mathcal{VSC}' = (1, 2, 3, 3, 3, 4, 5)$. They are incomparable since $\mathcal{VSC}$ can not solve $(5, 3)$-set-agreement and there is no $(3, 2)$-set-agreement protocol in $\mathcal{VSC}'$.

Solving Iterated Adaptive-set-Agreement

Let T_A be an iterated adaptive-set-agreement such that $A = k'_1, k'_2, ..., k'_{n+1}$. We show that T_A is solvable in the vector-set-consensus model $\mathcal{VSC} = (k_1, k_2, ..., k_{n+1})$ if $k_i \leq k'_i$ for each $i = 1, 2, ..., n+1$. The protocol is presented in Algorithm 2, which is adapted from [11]. Note that the algorithm solves one-shot adaptive set-agreement by making every process terminates after the first round.

The shard memory SM[1..][0...n] stores the input value for each set-agreement where SM[r][i] contains p_i's proposal at the rth round. V[1..][1...$n+1$] consists of the set-consensus objects where each $V[r][m]$ is a (m, k_m)-set-consensus object.

At lines 1-2, every process p_i updates its input value for the current round r with level $\ell = 0$ and then takes a snapshot of the memory. Next, p_i calculates the participating set and adopts an input value v_r from another process at the highest level (lines 5-7). At line 8, p_i proposes the input value to the $(|\text{part}|, k_{|\text{part}|})$-set-consensus object and updates v_r as an output estimation. Then, p_i updates SM[r][i] with $(v_r, |\text{part}|)$ to indicate it is at level $\ell = |\text{part}|$ and takes a snapshot of the memory to recalculate the participating set (lines 9-11). If the new participating is the same as the previous one, p_i exits the loop and writes its output estimation in out[r] at line 13. Otherwise, p_i repeatedly executes the loop until it observes the same participating set twice. At lines 14-15, p_i determines whether it terminates the computation or solves the next round adaptive set-agreement.

Lemma 4.8. *Algorithm 2 satisfies the Termination property.*

Proof. By way of contradiction, suppose that a process p_i participates round r and never writes its output value at line 13, p_i must execute the loop (lines 4–12) infinitely often. Hence, p_i observed part$' \neq$ part at line 12 infinitely often. Notice that once

Algorithm 2: Iterated adaptive-set-agreement algorithm: process p_i

Shared Variables: SM[1..][0..n], init $(\perp, \perp)$: single-writer multi-reader memory
V[1...][1..$n+1$]: vector-set-consensus objects

Local Variables: snap[1..][0..n], init $(\perp, \perp)$
out[1..]: adaptive-set-agreement ouput
part, part$' \subseteq \{0, 1, ..., n\}$
v_r: p_i's input at round r
$r, l \in \mathbb{N}$, init 0

```
1  repeat
2      SM[r][i] ← (v_r, 0)
3      snap[r] ← snapshot(SM[r])
4      repeat
5          part ← {p_j | snap[r][j] ≠ (⊥, ⊥)}
6          ℓ ← the greatest integer such that (−, ℓ) is in snap[r]
7          v_r ← any v such that snap[r][j] = (v, ℓ)
8          v_r ← propose v to the (|part|, k_|part|)-set-consensus in
               V[r][|part|]
9          SM[r][i] ← (v_r, |part|)
10         snap[r] ← snapshot(SM[r])
11         part' ← {p_j | snap[r][j] ≠ (⊥, ⊥)}
12     until part' = part
13     out[r] ← v_r
14     if Terminate(p_i) = true then
15         return decide(p_i)
16     r ← r + 1
17 forever
```

SM$[r][j]$ is updated by process p_j with a non-$\perp$ value, it is never changed to $(\perp, \perp)$. Therefore, it must be the case that part $\subsetneq$ part$'$ when p_i evaluates line 12. However, the growth of part$'$ is bounded since at most $n+1$ processes participate round r. Thus, p_i executes lines 4–12 at most $n+1$ times, a contradiction. □

Lemma 4.9. *Algorithm 2 satisfies the Validity property.*

Proof. Fix round r, let v_r be the output value written by p_i at line 13. Let part be the participate set p_i observed at line 11 before it exits the loop lines 4–12, and let $m = |\text{part}|$. The value v_r is either p_i's initial input value at line 2, an input value adopted from another process p_j at round r at line 7, or a value returned from accessing the set-consensus object $V[r][m]$ at line 8. By the inclusion property of immediate snapshot, at most m processes observe a participating set of m in round r. Therefore, at most m processes access the (m, k_m)-set-consensus object in round r, and so, the value returned from the set-consensus object is a valid input value. Hence, in either way, v_r satisfies the validity property. □

Lemma 4.10. *Algorithm 2 satisfies the Adaptive-agreement property*

Proof. Let p_i be the first process that produces an output value v_r at round r. In other words, p_i is the first process that observes part$'$ = part and exits the loop. Assume p_i takes the last snapshot at time t, and let part$_i$ be the participate set that p_i observed such that $m = |\text{part}_i|$. Notice that at time t, the number of processes that participate round r is m. Let O_m be the set of values ever written in SM$[r]$ at level m, i.e., for every $v \in O_m$, (v, m) is a value that appeared in SM$[r]$ in the execution of the algorithm. We say p_s returns at level m in round r if p_s participates round r and observes a participating set part$_s$ of size m when it exits the loop at line 12.

We show that if a value (v', ℓ') was ever written SM$[r]$ such that $\ell' > m$, then $v' \in O_m$. By way of contradiction, suppose that there is a process p_j participates round r and is the first to write a (v', ℓ') in SM$[r]$ such that $\ell' > m$ and $v' \notin O_m$. Before this write, p_j calculates a participating set part$_j$ with a size greater than m from its snapshot taken at line 3 or line 10. Notice that this snapshot is taken after any process p_s returns at level m. Since every such process p_s writes a value (v_s, m) in SM$[r]$ before it exits the loop, when p_j calculates part$_j$ at line 5, it must observe that m is the highest level in SM$[r]$. So, p_j will adopt a value from O_m at line 7, and likewise, any value proposed to the set-consensus object in $V[r][|\text{part}_j|]$ is a value taken from O_m. Therefore, p_j must write a value $v' \in O_m$ in SM$[r]$, contradicting the hypothesis.

Since every value in O_m is returned from the (m, k_m)-set-consensus object, there are at most k_m different values in O_m. Hence, the number of different outputs returned at round r is bounded by k_m, which shows the algorithm satisfies the adaptive-agreement property since $k_m \leq k'_m$. □

Theorem 4.11. *Let T_A be an iterated adaptive set-agreement task specified by $A = k', k'_2, ..., k'_{n+1}$. Algorithm 2 solves T_A in the vector-set-consensus model $\mathcal{VSC} = (k_1, k_2, ..., k_{n+1})$ if for $i = 1, 2, ..., n+1$, $k_i \leq k'_i$*

Chapter 5

Computability Theorem for Vector-Set-Consensus

The celebrated asynchronous computability theorem (ACT) theorem [27] says that the wait-free task solvability is equivalent to the existence of a certain simplicial map from a chromatic subdivision to the task's output complex.

Theorem 5.1. *Let $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ be a colored task. T is solvable in the wait-free model if and only if there is a chromatic subdivision of the input complex* $\mathrm{Div}(\mathcal{I})$ *and a color-preserving simplicial map $\phi : \mathrm{Div}(\mathcal{I}) \to \mathcal{O}$ carried by Γ.*

An equivalent formulation of ACT was given by Borowsky and Gafni [6] using the iterated standard chromatic subdivision instead of an arbitrary chromatic subdivision.

Theorem 5.2. *Let $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ be a colored task. T is solvable in the wait-free model if and only if there exists a number $N \in \mathbb{N}$ and a color-preserving simplicial map $\phi : \mathrm{Ch}^N(\mathcal{I}) \to \mathcal{O}$ carried by Γ.*

As the iterated standard chromatic subdivision is the protocol complex of IIS, this formulation provides insight into the connection between the operational argu-

ment of solving tasks and their topological interpretations. Subsequently, ACT was generalized to fault-tolerant models [32, 38] and k-set-consensus model [17].

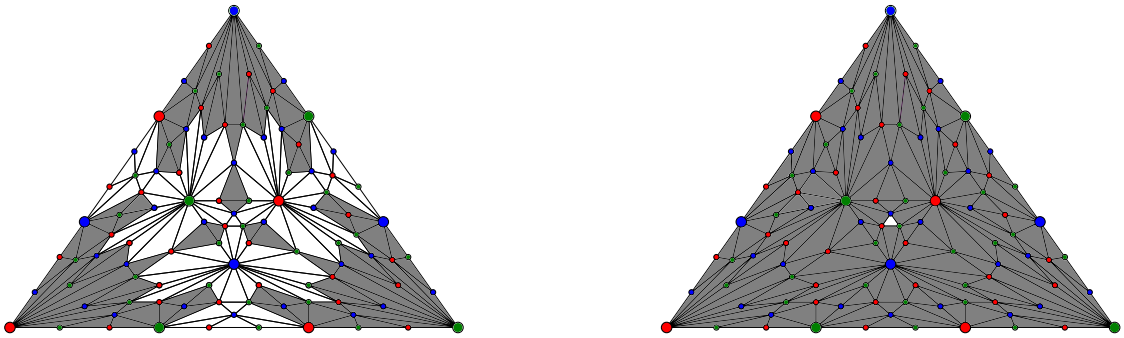
In this chapter, we present ACT for colorless task solvability in vector-set-consensus model. We first propose a protocol complex that characterizes solving $(n+1,k)$-set-agreement in a shared memory model. Then, using it as a building block, we construct the protocol complex that solves the adaptive set-agreement. We prove that solving colorless tasks in the vector-set-consensus model can be characterized by iterating this protocol complex: a task is solvable if and only if there exists a specific simplicial map from the iterated protocol complex to the task output complex. We also provide a characterization of the terminating iterated vector-set-consensus model.

Finally, for colored task solvability, we present an alternative formulation of ACT for k-set-consensus model [17]. Our characterization introduces a novel notion of color-projection, which shows an intriguing connection between tasks and objects.

5.1 Complex for $(n+1,k)$-set-agreement

Recall that in the input-less k-set-agreement task, every process has the input of its id and outputs a participating process id, such that the number of different ids decided is less or equal to k. Formally, the $(n+1,k)$-set-agreement is a task $T = (\Delta^n, \text{skel}^{k-1}(\Delta^n), \Gamma)$ where the input complex is the standard n-simplex Δ^n, and the output complex is the $(k-1)$ skeleton of Δ^n. For each simplex $\sigma \in \Delta^n$, the valid output simplices are $\Gamma(\sigma) = \text{skel}^{k-1}(\sigma)$. It is easy to see that Γ fixes the $(k-1)$ skeleton, i.e., $\Gamma(\text{skel}^{k-1}(\Delta^n)) = \text{skel}^{k-1}(\Delta^n)$.

We now present a model in which protocol complex characterizes the $(n+1,k)$-set-agreement. The complex is defined in two stages. First, we define a subcomplex



(a) $\mathrm{Ch}_1(\Delta^2)$ for $(3,1)$-set-agreement (b) $\mathrm{Ch}_2(\Delta^2)$ for $(3,2)$-set-agreement

Figure 5.1: Examples of protocol complex $\mathrm{Ch}_k(\Delta^n)$

in the barycentric subdivision $\mathrm{Bary}(\Delta^n)$. Then, we use it define the protocol complex $\mathrm{Ch}_k(\Delta^n)$.

Definition 5.3. Let $\mathrm{Bary}(\Delta^n)$ be the barycentric subdivision of the standard n-simplex, and let $k \in \{1, ..., n+1\}$. The complex $\mathrm{Bary}_k(\Delta^n)$ is defined as follows:

$$\mathrm{Bary}_k(\Delta^n) = \{\sigma \in \mathrm{Bary}(\Delta^n) \mid \forall v \in \sigma, \dim(\mathrm{Car}(v, \Delta^n) \geq k))\}.$$

Let $|\mathrm{Bary}_k(\Delta^n)|$and $|\mathrm{Ch}^2(\Delta^n)|$ be the standard geometric realization of $\mathrm{Bary}_k(\Delta^n)$ and $\mathrm{Ch}^2(\Delta^n)$. We now present the complex $\mathrm{Ch}_k(\Delta^n)$, which is the subcomplex of $\mathrm{Ch}^2(\Delta^n)$ whose simplices do not intersect with $|\mathrm{Bary}_k(\Delta^n)|$.

Definition 5.4. The complex $\mathrm{Ch}_k(\Delta^n)$ is a subcomplex of $\mathrm{Ch}^2(\Delta^n)$, defined by

$$\mathrm{Ch}_k(\Delta^n) = \left\{\sigma \in \mathrm{Ch}^2(\Delta^n) \mid |\sigma| \cap |\mathrm{Bary}_k(\Delta^n)| = \emptyset\right\}.$$

Operationally, the complex $\mathrm{Ch}_k(\Delta^n)$ can be considered as a subset of executions of two rounds immediate snapshot. Let M_k be an iterated model where in each round, the protocol complex is $\mathrm{Ch}_k(\Delta^n)$. We now prove that model M_k solves exactly

$(n+1,k)$-set-agreement: M_k solves $(n+1,k)$-set-agreement but not $(n+1,k-1)$-set-agreement.

To construct a protocol for $(n+1,k)$-set-agreement, we show that there is a continuous map from $|\operatorname{Ch}_k(\Delta^n)|$ to $|\operatorname{skel}^{k-1}(\Delta^n)|$ carried by the $(n+1,k)$-set-agreement carrier map Γ. Then, we turn the continuous map into a carrier-preserving simplicial map from the iterated protocol complex to output complex, which can be interpreted as a protocol in M_k operationally .

We first define a subspace in $|\Delta^n|$, which is useful for our construction.

Definition 5.5. The topological space $|\Delta^n|_k$ is the subspace of $|\Delta^n|$ by removing points in $|\operatorname{Bary}_k(\Delta^n)|$, defined by $|\Delta^n|_k = |\Delta^n| - |\operatorname{Bary}_k(\Delta^n)|$.

Here is the intuition behind this definition. In general, we can extend the definition of $\operatorname{Ch}_k(\Delta^n)$ in any iteration of standard chromatic subdivision. Indeed, for $N = 1,2,3...$, let $\operatorname{Ch}_{k,N}(\Delta^n)$ be the complex defined by

$$\operatorname{Ch}_{k,N}(\Delta^n) = \left\{\sigma \in \operatorname{Ch}^N(\Delta^n) \mid |\sigma| \cap |\operatorname{Bary}_k(\Delta^n)| = \emptyset\right\}.$$

It is not hard to see that $\operatorname{Ch}_k(\Delta^n) = \operatorname{Ch}_{k,2}(\Delta^n)$, and the sequence

$$|\operatorname{Ch}_{k,1}(\Delta^n)|, |\operatorname{Ch}_{k,2}(\Delta^n)|, ...$$

converges to $|\Delta^n|_k$. We now show that the space $|\operatorname{skel}^{k-1}(\Delta^n)|$ is a retract of $|\Delta^n|_k$.

Lemma 5.6. *There exists a retraction* $r : |\Delta^n|_k \to |\operatorname{skel}^{k-1}(\Delta^n)|$.

Proof. For $0 \le i \le n$, let $|\operatorname{skel}^i(\Delta^n)|_k$ be the subspace of $|\Delta^n|_k$ restrict to the ith skeleton of Δ^n, i.e.,

$$|\operatorname{skel}^i(\Delta^n)|_k = |\Delta^n|_k \cap |\operatorname{skel}^i(\Delta^n)|.$$

Specifically, we have $|\Delta^n|_k = |\operatorname{skel}^n(\Delta^n)|_k$ and $|\operatorname{skel}^{k-1}(\Delta^n)| = |\operatorname{skel}^{k-1}(\Delta^n)|_k$. Our strategy is to show that, for $k \leq i \leq n$, there is a sequence of retractions

$$r_i : |\operatorname{skel}^i(\Delta^n)|_k \to |\operatorname{skel}^{i-1}(\Delta^n)|_k.$$

By composing each retraction r_i, we obtain the desired retraction

$$r = r_k \circ r_{k+1} ... \circ r_n.$$

We construct each retraction by downward induction. For the base case n, we apply the radial projection from the barycenter of $|\Delta^n|$, denoted b_Δ. For every point $x \in |\Delta^n|_k$, let L_x be the line that connects x and b_Δ. By definition of $|\Delta^n|_k$, we have $b_\Delta \notin |\Delta^n|_k$ and therefore, L_x is well defined. The radial projection is thus the continuous map

$$r_n(x) = L_x \cap |\operatorname{skel}^{n-1}(\Delta^n)|_k,$$

that maps x to the intersection of L_x and the boundary of $|\Delta^n|$, which is a unique point in $|\operatorname{skel}^{n-1}(\Delta^n)|_k$. Clearly, this defines a retraction since r_n restrict to $|\operatorname{skel}^{n-1}(\Delta^n)|_k$ is the identity map. To verify the image of r_n is $|\operatorname{skel}^{n-1}(\Delta^n)|_k$, notice that for any x' in $|\operatorname{skel}^{n-1}(\Delta^n)|$ such that $x' \notin |\operatorname{skel}^{n-1}(\Delta^n)|_k$, the line $L_{x'}$ is contained in $|\operatorname{Bary}_k(\Delta^n)|$, which is not in $|\Delta^n|_k$.

For the inductive step, we construct a retract from $|\operatorname{skel}^i(\Delta^n)|_k$ to $|\operatorname{skel}^{i-1}(\Delta^n)|_k$. Let $\{\sigma\}$ be the enumeration of facets in $|\operatorname{skel}^i(\Delta^n)|$. Let b_σ be the barycenter of the each facet $|\sigma|$, and let

$$|\sigma|_k = |\sigma| \cap |\operatorname{skel}^i(\Delta^n)|_k.$$

Consider the radial projection $r_{i,\sigma}$ from b_σ, the projection is well defined on $|\sigma|_k$ since

$b_\sigma \notin |\operatorname{skel}^i(\Delta^n)|_k$. Moreover, $r_{i,\sigma}$ is a retraction

$$r_{i,\sigma} : |\sigma|_k \to |\operatorname{skel}^{i-1}(\sigma)|_k$$

since it fixes the boundary of σ. For two facets $\sigma, \tau \in |\operatorname{skel}^i(\Delta^n)|$ that have a non-empty intersection, it is obvious that $r_{i,\sigma}$ and $r_{i,\tau}$ agree on $|\sigma|_k \cap |\tau|_k$. By Theorem 3.1 and a simple induction, we construct r_i by gluing $r_{i,\sigma}$ for each facet σ, which completes the inductive step. □

Note that the induction argument in Lemma 5.6 stops at $i = k-1$ since for each $(k-1)$-simplex σ in Δ^n, we have $|\sigma| = |\sigma|_k$ as $\operatorname{Bary}_k(\Delta^n)$ does not contain any vertex in $\operatorname{Bary}(\sigma)$. We now show that M_k solves $(n+1, k)$-set-agreement.

Theorem 5.7. *There is a protocol that solves k-set-agreement in model M_k.*

Proof. By Lemma 5.6, there is a continuous map

$$r : |\Delta^n|_k \to |\operatorname{skel}^{k-1}(\Delta^n)|$$

such that r restrict to $|\operatorname{skel}^{k-1}(\Delta^n)|$ is the identify map. Since $|\operatorname{Ch}_k(\Delta^n)|$ is a subspace of $|\Delta^n|_k$, r restricts to $|\operatorname{Ch}_k(\Delta^n)|$ is a continuous map carried by the k-set-agreement carrier map Γ. Applying Theorem 3.3, there exists a number N and a carrier-preserving simplicial map

$$\phi : \operatorname{Ch}^N(\operatorname{Ch}_k(\Delta^n)) \to \operatorname{skel}^{k-1}(\Delta^n).$$

Without loss generality, we can assume N is an even number. Let $N' = (N+2)/2$. Since $\operatorname{Ch}_k^{N'}(\Delta^n)$ is a subcomplex of $\operatorname{Ch}^N(\operatorname{Ch}_k(\Delta^n))$, we can restrict ϕ on $\operatorname{Ch}_k^{N'}(\Delta^n)$

and obtain a carrier-preserving simplicial map

$$\phi' : \mathrm{Ch}_k^{N'}(\Delta^n) \to \mathrm{skel}^{k-1}(\Delta^n).$$

The simplicial map ϕ' implies a k-set-agreement protocol:

1. Every process p_i executes N' rounds in model M_k, and obtains a vertex in $\mathrm{Ch}_k^{N'}(\Delta^n)$.
2. p_i calculates $v = \phi'(v)$, a vertex in $\mathrm{skel}^{k-1}(\Delta^n)$.
3. p_i returns process p_j where $\chi(v) = p_j$.

The correctness of the algorithm follows from fact that ϕ' is simplicial and carrier-preserving. □

To show M_k does not solve $(n+1, k-1)$-set-agreement, we will need the following useful result from [24].

Theorem 5.8 (Herlihy et al.)**.** *Let M be a distributed computing model in which protocol complex is $(k-2)$-connected for any participating set. Then M can not solve $(k-1)$-set-agreement.*

To prove that the protocol complex $\mathrm{Ch}_k(\Delta^n)$ is $(k-2)$-connected, we first show that $|\Delta^n|_k$ is $(k-2)$-connected and then prove $|\mathrm{Ch}_k(\Delta^n)|$ and $|\Delta^n|_k$ are homotopy equivalent.

Lemma 5.9. *The space $|\Delta^n|_k$ is $(k-2)$-connected.*

Proof. We show that $|\Delta^n|_k$ is homotopy equivalent to $|\mathrm{skel}^{k-1}(\Delta^n)|$. Then, by Fact 3.8 and 3.10, $|\Delta^n|_k$ is $(k-2)$-connected.

Our strategy is to transform the retractions defined in Lemma 5.6 to deformation retractions from $|\Delta^n|_k$ onto $|\operatorname{skel}^{k-1}(\Delta^n)|$. Let $r_i : |\operatorname{skel}^i(\Delta^n)|_k \to |\operatorname{skel}^{i-1}(\Delta^n)|_k$ be a retraction in the proof of Lemma 5.6. For $i = k, k+1, ..., n$, we construct a sequence of deformation retraction

$$H_i : |\operatorname{skel}^i(\Delta^n)|_k \times [0,1] \to |\operatorname{skel}^{i-1}(\Delta^n)|_k.$$

defined by

$$H_i(x,t) = x + t(r_i(x) - x).$$

It is easy to verify that $H_i(x,0)$ is the identify map, and $H_i(x,1) = r_i$. Moreover, H_i is continuous. Hence, we can construct a deformation retraction

$$H : |\Delta^n|_k \times [0,1] \to |\operatorname{skel}^{k-1}(\Delta^n)|$$

defined by

$$H(x,t) = H_i(x, (n-k+1)(t-i+k))$$

when $t \in [\frac{i-k}{n-k+1}, \frac{i-k+1}{n-k+1}]$. Applying Theorem 3.14, we conclude that $|\Delta^n|_k$ is homotopy equivalent to $|\operatorname{skel}^{k-1}(\Delta^n)|$, and therefore, $|\Delta^n|_k$ is $(k-2)$-connected. □

Lemma 5.10. *The protocol complex* $\operatorname{Ch}_k(\Delta^n)$ *is* $(k-2)$*-connected .*

Proof. We first show that $|\Delta^n|_k$ is homotopy equivalent to $|\operatorname{Ch}_k(\Delta^n)|$. Then, $|\operatorname{Ch}_k(\Delta^n)|$ is $(k-2)$-connected is a simplex consequence of Theorem 3.14.

The idea is that we can iteratively apply the radial projections in the proof of Lemma 5.6 and then modify them to deformation retractions as in Lemma 5.9. Let $|\Delta^n|'_k$ be the space after applying the radial projection in Lemma 5.6. We check whether $|\Delta^n|'_k = |\operatorname{Ch}_k(\Delta^n)|$. If not, then there exists a simplex $\sigma \in \operatorname{Ch}^2(\Delta^n)$ such

that $\sigma \notin \mathrm{Ch}_k(\Delta^n)$ and $|\Delta^n|'_k$ contains an interior point of $|\sigma|$, i.e.,

$$\mathrm{int}\,\sigma \cap |\Delta^n|'_k \neq \emptyset.$$

Moreover, there exists a vertex $v \in |\sigma|$ such that $v \in \mathrm{Bary}_k(\Delta^n)$ and so, $v \notin |\Delta^n|'_k$. We define a radial projection r_v as follows. Let $L_{v,x}$ be the line that connects each point $x \in |\Delta^n|'_k$ and v. Let $r_v(x)$ be point of the intersection of $L_{v,x}$ and the boundary of $|\,\mathrm{Ch}_k(\Delta^n)|$. Then, we transform r_v to be a deformation retraction using the method in Lemma 5.9 and then obtain a subspace

$$|\Delta^n|''_k \subset |\Delta^n|'_k.$$

Clearly, we have

$$|\sigma| \cap |\Delta^n|''_k \subseteq |\,\mathrm{Ch}_k(\Delta^n)|$$

since the radial projection is defined on v and $|\sigma|$ is convex. Then, we iteratively apply this procedure until the space is equivalent to $|\,\mathrm{Ch}_k(\Delta^n)|$. In other words, we construct a sequence of subspaces $|\Delta^n|'_k, |\Delta^n|''_k, ..., |\,\mathrm{Ch}_k(\Delta^n)|$ and a sequence of deformation that continuously shrink $|\Delta^n|'_k$ to $|\,\mathrm{Ch}_k(\Delta^n)|$. Notice that the sequence has a finite length since there are only finitely many simplices removed in the definition of $\mathrm{Ch}_k(\Delta^n)$. By Theorem 3.14, $|\Delta^n|_k$ is homotopy equivalent to $|\,\mathrm{Ch}_k(\Delta^n)|$. □

Applying Theorem 5.8, we characterize the set-consensus power of M_k.

Theorem 5.11. *There is no protocol for $(k-2)$-set-agreement in model M_k.*

5.2 Task Solvability of Vector-Set-Consensus Model

In this section, we generalize the complex in the previous section to characterize the colorless task solvability in the vector-set-consensus model. We define a protocol complex that solves the adaptive-set-agreement task, which generalizes the construction of protocol complex for $(n+1, k)$-set-agreement. Then, we present a computability theorem in the vector-set-consensus model: a colorless task is solvable in the vector-set-consensus model if and only if there exists a certain simplicial map from the iterated protocol complex to the task output complex.

5.2.1 Protocol Complex

Let $\mathcal{VSC} = (k_1, k_2, ..., k_{n+1})$ be a vector-set-consensus model. By Theorem 4.5, $\mathcal{VSC}$ solves the adaptive set-agreement

$$T_A = (\Delta^n, \text{skel}^{k_{n+1}-1}(\Delta^n), \Gamma_A)$$

, where $A = (k_1, k_2, ..., k_{n+1})$. The input complex of T_A is the standard n-simplex, and its output complex is the $(k_{n+1} - 1)$-skeleton of the n-simplex. For each simplex $\sigma \in \Delta^n$, where $\dim(\sigma) = m - 1$, the task specification is

$$\Gamma_A(\sigma) = \text{skel}^{k_m-1}(\sigma).$$

To see Γ_A is indeed a carrier map, notice that Γ_A is monotonic since $k_m \leq k_{m'}$. We are now ready to define the protocol complex that characterizes T_A. The idea is that we can apply the construction in the previous section to characterize (m, k_m)-set-agreement in $m - 1$ skeleton.

Definition 5.12. For each simplex $\sigma \in \Delta^n$, where $\dim(\sigma) = m$, define the subcomplex

$$\mathrm{Bary}_{V,m}(\Delta^n) = \{\tau \in \mathrm{Bary}(\mathrm{skel}^m(\Delta^n)) \mid \forall v \in \tau, \dim(\mathrm{Car}(v, \Delta^n) \geq k_{m+1}))\}.$$

Let $\mathrm{Bary}_A(\Delta^n)$ denote the complex

$$\mathrm{Bary}_V(\Delta^n) = \bigcup_{m=0}^{n} \mathrm{Bary}_{V,m}(\Delta^n)$$

For each $\sigma \in \Delta^n$ where $\dim(\sigma) = m$, we define $\mathrm{Bary}_{V,m}(\sigma) = \mathrm{Bary}(\sigma) \cap \mathrm{Bary}_{V,m}(\Delta^n)$. Notice that there is a containment relation among different $\mathrm{Bary}_{V,m}$.

Proposition 5.13. *Let $m, m' \in \{1, 2, ..., n+1\}$ such that $k_{m+1} = k_{m'+1}$ and $m < m'$. $\mathrm{Bary}_{V,m}$ is a subcomplex of $\mathrm{Bary}_{V,m'}$.*

Proof. Let $\sigma, \sigma' \in \Delta^n$ be two simplices such that $\sigma \subseteq \sigma'$ and $\dim(\sigma) = m < m' = \dim(\sigma')$. Let τ be a simplex in $\mathrm{Bary}_{V,m}$. Since $\tau \in \mathrm{Bary}(\mathrm{skel}^{m'}(\Delta^n))$ and $\forall v \in \tau, \dim(\mathrm{Car}(v, \Delta^n)) \geq k_{m+1} = k_{m'+1}$, we have $\mathrm{Bary}_{V,m} \subsetneq \mathrm{Bary}_{V,m'}$. □

For each $\ell = 1, 2, ..., k_{n+1}$, let m_ℓ be the largest number such that $k_{m_\ell} = \ell$. By Proposition 5.13, $\mathrm{Bary}_V(\Delta^n)$ is completely characterized by Bary_{V,m_ℓ}, that is,

$$\mathrm{Bary}_V(\Delta^n) = \bigcup_{\ell=1}^{k_{n+1}} \mathrm{Bary}_{V,m_\ell}(\Delta^n).$$

We are now ready to define the protocol complex $\mathrm{Ch}_V(\Delta^n)$ that characterizes the adaptive set-agreement T_A.

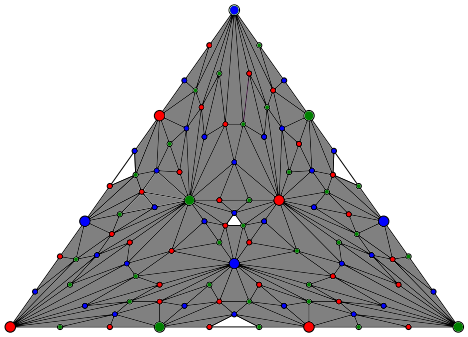


Figure 5.2: Example of $\mathrm{Ch}_V(\Delta^2)$ for $\mathcal{VSC} = (1, 1, 2)$

Definition 5.14. The complex $\mathrm{Ch}_V(\Delta^n)$ is a subcomplex of $\mathrm{Ch}^2(\Delta^n)$, defined by

$$\mathrm{Ch}_V(\Delta^n) = \{\sigma \in \mathrm{Ch}^2(\Delta^n) \mid |\sigma| \cap |\operatorname{Bary}_V(\Delta^n)| = \emptyset\}.$$

Let M_V be the iterated model that in each round, every process gets a view of two rounds immediate snapshot such that processes' views correspond to a simplex in $\mathrm{Ch}_V(\Delta^n)$. We now show that model M_V solves precisely the adaptive set-agreement T_V. As in the previous section, we define a subspace that will be useful in our proof. Let $|\operatorname{skel}^m(\Delta^n)|_{V,m}$ denote a subspace in the $(m-1)$-skeleton of $|\Delta^n|$, defined as

$$|\operatorname{skel}^m(\Delta^n)|_{V,m} = |\operatorname{skel}^m(\Delta^n)| - |\operatorname{Bary}_{V,m}(\Delta^n)|.$$

Lemma 5.15. *For $m = 1, 2, ..., n+1$, model M_V can solve k_m-set-agreement task, when m processes are participating.*

Proof. Notice that $|\operatorname{skel}^{m-1}(\Delta^n)|_{V,m-1}$ is a subspace of $|\Delta^n|_{k_m}$ in Definition 5.5. By Lemma 5.6, there is a continuous map:

$$f : |\operatorname{skel}^{m-1}(\Delta^n)|_{V,m-1} \to |\operatorname{skel}^{k_m-1}(\Delta^n)|$$

carried by Γ_A. When $m+1$ processes are participating, the protocol complex is $\mathrm{Ch}_V(\mathrm{skel}^m(\Delta^n))$, which geometric realization is a subspace of $|\Delta^n|_{V,m-1}$. Thus, considering f restricted on $|\mathrm{Ch}_V(\mathrm{skel}^{m-1}\Delta^n)|$, we obtained a carrier-preserving continuous map

$$f' : |\mathrm{Ch}_V(\mathrm{skel}^{m-1}\Delta^n)| \to |\mathrm{skel}^{k_m-1}(\Delta^n)|.$$

Applying the simplicial approximation theorem (Theorem 3.3), there exists a number N and a carrier-preserving simplicial map

$$\varphi : \mathrm{Ch}^N \mathrm{Ch}_V(\mathrm{skel}^{m-1}(\Delta^n)) \to \mathrm{skel}^{k_m-1}(\Delta^n).$$

Without loss of generality, we can assume N is an even number. Let $N' = (N+2)/2$, we obtain the following simplicial map

$$\varphi' : \mathrm{Ch}_V^{N'}(\mathrm{skel}^{m-1}(\Delta^n)) \to \mathrm{skel}^{k_m-1}(\Delta^n).$$

Therefore, model M_V solves k_m-set-agreement: every process p_i executes N' rounds in model M_V, and gets a vertex

$$v_i \in \mathrm{Ch}_V^{N'}(\mathrm{skel}^{m-1}\Delta^n) \subseteq \mathrm{Ch}^N \mathrm{Ch}_V(\mathrm{skel}^{m-1}(\Delta^n)).$$

Then, p_i calculates the vertex $w_j = \varphi(v_i) \in \mathrm{skel}^{k_m-1}(\Delta^n)$ and return process p_j where $\chi(w_j) = p_j$. The correctness follows from the fact that φ is simplicial and carrier-preserving. □

The next Lemma shows that we can combine each (m, k_m)-set-agreement to solve set-agreement adaptively.

Lemma 5.16. *There is a protocol in Model M_V that solves adaptive-set-agreement*

T_A.

Proof. The protocol complex of M_V can be identified as a subset of executions in two rounds of immediate snapshot. Therefore, M_V can simulate the atomic-snapshot model using the simulation in [21]. To solve the adaptive-set-agreement T_A, processes use Algorithm 2 in Chapter 4. More specifically, the set-consensus object access at line 8 in Algorithm 2 in Chapter 4 is replaced by the (m, k_m)-set-agreement protocol in Lemma 5.15. The correctness follows from the fact that the algorithm solves set-agreement adaptively, given the fact that each (m, k_m)-set-agreement can be solved when only m processes are participating. □

For the other direction, we show that model M_V can not solve $(m+1, k_{m+1}-1)$-set-agreement when $(m+1)$ processes are participating.

Lemma 5.17. *The complex* $\mathrm{Ch}_V(\mathrm{skel}^m(\Delta^n))$ *is* $(k_{m+1}-2)$*-connected .*

Proof. The protocol complex is $\mathrm{Ch}_V(\mathrm{skel}^m(\Delta^n))$ when $(m+1)$ processes participate. Our strategy is to show that $\mathrm{Ch}_V(\mathrm{skel}^m(\Delta^n))$ is $(k_{m+1}-2)$-connected, and then the claim is a simple consequence of Theorem 5.8.

First, we define a space that characterizes the topological property of $\mathrm{Ch}_V(\mathrm{skel}^m(\Delta^n))$. Let $|\,\mathrm{skel}^m(\Delta^n)|_V$ denote a subspace in $|\,\mathrm{skel}^m(\Delta^n)|$, defined as

$$|\,\mathrm{skel}^m(\Delta^n)|_V = |\,\mathrm{skel}^m(\Delta^n)| - |\,\mathrm{Bary}_V(\Delta^n)|.$$

Define the point set

$$X = \bigcup_{\ell=0}^{m-1} \mathrm{Bary}_{V,\ell}(\Delta^n).$$

We have

$$|\,\mathrm{skel}^m(\Delta^n)|_A = |\,\mathrm{skel}^m(\Delta^n)|_{V,m} - X.$$

Note that X only contains points on the boundary of $|\operatorname{skel}^m(\Delta^n)|_{V,m}$. Let X' be the closed ϵ neighborhood of X in $|\operatorname{skel}^m(\Delta^n)|_{V,m}$ for a sufficiently small $\epsilon > 0$. There is a deformation retraction from $(X' - X)$ onto the boundary of X' and thereby a deformation retraction from $|\operatorname{skel}^m(\Delta^n)|_V$ to a space that is homeomorphic to $|\operatorname{skel}^m(\Delta^n)|_{V,m}$. By Theorem 3.14, $|\operatorname{skel}^m(\Delta^n)|_V$ is homotopy equivalent to $|\operatorname{skel}^m(\Delta^n)|_{V,m}$. Since $|\operatorname{skel}^m(\Delta^n)|_{V,m}$ is $(k_{m+1} - 2)$-connected by Theorem 5.9, and by Fact 3.8, $|\operatorname{skel}^m(\Delta^n)|_V$ is $(k_{m+1} - 2)$-connected

To see that $|\operatorname{skel}^m(\Delta^n)|_V$ characterizes the protocol complex $\operatorname{Ch}_V(\operatorname{skel}^m(\Delta^n))$, notice that there is deformation retraction from $|\operatorname{skel}^m(\Delta^n)|_A$ to $|\operatorname{Ch}_V(\operatorname{skel}^m(\Delta^n))|$ by iteratively applying radial projections from points in X in the same way as in the proof of Lemma 5.10. Consequently, $|\operatorname{Ch}_V(\operatorname{skel}^m(\Delta^n))|$ is homotopy equivalent to $|\operatorname{skel}^m(\Delta^n)|_V$, and therefore, is $(k_{m+1} - 2)$-connected. □

We conclude the set-consensus power of M_V as follows.

Theorem 5.18. *Model M_V solves the adaptive-set-agreement T_A but not $(m+1, k_{m+1}-1)$-set-agreement when m processes participate.*

5.2.2 Computability Theorem

We now prove the main theorem in this chapter which characterizes the colorless task solvability in the vector-set-consensus model.

Theorem 5.19. *Let $\mathcal{VSC} = (k_1, k_2, ..., k_{n+1})$ be a vector-set-consensus model. A colorless task $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ is solvable in $\mathcal{VSC}$ if and only if there exists a number N and simplicial map $\phi : \operatorname{Ch}_V^N(\Delta^n) \to \mathcal{O}$ carried by Γ.*

For the necessity part of Theorem 5.19, we will show that model M_V can simulate a colorless task protocol in the vector-set-consensus model. Since the protocol complex of M_V is $\mathrm{Ch}_V(\Delta^n)$, the simulation implies the desired simplicial map ϕ.

Simulation

The simulation of the $\mathcal{VSC}$ is a variant of the BGG simulation [17, 30]. The set-consensus objects simulation is similar to the proof in Theorem 4.5. Without loss of generality, we assume that each process can perform three types of operations: writes a value, takes a snapshot of the memory, or requests a (m, k_m)-set-consensus object. We also assume that every process executes a full information protocol, in which every write is uniquely determine by the snapshots and the access of set-consensus objects. Hence, it suffices to simulate snapshot operations and set-consensus-object access.

Suppose that m processes are participating in M_V. In the simulation, processes solve the adaptive-set-agreement using the Algorithm in Lemma 5.16 and then execute the GG simulation [16] to emulate k_m simulators $s_1, s_2, ..., s_{k_m}$. The simulators then further use the BG simulation to simulate the snapshot operations and set-consensus objects in the $\mathcal{VSC}$ model. Specifically, each simulator s_i repeated executes the following steps.

1. Takes a snapshot of the current participating processes in M_A and obtains the input values for simulated processes.

2. Simulate one step of the participating processes in a round-robin way.

   (a) The snapshot operation of process p_j is simulated by invoking the safe-agreement protocol.

(b) For any $(m', k_{m'})$-set-consensus object access, s_i use k_m-safe-agreement to calculate an estimate of the returned value for each access by process p_j. Since simulators may get different estimates for each access, to ensure consistency, they further use safe-agreement to agree on the returned value.

3. If a safe-agreement or the ℓ-safe-agreement associated to process p_j is blocked, simulator s_i selects the next participating process to simulate in a round-robin manner.

If a process in M_V notices some output is produced in the BG simulation, it immediately adopts the output and terminates.

Lemma 5.20. *The BGG simulation produces a valid an execution of a colorless task protocol in the $\mathcal{VSC}$ model.*

Proof. The correctness of the snapshot follows from the agreement property of the safe-agreement. To show the access of set-consensus objects is correctly simulated, notice that for each access of $(m', k_{m'})$-set-consensus, simulators get at most $k_{m'}$ estimated value from $k_{m'}$-safe-agreement. Therefore, at most $k_{m'}$ values are proposed for each $(m', k_{m'})$-set-consensus simulation. Moreover, the returned values for each access is synchronized by executing the safe-agreement. Since every step is consistently simulated, it produces a valid execution in the $\mathcal{VSC}$ model.

To show the simulation eventually terminates, it suffices to prove that at least one output will be produced eventually. Since every process immediately adopts another process's output, every correct process eventually gets an output, and thus solves the colorless task.

Without loss of generality, suppose that m processes are participating. By properties of GG simulation [16], there are at most k_m simulators such that at least one

is always alive. If one simulator fails in the safe-agreement, at most one simulated process will be blocked. If k_t simulators fail in the simulation of (t, k_t)-set-consensus object, at most t processes will be blocked. Hence, we only need to show that at least one simulated process is not blocked. By way of contradiction, assume m processes are blocked. Then, there exists a multiset

$$\{(t_1, \ell_1), (t_2, \ell_2)...(t_q, \ell_q)\}$$

such that each $t_i \in \{1, 2, ...n+1\}$, $\ell_i = k_{t_i}$. Each (t_i, ℓ_i) represents a non-resolved ℓ_i-safe-agreement and blocks t_i simulated processes. Since at most $k_m - 1$ simulator can fail, we have

$$\sum_i \ell_i \leq k_m - 1, \ \sum_i t_i \geq m.$$

However, this contradicts the optimality property of the $\mathcal{VSC}$. Hence, at least one process is simulated infinitely often and eventually outputs, which shows the liveness property of the simulation. □

For the sufficiency part of Theorem 5.19, we will show that $\mathcal{VSC}$ can solve the colorless simplex agreement on $\mathrm{Ch}_V(\Delta^n)$. Formally, $(\Delta^n, \mathrm{Ch}_V(\Delta^n), \mathrm{Ch}_V)$ is a task such that for each $\sigma \in \Delta^n$, the valid output is any simplex in $\mathrm{Ch}_V(\sigma)$. Before we construct a protocol that solves this task, we first prove that there exists a certain continuous map if the task complex is sufficiently connected.

Lemma 5.21. *Let $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ be a colorless task such that for each $\sigma \in \mathcal{I}$, $\dim(\sigma) = m - 1$, $\Gamma(\sigma)$ is $(k_m - 2)$-connected. There exists a continuous map $f : |\operatorname{skel}^{k_m - 1}(\mathcal{I})| \to |\mathcal{O}|$ carried by Γ.*

Proof. Let τ be a simplex of $\mathcal{I}$ where $\dim(\tau) = m - 1$. We inductive define a contin-

uous map

$$f^t : |\operatorname{skel}^t(\tau)| \to |\Gamma(\tau)|$$

for $t = 0, 1, ..., k_m - 1$.

For the base case $t = 0$, since $|\Gamma(\tau)|$ is $(k_m - 2)$-connected and $k_m \geq 1$, $|\Gamma(\tau)|$ is at least (-1)-connected, or non-empty. For each $v \in \operatorname{skel}^0(\tau)$, define $f^0(v) = v'$ where v' is an arbitrary vertex in $|\Gamma(\tau)|$. Hence, we obtain a map

$$f^0 : |\operatorname{skel}^0(\tau)| \to |(\Gamma(\tau))|$$

carried by Γ, and f^0 is clearly continuous since it is a map between vertices. Assume that we have defined f^t for $t \leq k_m - 2$. For the inductive step, let $\{\omega_i\}$ be the enumeration of the facets in $\operatorname{skel}^{t+1}(\tau)$, and let $\partial\omega_i$ be the boundary of ω_i. Let $f^t|_{\partial\omega_i}$ denote the map f^t restricted to $\partial\omega_i$. Observe that $|\Gamma(\tau)|$ is t-connected since $|\Gamma(\tau)|$ is $(k_m - 2)$-connected and $t \leq k_m - 2$.

Since $|\partial\omega_i|$ is homeomorphic to t-sphere and $|\omega_i|$ is homeomorphic to $(t+1)$-disk, and by $|\Gamma(\tau)|$ is t-connected, $f^t|_{\partial\omega_i}$ can be extended into a continuous map

$$f^t_{\omega_i} : |\omega_i| \to |\Gamma(\tau)|.$$

Let ω_i and ω_j be two facets in $\operatorname{skel}^{t+1}(\tau)$. $f^t_{\omega_i}$ and $f^t_{\omega_j}$ agree on $|\omega_i| \cap |\omega_j|$. Applying the pasting lemma (Theorem 3.1), we can glue $\{f^t_{\omega_i}\}$ together and get a carrier-preserving continuous map

$$f^{t+1} : |\operatorname{skel}^{t+1}(\tau)| \to |\Gamma(\tau)|,$$

which completes the inductive step. By induction, for $\tau \in \operatorname{skel}^{k_m-1}(\mathcal{I})$, there is a continuous map

$$f_\tau : |\tau| \to |\Gamma(\tau)|.$$

Moreover, for any two simplices $\tau, \sigma \in \operatorname{skel}^{k_m-1}(\mathcal{I})$, f_τ and f_σ agree on $|\tau| \cap |\sigma|$. Apply pasting lemma again, we obtain the desired carrier-preserving continuous map

$$f : |\operatorname{skel}^{k_m-1}(\mathcal{I})| \to |\mathcal{O}|.$$

□

We now show that $\mathcal{VSC}$ can solve the colorless simplex agreement on $\operatorname{Ch}_V(\Delta^n)$.

Lemma 5.22. *There is a protocol in the vector-set-consensus model $\mathcal{VSC}$ that solve the colorless simplex agreement $(\Delta^n, \operatorname{Ch}_V(\Delta^n), \operatorname{Ch}_V)$.*

Proof. Assume m processes participate, and let $\sigma \in \Delta^n$ be the corresponding input simplex such that $\dim(\sigma) = m - 1$. By Lemma 5.17, the output complex $\operatorname{Ch}_V(\sigma)$ is $(k_m - 2)$-connected. By Lemma 5.21, there is a carrier-preserving continuous map

$$f : |\operatorname{skel}^{k_m-1}(\Delta^n)| \to |\operatorname{Ch}_V(\Delta^n)|.$$

Applying the simplicial approximation theorem (Theorem 3.3), there exists a number $N \in \mathbb{N}$ and a carrier-preserving simplicial map

$$\varphi : \operatorname{Bary}^N(\operatorname{skel}^{k_m-1}(\Delta^n)) \to \operatorname{Ch}_V(\Delta^n).$$

Thus, we obtain the following protocol:

1. Every process p_i solves the adaptive-set-agreement and gets a vertex $v_i \in \text{skel}^{k_m-1}(\Delta^n)$.

2. p_i executes N rounds immediate snapshot with the input v_i. Processes remove ids in their views in each round. The view of the Nth immediate snapshot is identified as a vertex $v_i' \in \text{Bary}^N(\text{skel}^{k_m-1}(\Delta^n))$.

3. p_i applies the simplicial map to calculate $v = \varphi(v_i')$, and then returns the vertex.

The correctness of the protocol follows from the fact that $\mathcal{VSC}$ solves the adaptive-set-agreement, and φ is a carrier-preserving simplicial map. □

We are now able to prove Theorem 5.19.

Proof. For sufficiency, by Lemma 5.20, model M_V can simulate any protocol in the $\mathcal{VSC}$ model. By Kőnig's lemma, if a $\mathcal{VSC}$ protocol solves a task, every process produces an output in a bounded number of steps. Hence, the simulation in M_A terminates in a bounded number of rounds. Since the protocol complex in M_A is $\text{Ch}_A^N(\Delta^n)$, the simulation implies a carrier-preserving simplicial map $\phi : \text{Ch}_V^N(\Delta^n) \to \mathcal{O}$.

In the other direction, by Lemma 5.22, processes in $\mathcal{VSC}$ can solve N rounds of colorless simplex agreement $(\Delta^n, \text{Ch}_V(\Delta^n), \text{Ch}_V)$ and obtain a vertex in $\text{Ch}_V^N(\Delta^n)$. Then, every process applies ϕ on its vertex and gets an output in $\mathcal{O}$. The correctness follows from the fact that ϕ is simplicial and carrier-preserving. □

5.2.3 Tasks solvability of terminating iterated vector-set-consensus model

Consider the iterated shared memory model that processes execute an immediate snapshot in each round and access the vector-set-consensus. In this model, once a

process has obtained an output, it simply terminates and stops the communication with other processes. We call this model the terminating iterated vector-set-consensus model. In each iteration, the set-consensus power of this model depends on the number of active processes, i.e., participating processes that do not have output. We now present a task solvability characterization of this model by introducing the terminating iterated complex, which generalizes the idea of terminating subdivision in [20].

Let Ch_V be the induced operator for the vector-set-consensus model $\mathcal{VSC} = (k_1, k_2, ..., k_{n+1})$. Let $\mathcal{I}$ be an arbitrary chromatic simplicial complex, and let N be a natural number. A N round terminating iterated complex is a specified by a sequence of chromatic complexes $\widetilde{\mathrm{Ch}_V^0(\mathcal{I})}, \widetilde{\mathrm{Ch}_V^1(\mathcal{I})}, ..., \widetilde{\mathrm{Ch}_V^N(\mathcal{I})}$ and a sequence of subcomplex $\Sigma_0 \subseteq \Sigma_1 \subseteq ... \subseteq \Sigma_N$ such that for $i \geq 0$:

1. Σ_i is a subcomplex of $\widetilde{\mathrm{Ch}_V^i(\mathcal{I})}$
2. $\widetilde{\mathrm{Ch}_V^0(\mathcal{I})} = \mathcal{I}$, and $\widetilde{\mathrm{Ch}_V^{i+1}(\mathcal{I})}$ is inductively constructed from $\widetilde{\mathrm{Ch}_V^i(\mathcal{I})}$ by partially applying Ch_V on simplices in $\mathrm{Ch}_V^i(\mathcal{I})$. Formally, we apply Ch_V on a simplex $\sigma \in \widetilde{\mathrm{Ch}_V^i(\mathcal{I})}$ that is not in Σ_k, and then connect vertices in $\mathrm{Ch}_V(\sigma)$ with vertices in Σ_k to turn it into a chromatic simplicial complex. The idea of the definition is that simplices in Σ_k are considered "fixed". In other words, processes with local states that can be identified to vertices in Σ terminate their computation. Additionally, we require that $\Sigma_N = \bigcup_i \Sigma_i = \widetilde{\mathrm{Ch}_V^N(\mathcal{I})}$.

We are now ready to characterize the task solvability in the terminating iterated vector-set-consensus model.

Theorem 5.23. *Let $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ be a task. T is solvable in the terminating iterated vector-set-consensus model $\mathcal{VSC} = (k_1, k_2, ..., k_{n+1})$ if and only if there exists a*

terminating iterated complex $\widetilde{\mathrm{Ch}_V^N(\mathcal{I})}$ *for some number* N *and a chromatic simplicial map* $\phi : \widetilde{\mathrm{Ch}_V^N(\mathcal{I})} \to \mathcal{O}$ *carried by* Γ.

Proof. Suppose there exists a protocol in the terminating vector-set-consensus model that solves T. Without loss of generality, assume every process obtains an output after N rounds. We inductively construct a terminating iterated complex $\widetilde{\mathrm{Ch}_V^N(\mathcal{I})}$ and a simplicial map ϕ as follows. Let $\widetilde{\mathrm{Ch}_V^0(\mathcal{I})} = \mathcal{I}$ for $N = 0$. At the ith stage of the construction, for each vertex $v \in \widetilde{\mathrm{Ch}_V^i(\mathcal{I})}$, we check process p_j's local state that can be identified as vertex v. If p_j has calculated an output val_j for the task in the protocol, we add v in Σ_k and set $\phi(v) = \mathrm{val}_j$. Then, we construct $\widetilde{\mathrm{Ch}_V^{i+1}(\mathcal{I})}$ based on the subcomplex Σ_k. Since every process obtains an output at the Nth round, the construction stops at $\widetilde{\mathrm{Ch}_V^N(\mathcal{I})}$ and ϕ is a well-defined simplicial map on $\widetilde{\mathrm{Ch}_V^N(\mathcal{I})}$. The correctness follows from the fact that $\mathrm{Ch}_V(\Delta^n)$ solves the adaptive set-agreement (Lemma 5.16), and simplices in $\widetilde{\mathrm{Ch}_V^N(\mathcal{I})}$ correspond to executions of IIS in which processes depart once obtained an output in the terminating iterated vector-set-consensus model.

For the other direction, suppose there exists a terminating iterated complex $\widetilde{\mathrm{Ch}_V^N(\mathcal{I})}$ and a carrier-preserving chromatic simplicial map $\phi : \widetilde{\mathrm{Ch}_V^N(\mathcal{I})} \to \mathcal{O}$. We construct a protocol in the terminating iterated vector-set-consensus model by inductively solving the simplex agreement on $\widetilde{\mathrm{Ch}_V^i(\mathcal{I})}$. The simplex agreement on $\widetilde{\mathrm{Ch}_V^0(\mathcal{I})} = \mathcal{I}$ is trivial for the base case. Assume that the simplex agreement on the ith stage is solved, process p_i obtains a vertex $v \in \widetilde{\mathrm{Ch}_V^i(\mathcal{I})}$ and then checks whether v in a vertex in Σ_i. If so, p_i calculates its output value $\mathrm{val}_i = \phi(v)$ and terminates. Otherwise, p_i uses the algorithm in Lemma 5.22 to solve the simplex agreement on $\mathrm{Ch}_V(\sigma)$ where σ is the simplex in $\widetilde{\mathrm{Ch}_V^i(\mathcal{I})}$ that includes vertex v. Since processes in Σ_i terminates, p_i gets a vertex in $\widetilde{\mathrm{Ch}_V^{i+1}(\mathcal{I})}$. The protocol terminates after N round, and every process

obtains an output since $\Sigma_N = \widetilde{\mathrm{Ch}_V^N(\mathcal{I})}$. The correctness of the protocol follows from the fact that ϕ is simplicial and carrier-preserving. □

5.3 Task Solvability of k-set-consensus Model

In the previous section, we characterized the colorless task solvability of the vector-set-consensus model. Gafni et al [17] presented ACT for colored task solvability in the k-set-consensus model. In this section, we provide an alternative characterization of k-set-consensus model using the notion of color-projection.

Color-Projection

Recall in section 5.1, we defined a protocol complex that characterizes the $(n+1, k)$-set-agreement. In the k-set-consensus model, a set of processes m processes can request a k-set-consensus object to solve k-set-consensus among themselves when more than m processes are participating. In this case, the (m, k)-set-agreement is not a colorless task since processes cannot adopt a decision value from another process that is not one of these m processes. Therefore, to characterize the colored task solvability in the k-set-consensus model, the protocol complex should be able to solve (m, k)-set-agreement regardless of the size of participating set.

In this section, we introduce the notion of color-projection. Suppose a subset Q of m participating processes wants to solve the (m, k)-set-agreement among processes in Q. They can reduce their views by recursively eliminating the views of processes that are not in Q. Every process in Q obtains a view from a reduced execution in which only processes in Q participate. Thus, they can solve (m, k)-set-agreement if such a protocol exists when only m processes participate.

The idea is formalized as a simplicial map from a standard chromatic subdivision to its faces that only contains vertices with colors in Q. Recall that in the definition of $\mathrm{Ch}^k(\Delta^n)$, each vertex $v \in \mathrm{Ch}^k(\Delta^n)$ can be identified as a tuple

$$v = (\chi(v), \mathrm{Car}(v, \mathrm{Ch}^{k-1}(\Delta^n)).$$

Further, we define $\mathrm{Ch}^0(\Delta^n) = \Delta^n$. To simplify the notation, assume the empty set $\emptyset$ is included as a simplex in any simplicial complex. Let $Q \subseteq \{0, ..., n\}$ be a subset of processes. Let $\sigma \subseteq \Delta^n$ be the face in n-simplex corresponds to Q, that is, $\chi(\sigma) = Q$.

Definition 5.24. For $N = 0, 1, 2, ...$, the Nth *color-projection on to* Q, is a vertex map

$$\Pi_Q^N : \mathrm{Ch}^N(\Delta^n) \to \mathrm{Ch}^N(\sigma),$$

defined as follows:

1. For $v \in \Delta^n$, $\Pi_Q^0(v) = \begin{cases} v & \text{if } \chi(v) \in Q \\ \emptyset & \text{if } \chi(v) \notin Q \end{cases}$
2. When $k \geq 1$, for $v \in \mathrm{Ch}^N(\Delta^n)$

$$\Pi_Q^N(v) = \begin{cases} (\chi(v), \{\Pi_Q^{N-1}(w) \mid w \in \mathrm{Car}(v, \mathrm{Ch}^{N-1}(\Delta^n)) \wedge \chi(w) \in Q\}) & \text{if } \chi(v) \in Q \\ \emptyset & \text{if } \chi(v) \notin Q \end{cases}$$

The Nth color-projection onto Q is a vertex map that sends a vertices v to a vertex v' such that $\chi(\mathrm{Car}(v', \Delta^n)) \subseteq Q$. We use Π_Q be denote the Nth color-projection when N is clear from the context. Next, we show that the map Π_Q^k is indeed a simplicial map, and moreover, it preserves the execution order of immediate snapshots.

Proposition 5.25. *The vertex map Π_Q^N is a simplicial map and preserves the linear order of immediate snapshots among processes in Q. Formally, let γ be a simplex in $\mathrm{Ch}^N(\Delta^n)$, and let $\gamma' = \Pi_Q^N(\gamma)$. γ' is a simplex in $\mathrm{Ch}^N(\sigma)$. Moreover, $\forall v_i, v_j \in \gamma$ such that $v_i = (p_i, \tau_i)$ and $v_j = (p_j, \tau_j)$, let $v_i' = \Pi_Q^N(v_i) = (p_i, \tau_i')$,* and let $v_j' = \Pi_Q^N(v_j) = (p_j, \tau_j')$, then the following properties are satisfied:

1. *(self-inclusion)* $p_i \in \chi(\tau_i')$
2. *(containment)* $\tau_i \subseteq \tau_j \Rightarrow \tau_i' \subseteq \tau_j'$
3. *(immediacy)* $p_i \in \chi(\tau_j) \Rightarrow p_i \in \chi(\tau_j')$ *and* $\tau_i' \subseteq \tau_j'$

Proof. We prove that Π_Q^N satisfies three properties by induction. The base case of $N = 0$ is easy to check, since $\Pi_Q^0(v_i) = v_i$ if $\chi(v_i) \in Q$ and γ' is simplex of σ. The self-inclusion property is satisfied, and the containment and immediacy properties are vacuously true. Assume Π_Q^{N-1} is simplicial and satisfies three properties. Consider the case of N, since γ is a simplex in $\mathrm{Ch}^N(\sigma)$ satisfies the self-inclusion property, $p_i \in \chi(\tau_i)$. Let vertex w_i be a vertex in $\mathrm{Ch}^{N-1}(\sigma)$, $w_i \in \tau_i$ and $\chi(w_i) = p_i$. Since $p_i \in Q$, $w_i' = \Pi_Q^{N-1}(w_i) \neq \emptyset$, we have $w_i' \in \tau'$. Therefore, $p_i \in \chi(\tau_i')$. For containment property, if $\tau_i \subseteq \tau_j$, τ_i' and τ_j' are simplices in $\mathrm{Ch}^N(\sigma)$ by the induction hypothesis. $\tau_i' \subseteq \tau_j'$ follows from the definition of Π_Q^N. To see Π_Q^N satisfies the immediacy property, notice that if $p_i \in \chi(\tau_j)$, then $\tau_i \subseteq \tau_j$ by the immediacy of $\mathrm{Ch}^{N-1}(\sigma)$. Since Π_Q^N preserves the containment property, we have $\tau_i' \subseteq \tau_j'$. Moreover, there exists a vertex $w_i' \in \tau_i'$ where $\chi(w_i') = p_i$ and so, $w_i' \in \tau_j'$ and $p_i \in \chi(\tau_j')$. Hence, $\gamma' = \Pi_Q^N(\gamma)$ is a simplex in $\mathrm{Ch}^N(\sigma)$ and satisfies the three properties, which completes the inductive step. □

In order to let processes in Q use color-projection to execute protocols for a restricted participating set, we need to ensure the projection of any simplex is valid in

the protocol complex. In other words, the protocol complex needs to be closed under color-projection.

Definition 5.26. Let $\mathcal{K}$ be a pure n-dimensional subcomplex in $\mathrm{Ch}^N(\Delta^n)$. Its *color-projection closed subcomplex* $\widehat{\mathcal{K}} = \mathrm{skel}^n(\widehat{\mathcal{K}})$, is inductively defined as follows:

1. $\mathrm{skel}^0(\widehat{\mathcal{K}}) = \mathrm{skel}^0(\mathcal{K})$
2. for $i = 1, 2, ..., n$, $\mathrm{skel}^i(\widehat{\mathcal{K}}) = \{\sigma \in \mathrm{skel}^i(\mathcal{K}) \mid \forall Q \subseteq \chi(\sigma), \Pi_Q(\sigma) \in \mathrm{skel}^{i-1}(\widehat{\mathcal{K}})\}$

From the definition, it is easy to verify that $\widehat{\mathcal{K}}$ is closed under color-projection, i.e., $\widehat{\widehat{\mathcal{K}}} = \widehat{\mathcal{K}}$.

Colored Task Solvability in k-set-consensus model

Gafni et al [17] proved the ACT for k-set-consensus model using the following protocol complex.

Definition 5.27. The k-set-consensus protocol complex $\mathcal{R}_k$, is defined by

$$\mathcal{R}_k = \left\{\sigma \in \mathrm{Ch}^2(\Delta^n) \mid \forall \tau \subseteq \sigma : (\forall v, v' \in \tau, \mathrm{Car}(v, \Delta^n) = \mathrm{Car}(v', \Delta^n)) \Rightarrow \dim(\tau) < k\right\}$$

The complex $\mathcal{R}_k$ can be operationally identified as a subset of executions of two round IS such that no more than $k+1$ processes see each other in their views. In [17] , the authors showed that a colored task is solvable if and only if there is a number N and a color-preserving simplicial map $\phi : \mathcal{R}_k^N(\mathcal{I}) \to \mathcal{O}$ carried by Γ.

We now present an alternative k-set-consensus protocol complex which is a color-projection closed. First, we present a different protocol complex that solves the $(n+1, k)$-set-agreement than the one in Section 5.1.

Definition 5.28. $\mathrm{Ch}_k(\Delta^n)'$ is a subcomplex in $\mathrm{Ch}^2(\Delta^n)$, defined as

$$\mathrm{Ch}_k(\Delta^n)' = \left\{\sigma \in \mathrm{Ch}^2(\Delta^n) \mid \exists v \in \sigma, \dim(\mathrm{Car}(v, \Delta^n)) \leq k-1\right\}$$

It is easy to check that every simplex in $\mathrm{Ch}_k(\Delta^n)'$ intersects with $\mathrm{skel}^{k-1}(\Delta^n)$. The following lemma shows the set-consensus power of $\mathrm{Ch}_k(\Delta^n)'$.

Lemma 5.29. *The protocol complex* $\mathrm{Ch}_k(\Delta^n)'$ *can solve* $(n+1,k)$*-set-agreement.*

Proof. We present an algorithm that solves $(n+1,k)$-set-agreement:

1. Every process p_i executes one round and gets a vertex $v_i \in \mathrm{Ch}_k(\Delta^n)'$.
2. Let IS_i^1 and IS_i^2 be the first and second immediate snapshot view corresponding to vertex v_i. p_i returns the process p_j such that j is the smallest index in which $\mathrm{IS}_j^1 \in \mathrm{IS}_i^2$ and $|\mathrm{IS}_j^1| \leq k$.

The validity of the protocol follows from the fact that $\mathrm{IS}_j^1 \in \mathrm{IS}_i^2$, and p_j is a participating process. For the agreement property, note that for each simplex $\sigma \in \mathrm{Ch}_k(\Delta^n)'$, at least one vertex is in $\mathrm{Ch}^2(\mathrm{skel}^{k-1}(\Delta^n))$. Let $\tau \subseteq \sigma$ be the maximum dimensional face that intersects $\mathrm{Ch}^2(\mathrm{skel}^{k-1}(\Delta^n))$. Let $L = \chi(\tau)$. For every process $p_j \in L$, we have $|\mathrm{IS}_j^1| \leq k$ and $|\mathrm{IS}_j^2| \leq k$.

We now argue that for every process p_i, there exists a $\mathrm{IS}_j^1 \in \mathrm{IS}_i^2$ and $|\mathrm{IS}_j^1| \leq k$. Clearly, processes in L see at least one $|\mathrm{IS}_j^1| \leq k$ because of the self-inclusion property of immediate snapshot. Every process p_i that is not in L sees at least one process in L in its second immediate snapshot. Hence, every process returns some p_j such that $|\mathrm{IS}_j^1| \leq k$. Moreover, the number of different processes returned is bounded by k since at most k processes have $|\mathrm{IS}^1| \leq k$. □

Let $\widehat{\mathrm{Ch}_k(\Delta^n)}$ be the color-projection closed complex constructed from $\mathrm{Ch}_k(\Delta^n)$, and let $\widehat{M_k}$ be the iterated model such that the protocol complex is $\widehat{\mathrm{Ch}_k(\Delta^n)}$ in each round. The following lemma shows that model $\widehat{M_k}$ can solve k-set-agreement regardless of the size of participating set.

Lemma 5.30. *Let P be the set of participating processes, $|P| = m$, and let $Q \subseteq P$ be a subset of participating processes where $|Q| = m' < m$. There is a (m', k)-set-agreement protocol for processes in Q in model $\widehat{M_k}$.*

Proof. Let τ, σ be simplices in Δ^n such that $\chi(\sigma) = P$ and $\chi(\tau) = Q$. Every process p_i in Q executes one round of $\widehat{M_k}$ and obtains a vertex v_i in $\widehat{\mathrm{Ch}_k(\sigma)}$. Then, p_i applies the color-projection on Q, and gets the vertex $v_i' = \Pi_Q(v_i)$. By Proposition 5.25, the set of vertices $\{v_i' \mid i \in \tau\}$ forms a simplex in $\widehat{\mathrm{Ch}_k(\tau)}$. Since $\widehat{\mathrm{Ch}_k(\tau)}$ is a subcomplex in $\mathrm{Ch}_k(\tau)'$, processes in Q use the protocol in Lemma 5.29 to solve the (m', k)-set-agreement. □

We now prove the ACT of the k-set-consensus model for colored task solvability. The proof is adapted from [17].

Lemma 5.31. *There exists a simulation of k-set-consensus model in model $\widehat{M_k}$.*

Proof. Without loss of generality, we assume that every process in the k-set-consensus model repeatedly executes the following three types of operations: writes a value, takes an atomic snapshot of the shared memory, or access a k-set-consensus object. It suffices to simulate the atomic snapshot memory and k-set-consensus object since they determines each write.

We use the simulation in [21] to simulate the atomic snapshot(AS) model in $\widehat{M_k}$ since its protocol complex can be identified as a subset of executions of IIS. By Lemma

5.30, there is a protocol that solves k-set-agreement among any subset of participating processes, and therefore, $\widehat{M_k}$ can directly simulates k-set-consensus objects access in a colored task protocol.

The AS and k-set-consensus objects simulations are further combined using the generic scheme in [17, 36]. The combined simulation is lock-free, and at least one process without outputs is simulated for infinitely many steps, finishing either a pending write, an atomic snapshot operation, or a k-set-consensus object access. Since a correct process eventually produces an output in the k-set-consensus model, every simulated process eventually terminates in the simulation. □

In the other direction, we show that k-set-consensus model solves the simplex agreement on $\widehat{\mathrm{Ch}_k(\tau)}$.

Lemma 5.32. *The k-set-consensus model can solve the simplex agreement on $\widehat{\mathrm{Ch}_k(\tau)}$.*

Proof. For task solvability, the k-set-consensus model is equivalent to a k-concurrent shared memory model [15, 17], i.e., a shared memory model that at most k processes are active at any point of time. We show that in k-concurrency model, an execution of two rounds of immediate snapshot (IS) solves the simplex agreement on $\widehat{\mathrm{Ch}_k(\tau)}$.

By way of contradiction, let σ denote the simplex corresponds to an k-concurrent execution of two rounds of IS such that $\sigma \notin \widehat{\mathrm{Ch}_k(\tau)}$. For any set $Q \subseteq \chi(\sigma)$, we have $\Pi_Q(\sigma) \cap \mathrm{skel}^{k-1}(\Delta^n) \neq \emptyset$. Otherwise, every process in Q observed more than k processes in its view of two round IS and thus, there exists a set $U \subseteq Q$ such that $|U| \geq k+1$ and every process in U sees each other in their views. However, this implies that at least $k+1$ processes are concurrently executing IS, contradicting the specification of k-concurrency model. □

We characterizes the colored task solvability in k-set-consensus model in the following theorem.

Theorem 5.33. *A task $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ is solvable in the k-set-consensus model if and only if there exists a number N and a color-preserving simplicial map $\phi : \widehat{\mathrm{Ch}_k^N(\mathcal{I})} \to \mathcal{O}$ carried by Γ.*

Chapter 6

Computability Theorem for General Adversary

In the previous chapter, we proved the colorless ACT for the vector-set-consensus model. Deplore et al. [12] proposed the general adversary to capture the non-uniform failure pattern in a distributed system. Vikram et al. [38] presented the ACT for the t-resilient model, which can be considered as an adversary containing all live sets with a size more than $n+1-t$. Kuznetsov et al. [32] later extended the result to fair adversaries, which generalizes the superset-closed and symmetric adversaries. The problem of characterizing task solvability for the general adversary, however, is still not settled.

This chapter partially answers this question by presenting a colorless ACT for general adversaries. Surprisingly, solving a task in the general adversary is not formally defined before. We close this gap by providing the first definition. Next, we show that the power of solving colorless tasks in an adversary is fully grasped by its ability to solve adaptive set-agreement. With this observation in mind, we prove the "wait-at-beginning" ACT for general adversary. The “wait-at-beginning” property of

our characterization captures the intuition that it is enough to use the set-consensus power at the beginning of a colorless task protocol.

6.1 Task Solvability and Set-Consensus Power

Task Solvability

Historically, the notion of solving a task in a general adversary is not formally defined. Once processes get outputs, it is unclear what the specification of processes behavior in the adversary is. The problem of whether processes stay or depart in the computation is a longtime issue. This leads to a more confusing question: if a process obtains an output but fails, should this output satisfy the task specification?

To clarify this confusion, we require that processes do not fail. A task is solvable in an adversary if a protocol exists such that it eventually produces outputs for every participating process under certain liveness condition. Let $P \subseteq \{0, 1, ..., n\}$ be a participating set, and let $Q \subseteq P$ be the set of processes that have obtained an output for the task. The adversary $\mathcal{A}$ restricted to a participating set P is the sub-adversary

$$\mathcal{A}|_P = \{S \in \mathcal{A} \mid S \subseteq P\}.$$

We also define the sub-adversary

$$A|_{P,\overline{Q}} = \{S \in \mathcal{A}|_P \mid S \not\subseteq Q\}.$$

$A|_{P,Q}$ characterizes the liveness condition when a subset of processes obtained outputs. Formally, solving a task in an adversary is defined as follows.

Definition 6.1. Let $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ be a task. Let P denote the set of participating processes, and let $Q \subseteq P$ be a set of processes obtained outputs. A task T is solvable in the adversary $\mathcal{A}$ if there exists a protocol such that

1. For any P and Q, the protocol eventually produces an output for every process in a live set $S \in A|_{P,\overline{Q}}$.
2. The outputs produced by the protocol satisfy the task specification Γ.

Set-consensus power

The set consensus power of the general adversary is defined by Gafni and Kuznetsov [19] to capture the best set-agreement that can be solved in the adversary. Let $\mathcal{A}|_{P,\bar{a}} = \{S \in \mathcal{A}|_P \mid a \notin S\}$ denote the subadversary of $\mathcal{A}$ that consists of live sets in $\mathcal{A}|_P$ that do not contain the element a.

Definition 6.2. The number setcon($\mathcal{A}$) is recursively defined as follows:

1. If $\mathcal{A} = \emptyset$, then setcon($\mathcal{A}$) $= 0$.
2. Otherwise, setcon($\mathcal{A}$) $= \max_{S \in \mathcal{A}} \min_{a \in S}$ setcon($\mathcal{A}|_{P,\bar{a}}$) $+ 1$.

It is not difficult to see that setcon($\mathcal{A}$) is the size of its minimum hitting set $H(\mathcal{A})$ for superset-closed adversaries, i.e., the minimum cardinality set that intersects with every live set in $\mathcal{A}$. For symmetric adversary, setcon($\mathcal{A}$) equals the number of different set sizes in $\mathcal{A}$. Moreover, Gafni and Kuznetsov proved [19] that $\mathcal{A}$ can solve setcon($\mathcal{A}$)-set-agreement but not (setcon($\mathcal{A}$) $- 1$)-set-agreement. Kuznetsov and Rieutord [30] used the notion of agreement function to capture the adaptive set-consensus power of the adversary. For every participating set P, the agreement function calculates

k_P as the lowest number such that k_P-set-agreement can be solved. Kuznetsov and Rieutord also [30] proposed an algorithm that solves (P, k_P)-set-agreement adaptively in an adversary in which $k_P = \text{setcon}(\mathcal{A}|_P)$.

6.2 Protocol Complex for General Adversary

In this section, we present a protocol complex that characterizes the set-consensus power of the general adversary. The construction generalized the protocol complex for the vector-set-consensus model in Chapter 5.

In the following discussion, we will abuse the notation $\sigma \in \Delta^n$ as a participating set when solving set-agreement. Let k_σ denote the set-consensus power associated to σ, i.e., $k_\sigma = \text{setcon}(A|_P)$ where $P = \chi(\sigma)$, and let $k = \text{setcon}(\mathcal{A})$. We first define a subcomplex in the barycentric subdivision of a standard n-simplex.

Definition 6.3. For any simplex $\sigma \in \Delta^n$, let $\text{Bary}_{\mathcal{A},\sigma}(\Delta^n)$ be a subcomplex of $\text{Bary}(\sigma)$, defined by

$$\text{Bary}_{\mathcal{A},\sigma}(\Delta^n) = \{\tau \in \text{Bary}(\sigma) \mid \forall v \in \tau, \dim(\text{Car}(v, \Delta^n) \geq k_\sigma))\}.$$

Define the subcomplex

$$\text{Bary}_{\mathcal{A}}(\Delta^n) = \bigcup_{\sigma \in \Delta^n} \text{Bary}_{\mathcal{A},\sigma}(\Delta^n).$$

The protocol complex for adversary $\mathcal{A}$ is a subcomplex in $\text{Ch}^2(\Delta^n)$ that does not intersect with $\text{Bary}_{\mathcal{A}}(\Delta^n)$.

Definition 6.4. The complex $\mathrm{Ch}_{\mathcal{A}}(\Delta^n)$ is a subcomplex of $\mathrm{Ch}^2(\Delta^n)$, defined as

$$\mathrm{Ch}_{\mathcal{A}}(\Delta^n) = \{\sigma \in \mathrm{Ch}^2(\Delta^n) \mid |\sigma| \cap |\operatorname{Bary}_{\mathcal{A}}(\Delta^n)| = \emptyset\}$$

It is easy to see that $\mathrm{Ch}_{\mathcal{A}}(\Delta^n)$ generalizes the protocol complex of vector-set-consensus model in Chapter 5. Indeed, let $\mathcal{VSC} = (k_1, k_2, ..., k_{n+1})$ be a vector-set-consensus model, its protocol complex $\mathrm{Ch}_{\mathcal{V}}(\Delta^n)$ is equivalent to the symmetric adversary $\mathcal{A}_{\mathcal{V}}$ that consists of all live sets with sizes in $\{k_1, k_2, ..., k_{n+1}\}$.

Let $M_{\mathcal{A}}$ be an iterated model such that the protocol complex has the form of $\mathrm{Ch}^N \mathrm{Ch}_{\mathcal{A}}(\Delta^n)$ for some number N. Operationally, model $M_{\mathcal{A}}$ is a restricted IIS model such that the executions of the first two rounds correspond to a simplex in $\mathrm{Ch}_{\mathcal{A}}(\Delta^n)$. For any round $r \geq 2$, $M_{\mathcal{A}}$ is equivalent to the ordinary IIS model. We now prove that $\mathrm{Ch}_{\mathcal{A}}(\Delta^n)$ characterizes the set-consensus power of $\mathcal{A}$. Let $\mathrm{Ch}_{\mathcal{A}}(\sigma) = \{\tau \in \mathrm{Ch}^2(\sigma) \mid \tau \in \mathrm{Ch}_{\mathcal{A}}(\Delta^n)\}$ be the complex that applies $\mathrm{Ch}_{\mathcal{A}}$ on a simplex $\sigma \in \Delta^n$.

Lemma 6.5. *For any simplex $\sigma \in \Delta^n$, model M_A solves (σ, k_σ)-set-agreement adaptively.*

Proof. Let $\sigma \in \Delta^n$ be an input simplex such that $k_\sigma \geq 1$. Observe that $|\mathrm{Ch}_{\mathcal{A}}(\sigma)|$ is a subspace of $|\Delta^n|_{k_\sigma}$ (Definition 5.5). By Lemma 5.6, there exists a continuous map

$$f : |\mathrm{Ch}_{\mathcal{A}}(\sigma)| \to |\operatorname{skel}^{k_\sigma - 1}(\sigma)|$$

carried by (σ, k_σ)-set-agreement task specification. Since for every $\tau \subseteq \mathrm{Ch}_{\mathcal{A}}(\sigma)$ such that $\dim(\tau) \leq k_\sigma$, f restrict to $|\tau|$ is the identity map. Applying Theorem 3.3, there exists a number N_σ and a carrier-preserving simplicial map

$$\varphi_\sigma : \mathrm{Ch}^{N_\sigma} \mathrm{Ch}_{\mathcal{A}}(\sigma) \to \operatorname{skel}^{k_\sigma - 1}(\sigma).$$

Thus, for each participating set P, there is a protocol in $M_{\mathcal{A}}$ that solves the (σ, k_σ)-set-agreement. Let

$$N = \max_{\sigma \in \Delta^n} N_\sigma,$$

and we obtain a simplicial map

$$\varphi : \mathrm{Ch}^N \mathrm{Ch}_{\mathcal{A}}(\Delta^n) \to \mathrm{skel}^{k-1}(\Delta^n)$$

such that φ restrict to σ agrees with φ_σ. Therefore, $M_{\mathcal{A}}$ solves (σ, k_σ)-set-agreement adaptively. □

The following lemma characterizes the connectivity of $\mathrm{Ch}_{\mathcal{A}}(\Delta^n)$.

Lemma 6.6. *For any simplex $\sigma \in \Delta^n$, $\mathrm{Ch}_{\mathcal{A}}(\sigma)$ is $(k_\sigma - 2)$-connected.*

Proof. Fix a simplex $\sigma \in \Delta^n$, define the topological space

$$|\sigma|_{\mathcal{A}} = |\sigma| - |\,\mathrm{Bary}_{\mathcal{A}}(\Delta^n)|.$$

Observe that the set-consensus power of general adversary is monotonically increasing. In other words, for any participating set P, P' such that $P \subseteq P'$, $\mathrm{setcon}(\mathcal{A}|_P) \leq \mathrm{setcon}(\mathcal{A}|_{P'})$. Let $|\sigma|_{k_\sigma} = |\sigma| - |\,\mathrm{Bary}_{\mathcal{A},\sigma}(\Delta^n)|$, and let

$$X = \bigcup_{\substack{\tau \subseteq \sigma: \\ k_\tau < k_\sigma}} |\,\mathrm{Bary}_{\mathcal{A},\tau}\,|.$$

We have

$$|\sigma|_{\mathcal{A}} = |\sigma|_{k_\sigma} - X.$$

Notice that X is on the boundary of $|\sigma|_{k_\sigma}$ and removing X from $|\sigma|_{k_\sigma}$ does not change its connectivity intuitively. Indeed, applying the argument in Lemma 5.17, it is not hard to see that $|\sigma|_{\mathcal{A}}$ is homotopic equivalent to $|\sigma|_{k_\sigma}$, and $|\sigma|_{\mathcal{A}}$ is homotopy equivalent to $|\operatorname{Ch}_{\mathcal{A}}(\sigma)|$. By Theorem 5.9, $|\sigma|_{k_\sigma}$ is $(k_\sigma - 2)$-connected and therefore, $\operatorname{Ch}_{\mathcal{A}}(\sigma)$ is $(k_\sigma - 2)$-connected. □

6.3 "Wait-at-beginning" Colorless ACT

This section presents the "wait-at-beginning" colorless ACT for general adversary. We show that for colorless task solvability, an adversary $\mathcal{A}$ is equivalent to $M_{\mathcal{A}}$. Since $M_{\mathcal{A}}$ is equivalent to an ordinary IIS except for the first two iterations, the "wait-at-beginning" property justifies the following phenomenon: processes only need the full power of the adversary at the beginning of a protocol and then proceed in a wait-free manner.

Theorem 6.7. *Let $\mathcal{A}$ be a general adversary. A colorless task $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ is solvable in $\mathcal{A}$ if and only if there exists a number N and a carrier-preserving simplicial map $\phi : \operatorname{Ch}^N \operatorname{Ch}_{\mathcal{A}}(\mathcal{I}) \to \mathcal{O}$*

Simplex Agreement

For the sufficiency of Theorem 6.7, it suffice to show that the general adversary $\mathcal{A}$ can solve the colorless simplex agreement $(\Delta^n, \operatorname{Ch}_{\mathcal{A}}(\Delta^n), \operatorname{Ch}_{\mathcal{A}})$. In this task, every process starts with a vertex in Δ^n and then convergence on a simplex $\tau \in \operatorname{Ch}_{\mathcal{A}}(\sigma)$ where σ is the corresponding input simplex in Δ^n. The protocol is presented in Algorithm 3.

Let $\sigma \in \Delta^n$ be the input simplex such that $\chi(\sigma)$ is the participating set. In Algorithm 3, every process p_i starts to write its id in Part and then waits to see a

Algorithm 3: SimplexAgreement $\left(\Delta^n, \mathrm{Ch}^N \mathrm{Ch}_{\mathcal{A}}(\Delta^n), \mathrm{Ch}^N \mathrm{Ch}_{\mathcal{A}}\right)$

Shared Variables: SM$[0..N-1][0..n]$, init $\perp$: single-writer multi-reader memory
Part$[0...N-1]$, init $\perp$

```
1  Part[i] ← i
2  repeat
3  |  snap ← snapshot(Part)
4  |  part_i ← {j | snap[j] ≠ ⊥}
5  until ∃S ∈ A : S ⊆ part_i
6  b_i ← barycenter of the simplex S : S ∈ A, S ⊆ part_i
7  b'_i ← adaptiveSetAgreement(b_i)
8  sview_i^0 ← {b'_i}
9  for ℓ ← 0 to N_A − 1 do
10 |  Immediate
11 |  |  SM[ℓ][i] ← sview_i^ℓ
12 |  |  snap ← snapshot(SM[ℓ])
13 |  sview_i^{ℓ+1} ← {sview_j^ℓ | sview_j^ℓ ∈ snap}
14 v_i ← φ(sview_i^{N_A})
15 return v_i
```

participating set that contains at least one live set (lines 1-5). At line 6, it selects a live set $S \in \mathcal{A}$ that is contained in the current participating set. Then, p_i choose a vertex b_i, the barycenter of a face in σ which can be identified as the live set S. Next, p_i uses the algorithm in [30] to solve the adaptive set-agreement with the input vertex b_i and obtains a vertex b'_i. Thus, every process gets a vertex in the complex $\mathcal{L}_{\mathcal{A}}(\sigma)$, defined by

$$\mathcal{L}_{\mathcal{A}}(\sigma) = \{\tau \in \mathrm{Bary}(\sigma) \mid \forall v \in \tau, \mathrm{Car}(v, \sigma) \in \mathcal{A}\}.$$

Note that $\mathcal{L}_{\mathcal{A}}(\sigma)$ is a subcomplex of the barycentric subdivision of σ such that the carrier of every vertex represents a live set in $\mathcal{A}$. Since at most k_σ vertices are returned from the adaptive-set-agreement, the set of vertices $\{b'_i\}$ chosen by participating processes at line 7 forms a $(k_\sigma - 1)$-dimensional simplex in $\mathcal{L}_{\mathcal{A}}(\sigma)$. At lines 8-13, p_i writes its vertex b'_i and repeatedly executes immediate snapshots $N_{\mathcal{A}}$ times. By ignoring processes' ids in each immediate snapshot, the view of the final immediate snapshot can be identified as a vertex in $\mathrm{Bary}^{N_{\mathcal{A}}}(\mathcal{L}_{\mathcal{A}}(\sigma))$. Then, p_i applies a simplicial map φ to obtain a vertex $v_i \in \mathrm{Ch}_{\mathcal{A}}(\sigma)$ as the output for the simplex agreement.

Lemma 6.8. *Algorithm 3 solves the colorless simplex agreement on* $\mathrm{Ch}_{\mathcal{A}}(\Delta^n)$ *in adversary* $\mathcal{A}$*.*

Proof. It is not hard to see that processes eventually terminate since processes exit the loop at lines 2-5 in an $\mathcal{A}$-compliant execution. Hence, it suffices to show the existence of $N_{\mathcal{A}}$ and φ in Algorithm 3. Notice that processes start with a $(k_\sigma - 1)$-dimensional simplex in $\mathcal{L}_{\mathcal{A}}(\sigma)$. By Lemma 5.17, the complex $\mathrm{Ch}_{\mathcal{A}}(\sigma)$ is $(k_{m+1} - 2)$-connected. Applying Lemma 5.21, there exists a carrier-preserving continuous map

$$f : |\,\mathrm{skel}^{k_\sigma - 1}(\mathcal{L}_{\mathcal{A}}(\sigma))| \to |\,\mathrm{Ch}_{\mathcal{A}}(\sigma)|$$

By Theorem 3.3, there exists a number $N_{\mathcal{A}} \in \mathbb{N}$ and a carrier-preserving simplicial map

$$\varphi : \mathrm{Bary}^{N_{\mathcal{A}}}(\mathrm{skel}^{k_\sigma - 1}(\Delta^n)) \to \mathrm{Ch}_A(\sigma).$$

Therefore, the number $N_{\mathcal{A}}$ and the simplicial map φ are indeed well defined. □

General Adversary Simulation

For the necessity of Theorem 6.7, we show that model $M_{\mathcal{A}}$ can simulate a colorless task protocol for a general adversary.

Algorithm 4: BGG simulation algorithm for process p_i

Local Variables: $\mathrm{val}_{\mathrm{in}}$: p_i's input value for task T
P, init $\perp$: participating set
$\mathrm{val}_{\mathrm{out}}$, init $\perp$: task output value

```
1 val_in ← adaptiveSetAgreement(val_in)
2 P ← current participating set in A
3 repeat
4 |   run one step BG simulation using val_in and participating set P
5 |   if ∃p_j : Outputed(p_j) = true then
6 |   |   val_out ← an output value produced in the BG simulation
7 until val_out ≠ ⊥
8 return val_out
```

The idea is to use a modified BGG simulation to produce an $\mathcal{A}$-compliant execution in $M_{\mathcal{A}}$, which combines the simulation schemes in [16] and [19]. The code of the simulation is presented in Algorithm 4 and Algorithm 5. In the beginning, processes solve the adaptive set-agreement using the algorithm in Lemma 6.5. Every process p_i proposes its task input value $\mathrm{val}_{\mathrm{in}}$ and adopts a new input value from the adaptive set-agreement (line 1). The number of different input values returned is bounded by the set-consensus power of the current participating set. At line 2, p_i takes a snapshot

Algorithm 5: Code for BG simulator with val_{in} and participating set P

Local Variables: $S_1, S_2, ..., S_k$, init $\perp$: simluiated live sets
$b_1, b_2, ...b_k$, init $\perp$: blocked processes
L, init 1 : current level of simulation
P : participating set in adversary $\mathcal{A}$
val_{in}: input value of the task

```
1  S_1 ← the first live set S ∈ A|_P such that setcon(A|_S) = setcon(A|_P)
2  L ← 1
3  repeat
4  |  ℓ ← 1
5  |  while ℓ < L and the simulation of b_ℓ is still blocked do
6  |  |  ℓ ← ℓ + 1
7  |  if ℓ < L then L ← ℓ
8  |  p_j ← the process in S_L with the least number of simulated steps
9  |  if p_j is not initiated then
10 |  |  run safe-agreement with input val_in to initialize p_j
11 |  else
12 |  |  run BG simulation to simulate the next step of p_j
13 |  if the safe-agreement in the simulation of p_j is blocked and
   |    L < setcon(A|_P) then
14 |  |  b_L ← p_j
15 |  |  S_{L+1} ← the first live set S ∈ A|_{S_L, b̄_L} such that
   |  |    setcon(A|_S) ≥ setcon(A|_P) − L
16 |  |  L ← L + 1
17 forever
```

of the memory and calculates a participating set P. Then, p_i repeatedly executes the BG simulation (described in Algorithm 5) using the updated input value val_{in} and participating set P.

In Algorithm 5, the BG simulation is implemented by values. Simulators ignore ids in their snapshots. Thus, two processes with the same initial values act as a single simulator .

Every simulator simulates a live set S under the participating set P using the input value val_{in}. The order of the simulation is based on the recursive structure of the set-consensus power. In line 1, a simulator starts the level 1 simulation by selecting a live set $S_1 \in \mathcal{A}|_P$ with the set-consensus power $\text{setcon}(A|_S) = \text{setcon}(A|_P)$. The existence of S_1 follows from the definition of setcon. Additionally, S_1 is chosen deterministically by some prefixed order.

Then, the simulator selects a process p_j in S_1 to simulate a step (lines 9-10). If the safe-agreement in the simulation is not resolved, the simulator sets p_j as the blocked process in the level 1 simulation (line 14). In this case, the simulator proceeds to level 2 simulation and chooses a different live set $S_2 \in \mathcal{A}|_{P,\overline{p_j}}$ such that $\text{setcon}(\mathcal{A}|_{S_2}) \geq \text{setcon}(A|_P) - 1$. The existence of the choice of S_2 follows from the fact that $\text{setcon}(\mathcal{A}|_{P,\overline{p_j}}) \geq \text{setcon}(A|_P) - 1$. Then, the simulator repeatedly simulates processes in S_2 until a new process is blocked. If the simulation of one step is finished, the simulator goes back to level 1 simulation to check whether the blocked safe-agreement is resolved. If so, the simulator selects S_1 to simulate again. Otherwise, it goes to the lowest level simulation that is not blocked.

At lines 3-7 in Algorithm 4, process p_i repeatedly runs one step BG simulation and checks whether a task output is produced in the simulation. If so, p_i immediately terminates the simulation and adopts the task output (line 8). Otherwise, it executes

one round BG simulation again.

Lemma 6.9. *The BGG simulation presented in Algorithm 4 and 5 simulate an $\mathcal{A}$-compliant execution of the protocol.*

Proof. In the BG simulation, every simulated process is initialized with an input value from some participating set. Therefore, the correctness of the simulation follows from the fact that every step is consistently simulated by running the safe-agreement.

To see the liveness of the simulation, we will show that there exists a live set $S \in \mathcal{A}$ such that S is a subset of the participating set P, and processes in S is simulated infinitely often. Therefore, the simulation produced an $\mathcal{A}$-compliant execution, and every simulated process in S eventually get an output. As soon as some outputs are produced in the simulation, other processes immediately adopt these outputs and then terminate.

Notice that the BG simulation in Algorithm 5 is implemented by values, processes that get the same input values from the adaptive-set-agreement act as a single simulator. When the set of participating processes is P, by Lemma 6.5, processes select at most k_P different input values. Hence, there are at most k_P different simulators such that most $k_P - 1$ of them are faulty.

In Algorithm 4, at least one process p_i observes the participating P and starts the BG simulation by selecting a live set from $A|_P$. In the simulation, each faulty simulator can block only one simulated process. Since at most $k_P - 1$ simulators are faulty, p_i observes at most $k_P - 1$ levels of the simulation are blocked. Thus, in line 13 of Algorithm 5 , p_i always evaluates $L < \text{setcon}(A|_P)$ to be true and selects a live set $S_L \in A|_P$ such that no safe-agreement in S_L is blocked forever. Since the live set S_L is simulated for infinitely many steps, the simulation produces an $\mathcal{A}$-compliant execution, and therefore, the simulation terminates eventually. □

We are now ready to prove Theorem 6.7.

Proof. For sufficiency, suppose there exists a number N and a simplicial map $\phi : \mathrm{Ch}^N \mathrm{Ch}_{\mathcal{A}}(\mathcal{I}) \to \mathcal{O}$. By Lemma 6.8, for any input simplex $\sigma \in \mathcal{I}$, there is a protocol in $\mathcal{A}$ that solves the colorless simplex agreement on $\mathrm{Ch}_{\mathcal{A}}(\sigma)$. To solve task T, every process p_i in $\mathcal{A}$ first solves the simplex agreement on $\mathrm{Ch}_{\mathcal{A}}(\sigma)$, and then executes N round immediate snapshot in a wait-free manner to get a vertex v_i in $\mathrm{Ch}^N \mathrm{Ch}_{\mathcal{A}}(\sigma)$. Next, p_i applies the simplicial map to get an output value $\varphi(v_i)$ in $\mathcal{O}$. The correctness follows from the fact that ϕ is simplicial and carrier-preserving.

For the other direction, assume T is a colorless task, and there exists a protocol in $\mathcal{A}$ that solves T. By Lemma 6.9, Algorithm 4 simulates an $\mathcal{A}$-compliant execution of the protocol in model $M_{\mathcal{A}}$. Since the simulation eventually terminates, every process in $M_{\mathcal{A}}$ terminates after N rounds for some $N > 0$. Since $\mathrm{Ch}^N \mathrm{Ch}_{\mathcal{A}}(\mathcal{I})$ is the N round protocol complex of $M_{\mathcal{A}}$, the simulation implies the desired carrier-preserving simplicial map $\phi : \mathrm{Ch}^N \mathrm{Ch}_{\mathcal{A}}(\mathcal{I}) \to \mathcal{O}$. □

Chapter 7

A Sufficient Topological Condition for Task Solvability in Vector-Set-Consensus Model

The original ACT [6, 27] reduces the wait-free task solvability problem to the existence of a specific simplicial map from a chromatic subdivision to the task output complex. However, showing a task is solvable requires an explicit construction of the subdivision and a simplicial map, which is usually not convenient. Herlihy et al. [24] proved a sufficient topological condition to ensure a task has a wait-free protocol. In practice, this characterization is more convenient for verification purposes as it only requires checking the output complex.

Theorem 7.1. *[24, Theorem 11.5.2] Let $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ be a colored task. If for each $\sigma \in \mathcal{I}$, $\Gamma(\sigma)$ is $(\dim(\sigma) - 1)$-connected, and $\mathcal{O}$ is link-connected. Then, there exists a wait-free protocol that solves T.*

In the proof of Theorem 7.1, the authors first show that there exists a chromatic subdivision $\mathrm{Div}(\mathcal{I})$ and a color-preserving simplicial map

$$\varphi : \mathrm{Div}(\mathcal{I}) \to \mathcal{O}.$$

According to wait-free ACT(Theorem 5.1), there exists a wait-free protocol that solves T.

Borowsky and Gafni [6] presented an alternative proof of ACT in an algorithmic way. In particular, their proof uses the convergence algorithm, which provides an algorithmic construction of the chromatic simplicial map

$$\varphi : \mathrm{Ch}^N(\mathcal{I}) \to \mathrm{Div}(\mathcal{I})$$

for some $N \in \mathbb{N}$. The original appearance of the convergence algorithm was sketchy, and no proof was given. Later, Vikram et al. [39] provided a detailed presentation of the algorithm and proof of correctness.

Observe that the convergence algorithm can be used as a universal wait-free protocol for a task if the topological condition in Theorem 7.1 is satisfied.

Theorem 7.2. *Let $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ be a colored task. If for each $\sigma \in \mathcal{I}$, $\Gamma(\sigma)$ is $(\dim(\sigma) - 1)$-connected, and $\mathcal{O}$ is link-connected, then T can be solved by the convergence algorithm in the wait-free IIS model.*

Borowsky [3] proposed a generalized version of the convergence algorithm in the active-resiliency model and set-consensus collection model. Furthermore, Borowsky presented a sufficient topological condition on task solvability in the set-consensus collection model.

Theorem 7.3 (Borowsky)**.** *Let $S = \{(m_1, \ell_1), (m_2, \ell_2), ..., (m_t, \ell_t)\}$ be a set-consensus collection model such that for $i = 1, ..., n + 1$, if $m_k \leq i$ and $\forall m_h \leq i$, it holds $m_k/\ell_j > m_h/\ell_h$. Then recursively define $k_0 = 0$, and $k_i = (i/m_k)\ell_k + k_{i_1}$ where $i_1 = i \mod m_k$. Let $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ be a task such that for each $\sigma \in \mathcal{I}$, $\dim(\sigma) = m$, $\Gamma(\sigma)$ is*

$(k_m - 2)$-connected and $\mathrm{Lk}(\tau, \Gamma(\sigma))$ *is* $(k_{(m-c)} - 2)$*-connected. Then T is solvable in the set-consensus collection model.*

However, the topological condition in Theorem 7.3 is not tight since the power of the set-consensus collection model is not fully utilized. In this chapter, we present a new sufficient topological condition to ensure a task is solvable in the vector-set-consensus model. Our result generalizes Theorem 7.2 and improves the result of Borowsky as our criteria is topologically weaker. Moreover, the generalized convergence algorithm formulated in [3] is still sketchy and missing a complete proof. We close the gap by presenting a full description and proving its correctness. We also present an application of the generalized convergence algorithm in chromatic loop agreement.

7.1 The Generalized Convergence Algorithm in Vector-Set-Consensus Model

In this section, we present the generalized convergence algorithm in the vector-set-consensus model. Given a colored task, we show that the algorithm solves the task if the output complex satisfies certain topological properties.

Theorem 7.4. *A task $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ can be solved by the generalized convergence algorithm in the vector-set-consensus model $\mathcal{VSC} = (k_1, k_2, ..., k_{n+1})$ if the following properties are satisfied: for each $\sigma \in \mathcal{I}$,* $\dim(\sigma) = m$*, $\Gamma(\sigma)$ is $(k_{m+1} - 2)$-connected and for every $\tau \in \Gamma(\sigma)$ where* $\dim(\tau) = c$*,* $\mathrm{Lk}(\tau, \Gamma(\sigma))$ *is $(k_{(m-c)} - 2)$-connected.*

We present the generalized convergence algorithm in Algorithm 6. The algorithm proceeds round by round. In each round, each process solves a colorless task using a

subroutine, and then determines whether it terminates or participates the next round. If a process decides, it chooses a vertex in the output complex with a matching color. Otherwise, it continues to the next round to solve a new colorless task.

In the first round, every process p_i starts with an input corresponding to vertex v_{in}^i and writes its input vertex in Part[i] (lines 1-2). Then, p_i solves the simplex agreement on the barycentric subdivision of the output complex, defined as follows.

Definition 7.5. The Bary-Simplex agreement task $T_B = (\mathcal{I}, \text{Bary}(\mathcal{O}), \Gamma_B)$ is a colorless task such that the input complex is the same as T, and the output complex is the barycentric subdivision Bary($\mathcal{O}$). For any simplex $\sigma \in \mathcal{I}$, the valid output complex is $\Gamma_B(\sigma) = \text{Bary}(\Gamma(\sigma))$.

In Algorithm 7, processes start with solving the adaptive set-agreement (line 1) using Algorithm 2. Every process p_i proposes its input vertex and gets a return vertex. Then, p_i executes N_B rounds of immediate snapshot. In each round, p_i ignores processes' id and only uses the set of views from the previous round (line 11). Next, it applies a decision map based on its view and obtains a vertex in Bary($\mathcal{O}$). The number of rounds N_B and the decision map φ depends on the output complex $\mathcal{O}$ and will be justified later. The vertex v in Bary($\mathcal{O}$) is identified as a simplex in $\mathcal{O}$, and used as the input in an immediate snapshot (line 5-6). At line 7, p_i checks the simplices that appear in its snapshot. If every simplex σ_j^1 in p_i's snapshot contains a vertex $\widetilde{v}_i$ with the matching color, p_i returns $\widetilde{v}_i$ and terminates. Otherwise, p_i prepares for round 2 by calculating view_i^1, an estimate of simplex that processes converged (line 10). view_i^1 is computed as the union of simplices σ_j^1 appears in p_i's snapshot snap_i^1, by removing any vertex with the same color as p_i.

At the beginning of round $r \geq 2$, process p_i recalculates the current participating set and then computes core_i^r, an estimate of decided vertices in previous rounds (lines

Algorithm 6: The generalized convergence algorithm: process p_i

Shared Variables: Part[0..n], init $\perp$
Simplexes[1..$n+1$][0..n], init $\perp$: single-writer multi-reader memory
View[1...$n+1$][0..n], init $\perp$

$v_{\text{in}}^i \leftarrow$ the vertex in $\mathcal{I}$ corresponds to p_i with input val_i
Part$[i] \leftarrow v_{\text{in}}^i$
$\sigma_i^1 \leftarrow$ **BarySimplexAgreement**$(v_{\text{in}}^i, \mathcal{I}, \text{Bary}(\mathcal{O}))$
Immediate
  Simplexes$[1][i] \leftarrow \sigma_i^1$
  $\text{snap}_i^1 \leftarrow$ **snapshot**(Simplexes[1])
if $\exists \tilde{v}_i : \tilde{v}_i \in \bigcap_{\sigma_j^1 \in \text{snap}_i^1} \sigma_j^1$ and $\chi(\tilde{v}_i) = p_i$ **then**
  return $\tilde{v}_i$
else
  $$\text{view}_i^1 \leftarrow \bigcup_{\sigma_j^1 \in \text{snap}_i^1} \sigma_j^1 - \left\{ \tilde{v}_i \mid \chi(v_i) = p_i, v_i \in \bigcup_{\sigma_j^1 \in \text{snap}_i^1} \sigma_j^1 \right\}$$
for $r \leftarrow 2$ **to** $n+1$ **do**
  Immediate
    $\text{View}[r][i] \leftarrow \text{view}_i^{r-1}$
    $\text{snapView}_i^r \leftarrow$ **snapshot**(View$[r]$)
    $\text{part}_i^r \leftarrow$ **snapshot**(Part)
  $$\text{core}_i^r \leftarrow \bigcap_{j \in \text{snapView}_i^r} \text{view}_j^{r-1}$$
  $\sigma_i^r, \overline{\text{core}_i^r} \leftarrow$ **LinkBarySimplexAgreement**$(\text{core}_i^r, \text{part}_i^r, \mathcal{I}, \mathcal{O}, \Gamma)$
  Immediate
    Simplexes$[r][i] \leftarrow (\sigma_i^r, \overline{\text{core}_i^r})$
    $\text{snap}_i^r \leftarrow$ **snapshot**(Simplexes$[r]$)
  if $\exists \tilde{v}_i : \tilde{v}_i \in \bigcap_{j \in \text{snap}_i^r} \sigma_j^r$ and $\chi(\tilde{v}_i) = p_i$ **then**
    return $\tilde{v}_i$
  else
    $$\text{view}_i^r \leftarrow \left(\bigcup_{j \in \text{snap}_i^r} \sigma_j^r \right) \bigcup \left(\bigcap_{j \in \text{snap}_i^r} \overline{\text{core}_i^r} \right) - \left\{ \tilde{v}_i \mid \chi(v_i) = p_i, v_i \in \bigcup_{\sigma_j^1 \in \text{snap}_i^r} \sigma_j^r \right\}$$

Algorithm 7: BarySimplexAgreement$(v_{\text{in}}^i, \mathcal{I}, \text{Bary}(\mathcal{O}))$

Shared Variables: SM$[0..N-1][0..n]$, init $\perp$: single-writer multi-reader memory

$v' \leftarrow$ **adaptiveSetAgreement**(v_i)
sview$_i^0 \leftarrow \{v'\}$
for $\ell \leftarrow 0$ **to** $N_B - 1$ **do**
  Immediate
    SM$[\ell][i] \leftarrow$ sview$_i^\ell$
    snap $\leftarrow$ **snapshot**(SM$[\ell]$)
  sview$_i^{\ell+1} \leftarrow \{\text{sview}_j^\ell \mid \text{sview}_j^\ell \in \text{snap}\}$
$v_{\text{out}} \leftarrow \varphi(\text{sview}_i^{N_B})$
$\sigma \leftarrow$ the simplex in $\mathcal{O}$ identified with $v_{\text{out}} \in \text{Bary}\,\mathcal{O}$
return σ

Algorithm 8: LinkBarySimplexAgreement$(\text{core}_i^r, \text{part}_i^r, \mathcal{I}, \mathcal{O}, \Gamma)$

Shared Variables: SM$[0..N-1][0..n]$, init $\perp$
// single-writer multi-reader memory

$\text{Lk}_i^r \leftarrow \text{Lk}(\text{core}_i^r, \Gamma(\text{part}_i^r))$
$v_i \leftarrow$ a vertex $v : v \in \text{Lk}_i^r, \chi(v) = p_i$
$v', \text{core}', \text{part}' \leftarrow$ **adaptiveSetAgreement**$(v_i, \text{core}_i^r, \text{part}_i^r)$
$\text{core}_i^r \leftarrow \text{core}_i^r \cap \text{core}'$; $\text{part}_i^r \leftarrow \text{part}_i^r \cup \text{part}'$
sview$_0^i \leftarrow \{v', \text{core}_i^r, \text{part}_i^r\}$
for $\ell \leftarrow 0$ **to** $N_r - 1$ **do**
  Immediate
    SM$[\ell][i] \leftarrow$ sview$_i^\ell$
    snap $\leftarrow$ **snapshot**(SM$[\ell]$)
  sview$_i^{\ell+1} \leftarrow \{\text{sview}_j^\ell \mid \text{sview}_j^\ell \in \text{snap}\}$
$\text{core}_i^r \leftarrow \bigcap_{\text{core}_j^r \in \text{sview}_i^N} \text{core}_j^r$; $\text{part}_i^r \leftarrow \bigcup_{\text{part}_j^r \in \text{sview}_i^N} \text{part}_j^r$
$v_{\text{out}} \leftarrow \phi(\text{sview}_i^N)$
$\sigma_i^r \leftarrow$ the simplex in $\text{Lk}(\text{core}_i^r, \Gamma(\text{part}_i^r)))$ identified with v_{out}
return $\sigma_i^r, \overline{\text{core}_i^r}$

12-16). core_i^r is defined as the intersections of all view_j^1 appeared in p_i's snapshot. Next, p_i executes Algorithm 8 to solve the Link-Bary-Simplex agreement (line 17). Roughly speaking, in the Link-Bary-Simplex agreement, processes converge on the barycentric subdivision of the link of decided vertices. The input complex and the output complex are determined dynamically based on previous rounds.

In Algorithm 8, p_i uses the updated participating set and core to calculate its convergence link

$$\text{Lk}_i^r = \text{Lk}(\text{core}_i^r, \Gamma(\text{part}_i^r))$$

where $\Gamma(\text{part}_i^r)$ denotes the valid output subcomplex, defined as

$$\Gamma(\text{part}_i^r) = \Gamma(\tau_i)$$

where τ_i is the input simplex observed by p_i in part_i^r.

Note that processes may compute different cores and links at the beginning. However, we will later show that at least one process calculates the largest link, i.e., the link of the decided vertices in previous rounds. Moreover, every Lk_i^r is a subcomplex of the largest link and is related by containment. Fix round r, the Link-Bary-Simplex agreement is formally defined as follows.

Definition 7.6. The Link-Bary-Simplex agreement $T_L = (\mathcal{I}_L, \mathcal{O}_L, \Gamma_L)$ is a colorless task such that the input complex $\mathcal{I}_L$ is a subcomplex of

$$\text{skel}^0(\mathcal{O}) \times \mathcal{O} \times \Delta^n$$

where $\text{skel}^0(\mathcal{O})$ consists of the vertices in $\mathcal{O}$. Each process p_i's initial state is a triple that contains an initial vertex v_{in}^i chosen from its convergence link, its initial core and the participating set it observed. The output complex $\mathcal{O}_L$ is the barycentric

subdivision Bary($\mathcal{O}$). The task specification requires that processes converge on a simplex in Bary($\mathcal{O}$) which is a simplex in the barycentric subdivision of the link of intersections of the cores. In other words, for each input simplex $\sigma \in \mathcal{I}$ such that $\sigma = \{(v_i, \text{core}_i^r, \text{part}_i^r)\}$,

$$\Gamma_L(\sigma) = \text{Lk}\left(\bigcap_i \text{core}_i^r, \Gamma(P)\right)$$

where $P = \bigcup_i \text{part}_i^r$.

Algorithm 8 solves the Link-Bary-Simplex agreement. In line 3, p_i submits its initial configuration and solves the adaptive-set-agreement. p_i adopts a new vertex and then runs N_r rounds immediate snapshot. As in the first round, processes ignore ids and only record the set of views. Next, p_i updates its participating set and its core_i^r by the intersection of the cores observed in the snapshot (line 11). After that, p_i applies a decision map based on its N_rth immediate snapshot view and gets a vertex v_{out} in the barycentric subdivision of the link. Then, p_i returns a simplex σ_i^r in $\mathcal{O}$ which can be identified as v_{out}.

With simplex σ_i^r and updated core, p_i executes an immediate snapshot again in round r. If there exists a vertex $\widetilde{v}_i$ that appears in every simplex in its snapshot, p_i chooses $\widetilde{v}_i$ as its output and terminates. Otherwise, p_i calculates view_i^r and then enters round $r+1$.

7.2 Proof of Correctness

We now prove the correctness of the generalized convergence algorithm. Throughout this section, we assume that the vector-set-consensus model $\mathcal{VSC} = (k_1, k_2, ..., k_{n+1})$ is

fixed beforehand. Also, the output complex of $T = (\mathcal{I}, \mathcal{O}, \Gamma)$ satisfies the topological condition in Theorem 7.4.

Lemma 7.7. *Let m processes participate Algorithm 7. After processes execute the loop N rounds (line 3-7) , the set of views*

$$\{\text{sview}_0^N, ..., \text{sview}_{m-1}^N\}$$

forms a simplex in $\text{Bary}^N(\text{skel}^{k_m-1}(\mathcal{I}))$.

Proof. Let $p_0, p_1, ..., p_{m-1}$ be the participating processes, and let σ be the corresponding input simplex. We show that for each round r, the set of views

$$\text{sview}^r = \{\text{sview}_0^r, ..., \text{sview}_{m-1}^r\}$$

forms a simplex in $\text{Bary}^r(\text{skel}^{k_m-1}(\sigma))$. The proof is by induction, the base case $r = 0$ is trivial. In line 1, every process submits its initial vertex and solves adaptive-set-agreement. The number of different vertices in sview^0 is bounded by k_m. Hence, sview^0 forms a simplex in

$$\text{skel}^{k_m-1}(\sigma) = \text{Bary}^0 \,\text{skel}^{k_m-1}(\sigma).$$

For the inductive step, by definition, sview_i^r is a subset of sview^{r-1}. By containment property of immediate snapshot, sets in sview^r can be linearly ordered by inclusion. Since sview^{r-1} is a simplex τ in

$$\text{Bary}^{r-1}(\text{skel}^{k_m-1}(\sigma))$$

by induction hypothesis, each sview_i^r can be identified with a subsimplex of τ and can be ordered by containment. By definition of Barycentric subdivision, sview^r can be identified with a simplex in $\text{Bary}(\tau)$, which is a simplex in $\text{Bary}^r(\text{skel}^{k_m-1}(\sigma))$. □

We now prove that Algorithm 7 solves the Bary-Simplex agreement T_B. Note that the output complex $|\,\text{Bary}(\mathcal{O})|$ is homeomorphic to$|\mathcal{O}|$ and therefore, they have the same connectivity. It suffices to show that there is a carrier-preserving continuous map $f : |\mathcal{I}| \to |\,\text{Bary}\,\Gamma(\sigma)|$.

Lemma 7.8. *Algorithm 7 solves the Bary-Simplex Agreement* $T_B = (\mathcal{I}, \text{Bary}(\mathcal{O}), \Gamma_B)$.

Proof. Without loss of generality, assume m processes $p_0, p_1, ..p_{m-1}$ participate. Let σ be the corresponding input simplex such that $\dim(\sigma) = m - 1$. By Lemma 5.21, there exists a carrier-preserving continuous map

$$f : |\,\text{skel}^{k_m-1}(\sigma))| \to |\,\text{Bary}\,\Gamma(\sigma)|.$$

Applying Theorem 3.3, there exists a number $N_B \in \mathbb{N}$ and a simplicial map

$$\varphi : \text{Bary}^N(\text{skel}^{k_m-1}(\sigma)) \to \text{Bary}\,\Gamma(\sigma)$$

carried by Γ_B. By lemma 7.7, after executing the loop (line 3–7) N_B rounds, the set $\text{sview}^N = \{\text{sview}_0^N, ..., \text{sview}_{m-1}^N\}$ forms a simplex in $\text{Bary}^N(\text{skel}^{k_m-1}(\mathcal{I}))$. Hence, each process applies the simplicial map φ and yields a vertex in $\text{Bary}(\mathcal{O})$. The correctness follows from the fact that φ is simplicial and carrier-preserving. □

At the end of Algorithm 7, every process returns a simplex σ_i in $\mathcal{O}$, which is identified with its output vertex in Bary-Simplex agreement. Moreover, by definition

of barycentric subdivision, the set of returned simplices $\{\sigma_i\}$ can be ordered by inclusion. We now show that if a vertex is decided in some round r, it will be observed in subsequent rounds.

Lemma 7.9. *If a process p_i decides vertex $\tilde{v}_i$ at round r where $\chi(\tilde{v}_i) = p_i$, $\tilde{v}_i$ appears in all $\text{view}_j^{r'}$ and $\text{core}_j^{r'+1}$ for every process p_j that participates round r', where $r' \geq r$.*

Proof. The proof is by induction. For the base case $r' = r$, let p_j be a process that does not decide at round r. We have two cases. Assume p_j takes the immediate snapshot at line 4 or 18 before p_i. In order for p_i to decide $\tilde{v}_i$, by line 21, p_i must observe $\tilde{v}_i$ appear in σ_k^r for every process p_k that executes line 4 or 18 before p_i. Therefore, we can conclude that $\tilde{v}_i \in \sigma_j^r$. Since $\chi(\tilde{v}_i) \neq p_j$, $\tilde{v}_i$ is included in view_j^r. In the other case, p_j takes the immediate snapshot after p_i and includes σ_i^r in its snapshot snap_j^r. Since $\tilde{v}_i \in \sigma_i^r$, view_j^r must contain $\tilde{v}_i$. Therefore, $\tilde{v}_i$ is included in view_j^r for every process p_j that does not decide at round r. Since core_j^{r+1} is calculated as the intersection of all $\text{view}_k^r \in \text{snap}_j^{r+1}$, we have $\tilde{v}_i \in \text{core}_j^{r+1}$.

Assume $\tilde{v}_i$ appears in $\text{view}_j^{r'}$ and $\text{core}_j^{r'+1}$ for $r \leq r'$. For the inductive step, assume p_j participates round $r'+1$ and does not decide. Observe that $\overline{\text{core}_i^{r'+1}}$ is calculated as the intersection of $\text{core}_j^{r'+1}$. Since the view of p_j is calculated by

$$\text{view}_i^{r'+1} \leftarrow \left(\bigcup_{j \in \text{snap}_i^{r'}} \sigma_j^1 \right) \bigcup \left(\bigcap_{j \in \text{snap}_i^{r'}} \overline{\text{core}_i^{r'}} \right) - \left\{ \tilde{v}_i \mid \chi(v_i) = p_i, v_i \in \bigcup_{\sigma_j^{r'} \in \text{snap}_i^{r'}} \sigma_j^1 \right\},$$

$\tilde{v}_i$ is included in $\text{view}_i^{r'+1}$. As $\text{core}_j^{r'+2}$ is calculated as the intersection of $\text{view}_k^{r'+1}$ for every p_k that participates round $r'+2$, we can conclude that $\tilde{v}_i \in \text{core}_j^{r'+2}$, which completes the inductive step. □

The following lemma shows that if a process p_i does not decide at some round r, it must compute a valid convergence link in Link-Bary-Simplex agreement.

Lemma 7.10. *If p_i does not decide on round r, then* view_i^r, core_i^{r+1} *does not contain any vertex v_i where $\chi(v_i) = p_i$, and* Lk_i^{r+1} *contains a vertex v_i such that $\chi(v_i) = p_i$.*

Proof. At line 10 or 24, p_i calculates view_i^r by taking the union of simplices in its snapshot and removing any vertices v_i with color p_i. As core_i^{r+1} is calculated as the intersection of view_j^r for processes that participate round $r+1$, core_i^{r+1} does not contain any vertex v_i such that $\chi(v_i) = p_i$. Since part_i^r contains p_i's initial vertex v_{in}^i and $\mathcal{O}$ is chromatic, $\text{Lk}_i^{r+1} = \text{Lk}(\text{core}_i^r, \Gamma(\text{part}_i^r))$ must contain a vertex v_i such that $\chi(v_i) = p_i$. □

In each round r, every process p_i calculates the core_i^r as an estimate of the decided vertices in previous rounds. Every core is a superset of decided vertices, and different processes may calculate different cores. However, the following lemma shows that at least one process calculates a core which is precisely the set of decided vertices.

Lemma 7.11. *Let τ be the the set of vertices decided in the first r rounds. At round $r+1$, at least one process p_i calculates* $\text{core}_i^{r+1} = \tau$.

Proof. Let τ_r be the set of vertices decided at first r rounds, and let p_i be the last process executes the immediate snapshot at lines 12-15 in round $r+1$. By Lemma 7.9, $\tau_r \subseteq \text{view}_j^r$ for every p_j that participates round $r+1$, and so, we conclude that $\tau_r \subseteq \text{core}_i^{r+1}$. On the other hand, by way of contradiction, assume $\text{core}_i^{r+1} \not\subseteq \tau_r$, then there exists a vertex v', such that $v' \in \text{core}_i^{r+1}$ and $v' \notin \tau_r$. Let p_k be the process such that $\chi(v') = p_k$. Since $v' \notin \tau_r$, p_k does not decide in the first r rounds and thus participates round $r+1$. By Lemma 7.10, we have $v' \notin \text{view}_k^r$. However, core_i^{r+1} is

the intersection of every view in round r, it must be the case that $v' \notin \text{core}_i^{r+1}$, a contradiction. □

We now show that the Link-Bary-Simplex agreement is well defined.

Lemma 7.12. *In each Link-Bary-Simplex Agreement, the convergence links* $\{\text{Lk}_i^r\}$ *can be linearly ordered by inclusion.*

Proof. The convergence link is calculated as

$$\text{Lk}_i^r = \text{Lk}(\text{core}_i^r, \Gamma(\text{part}_i^r)).$$

By definition of link and monotonicity of Γ, it suffices to prove that part_i^r is linearly ordered by inclusion and core_i^r is reversely linear ordered. For every p_i that participates round r, part_i^r and snapView_i^r are linearly ordered as the order of executing the immediate snapshot (IS) at line 12-15. Notice that p_i calculates

$$\text{core}_i^r = \bigcap_{j \in \text{snapView}_i^r} \text{view}_j^{r-1}$$

at line 16. Hence, the core_i^r is reversely linear ordered with respect to snapView_i^r. □

We are now ready to show that Algorithm 8 solves the Link-Bary-Simplex agreement.

Lemma 7.13. *Algorithm 8 solves the Link-Bary-Simplex agreement* $T_L = (\mathcal{I}_L, \mathcal{O}_L, \Gamma_L)$.

Proof. Without loss of generalization, let m processes participate T, and let $\sigma \in \mathcal{I}$ be the corresponding input simplex. Let $p_0, p_1, ..., p_{c-1}$ be processes that terminated in first $r-1$ rounds, and let τ be the decided simplex. Processes $p_c, p_{c+1}, ..., p_{m-1}$

participate round r. By Lemma 7.12, $\mathrm{Lk}^r_c, \mathrm{Lk}^r_{c+1}, ..., \mathrm{Lk}^r_{m-1}$ can be linearly ordered by containment. By Lemma 7.11, at least one process p_i calculates the largest convergence link, i.e.,

$$\mathrm{Lk}^r_i = \mathrm{Lk}(\tau, \Gamma(\sigma)).$$

Let Lk^r denote $\mathrm{Lk}(\tau, \Gamma(\sigma))$. Notice that Lk^r is $(k_{(m-c)} - 2)$ connected, and therefore, $\mathrm{Bary}(\mathrm{Lk}^r)$ is $(k_{(m-c)} - 2)$-connected since $|\,\mathrm{Bary}(\mathrm{Lk}^r)|$ is homeomorphic to $|\,\mathrm{Lk}^r\,|$. By Lemma 5.21, there is a carrier-preserving continuous map

$$f : |\,\mathrm{skel}^{k_m}(\Delta^m)| \to |\,\mathrm{Bary}(\mathrm{Lk}^r)|.$$

Applying Theorem 3.3, there exists a number N_r and a carrier-preserving simplicial map

$$\phi_r : \mathrm{Bary}^{N_r}(\mathrm{skel}^{k_m}(\Delta^m)) \to \mathrm{Bary}(\mathrm{Lk}^r).$$

In Algorithm 8, processes that participate round r first solve the adaptive-set-agreement and then execute N_r rounds immediate snapshot. By Lemma 7.7, each process obtains a vertex in $\mathrm{Bary}^{N_r}(\mathrm{skel}^{k_m}(\Delta^m))$ and then apply ϕ_r to get the output. The correctness follows from the fact that ϕ_r is simplicial and carrier-preserving. □

We are now ready to prove the termination and validity property of the generalized convergence algorithm.

Theorem 7.14. *Algorithm 6 terminates after at most $n + 1$ rounds.*

Proof. We will prove that in each round r, at least one process decides. By Lemma 7.8 and 7.13, each process p_i that participates round r returns a simplex $\sigma^r_i \in \mathcal{O}$. Let

the set $\{\sigma_i^r\}$ denote simplices returned from Bary-Simplex agreement or Link-Bary-Simplex agreement in round r. By definition of barycentric subdivision, simplices in $\{\sigma_i^r\}$ can linearly ordered by containment.

Consider the set

$$V_r = \bigcap_i \sigma_i^r,$$

which contains vertices that appears in every σ_i^r. Let U_r be the set of processes with matching color in V_r, i.e., $U_r = \chi(V_r)$. For each process $p_j \in U_r$, it must evaluate line 10 or line 21 to be true and then terminate since there is a vertex $v_j \in V_r$ such that $\chi(v_j) = p_j$.

Therefore, processes in U_r terminate at round r. Since $\{\sigma_i^r\}$ can be linearly ordered and it is not empty, V_r is also not empty. Therefore, at least one process decides in each round, and the algorithm terminates in at most $n + 1$ rounds. □

Theorem 7.15. *Let σ be the input simplex in $\mathcal{I}$ corresponding to the participating set. The set of decided vertices forms a simplex in $\Gamma(\sigma)$.*

Proof. For $r = 1, 2, ...n + 1$, let τ_r be the set of vertices decided at round r, and let

$$\widetilde{\tau_r} = \bigcup_{i=1}^{r} \tau_i$$

be the set of decided vertices in the first r rounds. We show that $\widetilde{\tau_r}$ is a simplex in $\Gamma(\sigma)$ by induction. For the base case $r = 1$, by Lemma 7.8, $\widetilde{\tau_1}$ consists of vertices in simplices of $\Gamma(\sigma)$ related by containment. By downward closure of simplicial complex, $\widetilde{\tau_1}$ is a simplex of $\Gamma(\sigma)$.

Assume that $\widetilde{\tau_r}$ is a simplex in $\Gamma(\sigma)$. For the inductive step, consider $\widetilde{\tau}_{r+1} = \widetilde{\tau_r} \cup \tau_{r+1}$. By Lemma 7.13, processes participate round $r + 1$ solve the Link-Bary-

Simplex agreement and return simplices in $\mathrm{Lk}(\widetilde{\tau}_r, \Gamma(\sigma))$ which can be ordered by containment. Let σ_ℓ^{r+1} be the largest simplex chosen at round $r+1$. By definition of link, the set

$$\sigma_\ell^{r+1} \cup \widetilde{\tau}$$

is a simplex of star $\mathrm{st}(\widetilde{\tau}_\mathrm{r})$, and therefore, is a simplex in $\Gamma(\sigma)$. Since vertices in τ_{r+1} is a subset of σ_ℓ^{r+1} and by downward inclusion, $\widetilde{\tau}_{r+1} = \widetilde{\tau}_r \cup \tau_{r+1}$ is a simplex. Notice that σ_ℓ^{r+1} is simplex in $\Gamma(\sigma)$ because of the specification of Link-Bary-Simplex agreement. Therefore, $\widetilde{\tau}_{r+1}$ is a simplex in $\Gamma(\sigma)$, which completes the induction step. □

By Theorem 7.14 and 7.15, the generalized convergence algorithm solves task T, which completes the proof of Theorem 7.4.

7.3 Application

In this section, we present an application of the generalized convergence algorithm on chromatic loop agreement.

Gafni and Koutsoupias [18] introduced the chromatic loop agreement, in which three processes solve the simplex agreement on a 2-dimensional chromatic simplicial complex $\mathcal{K}$ constrained by a look in $\mathcal{K}$. If one process p_i participates, then it decides on a predefined vertex in the loop. If p_i and p_j participate, they decide on a simplex in an edge path of the loop. If three processes participate, they converge on an arbitrary 2-dimensional simplex in $\mathcal{K}$. Herlihy and Rasjbaum [25] introduced the colorless loop agreement task, which extends the chromatic loop agreement for more than 3 processes. It is a well-known fact that the solvability of chromatic loop agreement and its colorless version is generally undecidable in the wait-free model [18, 25]. However,

Herlihy et al. [24] showed that in a model that solves 2-set-agreement, the solvability of colorless loop agreement is decidable.

We show that the chromatic loop agreement task is decidable in the vector-set-consensus model $\mathcal{VSC} = (1, 2, 2)$. Also, we show that the vector-set-consensus model $\mathcal{VSC} = (1, 1, 2)$ is universal for chromatic loop agreement, i.e., it solves any chromatic loop agreement.

Let $\mathcal{K}$ be a 2-dimensional chromatic simplicial complex and let v_0, v_1 be vertices in $\mathcal{K}$ such that $\chi(v_0) = p_0$ and $\chi(v_1) = p_1$. A simple chromatic edge path e_{01} is a sequence of distinct vertices $v_0 = u_0, u_1, ..., u_m = v_1$, where$\{u_i, u_{i+1}\}$ is a 1-simplex in $\mathcal{K}$ and $\chi(u_i) \in \{p_0, p_1\}$ for $0 \leq i \leq m$. A chromatic triangle loop is defined as a 6-tuple $\ell = (v_0, v_1, v_2, e_{01}, e_{12}, e_{20})$ where e_{ij} is a simple chromatic edge path between v_i and v_j.

Definition 7.16. Let Δ^2 be the standard chromatic 2-dimensional simplex, and let $\mathcal{O}$ be a path-connected[1] chromatic simplicial complex. The chromatic loop agreement task $T = (\Delta^2, \mathcal{O}, \Gamma, \ell)$ is a colored task for three processes, together with a predefined chromatic triangle loop ℓ. The carrier map Γ is defined by

$$\Gamma(\sigma) = \begin{cases} \{v_i\} & \sigma = \{p_i\}, i = 0, 1, 2 \\ e_{ij} & \sigma = \{p_i, p_j\}, 0 \leq i < j \leq 2 \\ \mathcal{O} & \sigma = \{p_0, p_1, p_2\} \end{cases}.$$

The following theorem show that the solvability of the chromatic loop agreement is decidable in some vector-set-consensus model.

[1]Here is the intuition of requiring $\mathcal{O}$ to be path-connected: when two processes are participating, they decide a simplex in the chromatic loop. If the third process participates after the first two processes decided, their outputs must form a simplex in the path-connected component that contains the chromatic loop.

Theorem 7.17. *Let $T = (\Delta^2, \mathcal{O}, \Gamma)$ be a chromatic loop agreement. Whether T is solvable in the vector-set-consensus model $\mathcal{VSC} = (1, 2, 2)$ is decidable .*

Proof. Assume $m + 1$ processes are participating, and let σ be corresponding the input simplex such that $\dim(\sigma) = m$. Since $\mathcal{O}$ is path-connected, we conclude that $\Gamma(\sigma)$ is $(k_{m+1} - 2)$-connected. To check whether T satisfies the condition in Theorem 7.4, it remains to check whether the $\mathrm{Lk}(v, \mathcal{O})$ is path-connected for any vertex $v \in \mathcal{O}$. Notice that this problem is equivalent to determine whether there is a path in $\mathrm{Lk}(v, \mathcal{O})$ connecting any two vertices, which is the st-connectivity problem in an undirected graph. □

The following theorem shows that the vector-set-consensus model $\mathcal{VSC} = (1, 1, 2)$ is universal for chromatic loop agreement.

Theorem 7.18. *Let $T = (\Delta^2, \mathcal{O}, \Gamma)$ be a chromatic loop agreement. T is solvable in the vector-set-consensus model $\mathcal{VSC} = (1, 1, 2)$.*

Proof. By definition, $\mathcal{O}$ is a path-connected chromatic simplicial complex, and therefore, $\mathrm{Lk}(v, \mathcal{O})$ is (-1)-connected. Hence, the condition in Theorem 7.4 is satisfied, and therefore, the generalized convergence algorithm solves T in $\mathcal{VSC} = (1, 1, 2)$. □

Chapter 8

Conclusion and Open Problems

In this dissertation, we generalized ACT for more general models. We present the colorless ACT for the vector-set-consensus model. We rephrased the ACT for colored task solvability in the k-set-consensus model using the notion of color-projection. We also proved the "wait-at-beginning" colorless ACT for the general adversary model, which shows that the full power of the adversary is only needed at the beginning of a colorless task protocol. For task solvability in the vector-set-consensus model, we show a sufficient topological condition in an algorithmic way.

Our generalizations of ACT reaffirms the phenomenon that many models are equivalent to "restricted" IIS models. Thus, we infer that the IIS model is analogous to Turing machine in a unified distributed computability theory: solving a task in any reasonable distributed system can be analyzed in a "restricted" IIS model.

Colored ACT for vector-set-consensus

In Chapter 5, we show that to characterize colored task solvability in the vector-set-consensus model, the protocol complex should enable processes to solve set-agreement

regardless of the participating set. We use the color-projection closed complex to characterize the k-set-consensus model. It is natural to try to apply the same method for vector-set-consensus protocol complex $\mathrm{Ch}_V(\Delta^n)$. However, it is not hard to see that this approach does not work for every vector-set-consensus model. For example, in the vector-set-consensus model $\mathcal{VSC} = (1, 1, 2)$, the color-projection-closed complex $\widehat{\mathrm{Ch}_V(\Delta^n)}$ does solve $(2, 1)$-consensus and $(3, 2)$-set-consensus regardless of the participating set. However, this approach fails in the model $\mathcal{VSC} = (1, 1, 1, 2)$. In particular, the color-projection closed complex $\widehat{\mathrm{Ch}_V(\Delta^3)}$ is not path-connected, which implies that $(4, 1)$-consensus can be solved, contradicting the set-consensus power of the model.

Thus, if there exists a color-projection closed complex with the same connectivity as $\mathrm{Ch}_V(\Delta^n)$, it characterizes the colored task solvability of vector-set-consensus model. However, our experience suggests that there is no such complex in $\mathrm{Ch}^2(\Delta^n)$. It is an intriguing problem whether such protocol complex exists in more iterations of the standard chromatic subdivision.

Problem 8.1. Given a vector-set-consensus model $\mathcal{VSC}$, is there a pure n-dimensional color-projection closed complex $\widehat{\mathcal{K}} \subseteq \mathrm{Ch}^N(\Delta^n)$ with the same connectivity as $\mathrm{Ch}_V(\Delta^n)$ for some $N \geq 2$?

Theorem 7.4 shows a topological "upper bound" for solving a task in the vector-set-consensus model. We ask whether there exists a protocol complex in $\mathrm{Ch}^N(\Delta^n)$ which connectivity is precisely the "upper bound".

Problem 8.2. Let $\mathcal{VSC} = (k_1, k_2, ..., k_{n+1})$ be a vector-set-consensus model. Is there a pure n-dimensional complex $\mathcal{K} \subseteq \mathrm{Ch}^N(\Delta^n)$ that satisfies exactly the topological condition in Theorem 7.4 ? In other words, is there a complex such that for $m =$

$0, 1, ..., n$, $\mathcal{K} \cap \mathrm{Ch}^N(\mathrm{skel}^m(\Delta^n))$ is $(k_{m+1}-2)$-connected but not $(k_{m+1}-1)$-connected, and for any simplex $\tau \in \mathcal{K}$ where $\dim(\tau) = c$, $\mathrm{Lk}(\tau, \mathcal{K} \cap \mathrm{Ch}^N(\mathrm{skel}^m(\Delta^n)))$ is $(k_{(m-c)}-2)$-connected but not $(k_{(m-c)}-1)$-connected?

ACT theorem for Objects

It is a well-known fact that a shared-memory object can be classified by the consensus number and set-consensus number [2, 23]. For example, the test-and-set object has the consensus number 2, and the test-and-set model can be characterized by a complex in $\mathrm{Ch}(\Delta^n)$ [31]. We believe that every meaningful deterministic object can be characterized by a complex in $\mathrm{Ch}^N(\Delta^n)$ for some N.

General Adversary Structure

Gafni and Kuznetsov [19] showed that, for superset-closed adversary, the set-consensus number is the size of the minimal hitting set of $\mathcal{A}$. However, it is not clear how to extend this observation to general adversary. We provide a combinatorial conjecture on the structure of a generalized adversary.

A *partially ordered set* or a *poset*, is a pair $(P, \leq)$ that consists of a set P together with a relation $\leq$ that satisfies the following properties:

1. for any $x \in P$, we have $x \leq x$.
2. for any $x, y \in P$, if $x \leq y$ and $y \leq x$, then $x = y$.
3. for any $x, y, z \in P$, if $x \leq y$ and $y \leq z$, then $x \leq z$.

For any $x, y \in P$, we say that y *covers* x, denoted by $x \prec y$, if there is no $z \in P$ such that $x \leq z \leq y$. For any $y \in P$, let $C(y)$ denote the set of elements that are covered

by x, i.e., $C(y) = \{x \in P \mid x \prec y\}$. An adversary $\mathcal{A}$ is a poset ordered by inclusion. We say an adversary $\mathcal{A}$ is *regular* if for every set $S \in \mathcal{A}$, $C(S)$ satisfies one of the two following properties.

1. (subset structure) For any set $S' \subseteq S_v$ such that $|S'| = |S| - 1$, $S' \in \mathcal{A}$
2. (binary structure) $C(S) = \{U_1, U_2\}$ such that for $i = 1, 2$, $|U_i| \geq |U|/2$, and U_1, U_2 is a partition of U.

The intuition behind the definition is that an adversary is regular if the live sets are related by mixing the subset and binary structures. We conjecture that any general adversary $\mathcal{A}$ contains a regular sub-adversary.

Problem 8.3. For any general adversary $\mathcal{A}$ such that $\text{setcon}(\mathcal{A}) = k$, does $\mathcal{A}$ contains a regular subadversary $\mathcal{A}'$ such that $\text{setcon}(\mathcal{A}') = k$?

Bibliography

[1] Yehuda Afek, Hagit Attiya, Danny Dolev, Eli Gafni, Michael Merritt, and Nir Shavit. Atomic snapshots of shared memory. *J. ACM*, 40(4):873–890, 1993.

[2] Yehuda Afek, Faith Ellen, and Eli Gafni. Deterministic objects: Life beyond consensus. In George Giakkoupis, editor, *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, pages 97–106. ACM, 2016.

[3] Elizabeth Borowsky. *Capturing the Power of Resiliency and Set Consensus in Distributed Systems*. PhD thesis, 1995.

[4] Elizabeth Borowsky and Eli Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 91–100. ACM, 1993.

[5] Elizabeth Borowsky and Eli Gafni. Immediate atomic snapshots and fast renaming (extended abstract). In Jim Anderson and Sam Toueg, editors, *Proceedings of the Twelth Annual ACM Symposium on Principles of Distributed Computing, Ithaca, New York, USA, August 15-18, 1993*, pages 41–51. ACM, 1993.

[6] Elizabeth Borowsky and Eli Gafni. A simple algorithmically reasoned characterization of wait-free computations (extended abstract). In James E. Burns and Hagit Attiya, editors, *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, Santa Barbara, California, USA, August 21-24, 1997*, pages 189–198. ACM, 1997.

[7] Elizabeth Borowsky, Eli Gafni, Nancy A. Lynch, and Sergio Rajsbaum. The BG distributed simulation algorithm. *Distributed Comput.*, 14(3):127–146, 2001.

[8] Zohir Bouzid, Eli Gafni, and Petr Kuznetsov. Strong equivalence relations for iterated models. In Marcos K. Aguilera, Leonardo Querzoni, and Marc Shapiro, editors, *Principles of Distributed Systems - 18th International Conference, OPODIS 2014, Cortina d'Ampezzo, Italy, December 16-19, 2014. Proceedings*, volume 8878 of *Lecture Notes in Computer Science*, pages 139–154. Springer, 2014.

[9] Soma Chaudhuri. Agreement is harder than consensus: Set consensus problems in totally asynchronous systems. In Cynthia Dwork, editor, *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing, Quebec City, Quebec, Canada, August 22-24, 1990*, pages 311–324. ACM, 1990.

[10] Soma Chaudhuri and Paul Reiners. Understanding the set consensus partial order using the borowsky-gafni simulation (extended abstract). In Özalp Babaoglu and Keith Marzullo, editors, *Distributed Algorithms, 10th International Workshop, WDAG '96, Bologna, Italy, October 9-11, 1996, Proceedings*, volume 1151 of *Lecture Notes in Computer Science*, pages 362–379. Springer, 1996.

[11] Carole Delporte-Gallet, Hugues Fauconnier, Eli Gafni, and Petr Kuznetsov. Set-consensus collections are decidable. In Panagiota Fatourou, Ernesto Jiménez,

and Fernando Pedone, editors, *20th International Conference on Principles of Distributed Systems, OPODIS 2016, December 13-16, 2016, Madrid, Spain*, volume 70 of *LIPIcs*, pages 7:1–7:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[12] Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, and Andreas Tielmann. The disagreement power of an adversary. *Distributed Comput.*, 24(3-4):137–147, 2011.

[13] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

[14] Eli Gafni. The extended bg-simulation and the characterization of t-resiliency. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 85–92. ACM, 2009.

[15] Eli Gafni and Rachid Guerraoui. Simulating few by many: Limited concurrency = set consensus. `http://web.cs.ucla.edu/~eli/eli/kconc.pdf`.

[16] Eli Gafni and Rachid Guerraoui. Generalized universality. In Joost-Pieter Katoen and Barbara König, editors, *CONCUR 2011 - Concurrency Theory - 22nd International Conference, CONCUR 2011, Aachen, Germany, September 6-9, 2011. Proceedings*, volume 6901 of *Lecture Notes in Computer Science*, pages 17–27. Springer, 2011.

[17] Eli Gafni, Yuan He, Petr Kuznetsov, and Thibault Rieutord. Read-write memory and k-set consensus as an affine task. In Panagiota Fatourou, Ernesto Jiménez, and Fernando Pedone, editors, *20th International Conference on Principles of*

Distributed Systems, OPODIS 2016, December 13-16, 2016, Madrid, Spain, volume 70 of *LIPIcs*, pages 6:1–6:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[18] Eli Gafni and Elias Koutsoupias. Three-processor tasks are undecidable. *SIAM J. Comput.*, 28(3):970–983, 1999.

[19] Eli Gafni and Petr Kuznetsov. Turning adversaries into friends: Simplified, made constructive, and extended. In Chenyang Lu, Toshimitsu Masuzawa, and Mohamed Mosbah, editors, *Principles of Distributed Systems - 14th International Conference, OPODIS 2010, Tozeur, Tunisia, December 14-17, 2010. Proceedings*, volume 6490 of *Lecture Notes in Computer Science*, pages 380–394. Springer, 2010.

[20] Eli Gafni, Petr Kuznetsov, and Ciprian Manolescu. A generalized asynchronous computability theorem. In Magnús M. Halldórsson and Shlomi Dolev, editors, *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 222–231. ACM, 2014.

[21] Eli Gafni and Sergio Rajsbaum. Distributed programming with tasks. In Chenyang Lu, Toshimitsu Masuzawa, and Mohamed Mosbah, editors, *Principles of Distributed Systems - 14th International Conference, OPODIS 2010, Tozeur, Tunisia, December 14-17, 2010. Proceedings*, volume 6490 of *Lecture Notes in Computer Science*, pages 205–218. Springer, 2010.

[22] Allen Hatcher. *Algebraic topology*. Cambridge Univ. Press, Cambridge, 2000.

[23] Maurice Herlihy. Wait-free synchronization. *ACM Trans. Program. Lang. Syst.*, 13(1):124–149, 1991.

[24] Maurice Herlihy, Dmitry N. Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, 2013.

[25] Maurice Herlihy and Sergio Rajsbaum. The decidability of distributed decision tasks (extended abstract). In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 589–598. ACM, 1997.

[26] Maurice Herlihy and Nir Shavit. The asynchronous computability theorem for t-resilient tasks. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 111–120. ACM, 1993.

[27] Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *J. ACM*, 46(6):858–923, 1999.

[28] Dmitry N Kozlov. Chromatic subdivision of a simplicial complex. *Homology, Homotopy and Applications*, 14(2):197–209, 2012.

[29] Dmitry N Kozlov. Topology of the view complex. *Homology, Homotopy & Applications*, 17(1), 2015.

[30] Petr Kuznetsov and Thibault Rieutord. Agreement functions for distributed computing models. In Amr El Abbadi and Benoît Garbinato, editors, *Networked Systems - 5th International Conference, NETYS 2017, Marrakech, Morocco, May 17-19, 2017, Proceedings*, volume 10299 of *Lecture Notes in Computer Science*, pages 175–190, 2017.

[31] Petr Kuznetsov and Thibault Rieutord. Affine tasks for k-test-and-set. In Stéphane Devismes and Neeraj Mittal, editors, *Stabilization, Safety, and Secu-*

rity of Distributed Systems - 22nd International Symposium, SSS 2020, Austin, TX, USA, November 18-21, 2020, Proceedings, volume 12514 of *Lecture Notes in Computer Science*, pages 151–166. Springer, 2020.

[32] Petr Kuznetsov, Thibault Rieutord, and Yuan He. An asynchronous computability theorem for fair adversaries. In Calvin Newport and Idit Keidar, editors, *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 387–396. ACM, 2018.

[33] James R Munkres. *Topology*. Prentice Hall, 2000.

[34] James R Munkres. *Elements of algebraic topology*. CRC press, 2018.

[35] Paul Reiners. Understanding the set consensus partial order using the borowsky-gafni simulation, 1996.

[36] Thibault Rieutord. *Combinatorial Characterization of Asynchronous Distributed Computability*. PhD thesis, 2018.

[37] Michael E. Saks and Fotios Zaharoglou. Wait-free k-set agreement is impossible: the topology of public knowledge. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 101–110. ACM, 1993.

[38] Vikram Saraph, Maurice Herlihy, and Eli Gafni. Asynchronous computability theorems for t-resilient systems. In Cyril Gavoille and David Ilcinkas, editors, *Distributed Computing - 30th International Symposium, DISC 2016, Paris,*

France, September 27-29, 2016. Proceedings, volume 9888 of *Lecture Notes in Computer Science*, pages 428–441. Springer, 2016.

[39] Vikram Saraph, Maurice Herlihy, and Eli Gafni. An algorithmic approach to the asynchronous computability theorem. *J. Appl. Comput. Topol.*, 1(3-4):451–474, 2018.

[40] Edwin H Spanier. *Algebraic topology*. Springer Science & Business Media, 1989.