A Simple Algorithmically Reasoned Characterization of Wait-free Computations *

(Extended Abstract)

Elizabeth Borowsky (borowsky@hpl.hp.com)

Eli Gafni (eli@cs.ucla.edu)

Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90024
U.S.A.

Abstract

This paper introduces two new novel tools for the study of distributed computing and shows their utility by using them to exhibit a simple derivation of the Herlihy and Shavit characterization of wait-free shared-memory computation. The first tool is the notion of the *iterated* version of a given model. We show that the topological structure that corresponds to an iterated model has a nice recursive structure, and that the iterated version of the atomic snapshot memory solves any task solvable by the non-iterated model. The second tool is an iterated explicit simple convergence algorithm. In the Ph.D. Thesis of the first author these tool were used to characterize models more complex than read-write shared-memory.

1 Introduction

We consider the standard Single Writer Multi-Reader (SWMR) read-write shared-memory model where read is done via atomic snapshots. This model is considered w.l.o.g. since all standard variations of the shared-memory model are equivalent to it [1]. As early as 1983, Fisher, Paterson, and Lynch [2] showed that in this model there are problems that are not solvable if even one processor may fail-stop, let alone if any number may fail-stop.

In 1988 Biran, Moran, and Zaks [3] introduced the notion of *tasks* to generalize and formalize the problem instances in [2]. They specified the necessary and sufficient conditions that render tasks solvable in spite of a single failure. They also gave an algorithm that when given a task, checks for the conditions and returns the answer to whether the task is solvable in the presence of a single failure.

Just as the problems in [2] helped in characterizing a single failure, the identification of a problem that might be solvable in spite of k failures but not in spite of $k+1$ failures was perceived as instrumental for any advancement in characterizing other models. Such a problem, k-set consensus, was proposed by Chaudhuri [4] in 1990. In 1993, three teams [5, 6, 7] independently proved her conjecture correct. The team of Herlihy and Shavit went further and, in a conceptual breakthrough, gave necessary conditions for wait-free solvability. In a subsequent paper, building on the immediate snapshot implementation of [8], they proved these conditions sufficient.

They did not, however, address the question of whether the conditions are effective. Recently, it was shown, based on their conditions, that answering the solvability question for three or more

*Work supported by UCLA Academic Senate Grant .

processors is undecidable [9]. Thus, since we cannot say that one complete characterization is more complex than another, the way to compare characterizations is qualitative, that is, by the ease with which they allow various interesting task *instances* to be proven solvable or unsolvable. Two important such instances are set consensus and renaming.

Set consensus was proven impossible by the three teams, and the arguments in [7] are elementary. Proving renaming impossible was done in [6] using machinery from a branch of topology called *homology*. Moreover, in [6] conditions necessary for wait-free solvability were derived by homology too.

An interesting question is whether homology is indispensable? It may be indispensable for proving renaming impossible, or, more precisely tasks may exist that we cannot conceive of proving impossible without homology (since this might mean proving facts about homology without homological machinery). Yet, it would be disappointing if what is currently perceived to be the main central result in the area of distributed algorithms, namely, characterizing wait-free computations, could not be derived from elementary, mostly algorithmic arguments.

This paper rederives the characterization for wait-free solvability by mainly algorithmic arguments. Furthermore, it introduces the reader to the tools and techniques that enabled us to characterize additional models in [10] and in a forthcoming paper [11].

Our primary tool is a new model of computation, called *iterated immediate snapshots*, a variation of the immediate snapshot model [8]. Implicitly, this model was introduced and characterized in [12]. Here, however, we consider and investigate it as a *model* of computation. The characterization of problems solvable in this model is surprisingly simple. The full information protocol in this model has the structure of a recursive standard chromatic subdivision [12]. Since a decision function is a simplicial map from a local state to an output value, the implication is that the necessary and sufficient condition for a task to be solvable in the model is the existence of a simplicial map from the input that is subdivided by some level of standard chromatic subdivision to the output.

Once we have a characterization of the model on our hands, it is natural to ask how its wait-free solvability power relates to the power of the standard shared-memory model. It is easy to see that any task wait-free solvable in the iterated immediate snapshots model is solvable in the shared memory model [8]. The main result of this paper is to show the *opposite* direction: that any task solvable wait-free in the atomic snapshot model is wait-free solvable in the iterated immediate snapshot model. Consequently, the characterization of wait-free solvability in the atomic snapshot model will be the same as in iterated immediate snapshots. We do this by implementing a simple emulation scheme which allows wait-free atomic snapshot protocols to run in the iterated immediate snapshot model.

The characterization in [6, 12] goes further, however; it replaces the standard chromatic subdivision by any chromatic subdivision. This is possible due to the geometrical fact that for any chromatic subdivided simplex there exists a high-enough level recursive standard chromatic subdivision that maps simplicially to it. This fact is proven in [12] using complex geometrical arguments. Here, we show that these arguments can be replaced by a corollary to a standard elementary result from topology, the *simplicial approximation theorem*, and our simplex convergence algorithm from [10]. In this instance, then, a topological theorem is proven via a distributed algorithm.

The emulation algorithm we present in this paper is a logical extension of our previous work. We derived it by observing the working of our simplex convergence algorithm on the history complex of a shared-memory protocol, and taking advantage of the fact that each "next step" locally contains a chromatic subdivided simplex. This indicates an alternate route we might have taken: after showing that our convergence algorithm converges on the history complex of a shared-memory protocol, show that the convergence algorithm can be implemented in the iterated immediate snapshot model. This route is taken in a follow-up paper that characterizes resiliency and set-consensus models [11].

Recently, Attiya and Rajsbaum [13] and Mavronicolas [14] proposed *combinatorial* framework for wait-free computability. Some impossibility results may rely only on subset of the properties that hold for wait-free computations. In fact, the impossibility proofs in [5, 7] rely only on the fact that wait-free computations produce a manifold, rather than on the fact that they produce a manifold that is also a subdivided simplex. Such a framework may simplify proofs of the impossibility of instance tasks, as it did in [5, 7], but it does not capture the true nature of wait-free computations. Solvability is essentially ϵ-agreement [3], and ϵ-agreement has no meaning without an underlying topological space. The combinatorial framework precludes the notion of a subdivided simplex, a notion that requires an embedding in a Euclidean space. Thus, it seems that any characterization in this framework is bound to be *incomplete*.

This paper is organized as follows. After a short section on elementary topological notions, we introduce a formal definition of the models and discuss their topological structure. We then show a direct emulation of the atomic snapshot memory wait-free protocol in the iterated immediate snapshot model. Finally, we introduce the simplex convergence algorithm to prove the geometrical fact of the *universality* of the iterated standard chromatic subdivision. This involves replacing the direct geometrical arguments in [12] by the simplicial approximation theorem and our convergence algorithm from [10].

2 Topological Preliminaries

An n-dimensional simplex is a set of $n+1$ vertices. A simplicial complex is a set of simplices closed under intersection and containment. A complex C^n is pure of dimension n if every simplex is a subset of an n-dimensional simplex, and if the n-dimensional simplices are only subset of themselves.

A complex C has an embedding in a Euclidean space of high-enough dimension such that vertices in a simplex are affinely independent. All our complexes will be embedded.

A complex $B(A)$ is a subdivision of a complex A if there is an embedding of both in the same space such that: 1. The convex hull of any simplex of $B(A)$ is contained in the convex hull of a simplex of A and, 2. The union of the convex hull of simplices in B is the same as A. If s is a simplex of $B(A)$, then $carrier(s, A)$ is the unique smallest simplex of A whose convex hull contains the convex hull of s. For a complex $C \subseteq B(A)$, $carrier(C, A) = \cup_{s \in C}\ carrier(s, A)$.

A map of vertices of complex A to vertices of complex B is simplicial if any simplex in A maps to a simplex in B. The map is dimension preserving if the image of any simplex has the same dimension as that of the source. A coloring of a complex C^n is a dimension-preserving simplicial map $\mathcal{X}$ from C^n to some simplex s^n of dimension n. A complex together with coloring is a chromatic complex. A simplicial map μ from a chromatic complex A to B is *color preserving* if for all $v \in A$, $\mathcal{X}(v) = \mathcal{X}(\mu(v))$. A complex A is a subcomplex of B if every simplex in A is a simplex of B. If B colored, then A inherits the colors from B. For $C \subseteq A$, where A is chromatic, $\mathcal{X}(C) = \cup_{v \in C}\mathcal{X}(v)$. If $A(C)$ and $B(C)$ are subdivisions of C then a simplicial map μ from $A(C)$ to $B(C)$ is carrier preserving, if for all $v \in A(C)$, $carrier(v, C) = carrier(\mu(v), C)$. the source simplex.

A subdivided simplex $A(s^n)$ is a complex that is a subdivision of some simplex s^n. A face of A corresponding to $s^q \subseteq s^n$, denoted by $A(s^q)$, is the subcomplex of the simplices of A whose carrier is a subset of s^q. The complex of the $n-1$ dimensional faces of a subdivided simplex $A(s^n)$, called $boundary(A(s^n))$, is an $(n-1)$-dimension *sphere*. Let μ be a simplicial map from $boundary(A(s^n))$ to C. We say that $\mu(boundary(A(s^n)))$ has a fill-in (span), if there exists $A'(s^n)$ and a simplicial map μ' from $A'(s^n)$ to C, such that $boundary(A'(s^n)) = boundary(A(s^n))$ and μ' agrees with μ on the boundary. The simplicial image in C of $(n-1)$-sphere has a fill-in (span) if there exist a simplicial map from the subdivided simplex to C that extends the map of the sphere.

A complex C has no hole of dimension k if every image of a $(k-1)$-sphere in C has a fill-in. For $s^q \in C$, $star(s^q, C)$ is union of the convex hull of all the simplices in C that contain s^q. $Link(s^q, C)$ is the union of the convex hull of all the simplices in $star(s^q, C)$ that do not contain any vertex of s^q. The first barycentric subdivision of an embedding of s^n, denoted $Bsd(s^n)$, is obtained recursively, by planting a vertex in the center of s^q and considering it a simplex when added to any simplex of the recursively barycentrically subdivided $n-1$ faces of s^n, where the subdivision of the zero-dimension simplex is itself. The $Bsd(C^n)$ of a complex C^n is the subdivision of C^n obtained by replacing each $s^q \in C^n$ with $Bsd(s^q)$. $Bsd^k(C) = Bsd^{k-1}(Bsd(C))$, where $Bsd^0(C) = C$.

In this paper we use two results from topology

[15]. One is a corollary to the simplicial approximation theorem:

Lemma 2.1 *Given a subdivided simplex $A(s^n)$ then for all k that are large enough there exists a carrier-preserving simplicial map from $Bsd^k(s^n)$ to $A(s^n)$.*

The other says that a subdivided simplex is a "nice" structure.

Lemma 2.2 *A subdivided simplex $A(s^n)$ has no hole of any dimension, and the $link(s^q, A(s^n))$ for any $s^q \in A(s^n)$ of dimension q has no hole of dimension less than or equal to $n-(q+1)$.*

3 Models

3.1 SWMR Atomic Snapshot Memory Model and the Full-information Protocol

In the SWMR atomic snapshot memory model [1], each processor P_i, $i = 0, 1, ..., n$, has a cell C_i that is exclusively written by it and is read by all processors. Each processor alternates between writing its cell and reading all cells in a snapshot.

An execution of the full-information protocol in the model is an infinite sequence of processor IDs where the first appearance of a processor is interpreted as write, the second as snapshot, and so on. In its first write, a processor P_i writes its input to C_i. In subsequent writes, it writes the precise encoding of the whole memory it last read in a snapshot. This last snapshot and whether P_i has written this last snapshot back to C_i is P_i's local state.

Given the possible input combinations that processors may hold, and given an integer k, we may define the complex of the full-information protocol corresponding to an execution prefix in which all processors have each gone through k rounds of writes and snapshots before any processor writes for the $k+1$ time. The vertices of the complex are all the possible pairs $(P_i, view_i)$, where P_i is a processor ID and $view_i$ is the local state of P_i after k rounds in any of the prefixes above, for all input combinations. A set of vertices form a simplex when their associated local states are consistent with a single execution prefix.

We assume that processors IDs are identified with the vertices of a simplex s^n and thus, since a processor has a unique view in each execution prefix, the complex is chromatic.

3.2 Tasks

A task is an input-output relation. For each subset of processors, each with its associated input, constituting an input tuple, a task specifies all the output tuples that may result when each processor has decided on an output value.

For instance, in the $(n+1, k)$ set-consensus task [4], each processor has its ID as an input and, for each set of processors, each processor decides on an ID from the set, such that the number of distinct IDs decided on does not exceed k.

Tasks over $n+1$ processors can be represented by two chromatic simplicial complexes [6], I^n and O^n, and a map Δ. I^n is a complex whose vertices are all pairs (P_i, val_i) where P_i may hold the input value val_i. A set of vertices is a simplex if it corresponds to an input tuple. O^n is a complex whose vertices are all pairs $(P_i, DecisionValue_i)$ that appear in any output tuple. A set of vertices is a simplex if there is an output tuple containing them. The map Δ is a point-to-set mapping from I^n to O^n such that if $s_i \in I^n$, $s_o \in O^n, s_o \in \Delta(s_i)$, then $\mathcal{X}(s_i)=\mathcal{X}(s_o)$. The map Δ captures the possible output tuples that may result from a possible input tuple associated with a set of processors.

3.3 Wait-free Solvability

Given an execution of the full-information protocol, the set of the processors which appear at least once is called the participating set. A partial map from local states of processors in the full information protocol to values is a decision function if in any execution after a local state of a processor is mapped to a value, all subsequent local states of the processor are mapped and to the same value. A processor's appearance in an infinite execution is (or has) decided if the corresponding local state in the execution is mapped to a value. A processor is decided in an execution if it has an appearance in the execution which is decided. A task is wait-free solvable in the atomic snapshot model if, for any infinite execution of the full-information protocol, with participating set P and input that correspond to an input simplex $s_i \in I^n$, all processors that appear infinitely often are decided, and the output tuple $s`_o \in O^n$ that results in from the decisions in the execution can be extended to an output simplex $s_o \in O^n$, $s`_o \subseteq s_o$ and $s_o \in \Delta(s_i)$.

Let T be a task that is wait-free solvable and that has a finite number of inputs tuples. A task is bounded wait-free solvable if there exists a bound b such that in any execution, any processor that appears b times in the sequence has decided. Does the fact that T is wait-free solvable imply that it is bounded wait-free solvable?

Lemma 3.1 *If T is wait-free solvable, then it is bounded wait-free solvable.*

Proof Let DT be a decision function that solves T. Consider the tree of all executions where a processor does not appear in the sequence after the processor has decided. This tree has finite branching. Consequently, by Konig's lemma, if the tree is unbounded, then there is an infinite path, contradicting the wait free solvability of T. Thus, the tree must be bounded by some bound b. Consider now a new decision function, NDT, which is DT only that a processor discards from its local state the steps taken by processors after they are decided. $\Box$

3.4 Immediate Snapshot Model

The immediate snapshot model introduced in [7, 5] is a restriction of the atomic snapshot model (and thus potentially more powerful) in the sense that its set of executions is a subset of the atomic snapshot model. It comprises executions in the atomic snapshot model in which each maximal run of writes is followed by a maximal run of snapshots by the same processors. Consequently, we can condense the odd-followed-by-even appearance of a processor as a single operation called WriteRead($value$). Thus, an execution is a sequence of sets of processors. In [8] it was proven that immediate snapshots model can be simulated by the atomic snapshot model and therefore cannot solve anything not solvable in the atomic snapshot model.

3.5 Iterated Immediate Snapshot Model

A one-shot immediate snapshot is an immediate snapshot model that allows each processor to WriteRead only once. In the iterated immediate snapshot model, we have a sequence of one-shot immediate snapshot memories, $M_0, M_1, \ldots$. The full-information protocol execution starts by P_i WriteReading its input to M_0. It then applies the output from M_i, $i > 0$, as an input value to M_{i+1}, ad infinitum.

Formally, one-shot immediate snapshot is specified as follows. For any input val_i to P_i (where w.l.o.g. $\forall i, j,\ val_i \neq val_j$ by adding processor's Id to the input), if the set of participating processors is P, then P_i's output is a set S_i of inputs to processors, such that:

1. $val_i \in S_i, \forall P_i \in P$
2. $S_i \subseteq S_j$ or $S_j \subseteq S_i, \forall P_i, P_j \in P$
3. $val_i \in S_j \Rightarrow S_i \subseteq S_j,\ \forall P_i, P_j \in P$

A full-information execution in the model is an infinite sequence, each element of which is an ordered partition of the set of processors P_i, $i = 0, \ldots, n$. Inductively, if the local state of processor P_i after its appearance in element j is v_{ij}, then its local state after its $j+1$ appearance is the result of inputing (P_i, v_{ij}) to a one-shot immediate snapshot with S_i including all the tuples of processors in the $j+1$ partition, which appear in the order in sets that precede or include P_i.

In the full paper we generalize the iterated notion to many other models.

A task is solvable wait-free in the iterated immediate snapshot model if for j_i large enough, the output of P_i from M_{j_i} can be mapped to a decision-value satisfying the task. It is easy to see that wait-free solvability here also implies bounded wait-free solvability.

Obviously, the immediate snapshot model can simulate the iterated one, and consequently the atomic snapshot model can also simulate it. The main result of this paper is that the *converse* is true. The iterated immediate snapshot model can simulate any b-shot execution of the atomic snapshot model. Since, by lemma 3.1 if a task is solvable in the atomic snapshot model, it is solvable in a bounded number of steps, this implies that the task is solvable in the iterated immediate snapshot model.

3.6 Characterization of the b-shot Iterated Immediate Full-Information Protocol

Let $V = \{(P_i, S_i) | i = 0, ..., n, S_i \subseteq \{P_0, ..., P_n\}, P_i \in S_i\}$ be all the possible tuples where S_i is an output of P_i in the immediate snapshot model when each processor inputs its ID. Consider the complex, called the one-shot immediate snapshot complex, whose vertices are the elements of V, and a set of vertices is a simplex if the S_i's satisfy the immediate snapshot relation with respect to some participating set P.

Lemma 3.2 *The one-shot immediate snapshot complex over $n+1$ processors with each processor inputing its ID, is a chromatic subdivided simplex, $SDS(s^n)$, of dimension n, with $\mathcal{X}(P_i, S_i) = P_i$. Moreover, for $v = (P_i, S_i)$, and $S_i = P$, we have $carrier(v, SDS(s^n)) = P$.*

This subdivided simplex is called by Herlihy and Shavit [12] a standard chromatic subdivided simplex of dimension n.

For completeness, we give a construction of its embedding here. Let s^n be a colored simplex of dimension n. Let a be the barycenter of s^n, and let b_i be the barycenter of the $(n-1)$-dimensional face across (opposite) from the vertex of s^n of color i. For each i, we plant a vertex m_i in the middle of the (a, b_i) interval and let its color be i. Let s be any chromatic simplex defined recursively on the faces of s^n of dimension less than n. If $carrier(s, s^n) = s^q \subseteq s^n$ then s together with any combination of m_i, $\mathcal{X}(m_i) \notin \mathcal{X}(s^q)$, is a simplex. It is simple to see that this structure has a subdivided simplex embedding [16].

The b-shot counterpart is complex in which we take as vertices all the possible outputs of the full-information protocol after b stages of one-shot immediate snapshot, with P_i's input being its own ID.

Let C be a pure complex of dimension n, and let $SDS(C)$ be the complex that one gets by subdividing each n-dimensional simplex of C by the Standard Chromatic Subdivision.

Lemma 3.3 *The b-shot immediate snapshot complex over $n+1$ processors with each processor inputing its ID, is a chromatic subdivided $SDS^b(s^n) = SDS(SDS^{b-1}(s^n))$, $SDS^0(s^n) = s^n$.*

What is the b-shot immediate snapshot complex for a general input complex I^n? It is easy to see that since each simplex $s^q \in I^n$ gives rise to $SDS^b(s^q)$, and when two simplexes share a face, the local states corresponding to the face involves only inputs associated with the face, we deduce that the b-shot full-information protocol complex is $SDS^b(I^n)$. Since by lemma 3.1 a finite input task is wait-free solvable only if it is bounded wait-free solvable we obtain the following proposition.

Proposition 3.1 *Bounded input task T is wait-free solvable in the iterated immediate snapshot model iff for all integer b large enough there exits a color-preserving simplicial map $\mu_b : SDS^b(I^n) \to O^n$, such that $\forall s \in SDS^b(I^n)$, $\mu_b(s) \in \Delta(carrier(s, SDS^b(I^n)))$.*

This elegant way of putting down the conditions is due to Herlihy and Shavit.

4 Emulation of Atomic Snapshot Memory by Iterated Immediate Snapshot Memory

W.l.o.g we may assume that a protocol Π in SWMR atomic snapshot memory proceeds by each processor P_i executing a $WriteRead_i(v)$ operation in which it writes v to C_i and then returns a snapshot all in a single operation.

To show that a given model M can emulate an atomic-snapshot full-information protocol we may assume w.l.o.g. that processor the value v that P_i writes in its jth $WriteRead_i(v)$ operation is just j. Thus an operation returns an n-vector of integers (including 0). M emulates correctly if we can argue that the vectors returned satisfy:

1. In the vector returned by the operation $WriteRead_i(k)$, $v_i = k$.
2. Any two vectors returned are either the same or one majorizes the other.
3. No value j for P_i is returned in an execution in which P_i has not $WriteRead$ j times.

To emulate $WriteRead_i(j)$, we assume inductively that processor P_i just finished emulating $WriteRead_i(j-1)$ and returned a vector of integers $(v_1, ..., v_n)$ between 0 and k. The initial vector for P_i is 0 everywhere.

Processor P_i increment v_i to j and submits this vector to the next stage of the immediate snapshot. We call this vector its "estimate" of the values of cells. In the immediate snapshot it obtains a set of estimate vectors. Its update its estimate by taking the maximum entry-wise on all these vectors.

It repeats submitting its estimate vector to the next stage of the iterated immediate snapshot until all estimate vectors returned to it by the immediate snapshot are the same. It then finished the operation $WriteRead_i(j)$. It returns the estimate vector of the stage as the output for the operation and proceeds to emulate $WriteRead_i(j+1)$.

The correctness of the algorithm hinges on two simple facts: 1. Only a single estimate vector may be adopted as an output at a stage, and all processors returning that vector have the correct required value for their own entry. 2. By The virtue of estimates being entry-wise maximum, if a vector is returned as output in one stage, all estimate vectors input to the next stage majorize it.

The liveness of the algorithm follows since if in a stage no processor outputs a value then there must be a processor that strictly increased its estimate. Since processors do not increase there own value without obtaining an output, the number of possible estimate increase is bounded.

Notice that this emulation is not as strong as the emulation of snapshot memory in [1]. When we emulate a nonterminating protocol with no decision involved, we cannot bound the number of steps taken by a snapshot procedure of a specific emulator. The emulation is just nonblocking, which is equivalent to wait-free Solvability for wait-free tasks. This is analogous to a snapshot procedure in [1] with no snapshot "embedding," i.e. "double collect" until one double collect succeeds.

Once we have the emulation, we have established that the conditions for wait-free solvability of the atomic snapshot shared memory model are identical to those of the iterated immediate snapshot model.

For most practical purposes, we could stop here. No conditions we will give are effective. Yet, if one set of conditions is by definition more relaxed than the other, then the latter is obviously superior. One obtains such conditions by replacing the iterated standard chromatic subdivision by any chromatic subdivision. This follows from the fact that a lemma analogous to Lemma 2.1 holds for the standard chromatic subdivision. This lemma is proven in the next section.

5 Approximating a Chromatic Subdivided Simplex by the Iterated Standard Chromatic Subdivision

In this section we prove a theorem in geometry. This theorem was proven in [12] using geometry. The simplex convergence algorithm in [10] is an algorithmically alternative proof that relies on repeated solution to the ϵ-agreement problem. Here we present this instrumental convergence algorithm replacing ϵ-agreement by direct reference to the Simplicial Approximation Theorem. That this replacement in [10] is possible was independently observed by Herlihy and Rajsbaun (private communication 1995).

Theorem 5.1 *Let A be a chromatic subdivision of the simplex s^n. Then for all k large enough there exists a color and carrier preserving simplicial map μ_k from $SDS^k(s^n)$ to A.*

Corollary 5.2 $T = (I^n, O^n, \Delta)$ *is wait-free solvable iff there a chromatic subdivision $\sigma(I^n)$, and a color preserving simplicial map $\mu : \sigma(I^n) \to O^n$, such that $\forall s \in \sigma(I^n)$, $\mu(s) \in \Delta(carrier(s, \sigma(I^n)))$.*

Proof $\Rightarrow$ Let $\sigma(I^n)$ be $SDS^b(I^n)$ in 3.1
$\Leftarrow$ By 5.1 for large enough b there is a color carrier preserving simplicial map μ_1 from $SDS^b(I^n)$ to $\sigma(I^n)$. Compose μ and μ_1 to obtain the result. □

5.1 Proof of Theorem 5.1

Proof We prove Theorem 5.1 by presenting a wait-free algorithm that solves the chromatic simplex agreement problem [12], below. Since we proved that a wait-free algorithm is a chromatic simplicial map from the Iterated Standard Chromatic Subdivision to the output, and the output in this case will be the chromatic subdivided simplex A, we have proved Theorem 5.1.

CSASS Task Chromatic simplex convergence over a subdivided simplex:

In this input-less task each processor P_i, $i = 0, \ldots, n$ is associated with a distinct corner v_i of a chromatic subdivided simplex A. Let P be the participating set. Each processor P_i outputs a vertex w_i such that $\mathcal{X}(w_i) = P_i$, $W = \{w_i | P_i \in P\}$ is a simplex in A, and $carrier(W, A) \subseteq carrier(\{v_i | P_i \in P\}, A)$.

The algorithm relies on the following topological fact.

The algorithm to solve the colored simplex convergence uses the following fact.

Lemma 5.3 *Let A be a subdivision of s^n, then for all k large enough there exists a carrier preserving simplicial map from $SDS^k(s^n)$ to A.*

Proof By the simplicial approximation theorem for all k_1 large enough there exists a carrier preserving simplicial map from $Bsd^k(s^n)$ to A. There exists the obvious carrier preserving simplicial map μ from $SDS(s^n)$ to $Bsd(s^n)$. Consequently, there exist a carrier preserving simplicial map from $SDS^k(s^n)$ to $Bsd^k(S^n)$ for all k. Since composition of carrier preserving simplicial map is a carrier preserving simplicial map, the lemma follows. □

Corollary 5.4 *The non-chromatic simplex agreement over a subdivided simplex below, is wait-free solvable.*

NCSASS Task: The non-chromatic simplex agreement over a subdivided simplex:

In this input-less task each processor P_i, $i = 0, \ldots, n$ is associated with a distinct corner v_i of a subdivided simplex A. Let P be the participating set. Each processor P_i outputs a vertex w_i such that $W = \{w_i | P_i \in P\}$ is a simplex in A and $carrier(W, A) \subseteq carrier(\{v_i | P_i \in P\}, A)$.

We now want to show that we can use the non-chromatic simplex agreement over a subdivided simplex to solve the following problem.

NCSAC Task: The non-chromatic simplex agreement over a complex with no holes:

In this task each processor P_i, $i = 0, \ldots, n$ holds as input any vertex v_i of a finite complex C that has no holes of dimension less than $n+1$. Let P be the participating set. Each processor P_i outputs a vertex w_i such that $W = \{w_i | P_i \in P\}$ is a simplex in C and if $P = \{P_i\}$ then $w_i = v_i$.

To solve this task we reduce it to the subdivided simplex version. Assume inductively that we solved the problem when the cardinality of the participating set is less than $n+1$ processors. This means that for a given combination c of n processors with inputs we have a simplicial map μ_c from $SDS^{k_c}(s^{n-1})$ to C that satisfies the conditions of the task. Since C is finite we can w.l.o.g. define $K_n = max_c k_c$ and for each c have a simplicial map $\bar{\mu}_c$ from $SDS^{K_n}(s^{n-1})$ to C for all c.

To solve the problem when the participating set is of size $n+1$ we notice that the $n+1$ combinations $c_0, \ldots, c_n$ of n inputs, under the maps $\bar{\mu}_{c_i}$ define an $n-1$ image of a sphere S. Since C has no hole of dimension n the maps can be extended to the interior of S. We choose such an extension to get a simplicial map μ from $SDS^d(s^n)$ to c for some $d \geq K_n$. Taking K_n to be the maximum over the different d's for the different full input combinations we get the desired algorithm.

Unfolding the recursion we see that it involves a large implicit table each processor has to have by which each two vertices in C determine a (not necessarily simple) path. Each three vertices with the associated 3 paths connecting each pair determine an image in C of a 2-dimensional subdivided simplex whose boundary maps in a carrier preserving manner to the paths and the vertices, etc.

We may consider a variation on the problem, as we will do below, in which the input to a processor in addition to a vertex of C contains additional information that constrains where the convergence vertex may live. As long as the total number of inputs is finite, and as long as the restriction appropriately comply with the no-holes conditions the algorithm will go through.

We are now ready for the algorithm that will prove theorem 5.1.

The algorithm proceeds in stages. In each stage there is an implicit chromatic complex C on the barycenters of which we solve the non-chromatic simplex agreement over a complex with no holes. A vertex of the barycenter is associated in the obvious way with a simplex of C. Thus processors decide on simplexes in C which are related by containment. Each processor P_i posts its simplex s_i in shared memory. It then takes a snapshot and takes the intersection $\cap s_j$ of all the simplexes it observed posted in it snapshot. If a vertex of its color is in $\cap s_j$ it decides on it. Otherwise it takes the simplex (by the containment property) $core_i = \cup s_j - \{w | w \in \cup s_j,\ \mathcal{X}(w) = P_i\}$ of the posted simplexes minus any vertex of its color if it is in $\cup s_j$ and proceeds to the next stage. At each stage at least one processor will be decided since $\cap s_j \neq \emptyset$.

The core that a processor carries is the set of vertices to which other processors may have converged. Inductively any vertex on which a processor has converged at stage i appears in all the cores of later stages. Thus a processor at stage i has to restrict itself to the link of the intersection of the cores at stage i. Any processor's core is a superset of the intersection. Thus as more processors participate at stage i the larger possibly the link on which they may converge. This is the signaling eluded to above. At stage i a processor posts as input, a vertex of its color in the link of the core from stage $i-1$, and its core. If two processors show up there is a predefined path that lives in the face of A that carries the two cores and the two starting vertices, and is entirely in the link of the intersection of the cores. Such a path exists since the face is a subdivided simplex, and the link satisfies the appropriate no-hole condition. When three processors participate in stage i there is a fill-in to the three paths defined by each pair of processors that lives in the link of the intersection of the three cores, etc. After processors converged to the barycentric subdivision of the link, the new core of a processor, i.e. the vertices it may suspect processors have converged to are the intersection of the cores of stage i it observed plus the union of the posted barycenters at the end of stage i. This is the core (minus itself if not in the intersection) that the processor carries to stage $i+1$.

The algorithm starts with empty cores. □

6 Conclusion

This paper, an outgrow of our preconference talk at Principles Of Distributed Computing 1995, is an attempt to elucidate the pioneering results of Herlihy and Shavit using familiar techniques from distributed computations theory. By so doing, we believe that we are making the results more accessible and, more important, more extendible: a result can be strengthened and expanded when it is derived in many different ways.

Neither these results nor the direction of investigation is "esoteric." Rather, they lay bare the heart of what asynchronous computations mean. Answering the questions that arise will, we believe, change our views of the modeling of asynchronous computations and the way we analyze them, in a profound way.

Acknowledgment: The second author would like to acknowledge uncountable coffee sessions with Elias Koutsoupias. David Taylor participated in a direct proof that the immediate-snapshot protocol complex is a subdivided simplex. That proof is discarded in this paper in favor of the simplicity of emulation from iterated immediate snapshots.

References

[1] Y. Afek, H. Attiya, D. Dolev, E. Gafni, M. Merrit, and N. Shavit, "Atomic Snapshots of Shared Memory," in *Proceedings of the 9th ACM Symposium on Principles of Distributed Computing*, pp. 1–13, 1990.

[2] M. Fischer, N. Lynch, and M. Paterson, "Impossibility of Distributed Consensus with One Faulty Process," *Journal of the ACM*, vol. 32, no. 2, pp. 374–382, 1985.

[3] O. Biran, S. Moran, and S. Zaks, "A Combinatorial Characterization of the Distributed Tasks which Are Solvable in the Presence of One Faulty Processor," in *Proceedings of the 7th ACM Symposium on Principles of Distributed Computing*, pp. 263–275, 1988.

[4] S. Chaudhuri, "More *choices* allow more *faults*: Set consensus problems in totally asynchronous systems," *Information and Compu-*

tation, vol. 105, pp. 132 – 158, jul 1993. supercedes 1990 PODC version.

[5] M. Saks and F. Zaharoglou, " Wait-Free k-Set Agreement is Impossible: The Topology of Public Knowledge," in *Proceedings of the 26th ACM Symposium on the Theory of Computing*, pp. 101–110, 1993.

[6] M. Herlihy and N. Shavit, "The Asynchronous Computability Theorem for t-Resilient Tasks," in *Proceedings of the 25th ACM Symposium on the Theory of Computing*, pp. 111–120, 1993.

[7] E. Borowsky and E. Gafni, "Generalized FLP Impossibility Result for t-Resilient Asynchronous Computations," in *Proceedings of the 25th ACM Symposium on the Theory of Computing*, pp. 91–100, 1993.

[8] E. Borowsky and E. Gafni, "Immediate Atomic Snapshots and Fast Renaming," in *Proceedings of the 12th ACM Symposium on Principles of Distributed Computing*, pp. 41–51, 1993.

[9] E. Gafni and E. Koutsoupias, "3-processor tasks are undecidable," in *Proceedings of the 14th Annual ACM Symposium on Principles of Distributed Computing*, p. 271, ACM, Aug. 1995.

[10] E. Borowsky, "Capturing the Power of Resiliency and Set Consensus in Distributed Systems," tech. rep., University of California, Los Angeles, Oct 15, 1995.

[11] E. Gafni and E. Koutsoupias, "New characterization of resiliency and consensus,"

[12] M. Herlihy and N. Shavit, "A Simple Constructive Computability Theorem for Wait-free Computation," in *Proceedings of the 26th ACM Symposium on the Theory of Computing*, 1994.

[13] H. Attiya and S. Rajsbaum, "The combinatorial structure of wait-free solvable tasks," *Lecture Notes in Computer Science*, vol. 1151, pp. 322–??, 1996.

[14] M. Mavronicolas, "Wait-free solvability via combinatorial topology," *PODC96, Brief Announcement*, 1996.

[15] C. P. Rourke and B. J. Sanderson, *Introduction to Piecewise-Linear Topology.* Springer Verlag, 1982.

[16] M. Herlihy and N. Shavit, "The topological structure of asynchronous computability," *preprint*, January 1996. stoc93, stoc94 of authors, long-awaited-new-improved.