COMPUTABLE OBSTRUCTIONS TO WAIT-FREE COMPUTABILITY

John Havlicek

March 31, 1997

Abstract. We show how to associate effectively computable obstructions to a wait-free distributed decision task $(\mathcal{I}, \mathcal{O}, \Delta)$ in the asynchronous shared-memory, read-write model. The key new ingredient of this work is the association of a simplicial complex $\mathcal{T}$, the *task complex*, to the input-output relation Δ. There is a simplicial map α from the task complex to the input complex $\mathcal{I}$, and α is determined by the task. The existence of a wait-free protocol solving the task implies that the map α_* induced in homology must surject, and thus the non-zero elements of the cokernel of α_* are obstructions to solvability of the task. These obstructions are effectively computable when using suitable homology theories, such as mod-2 simplicial homology. Functors other than homology can be substituted, although the obstructions obtained may not be computable. We also extend the Herlihy-Shavit Theorem on Spans to the case of protocols that are anonymous relative to the action of a group, provided the action is suitably *rigid*. For such rigid actions, the quotients of the input complex and the task complex by the group are well behaved, and obstructions to anonymous solvability of the task are obtained analogously, using the homology of the quotient complexes.

1. Introduction

Given a model of computation, a basic theoretical problem is to gain an understanding of tasks computable in the model and to prescribe some meaningful measure of the complexity of these tasks. This problem has proved quite difficult for models of distributed computation that are expected to tolerate faults but in which lack of synchronization prevents the detection of faulty processes. The seminal paper of Fischer, Lynch, and Paterson [FLP] shows that in the asynchronous message-passing model, no deterministic protocol that tolerates even a single faulty process can solve the basic task of consensus. Subsequent work of Biran, Moran, and Zaks [BMZ] generalizes the approach of [FLP] to give necessary and sufficient conditions for solvability of distributed decision tasks assuming asynchronous message passing and resilience to a single faulty process. Their techniques use adjacency graphs associated to configurations of inputs, outputs, and protocol outputs. Elements of these graphs appear also in [FLP]. Among the necessary conditions for solvability is that connectivity of the graphs be preserved in a suitable sense when passing from inputs to outputs. It should be noted that in the adjacency

The author was supported by an MCD Fellowship granted by the University of Texas at Austin during the research and writing of this paper.

graphs of [BMZ], a vertex corresponds to a configuration of all processes, while two vertices are connected by an edge if the corresponding configurations agree except at a single process.

More recent work ([BG2, HS1-2, SZ] are just a few examples) has focussed on the asynchronous read-write shared-memory model, in which processes communicate by atomic (snapshot) read and atomic write operations on shared memory locations. For this model, Herlihy and Shavit [HS1] introduced a framework for describing and reasoning about solvability of distributed decision tasks that uses the language of simplicial complexes from combinatorial and algebraic topology. The simplicial complexes $\mathcal{I}$ and $\mathcal{O}$ that describe the inputs and outputs, respectively, for a decision task are "dual" to the adjacency graphs of inputs and outputs in [BMZ]. Indeed, a vertex in one of these complexes represents a configuration of a single process, and a simplex of dimension k represents a configuration of $k+1$ processes. If two configurations share a common sub-configuration of size $k+1$, then the corresponding simplices intersect in a sub-simplex (i.e., face) of the dimension k. The use of simplicial complexes provides the flexibility to represent configurations of any positive number of processes, which is important when entertaining the possibility of failure of more than one process, and it represents the collection of configurations of a given type efficiently, at least in the sense that each configuration corresponds to a unique simplex. (In the adjacency graph representation, multiple edges may correspond to the same configuration.)

In addition to being notationally convenient, simplicial complexes provide the "right" topology for studying wait-free solvability of decision tasks. Wait freedom in this context amounts to resilience to failure of up to all but one process (since processes run asynchronously and fail only by crashing). The correctness of the topology is justified by the Asynchronous Computability Theorem of Herlihy and Shavit [HS2]. This theorem gives elegant necessary and sufficient conditions for the existence of a wait-free protocol to solve a decision task. A decision task is specified by a triple $(\mathcal{I}, \mathcal{O}, \Delta)$. As above, $\mathcal{I}$ and $\mathcal{O}$ are the complexes of inputs and outputs, respectively, and Δ is the relation that associates to each input configuration the collection of output configurations that are satisfactory under the task. In very loose terms, the Asynchronous Computability Theorem says that $(\mathcal{I}, \mathcal{O}, \Delta)$ admits wait-free solution if and only if there is a suitable continuous map $f\colon |\mathcal{I}| \to |\mathcal{O}|$ that respects the relation Δ. Here, $|\mathcal{I}|$ denotes the topological space that is obtained in a standard fashion from the combinatorial object $\mathcal{I}$, and similarly for $|\mathcal{O}|$. In slightly more precise terms, the condition is that there exists a subdivision $\chi(\mathcal{I})$ that respects processes and there exists a simplicial map $\mu\colon \chi(\mathcal{I}) \to \mathcal{O}$ that respects both processes and the relation Δ. The map f can be taken as $|\mu|$ when $|\chi(\mathcal{I})|$ is identified with $|\mathcal{I}|$. (A precise statement of the Asynchronous Computability Theorem is given in Subsection 2.5 below.)

In order to prove the Asynchronous Computability Theorem, Herlihy and Shavit introduce an auxiliary complex, the *protocol complex*, whose topology captures the capabilities of the protocol. Given a protocol that is wait-free executable on $\mathcal{I}$, the protocol complex $\mathcal{P}(\mathcal{I})$ is the simplicial complex of configurations of final views of finishing processes in all possible executions of the protocol with input configurations from $\mathcal{I}$. If the protocol solves the task, then there is a simplicial map $\delta\colon \mathcal{P}(\mathcal{I}) \to \mathcal{O}$ that respects process identifiers and the task relation. The map δ simply maps finishing processes to their output values. The proof of necessity

proceeds by showing that for a fine enough subdivision $\chi(\mathcal{I})$, there exists a simplicial map $\varphi\colon \chi(\mathcal{I}) \to \mathcal{P}(\mathcal{I})$ that respects both process identifiers and, in an appropriate fashion, carriers of the subdivision. Herlihy and Shavit call such a map φ a *span*. Their Theorem on Spans is stated precisely in Subsection 2.5 below. The map μ can then be obtained as the composition $\delta \circ \varphi$. The proof of sufficiency relies essentially on Borowsky and Gafni's clever Participating Set Protocol [BG1] for solving the *simplex agreement task*. This protocol serves as a "universal" protocol for wait-free decision tasks, and the existence of the map μ allows specialization of the universal protocol to the task at hand.

The Asynchronous Computability Theorem can be used to give proofs of the impossibility of solution of Chaudhuri's *set consensus task* [Ch] and Attiya et al.'s *renaming task* [At+].[1] Such proofs can be found in [HS1, HS3], although it should be noted that independent proofs of the impossibility of solution of the set consensus task appear in [BG2, SZ]. Arguments of this kind tend to focus on a single input configuration that is challenging for the task (such as a configuration of distinct inputs in the case of set consensus) and show that there is a topological obstruction to arranging μ on that simplex despite the refinement of the subdivision χ. While they are concise and æsthetically pleasing applications of topology, these arguments are ad hoc and seem ill suited to automation. For example, a basic problem in automating a search for μ is that it is not known how fine a subdivision χ should suffice. It would therefore be interesting to find a more systematic and computationally feasible method for approaching the question of wait-free solvability of a decision task.[2]

In this paper, we propose such a method. Our approach takes advantage of the global topology of the input complex and the previously unexplored topology of the task specification itself. We associate a simplicial complex $\mathcal{T}$, the *task complex*, to the input-output relation Δ of a decision task $(\mathcal{I}, \mathcal{O}, \Delta)$. The task complex is determined in a simple way from Δ, and there is a simplicial map $\alpha\colon \mathcal{T} \to \mathcal{I}$, also determined from Δ. From the Herlihy-Shavit Theorem on Spans, it follows that if the task admits wait-free solution, then $|\alpha|$ has a right homotopy inverse, i.e. a continuous map $\beta\colon |\mathcal{I}| \to |\mathcal{T}|$ such that the composition $|\alpha| \circ \beta$ is homotopic to the identity map of $|\mathcal{I}|$. The existence of a right homotopy inverse implies that the map

$$\alpha_*\colon H_*(\mathcal{T}) \to H_*(\mathcal{I})$$

induced in simplicial homology must surject in all dimensions. (A more general consequence of the existence of a right homotopy inverse for $|\alpha|$ is given in Theorem 3.2.2.) Of course, the homology map α_* and its image can be computed whether or not there is a wait-free solution of the task, and the existence of nonzero elements in the cokernel of α_* (equivalently, the existence of elements of $H_*(\mathcal{I})$ that are not in the image of α_*) implies that *no wait-free solution exists*. It is customary in such a situation to say that the cokernel of α_* consists of *obstructions* to wait-free solution of the decision task.[3] It is well known that simplicial homology of a finite complex

[1] Precise descriptions of these tasks are given in Subsections 5.2 and 5.3.

[2] Any such method that is computable is necessarily incomplete, since it is known [GK, HR3] that the question of wait-free solvability of a decision task is undecidable in general for three or more processes.

[3] We do not mean that $\operatorname{coker} \alpha_*$ is a complete set of obstructions in the sense that $\operatorname{coker} \alpha_* = 0$ implies the existence of a wait-free solution. See the preceding footnote.

can be computed in time that is a polynomial function of the number of simplices of the complex. The image of a homology map such as α_* can also be computed in time that is polynomial in the number of simplices of the domain and the target complexes. In this sense, the obstructions we propose are effectively computable.

Of course, having defined obstructions to wait-free computability does not guarantee that computation of these obstructions actually detects the impossibility of solution of any decision tasks. But, in fact, the obstruction method seems to be powerful enough to detect the impossibility of solution of tasks such as consensus, set consensus, and renaming, the last after suitably adapting the definition of obstructions to the situation of anonymous protocols. Our results in this regard are somewhat preliminary and technical, though, and it is, perhaps, more convincing simply to automate the method of obstructions and let it work on examples.

The balance of this section presents a detailed overview of the present paper. Considerable effort has been made to ensure accessibility both to computer scientists and to mathematicians, and it is hoped that the writing is palatable to readers from both groups. The main text assumes that the reader is familiar with the material from algebraic topology that has already been used to study distributed computability. This material includes the language of simplicial complexes and subdivisions and the simplicial homology functor. Some elements of homotopy theory have been used in the proofs of major theorems, such as the Theorem on Spans. The reader unfamiliar with these topics is encouraged to skim the appendix. The primer by Herlihy and Rajsbaum [HR2] provides another good introduction, and the first chapter of Munkres standard textbook [Mu] is an excellent reference.

In Section 2, we set down preliminaries for our work. Brevity has been sacrificed for completeness. The asynchronous read-write shared-memory model is discussed in Subsection 2.1. Subsection 2.2 gives formal definitions of *well-posed decision task* and of the properties of *extensibility* and *reducibility* of such tasks. All of these ideas exist, at least in spirit, in prior work, but the treatment here is a bit different. Rather than accounting for the possibility of faulty processes when defining what it means for a protocol to solve a task, we demand that the task specification itself allow for failures by pairing input and output simplices of different dimensions. Subsection 2.3 describes wait-free protocols. The point of view is to understand a protocol by considering all of its possible (linear) execution sequences and the relation of input-output pairs that can be generated by these executions. This relation is called the *executable relation* of the protocol. We give axioms for the existence of execution sequences, and we show that, under suitable assumptions, these axioms ensure that the executable relation is well-posed, extensible, and reducible in the sense of Subsection 2.2. We use the executable relation to give a simple and, hopefully, natural definition of what it means for a protocol to solve a wait-free decision task. Again, these ideas exist already in the literature, although the definition of solution of a wait-free decision task has been murky. Subsection 2.4 describes the protocol complex associated to a wait-free protocol. In Subsection 2.5, we give a precise statement of the Theorem on Spans proved by Herlihy and Shavit [HS1], which is the launching point for our work. We also cite the necessary and sufficient conditions of the Asynchronous Computability Theorem [HS2].

With all of this background, the definition of the task complex $\mathcal{T}$ in Section 3 is quite easy. Subsection 3.1 gives the definition and a few very simple examples, motivated from [HS2]. We show how to use $\mathcal{T}$ to associate obstructions to a decision

task in Subsection 3.2, and we give our main result as Theorem 3.2.2. Applications are deferred until Section 5.

In Section 4, we enhance the topological structures already introduced by adding the action of a group of symmetries. The objective is to give a precise description of protocols that are *anonymous* in the informal sense of making no "essential" use of process identifiers. Group actions on topological spaces form a standard topic in algebraic topology. The idea of characterizing anonymous protocols in terms of such actions appears in [HR1], although in a somewhat different form. Subsection 4.1 presents the rigidity properties of group actions on complexes that are appropriate for our work and shows how to form quotient spaces under such actions. In Subsection 4.2, we restrict attention to groups of symmetries that arise by permuting processes. We give a simple condition for such group actions to be rigid and make a precise definition of anonymity. Theorem 4.2.1 is a generalization of the Theorem on Spans, and it leads immediately to a generalization of our main result to the situation of anonymous protocols, Theorem 4.2.2.

Section 5 gives non-trivial applications of the obstruction method. We show that the obstruction method is powerful enough to detect the impossibility of solving ordinary consensus and of solving certain non-trivial cases of set consensus and anonymous renaming. These results are somewhat preliminary, and the arguments make rather heavier use of algebraic topology than needed in the preceding sections. On the other hand, the obstruction method opens the door to computer-aided proofs of impossibility of solution of wait-free decision tasks.

Acknowledgment. The author wishes to thank Lorenzo Alvisi for many useful and encouraging discussions during the research and writing of this paper.

2. Preliminaries

2.1 Model of computation.

The model of computation for the present work is the asynchronous read-write shared-memory model that has been common in recent studies of wait-free decision tasks [BG3, GK, HS2, HS3, SZ]. We assume a system of $n+1$ processes, $p_0, \ldots, p_n$. These processes run asynchronously, which means that no assumption is made about their relative speeds. The processes may therefore experience arbitrary delays relative to one another. Each process is assumed to have private memory that can be used for internal computations and in which private input values can appear at the outset of a distributed computation. The private memory is assumed to be unbounded.

The processes communicate through shared memory locations. The shared memory is organized as an array $S[0..n]$ of non-overlapping regions. The region $S[i]$ can be written only by p_i, but can be read by any process. We assume that each region $S[i]$ is unbounded. Processes interact with shared memory via atomic *write* and *scan* operations.[4] In a write operation, process p_i appends a value to $S[i]$ without destroying the contents of any previously written location in $S[i]$. The shared memory is assumed to support atomic snapshots [Af+]. A scan operation by p_i

[4] An operation by a process is *atomic* if it is always executed without interruption.

therefore returns to the private memory of p_i a copy of the entire shared memory. This copy is called the *view* returned to the process by the scan operation.

The processes execute deterministic protocols. A protocol is a collection of programs running on the processes that direct all shared-memory write and scan operations. A protocol may also direct internal computations of a process, although such computations can influence other processes only through the use of writes and scans. The internal state of a process consists of the values stored in its private memory. A deterministic protocol is one such that the internal state of p_i at any point of the computation determines recursively a unique ensuing sequence of actions and internal state changes of p_i up to and including the next scan of shared memory. The view returned from the next scan is not determined and depends on the intervening writes by other processes to shared memory.

Processes may fail only by crashing. A process crashes when it stops execution without having been directed to halt by the protocol. A crashed process remains crashed, taking no further action. A process may crash at any time, except in a way that disintegrates an atomic operation. A crashing process makes no terminal write to shared memory to notify the other processes of its demise. Since processes run asynchronously, they cannot distinguish a crashed process from one that has been severely delayed. With the distinction between crashed and severely delayed processes blurred, the condition that a protocol run wait-free is tantamount to the condition that the protocol tolerate faults by up to n of the $n+1$ processes.

2.2 Decision tasks.

In a distributed decision task, each of the $n+1$ processes begins with a private input value, executes a protocol, during which it may communicate with the other processes, and, barring failure, concludes by writing an output value to shared memory and then halting.[5] For this paper, we assume that the input values and output values come from finite sets V_I and V_O (respectively) that are fixed from the outset. An assignment of input (respectively, output) values to one or more of the processes will be called an *input* (respectively, *output*) *configuration*. An input or output configuration which assigns values to all $n+1$ processes will be called *full*. We can represent assignment of the value v to the process with identifier p by the ordered pair $x = (p, v)$. According to the notation in [HS3], for such a pair we write $\mathrm{id}(x) = p$ and $\mathrm{val}(x) = v$.[6] An input (respectively, output) configuration is then written as a set $X = \{x_0, \ldots, x_r\}$ of pairs such that $\mathrm{id}(x_0), \ldots, \mathrm{id}(x_r)$ are distinct process identifiers and each value $\mathrm{val}(v_i)$ comes from V_I (respectively, V_O). Such a configuration can be thought of as a chromatic r-simplex with vertices x_i and coloring $x_i \mapsto \mathrm{id}(x_i)$. The vertices of X are additionally labelled by the various values. If X and X' are configurations such that $X \subseteq X'$, then we say that X is *extensible* to X' and X' is an *extension* of X.

A decision task for the $n+1$ processes must specify which full input configurations are *admissible* for the task. In this paper, we focus on tasks that are genuinely

[5] In [HS3] the authors require that the output values be private. However, for the normalized full-information protocols in [HS3], any process that (1) scans the shared memory after the final write of process p, (2) can detect that this write is the final write of process p (as in the case of protocols that proceed in a fixed number of asynchronous rounds), and (3) has access to the decision function δ can deduce the output of p.

[6] If $v \in V_I$ and it is useful to emphasize this fact, we may write $\mathrm{inval}(x) = v$. Similarly, if $v \in V_O$, we may write $\mathrm{outval}(x) = v$.

intended for all the processes, and so any admissible input configuration should be extensible to an admissible full input configuration. In order for the task to be well-posed for wait-free computation, it must also address the possibility of failure of some of the processes, perhaps from the outset. A process is said to be *participating* in an execution of a protocol if it at least manages to write its input value to shared memory. The process is said to be *effectively participating* in an execution if it manages to write its input value to shared memory before the final scan by some non-faulty process. Assuming that the protocol directs each process to make at least one scan of shared memory, every non-faulty process is effectively participating. The input values of non-participating processes cannot influence the execution, and the input values of processes that are not effectively participating cannot influence the execution of any non-faulty process. We say that the *effective input configuration* for the execution is the configuration of inputs of the effectively participating processes. In order to articulate that tasks be intended for all processes and to allow for the possibility of processes that do not participate, or that do not participate effectively, we make the following requirement of a wait-free decision task:

(T1): *Any admissible input configuration is extensible to an admissible full input configuration. Any input configuration that is extensible to an admissible input configuration is itself admissible.*

With this requirement, the admissible input configurations form a pure n-dimensional chromatic simplicial complex, the input complex, which is denoted $\mathcal{I}$.

Again following the notation of [HS3], we write $\mathrm{ids}(X)$ for the set $\{\mathrm{id}(x) : x \in X\}$. A decision task must specify for each admissible input configuration the collection of output configurations that are considered satisfactory solutions to the task for the given input. The decision task must therefore specify an input-output relation Δ consisting of pairs (X, Y) where X is an admissible input configuration and Y is an output configuration satisfactory for input X. It is unreasonable for a decision task to ask processes that are not effectively participating to reach decision values. We therefore require

(T2): *If* $(X, Y) \in \Delta$, *then* $\mathrm{ids}(Y) \subseteq \mathrm{ids}(X)$.[7]

If all participating processes in an execution are non-faulty, then there should be an acceptable way for them all to write output values. The decision task needs, therefore, to specify for an admissible input configuration X at least one satisfactory output configuration involving the same processes. Since effectively participating processes may fail, there must also be output configurations involving as few as one process that are satisfactory for the corresponding input configuration. Note that the last condition is not the same as specifying an output configuration for an input with fewer processes, since the input values of effectively participating processes that fail can influence the outputs of non-faulty processes. Thus, we require

[7]Herlihy and Shavit [HS3] require that $\mathrm{ids}(Y) = \mathrm{ids}(X)$ for $(X, Y) \in \Delta$. With this stricter condition, however, it is no longer correct to consider Δ as specifying *all* satisfactory input-output pairs for a wait-free task, since effectively participating processes are then never allowed to fail.

(T3): *For each admissible input configuration X and for each non-empty subset P of* $\mathrm{ids}(X)$*, there is at least one pair (X,Y) in Δ such that* $\mathrm{ids}(Y)=P$*.*

DEFINITION 2.2.1: A decision task that satisfies (T1), (T2), and (T3) will now formally be called *well-posed*.

Since processes cannot distinguish a faulty process from one which is running slowly, it is desirable for a wait-free decision task to exhibit *extensibility* in the relation Δ. Consider, for example, a situation in which a subset of the non-faulty processes run very quickly, so fast, say, that they finish the protocol before others among the non-faulty processes write their input values to shared memory. A reasonable wait-free task must allow the slow processes to finish the protocol in a satisfactory way. We may therefore require

(T4): *If $(X,Y)\in\Delta$ and X' is an admissible extension of X, then there is an extension Y' of Y with* $\mathrm{ids}(Y')=\mathrm{ids}(X')$ *such that $(X',Y')\in\Delta$.*

A well-posed decision task that satisfies (T4) will be called *extensible*.

Remark. Any decision task can be modified to satisfy (T4) by adding a special output value $\perp$ and creating any missing extensions Y' by assigning the value $\perp$ to processes with no value defined from the original set V_O. Processes halting with value $\perp$ are then understood to have withdrawn from the decision process without choosing one of the original output values. Such a modified decision task may or may not preserve the spirit of the original task. □

Another desirable property of a wait-free decision task is *reducibility*, meaning that the input-output relation Δ admits the assembly of a satisfactory solution from partial solutions. If a collection P of processes has finished a protocol at some point and the output configuration of the processes in P does *not* represent a satisfactory solution for the effective input configuration at that moment, then an immediate crash of all processes not in P leaves the protocol with an unsatisfactory output. We may therefore require

(T5): *Let $(X,Y)\in\Delta$. Then there is a strictly ascending chain $Y_1\subset\cdots\subset Y_r=Y$ of output configurations with $|Y_j|=j$ and an ascending chain $X_1\subseteq\cdots\subseteq X_r=X$ of input configurations such that* $\mathrm{ids}(Y_j)\subseteq\mathrm{ids}(X_j)$ *and each $(X_j,Y_j)\in\Delta$.*

A well-posed decision task that satisfies (T5) will be called *reducible*. We may be less restrictive in the decision task by admitting as acceptable *any* partial solution of an acceptable solution. We will say that a well-posed decision task is *strongly reducible* if it satisfies

(T6): *If $(X,Y)\in\Delta$ and Y_1 is extensible to Y, then $(X,Y_1)\in\Delta$.*

It seems that most decision tasks considered in previous research have been treated as strongly reducible. The input-output relation Δ of a well-posed task can be extended to a relation Δ' satisfying (T6) simply by adding any missing pairs. Our construction of the task complex $\mathcal{T}$ will not distinguish between Δ and Δ'.

If Δ is the input-output relation of a decision task, then we say that an output configuration Y is *admissible* if there is an admissible input configuration X such

that $(X, Y) \in \Delta$. Note that the collection of admissible output configurations need not be a simplicial complex, even if the task is well-posed. We call an output configuration *sub-admissible* if it is extensible to an admissible output configuration. The collection of sub-admissible output configurations does form a chromatic simplicial complex, the output complex, which is denoted $\mathcal{O}$. If the decision task is strongly reducible, then all sub-admissible output configurations are themselves admissible, and so, in this case, $\mathcal{O}$ consists entirely of admissible configurations.

We close this subsection with two very simple examples, taken from [HS2]. We will revisit these examples to illustrate the construction of the task complex in Section 3.

EXAMPLE 2.2.2. *Two-process binary consensus.* In this task, $V_I = V_O = \{0, 1\}$. There are two processes, p and q, and each can begin with either of the two possible input values. The task requires that the processes reach consensus in the sense that all finishing processes choose the same output value. Furthermore, this common value must be the input value of at least one effectively participating process.

We will use the following notation. A configuration is written as a sequence of values within angle brackets, the value of p first and the value of q second. If a process has no value in the configuration, then we write $*$ in place of its value. Thus, $\langle 01 \rangle$ represents the configuration in which p has value zero and q has value 1, while $\langle *0 \rangle$ represents the configuration in which q has the value 0 and p does not appear.

The input complex $\mathcal{I}$ has four vertices, $\langle *0 \rangle$, $\langle *1 \rangle$, $\langle 0* \rangle$, and $\langle 1* \rangle$, and four 1-simplices $\langle 00 \rangle$, $\langle 01 \rangle$, $\langle 10 \rangle$, and $\langle 11 \rangle$. $\mathcal{I}$ can be realized as the boundary of a square, and $|\mathcal{I}|$ is thus homeomorphic to a 1-sphere. The output complex $\mathcal{O}$ can be identified with the subcomplex of $\mathcal{I}$ obtained by removing the two 1-simplices with mixed values, $\langle 01 \rangle$ and $\langle 10 \rangle$. $|\mathcal{O}|$ is therefore a pair of disjoint line segments.

In order to describe Δ, we abbreviate the notation for a pair from $(\langle ab \rangle, \langle cd \rangle)$ to simply $\langle ab, cd \rangle$. Then Δ is enumerated as

$\langle 0*, 0* \rangle$	$\langle 1*, 1* \rangle$	$\langle *0, *0 \rangle$	$\langle *1, *1 \rangle$		
$\langle 00, 0* \rangle$	$\langle 00, *0 \rangle$	$\langle 00, 00 \rangle$	$\langle 11, 1* \rangle$	$\langle 11, *1 \rangle$	$\langle 11, 11 \rangle$
$\langle 01, 0* \rangle$	$\langle 01, 1* \rangle$	$\langle 01, *0 \rangle$	$\langle 01, *1 \rangle$	$\langle 01, 00 \rangle$	$\langle 01, 11 \rangle$
$\langle 10, 0* \rangle$	$\langle 10, 1* \rangle$	$\langle 10, *0 \rangle$	$\langle 10, *1 \rangle$	$\langle 10, 00 \rangle$	$\langle 10, 11 \rangle$

Notice that this task is both extensible and strongly reducible.

EXAMPLE 2.2.3. *Two-process binary almost-consensus.* This task has the same input complex as two-process binary consensus, and any input-output pair satisfactory for binary consensus is still allowed as satisfactory. In addition, if the processes begin with distinct inputs, then $\langle 10 \rangle$ is allowed as a satisfactory output. The output complex $\mathcal{O}$ has, therefore, the additional 1-simplex $\langle 10 \rangle$ and $|\mathcal{O}|$ is three sides of a square. The input-output relation is obtained from that for binary consensus by adding $\langle 01, 10 \rangle$ and $\langle 10, 10 \rangle$. As with binary consensus, binary almost-consensus is both extensible and strongly reducible.

2.3 Executions of wait-free protocols.

We assume all protocols to be structured so that the first action of a process is to write its input value to shared memory, and its last action before halting is to write its output value to shared memory. A process is said to *start/finish* the protocol if

and when it writes its input/output value to shared memory. A process scanning shared memory after process p has finished the protocol can therefore deduce that p has finished. We assume that all protocols direct each process to make at least one scan of shared memory between starting and finishing. Protocols are also assumed to be deterministic, and processes can fail only by crashing. All executions are assumed to have at least one non-faulty process.

Let e be an execution of a protocol. We assume that it is meaningful to speak of a *clairvoyant view* of e from which can be determined

(1) The full input configuration describing the input values offered to all $n+1$ processes.
(2) The sequence $\langle e_i \rangle$ of "events" of e.
(3) A partition of the $n+1$ processes into the sets F_e and NF_e of processes that are faulty in e and that are non-faulty in e, respectively.

We do not assume that any process is able to achieve a clairvoyant view.

At this point we have not said what constitutes an event, nor have we said exactly which events are reflected in the sequence $\langle e_i \rangle$. In general terms, an event should represent an atomic action of some process. We assume that all scans of and writes to shared memory are reflected in the sequence $\langle e_i \rangle$. Recall that writes are assumed to append to shared memory, not overwrite, and thus if w is a write event in the sequence and s is a scan event appearing later in the sequence, then the value written in w appears in the view returned in s. We also assume that if the protocol directs internal actions of processes, sufficiently many internal actions are reflected in the sequence to ensure that a non-faulty process cannot appear faulty.[8] It is then possible to define the sets NF_e and F_e in terms of the sequence of events, as in [HS3]: NF_e consists of the processes that finish or to which are associated infinitely many events.

A clairvoyant view of e determines the effective input configuration for the execution, which we will denote by X_e. Recall that X_e is the configuration of input values of processes with a start event in e that precedes a scan event by some process that is non-faulty in e. A clairvoyant view also determines the configuration Y_e of output values of all processes that finish the protocol in e. We have

$$\mathrm{ids}(Y_e) \subseteq NF_e \subseteq \mathrm{ids}(X_e) \ .$$

In order to characterize more precisely the collection of all wait-free executions of a protocol, we propose the axioms (E1) through (E5) below. We will use these axioms to establish properties of the executable relation of the protocol (defined below), showing that this relation determines a well-posed, extensible, reducible decision task.

(E1): *For any full input configuration and for any non-empty sets of processes $R \subseteq Q \subseteq P$, there exists an execution in which P is the set of participating processes, Q is the set of effectively participating processes, and R is the set of non-faulty processes.*
(E2): *If e is an execution and X is a full input configuration that extends the input configuration of processes participating in e, then there is an execution e'*

[8] A protocol could conceivably direct a process to enter a non-terminating internal computation whose actions are not reflected in the sequence of events.

with X as full input configuration and whose sequence of events is the same as that of e.

(E3): *Two successive events of an execution may be interchanged to yield another execution provided the events are associated to different processes and* (1) *the events are both scans of shared memory,* (2) *the events are both writes to shared memory, or* (3) *at least one of the events is internal to a process.*

(E4): *Let e be an execution, and let P be a set of processes that does not contain all processes in NF_e. For any event e_c that does not follow the finishing event of any process in P, there is an execution e' such that* (1) $e'_i = e_i$ *for* $i < c$, (2) *no process in P has an event in $e'_c e'_{c+1} \cdots$, hence $P \subseteq F_{e'}$, and* (3) $NF_e - P \subseteq NF_{e'}$.

(E5): *Let e be an execution, and let p be a process in F_e whose last event in e is e_j (we understand $j = -\infty$ if p has no event in e). For any event e_r with $r > j$, there is an execution e' such that* (1) $e'_i = e_i$ *for* $i < r$, (2) e'_r *is an event associated to p,* (3) *processes other than p that have no event in $e_r e_{r+1} \cdots$ have no event in $e'_r e'_{r+1} \cdots$, and* (4) $NF_e \cup \{p\} \subseteq NF_{e'}$.

Loosely speaking, (E4) represents crashing all processes in P by the instant before event e_c. Since the failure of a process to write might allow another process to finish more quickly, (E4)(3) allows processes in $F_e - P$ to become non-faulty in e', although such processes could be crashed by another application of (E4). Notice that (E4) can be used to eliminate all events of processes that are participating but not effectively participating in e. In the resulting execution e', $(X_{e'}, Y_{e'}) = (X_e, Y_e)$ and all participating processes are effectively participating. (E5) represents "resurrecting" process p at the instant before event e_r. (E5)(3) ensures that processes that crashed before e_r are not resurrected along with p. By suitably crashing and resurrecting processes, these axioms can be used to obtain executions that reflect delays or accelerations of the various processes.

Definition 2.3.1: Let X be an input configuration. A protocol is *wait-free executable* on X if in any execution e of the protocol for which the effective input configuration X_e is extensible to X, all non-faulty processes finish the protocol, i.e., if

$$X_e \subseteq X \Rightarrow \mathrm{ids}(Y_e) = NF_e \ .$$

Notice that if a protocol is wait-free executable on X and if X_1 is extensible to X, then the protocol is wait-free executable on X_1. A protocol is wait-free executable on a complex of input configurations if it is wait-free executable on each simplex of the complex.

Definition 2.3.2: Let $\mathcal{K}$ be a complex of input configurations on which a protocol is wait-free executable. By $\Delta_{\mathrm{ex}}(\mathcal{K})$ we mean the relation consisting of all pairs (X_e, Y_e) arising from executions of the protocol with X_e a simplex of $\mathcal{K}$. $\Delta_{\mathrm{ex}}(\mathcal{K})$ is the *executable relation* that arises by applying the protocol to $\mathcal{K}$.

Definition 2.3.3: A protocol is a *wait-free solution* of a well-posed decision task $(\mathcal{I}, \mathcal{O}, \Delta)$ if (1) the protocol is wait-free executable on $\mathcal{I}$, and (2) the executable relation $\Delta_{\mathrm{ex}}(\mathcal{I})$ of the protocol is a subset of Δ.

Remark. In [HS3], the authors give different definitions of what it means for a protocol to be a wait-free solution of a decision task $(\mathcal{I}, \mathcal{O}, \Delta_{\mathrm{HS}})$. (In [HS3], the

input-output relation consists only of pairs (X, Y) for which $\mathrm{ids}(X) = \mathrm{ids}(Y)$, and we write Δ_{HS} to emphasize this restriction.) On p. 6 of [HS3], the authors define a protocol to be a wait-free solution of $(\mathcal{I}, \mathcal{O}, \Delta_{\mathrm{HS}})$ if (1) the protocol is wait-free executable on $\mathcal{I}$, and (2) for any execution e in which the input configuration X of participating processes is a simplex of $\mathcal{I}$, there is an extension Y of Y_e such that $(X, Y) \in \Delta_{\mathrm{HS}}$. Condition (2) amounts to allowing that if $(X, Y) \in \Delta_{\mathrm{HS}}$, then any non-empty $Y_1 \subseteq Y$ also represents a satisfactory solution for input X. In other words, the task should be treated as strongly reducible. The specification of X as an input configuration of participating processes instead of just effectively participating processes is unimportant if the task is also understood to be extensible, as in [HS3, p. 4]. It is not difficult to check that if Δ_{HS} is extensible in the sense of [HS3, p. 4], then there is a unique minimal extension Δ of Δ_{HS} that is extensible and strongly reducible, and a wait-free solution of $(\mathcal{I}, \mathcal{O}, \Delta_{\mathrm{HS}})$ in the sense of [HS3, pp. 4–6] is equivalent to a wait-free solution of $(\mathcal{I}, \mathcal{O}, \Delta)$ according to our definition. □

We can view the executable relation $\Delta_{\mathrm{ex}}(\mathcal{K})$ of a protocol on $\mathcal{K}$ as the input-output relation of a decision task. We write $\mathcal{O}_{\mathrm{ex}}(\mathcal{K})$ for the associated output complex. Notice that if the protocol is a wait-free solution of $(\mathcal{I}, \mathcal{O}, \Delta)$, then $\mathcal{O}_{\mathrm{ex}}(\mathcal{I})$ is a subcomplex of $\mathcal{O}$.

Proposition 2.3.4: *Let $\mathcal{K}$ be a pure n-dimensional complex of input configurations on which a given protocol is wait-free executable. Then $(\mathcal{K}, \mathcal{O}_{\mathrm{ex}}(\mathcal{K}), \Delta_{\mathrm{ex}}(\mathcal{K}))$ is a well-posed, extensible, and reducible decision task of which the protocol is a wait-free solution.*

Proof: Throughout, we use the assumption that the protocol is wait-free executable on $\mathcal{K}$. The task satisfies (T1) because $\mathcal{K}$ is a pure complex. (T2) is satisfied because $\mathrm{ids}(Y_e) \subseteq \mathrm{ids}(X_e)$ for any execution e, and (T3) follows by applying (E1). Thus, the task is well-posed. Let $(X, Y) \in \Delta_{\mathrm{ex}}(\mathcal{K})$. Using (E4), there is an execution e such that $(X_e, Y_e) = (X, Y)$ and such that all participating processes are effectively participating. Suppose that X' is a simplex of $\mathcal{K}$ that extends X. By using (E2), we can assume that the full input configuration for e is an extension of X'. Let e_j be the final scan event for a process in $\mathrm{ids}(Y_e)$. Using (E5) successively to resurrect all processes in $\mathrm{ids}(X')$ after e_j, we obtain an execution e' satisfying $\mathrm{ids}(Y_{e'}) = \mathit{NF}_{e'} = \mathrm{ids}(X')$, $X_{e'} = X'$, and such that $Y_{e'}$ extends Y_e. Thus, (T4) is satisfied and the task is extensible. Recall now the execution e such that $(X_e, Y_e) = (X, Y)$ and such that all participating processes are effectively participating. We apply (E4) successively to crash increasing sets of processes. Let p be the non-faulty process whose finish event f occurs latest in e. We use (E4) to crash p and all processes in F_e by the instant before f. Let the resulting execution be e^{r-1}, and write $X_{r-1} = X_{e^{r-1}}$ and $Y_{r-1} = Y_{e^{r-1}}$. Then $X_{r-1} \subseteq X$, $Y_{r-1} \subseteq Y$, and $\mathrm{ids}(Y_{r-1}) = \mathrm{ids}(Y_e) - \{p\}$. (The difference $X - X_{r-1}$ will be non-empty if the final scan s of p is the last among the processes in NF_e and there are processes with start events in e falling between s and the preceding final scan of a process in NF_e.) Now we repeat this procedure with e^{r-1} in place of e. Continuing in this fashion, we obtain the chains of configurations required by (T5). Thus, the task is reducible. □

2.4 The protocol complex.

Consider a protocol and let e be an execution. A process that finishes the protocol in e is understood to have as *final view* the view from the last scan it

made of shared memory together with the output value it wrote before halting. We can represent the fact that a process p achieves final view v by the pair $z = (p, v)$. For such a pair, we write $p = \mathrm{id}(z)$ and $v = \mathrm{view}(z)$, and we say that the pair is *derivable* from e. We also write $\mathrm{inval}(z)$ and $\mathrm{outval}(z)$ to denote the input value and output value (respectively) of process $\mathrm{id}(z)$. The same pair might be derivable from many different executions of the protocol. A *configuration of the protocol* is a non-empty set $Z = \{z_0, \ldots, z_r\}$ of pairs of process identifiers and final views such that $\mathrm{id}(z_0), \ldots, \mathrm{id}(z_r)$ are distinct and such that there exists a single execution e from which all the pairs $z_0, \ldots, z_r$ are derivable. In this case, we say that Z is derivable from e. Such a configuration can be thought of as a chromatic r-simplex. If Z is derivable from e, then there are chromatic simplicial maps $\psi\colon Z \to X_e$ and $\delta\colon Z \to Y_e$ defined by

$$\psi\colon z \mapsto (\mathrm{id}(z), \mathrm{inval}(z)) \quad \text{and} \quad \delta\colon z \mapsto (\mathrm{id}(z), \mathrm{outval}(z)) \ .$$

Notice that Z does not identify an execution e from which it is derivable. A given configuration of the protocol may be derivable from many executions.

Let z be a pair associating a process and final view in some execution. Let $X(z)$ denote the input configuration of all processes and corresponding input values that appear in $\mathrm{view}(z)$. Similarly, let $Y(z)$ denote the output configuration of all processes and corresponding output values that appear in $\mathrm{view}(z)$, including process $\mathrm{id}(z)$. If z is derivable from the execution e, then $X(z) \subseteq X_e$, $Y(z) \subseteq Y_e$, and

$$\mathrm{id}(z) \in \mathrm{ids}(Y(z)) \subseteq \mathrm{ids}(X(z)) \ .$$

For a configuration Z of the protocol, we write

$$X(Z) = \bigcup_{z \in Z} X(z) \ , \quad Y(Z) = \bigcup_{z \in Z} Y(z) \ .$$

If Z is derivable from e, then $X(Z) \subseteq X_e$, $Y(Z) \subseteq Y_e$, and

$$\mathrm{ids}(Z) \subseteq \mathrm{ids}(Y(Z)) \subseteq \mathrm{ids}(X(Z)) \ .$$

The *maximal* configuration for an execution e is the configuration consisting of all final views of non-faulty processes. If Z is the maximal configuration for e, then $X(Z) = X_e$, $Y(Z) = Y_e$, and

$$\mathrm{ids}(Z) = \mathrm{ids}(Y(Z)) \subseteq \mathrm{ids}(X(Z)) \ .$$

If the protocol is wait-free executable on the input configuration X, then the protocol complex $\mathcal{P}(X)$ is defined to be the complex consisting of all configurations Z derivable from executions e satisfying $X_e \subseteq X$. Similarly, if the protocol is wait-free executable on the complex $\mathcal{K}$ of input configurations, then the protocol complex $\mathcal{P}(\mathcal{K})$ is the union of the complexes $\mathcal{P}(X)$ as X runs through the simplices of $\mathcal{K}$. In this case, there are chromatic simplicial maps

$$\psi\colon \mathcal{P}(\mathcal{K}) \to \mathcal{K} \quad \text{and} \quad \delta\colon \mathcal{P}(\mathcal{K}) \to \mathcal{O}_{\mathrm{ex}}(\mathcal{K})$$

defined as above.

Proposition 2.4.1: *Let $\mathcal{K}$ be a complex of input configurations, and consider a protocol that is wait-free executable on $\mathcal{K}$. Then*

$$\Delta_{\mathrm{ex}}(\mathcal{K}) = \{(X(Z), Y(Z)) : Z \in \mathcal{P}(\mathcal{K})\} \ .$$

Proof: If e is an execution with X_e a simplex of $\mathcal{K}$, let Z be the maximal configuration for e. Then $(X_e, Y_e) = (X(Z), Y(Z))$. This proves the inclusion $\subseteq$.

Now let Z be a configuration in $\mathcal{P}(\mathcal{K})$. Then Z is derivable from an execution e with X_e a simplex of $\mathcal{K}$. We have $\mathrm{ids}(Z) \subseteq \mathrm{ids}(Y(Z)) \subseteq NF_e$. Let s be the last scan event of any process in $\mathrm{ids}(Z)$. Then any process with finishing event before s is in $\mathrm{ids}(Y(Z))$, and any process in $\mathrm{ids}(Y(Z)) - \mathrm{ids}(Z)$ has finishing event before s. Since the protocol is deterministic and wait-free executable on $\mathcal{K}$, failures by processes not in $\mathrm{ids}(Y(Z))$ after s cannot prevent any process in $\mathrm{ids}(Z)$ from finishing with the same output value as it would have otherwise. We can therefore use (E4) to crash all processes not in $\mathrm{ids}(Y(Z))$ just after s, leaving the processes in $\mathrm{ids}(Z)$ to finish. Let the resulting execution be e'. Then $NF_{e'} = \mathrm{ids}(Y(Z))$ and $Y_{e'} = Y(Z)$. Furthermore, any process that has not started by s does not start in e', so $X_{e'} = X(Z)$. Therefore, $(X(Z), Y(Z)) = (X_{e'}, Y_{e'}) \in \Delta_{\mathrm{ex}}(\mathcal{K})$. □

Corollary 2.4.2: *Let $(\mathcal{I}, \mathcal{O}, \Delta)$ be a well-posed decision task, and consider a protocol that is wait-free executable on $\mathcal{I}$. The protocol is a solution of the task if and only if*

$$\{(X(Z), Y(Z)) : Z \in \mathcal{P}(\mathcal{I})\} \subseteq \Delta \ .$$

Remark. On p. 25 of [HS3] the authors give the following definition of wait-free solution of a task $(\mathcal{I}, \mathcal{O}, \Delta_{\mathrm{HS}})$: a protocol that is wait-free executable on $\mathcal{I}$ is a solution of the task provided that, for every input configuration X in $\mathcal{I}$ and every protocol configuration Z in $\mathcal{P}(X)$,

$$\mathrm{ids}(Z) = \mathrm{ids}(X) \Rightarrow (X, \delta(Z)) \in \Delta_{\mathrm{HS}} \ . \tag{A}$$

For Z in $\mathcal{P}(X)$, though, $X(Z) \subseteq X$ and

$$\mathrm{ids}(Z) \subseteq \mathrm{ids}(Y(Z)) \subseteq \mathrm{ids}(X(Z)) \subseteq \mathrm{ids}(X) \ .$$

Thus, $\mathrm{ids}(Z) = \mathrm{ids}(X)$ implies that each of these inclusions is an equality, and it follows that $Y(Z) = \delta(Z)$ and $X(Z) = X$. The condition (A) is therefore equivalent to the requirement that, for all $X \in \mathcal{I}$ and for all $Z \in \mathcal{P}(X)$,

$$\mathrm{ids}(Z) = \mathrm{ids}(X) \Rightarrow (X(Z), Y(Z)) \in \Delta_{\mathrm{HS}} \ . \tag{B}$$

If we understand Δ to be the the minimal extensible and strongly reducible extension of Δ_{HS}, then the condition (B) is equivalent to the inclusion in the preceding corollary. □

Notice that if $\mathrm{ids}(Z) = \mathrm{ids}(Y(Z))$ in the second paragraph of the proof of the proposition, then in fact Z is the maximal configuration for the execution e'. Thus, we have the

Corollary 2.4.3: *Consider a protocol that is wait-free executable on a complex $\mathcal{K}$ of input configurations. A configuration Z in $\mathcal{P}(\mathcal{K})$ is maximal for some execution e with X_e a simplex of $\mathcal{K}$ if and only if* $\mathrm{ids}(Z) = \mathrm{ids}(Y(Z))$.

2.5 Theorems of Herlihy–Shavit.

Throughout this subsection, we assume that the protocol considered is wait-free executable on $\mathcal{K}$. In citing and whenever using the results of Herlihy and

Shavit, we assume without further remark that the protocol involved takes the normalized full-information form assumed by those authors. Roughly speaking, in a full-information protocol each process accumulates in private memory the entire history of views that it has received from scans of shared memory, and in each write to shared memory a process includes a copy of the history it has accumulated up to that point. It would be interesting to investigate the extent to which their results depend upon this assumption. A key theorem behind the work in [HS1, HS2, HS3] is the following

Theorem on Spans 2.5.1 ([HS1]): *Let $\mathcal{K}$ be a complex of input configurations, and consider a protocol that is wait-free executable on $\mathcal{K}$. Then there exists a chromatic subdivision $\chi(\mathcal{K})$ and a chromatic simplicial map $\varphi\colon \chi(\mathcal{K}) \to \mathcal{P}(\mathcal{K})$ satisfying $\varphi(S) \in \mathcal{P}(\mathrm{carrier}(S))$ for all simplices S in $\chi(\mathcal{K})$.*

Following Herlihy and Shavit, we call the map φ given by the theorem a *span* associated to the protocol on $\mathcal{K}$. Let S be a simplex of $\chi(\mathcal{K})$, so that $\varphi(S)$ is a configuration of the protocol, and let e be an execution such that $X_e \subseteq \mathrm{carrier}(S)$ and $\varphi(S)$ is derivable from e. Then $X(\varphi(S)) \subseteq X_e$ and $Y(\varphi(S)) \subseteq Y_e$. Since φ is chromatic, $\mathrm{ids}(S) = \mathrm{ids}(\varphi(S))$, and so

$$\mathrm{ids}(S) \subseteq \mathrm{ids}(Y(\varphi(S))) \subseteq \mathrm{ids}(X(\varphi(S))) \subseteq \mathrm{ids}(X_e) \subseteq \mathrm{ids}(\mathrm{carrier}(S)) \ .$$

If S happens to be of full dimension in $\mathrm{carrier}(S)$, then $\mathrm{ids}(S) = \mathrm{ids}(\mathrm{carrier}(S))$, and so each inclusion above is an equality. In this case, it follows that $X(\varphi(S)) = \mathrm{carrier}(S)$ and that $\varphi(S)$ is the maximal configuration for some execution e' with $X_{e'} = \mathrm{carrier}(S)$.

If $\mathcal{K}$ is a chromatic complex and $\chi(\mathcal{K})$ is a chromatic subdivision, then there is a unique chromatic simplicial map $f_\chi\colon \chi(\mathcal{K}) \to \mathcal{K}$ satisfying $f_\chi(S) \subseteq \mathrm{carrier}(S)$ for every simplex S in $\chi(\mathcal{K})$, and $|f_\chi|$ is homotopic to the identity map when we identify $|\chi(\mathcal{K})|$ with $|\mathcal{K}|$. (See Subsection A.6.) If $\varphi\colon \chi(\mathcal{K}) \to \mathcal{P}(\mathcal{K})$ is a span, then $\psi \circ \varphi\colon \chi(\mathcal{K}) \to \mathcal{K}$ is chromatic simplicial and satisfies $\psi \circ \varphi(S) \subseteq \mathrm{carrier}(S)$ for every simplex $S \in \chi(\mathcal{K})$. Thus,

$$\psi \circ \varphi = f_\chi \ ,$$

and $|\varphi|$ is a right homotopy inverse for $|\psi|$.

Following [HS2], we define $\mu = \delta \circ \varphi$. In general $\delta(Z) \subseteq Y(Z)$ for any configuration Z of the protocol, and, according to Corollary 2.4.3, equality holds if and only if Z is maximal for some execution. Thus, it follows that $\mu(S) \subseteq Y(\varphi(S))$, with equality if and only if S is of the same dimension as $\mathrm{carrier}(S)$. From Proposition 2.4.1 we also have

$$(X(\varphi(S)), Y(\varphi(S))) \in \Delta_{\mathrm{ex}}(\mathcal{K}) \ .$$

Since $\Delta_{\mathrm{ex}}(\mathcal{K})$ is extensible, there is an extension Y of $Y(\varphi(S))$, and hence also of $\mu(S)$, so that

$$(\mathrm{carrier}(S), Y) \in \Delta_{\mathrm{ex}}(\mathcal{K}) \ .$$

If S has the same dimension as $\mathrm{carrier}(S)$, then

$$(\mathrm{carrier}(S), \mu(S)) \in \Delta_{\mathrm{ex}}(\mathcal{K}) \ .$$

The following corollary, the necessary condition of the Asynchronous Computability Theorem, now follows from the Theorem on Spans and the preceding discussion.

Corollary 2.5.2 ([HS2]): *Let $\mathcal{K}$ be a complex of input configurations, and consider a protocol that is wait-free executable on $\mathcal{K}$. Then there exist a chromatic subdivision $\chi(\mathcal{K})$ and a chromatic simplicial map $\mu\colon \chi(\mathcal{K}) \to \mathcal{O}_{\text{ex}}(\mathcal{K})$ such that for each simplex S in $\chi(\mathcal{K})$ there exists an extension Y of $\mu(S)$ such that $(\text{carrier}(S), Y) \in \Delta_{\text{ex}}(\mathcal{K})$. If S has the same dimension as $\text{carrier}(S)$, then $(\text{carrier}(S), \mu(S)) \in \Delta_{\text{ex}}(\mathcal{K})$.*

In order to obtain the sufficient condition of the Asynchronous Computability Theorem, one uses Borowsky and Gafni's Participating Set Protocol for the simplex agreement task [BG1] as a "universal" protocol. The existence of an iterated standard chromatic subdivision $\chi(\mathcal{I})$ and a simplicial map $\mu\colon \chi(\mathcal{I}) \to \mathcal{O}$ respecting the task relation Δ enables specialization of the universal protocol to solve $(\mathcal{I}, \mathcal{O}, \Delta)$. There is a technical argument involved in allowing $\chi(\mathcal{I})$ to be an arbitrary chromatic subdivision; see [BG3, HS3].

Theorem 2.5.3 ([HS2]): *Let $(\mathcal{I}, \mathcal{O}, \Delta)$ be a well-posed, extensible, and strongly reducible decision task. Suppose that there is a chromatic subdivision $\chi(\mathcal{I})$ and a chromatic simplicial map $\mu\colon \chi(\mathcal{I}) \to \mathcal{O}$ such that $(\text{carrier}(S), \mu(S)) \in \Delta$ for every simplex S of $\chi(\mathcal{I})$. Then $(\mathcal{I}, \mathcal{O}, \Delta)$ has a wait-free solution.*

3. The Task Complex

We define a simplicial complex from the input-output relation Δ of a well-posed decision task. Our construction is quite simple and loses information about Δ. In particular, the construction does not discriminate between Δ and the relation Δ' obtained by extending Δ to be strongly reducible. More careful constructions that better reflect the conditions imposed by Δ may be possible.

3.1 Definition and simple examples.

An input-output pair (X, Y) will be called *matched* if $\text{ids}(X) = \text{ids}(Y)$. We can think of a matched pair as a single simplex T as follows. For each p in $\text{ids}(X)$, form a triple $t = (p, v, w)$, where $(p, v) \in X$ and $(p, w) \in Y$. Then let T be the set of triples t formed in this fashion from (X, Y). T can be thought of as a chromatic simplex with $\text{ids}(T) = \text{ids}(X) = \text{ids}(Y)$. For t as above, we write $p = \text{id}(t)$, $v = \text{inval}(t)$, and $w = \text{outval}(t)$. Having said carefully how to form the simplex T from a matched pair (X, Y), we will typically identify (X, Y) with T whenever convenient.

Definition 3.1.1: Let Δ be the input-output relation of a well-posed decision task. The *task complex* $\mathcal{T}$ associated to Δ is the chromatic complex consisting of all matched pairs (X, Y) for which there exist extensions X' of X and Y' of Y such that $(X', Y') \in \Delta$.

Let $\mathcal{I}$ and $\mathcal{O}$ be the input and output complexes associated to Δ. Then there are chromatic simplicial maps

$$\alpha\colon \mathcal{T} \to \mathcal{I} \quad \text{and} \quad \omega\colon \mathcal{T} \to \mathcal{O}$$

defined by

$$\alpha(t) = (\text{id}(t), \text{inval}(t)) \quad \text{and} \quad \omega(t) = (\text{id}(t), \text{outval}(t)) \ .$$

Notice that $\alpha(X,Y) = X$ and $\omega(X,Y) = Y$. If Δ' is the relation of another well-posed task such that $\Delta \subseteq \Delta'$, then $\mathcal{I} \subseteq \mathcal{I}'$, $\mathcal{O} \subseteq \mathcal{O}'$, and $\mathcal{T} \subseteq \mathcal{T}'$, all in the obvious way.

EXAMPLE 3.1.2. *Two-process binary consensus.* Recall the input-output relation Δ from Example 2.2.2. Each matched pair from Δ is itself a simplex of $\mathcal{T}$. These pairs are

$$\begin{array}{llllll} \langle 0*, 0* \rangle & \langle 1*, 1* \rangle & \langle *0, *0 \rangle & \langle *1, *1 \rangle & & \\ \langle 00, 00 \rangle & \langle 11, 11 \rangle & \langle 01, 00 \rangle & \langle 01, 11 \rangle & \langle 10, 00 \rangle & \langle 10, 11 \rangle \end{array}$$

In addition, $\mathcal{T}$ has the following matched pairs that are not in Δ but that are extensible to pairs in Δ:

$$\langle 0*, 1* \rangle \qquad \langle 1*, 0* \rangle \qquad \langle *0, *1 \rangle \qquad \langle *1, *0 \rangle$$

Such pairs correspond to decisions by a single process that are allowed provided the other process participates effectively and with the appropriate input value. For example, $\langle 0*, 1* \rangle$ is an unacceptable input-output pair if q does not participate effectively, since p is not allowed to decide 1 when the only effective input, p's own, is 0. This pair is acceptable, however, if q writes input value 1 before the final scan of p and then, perhaps, q crashes and p finishes with output value 1.

$|\mathcal{T}|$ separates into two components according to the output values of the simplices. Each component is the realization of a three-edge path. Thus, $|\mathcal{T}|$ is homotopy equivalent to a two-point space, as is $|\mathcal{O}|$, and $|\omega|\colon |\mathcal{T}| \to |\mathcal{O}|$ is a homotopy equivalence. On the other hand, $|\mathcal{I}|$ is homeomorphic to a 1-sphere, and the map $|\alpha|\colon |\mathcal{T}| \to |\mathcal{I}|$ is homotopic to a constant map.

EXAMPLE 3.1.3. *Two-process binary almost-consensus.* In addition to the matched pairs given in the preceding example, the task complex for binary almost-consensus has the 1-simplices $\langle 01, 10 \rangle$ and $\langle 10, 10 \rangle$. As a result, $|\mathcal{T}|$ is connected, and in fact $|\alpha|\colon |\mathcal{T}| \to |\mathcal{I}|$ is a homotopy equivalence. Since $|\mathcal{O}|$ is now contractible to a point, $|\omega|$ is homotopic to a constant map.

3.2 Obstructions.

Consider a protocol that is wait-free executable on a complex $\mathcal{K}$ of inputs. We write $\mathcal{T}_{\mathrm{ex}}(\mathcal{K})$ for the task complex associated to $\Delta_{\mathrm{ex}}(\mathcal{K})$. If z is a vertex of $\mathcal{P}(\mathcal{K})$, representing a process and final view from some execution, we can define

$$\vartheta\colon z \mapsto (\mathrm{id}(z), \mathrm{inval}(z), \mathrm{outval}(z)) \ .$$

If Z is a simplex of $\mathcal{P}(\mathcal{K})$, then $\vartheta(Z)$ is the matched pair $(\psi(Z), \delta(Z))$. Since $(\psi(Z), \delta(Z))$ is extensible to $(X(Z), Y(Z)) \in \Delta_{\mathrm{ex}}(\mathcal{K})$, it follows that $(\psi(Z), \delta(Z))$ is a simplex of $\mathcal{T}_{\mathrm{ex}}(\mathcal{K})$. Thus, ϑ defines a chromatic simplicial map from $\mathcal{P}(\mathcal{K})$ to $\mathcal{T}_{\mathrm{ex}}(\mathcal{K})$ such that the following diagrams commute:

$$\begin{array}{ccc} \mathcal{P}(\mathcal{K}) & \xrightarrow{\vartheta} & \mathcal{T}_{\mathrm{ex}}(\mathcal{K}) \\ \downarrow{\scriptstyle \psi} & & \downarrow{\scriptstyle \alpha} \\ \mathcal{K} & \xrightarrow{=} & \mathcal{K} \end{array} \quad , \qquad \begin{array}{ccc} \mathcal{P}(\mathcal{K}) & \xrightarrow{\vartheta} & \mathcal{T}_{\mathrm{ex}}(\mathcal{K}) \\ \downarrow{\scriptstyle \delta} & & \downarrow{\scriptstyle \omega} \\ \mathcal{O}_{\mathrm{ex}}(\mathcal{K}) & \xrightarrow{=} & \mathcal{O}_{\mathrm{ex}}(\mathcal{K}) \end{array} \quad .$$

Now suppose that the protocol is a wait-free solution of the well-posed decision task $(\mathcal{I}, \mathcal{O}, \Delta)$ with task complex $\mathcal{T}$. Then $\Delta_{\mathrm{ex}}(\mathcal{I}) \subseteq \Delta$, and so $\mathcal{T}_{\mathrm{ex}}(\mathcal{I}) \subseteq \mathcal{T}$ and $\mathcal{O}_{\mathrm{ex}}(\mathcal{I}) \subseteq \mathcal{O}$ as subcomplexes. We therefore obtain commutative diagrams

$$\begin{array}{ccc} \mathcal{P}(\mathcal{I}) & \xrightarrow{\vartheta} & \mathcal{T} \\ \downarrow{\scriptstyle\psi} & & \downarrow{\scriptstyle\alpha} \\ \mathcal{I} & \xrightarrow{=} & \mathcal{I} \end{array}\quad , \qquad \begin{array}{ccc} \mathcal{P}(\mathcal{I}) & \xrightarrow{\vartheta} & \mathcal{T} \\ \downarrow{\scriptstyle\delta} & & \downarrow{\scriptstyle\omega} \\ \mathcal{O}_{\text{ex}}(\mathcal{I}) & \xrightarrow{\subseteq} & \mathcal{O} \end{array}\quad .$$

Applying the Theorem on Spans, there is a commutative diagram

$$\begin{array}{ccccc} \chi(\mathcal{I}) & \xrightarrow{\varphi} & \mathcal{P}(\mathcal{I}) & \xrightarrow{\vartheta} & \mathcal{T} \\ \downarrow{\scriptstyle f_\chi} & & \downarrow{\scriptstyle\psi} & & \downarrow{\scriptstyle\alpha} \\ \mathcal{I} & \xrightarrow{=} & \mathcal{I} & \xrightarrow{=} & \mathcal{I} \end{array}\quad . \tag{3.2.1}$$

Thus, $\alpha \circ \vartheta \circ \varphi = f_\chi$, and since $|f_\chi|$ is homotopic to the identity, $|\vartheta \circ \varphi|$ is a right homotopy inverse for $|\alpha|$. The following theorem is now immediate.

Theorem 3.2.2: *Let $(\mathcal{I}, \mathcal{O}, \Delta)$ be a well-posed decision task that admits a wait-free solution, and let $\mathcal{T}$ be the associated task complex.*

(1) *Let $\mathfrak{F}_*$ be a covariant functor on the category of finite simplicial complexes and simplicial maps, and assume that $\mathfrak{F}_*$ transforms any simplicial map whose realization is a homotopy equivalence into a surjection. Then the morphism $\mathfrak{F}_*(\alpha)\colon \mathfrak{F}_*(\mathcal{T}) \to \mathfrak{F}_*(\mathcal{I})$ is a surjection.*

(2) *Let $\mathfrak{F}^*$ be a contravariant functor on the category of finite simplicial complexes and simplicial maps, and assume that $\mathfrak{F}^*$ transforms any simplicial map whose realization is a homotopy equivalence into an injection. Then the morphism $\mathfrak{F}^*(\alpha)\colon \mathfrak{F}^*(\mathcal{I}) \to \mathfrak{F}^*(\mathcal{T})$ is an injection.*

Note that the conditions on the covariant functor are satisfied when $\mathfrak{F}_*$ is simplicial homology or the ordinary homotopy group functor; the conditions on the contravariant functor are satisfied when $\mathfrak{F}^*$ is simplicial cohomology.

By an *obstruction* to wait-free computability of a well-posed task $(\mathcal{I}, \mathcal{O}, \Delta)$, we mean a mathematical object whose existence or "non-triviality" implies that no wait-free solution for the task can exist. Non-trivial usually means non-zero. An obstruction is (*effectively*) *computable* if it is (effectively) computable from $(\mathcal{I}, \mathcal{O}, \Delta)$.

Effectively computable obstructions to wait-free computability can be obtained by choosing $\mathfrak{F}_*$ to be simplicial homology with field coefficients. When the coefficients are understood, it is customary to write $\mathfrak{F}_*(\mathcal{K}) = H_*(\mathcal{K})$ and $\mathfrak{F}_*(\alpha) = \alpha_*$. If a wait-free solution for the task exists, then

$$\alpha_*\colon H_*(\mathcal{T}) \to H_*(\mathcal{I})$$

must surject, and so any element of $H_*(\mathcal{I})$ not in the image $\operatorname{im}\alpha_*$ is an obstruction to wait-free computability. Equivalently, any non-zero element of the quotient

$$\operatorname{coker}\alpha_* = H_*(\mathcal{I}) \Big/ \operatorname{im}\alpha_*$$

is an obstruction.

In the simple examples below, we use simplicial homology with coefficients in the field $\mathbb{F}$.

EXAMPLE 3.2.3. *Two-process binary consensus.* From Example 3.1.2, the map $|\alpha|:|\mathcal{T}| \to |\mathcal{I}|$ is homotopic to a constant map, and therefore the map $\alpha_*: H_*(\mathcal{T}) \to H_*(\mathcal{I})$ induced in homology is zero in dimension one. But $H_1(\mathcal{I}) \cong \mathbb{F}$, and so a generator of this homology is an obstruction to wait-free computability.

EXAMPLE 3.2.4. *Two-process binary almost-consensus.* From Example 3.1.3, the map $|\alpha|:|\mathcal{T}| \to |\mathcal{I}|$ is a homotopy equivalence, and therefore $\alpha_*: H_*(\mathcal{T}) \to H_*(\mathcal{I})$ is an isomorphism, hence a surjection. Thus, there are no obstructions to wait-free computability. Consider the subcomplex $\mathcal{T}'$ of $\mathcal{T}$ consisting of the 1-simplices

$$\langle 00, 00\rangle \qquad \langle 10, 10\rangle \qquad \langle 11, 11\rangle \qquad \langle 01, 11\rangle \qquad \langle 01, 10\rangle \qquad \langle 01, 00\rangle$$

and their vertices. $\mathcal{T}'$ is isomorphic to a subdivision $\chi(\mathcal{I})$. It is not difficult to see that an isomorphism $f: \chi(\mathcal{I}) \to \mathcal{T}'$ can be chosen so that $\mu = \omega \circ f$ satisfies the hypotheses of Theorem 2.5.3, and so this task admits wait-free solution. (See also [HS2], where an explicit protocol is given.)

Less trivial applications of the obstruction method are given in Section 5.

4. Symmetric Complexes and Anonymous Protocols

In this section, complexes of configurations are enhanced by adding the action of a group of symmetries that arise by permuting process identifiers. The objective is to treat anonymous wait-free protocols. Theorem 4.2.1 generalizes the Theorem on Spans and Theorem 4.2.2 generalizes Theorem 3.2.2 to this situation. Only the renaming application in Subsection 5.3 uses the ideas from this section.

4.1 Group actions on complexes.

Let $\mathcal{K}$ be a finite simplicial complex, and let G be a group of permutations of the vertices of $\mathcal{K}$. If $X = \{x_0, \ldots, x_r\}$ is a simplex of $\mathcal{K}$ and $g \in G$, then we write gX for the set $\{gx_0, \ldots, gx_r\}$. We say that G *acts simplicially* on $\mathcal{K}$, and that $\mathcal{K}$ is a *G-complex*, if for every $g \in G$ and every simplex X of $\mathcal{K}$, gX is also a simplex of $\mathcal{K}$. In this case, $g: \mathcal{K} \to \mathcal{K}$ is a simplicial automorphism for every $g \in G$, and G can be thought of as a group of symmetries of $\mathcal{K}$. The *orbit* of a simplex X under the action of G is the set of simplices $GX = \{gX : g \in G\}$. Suppose $\mathcal{K}$ and $\mathcal{L}$ are both G-complexes. A simplicial map $f: \mathcal{K} \to \mathcal{L}$ is a *G-equivariant map*, or simply a *G-map*, if $f(gX) = gf(X)$ for all $g \in G$ and $X \in \mathcal{K}$. The action of G on $\mathcal{K}$ can be extended piecewise-linearly to an action of G on the realization $|\mathcal{K}|$. A G-map $f: \mathcal{K} \to \mathcal{L}$ then gives rise to a G-equivariant piecewise linear map $|f|:|\mathcal{K}| \to |\mathcal{L}|$. A homotopy $F: |\mathcal{K}| \times [0,1] \to |\mathcal{L}|$ is called *G-equivariant* if for each fixed value $t_0 \in [0,1]$ of the homotopy parameter, $F(-, t_0): |\mathcal{K}| \to |\mathcal{L}|$ is a G-equivariant map.

For the present work, it is useful to focus attention on a restricted class of actions that induce no non-trivial self-maps of simplices. We say that G acts *rigidly* on $\mathcal{K}$, or that $\mathcal{K}$ is a *rigid* G-complex, if for any $g \in G$ and any simplex $X = \{x_0, \ldots, x_r\}$ of $\mathcal{K}$, $gX = X$ implies that $gx_i = x_i$ for $0 \le i \le r$. In other words, if g maps X to itself, then it does so by the identity map. Assume that G acts rigidly and that $g, h \in G$ are such that $gX = Y = hX$. Then $g^{-1}hX = X$, so that $g^{-1}h$ restricts to the identity map $X \to X$. It follows that each of g and h restricts to the same

simplicial map $X \to Y$. Thus, for any simplex Y in the orbit GX, there is a *unique* simplicial map $X \to Y$ induced by every $g \in G$ that maps X to Y.

Lemma 4.1.1: *Let $\mathcal{K}$ and $\mathcal{L}$ be G-complexes and let $f\colon \mathcal{K} \to \mathcal{L}$ be a G-map such that $\dim f(X) = \dim X$ for every simplex X in $\mathcal{K}$. If $\mathcal{L}$ is rigid, then $\mathcal{K}$ is rigid.*

Proof: Suppose $gX = X$, where $g \in G$ and $X = \{x_0, \ldots, x_r\}$ is a simplex of $\mathcal{K}$. Then $f(X) = f(gX) = gf(X)$. Since $\mathcal{L}$ is rigid, it follows that the restriction of g to $f(X) = \{f(x_0), \ldots, f(x_r)\}$ is the identity. Thus $gf(x_i) = f(x_i)$, and hence $f(gx_i) = f(x_i)$, for $0 \le i \le r$. The dimension condition ensures that $f(x_0), \ldots, f(x_r)$ are distinct, and it follows that $gx_i = x_i$ for $0 \le i \le r$. $\square$

Notice that the dimension condition of the lemma is satisfied whenever $\mathcal{K}$, $\mathcal{L}$, and f are all chromatic.

Let $\mathcal{K}$ be a G-complex. We say that a subdivision $\sigma(\mathcal{K})$ is a *G-subdivision* if $\sigma(\mathcal{K})$ is a G-complex and $\operatorname{carrier}(gS) = g \operatorname{carrier}(S)$ for any $g \in G$ and S in $\sigma(\mathcal{K})$. Because of its uniformity, a barycentric or standard chromatic subdivision of a G-complex can be given the structure of a G-subdivision. If $\mathcal{K}$ is a rigid G-complex, then G-subdivisions can be generated inductively on the skeleta of $\mathcal{K}$ as follows. There is no subdivision on the 0-skeleton. Suppose X is a simplex of $\mathcal{K}$ that has not yet been subdivided and that the lower dimensional skeleta have been G-subdivided. Choose any subdivision of X that is consistent with the subdivision on the boundary of X. By rigidity, for each Y in the orbit GX, there is a unique simplicial map $X \to Y$ induced by elements of G, and so the subdivision of X gives a well-defined subdivision of Y. This subdivision of Y is consistent with the subdivision on the boundary of Y because the lower skeleta are G-subdivided. It is not difficult to see that a subdivision constructed in this fashion is itself a rigid G-complex.

Lemma 4.1.2: *Let $\mathcal{K}$ be a finite rigid G-complex, and let $\sigma(\mathcal{K})$ be a G-subdivision. Then there exists a G-map $f\colon \sigma(\mathcal{K}) \to \mathcal{K}$ such that $f(S) \subseteq \operatorname{carrier}(S)$ for every simplex S in $\sigma(\mathcal{K})$. For any such f, the map $|f|\colon |\sigma(\mathcal{K})| \to |\mathcal{K}|$ is G-equivariantly homotopic to the identity map $|\sigma(\mathcal{K})| \to |\mathcal{K}|$ relative to the vertices of $\mathcal{K}$.*

Proof: We can assume $|\sigma(\mathcal{K})| = |\mathcal{K}|$ and that the vertex set of $\mathcal{K}$ is contained in the vertex set of $\sigma(\mathcal{K})$. A map f can be constructed as follows. First, let f map any vertex of $\mathcal{K}$ to itself. The definition of f is G-equivariant so far. Now pick any vertex x on which f has not yet been defined. Pick some vertex of $\operatorname{carrier}(x)$ and let f map x to this vertex. Then extend f to be G-equivariant over the orbit Gx by letting $f(gx) = gf(x)$. We need to check that this extension is well-defined. If $gx = hx$, then, since $\sigma(\mathcal{K})$ is a G-subdivision,

$$g \operatorname{carrier}(x) = \operatorname{carrier}(gx) = \operatorname{carrier}(hx) = h \operatorname{carrier}(x) \ .$$

By rigidity, g and h induce the same simplicial map on $\operatorname{carrier}(x)$. Since $f(x)$ is chosen in $\operatorname{carrier}(x)$, $gf(x) = hf(x)$, and this proves that the equivariant extension of f to Gx is well-defined. Continuing in this fashion, f is defined and G-equivariant on the entire vertex set of $\sigma(\mathcal{K})$. If S is a simplex of $\sigma(\mathcal{K})$, then for each vertex x of S, $\operatorname{carrier}(x)$ is a face of $\operatorname{carrier}(S)$. Since $f(x)$ is a vertex of $\operatorname{carrier}(x)$, $f(x)$ is also a vertex of $\operatorname{carrier}(S)$, and this ensures that $f(S)$ is a face of $\operatorname{carrier}(S)$. Thus, f is simplicial and satisfies the carrier requirement. Finally, since f is a simplicial map that is G-equivariant on vertices, it follows that f is G-equivariant on simplices.

For the homotopy, we let

$$F(x,t) = tx + (1-t)(|f|(x))$$

for $x \in |\sigma(\mathcal{K})|$ and $t \in [0,1]$. Note that x and $|f|(x)$ both lie in $|\text{carrier}(x)|$, and $F(x,-)$ is simply a parameterization of the line segment joining these two points. F is continuous and is therefore a homotopy from $|f|$ to the identity. Since $|f|$ maps vertices of $\mathcal{K}$ to themselves, F is relative to these vertices (i.e., leaves them fixed). Fix $t_0 \in [0,1]$, let $g \in G$, and let $x \in |\sigma(\mathcal{K})|$. Since $|f|$ is G-equivariant, we have $|f|(gx) = g|f|(x)$. Since x and $|f|(x)$ both lie in $|\text{carrier}(x)|$, gx and $g|f|(x)$ both lie in $g\,|\text{carrier}(x)| = |\text{carrier}(gx)|$. It follows that

$$F(gx,t_0) = t_0(gx) + (1-t_0)(g|f|(x)) = g\big(t_0 x + (1-t_0)(|f|(x))\big) = gF(x,t_0)\ .$$

Thus, F is a G-equivariant homotopy. □

If $\mathcal{K}$ is a rigid G-complex, then we can form a quotient space $\mathcal{K}/G$ by identifying points that can be mapped to one another by the action of the group.[9] The resulting space may not be a simplicial complex, but will always be a more general *cell complex* [Mu, §38]. If X is an r-simplex of $\mathcal{K}$, then every simplex in the orbit GX is an r-simplex that can be uniquely identified with X via a simplicial map induced by G. We let the cell complex $\mathcal{K}/G$ have one r-dimensional cell for each r-dimensional orbit. We write $[X]$ for the cell associated to the orbit GX, and so $[X] = [Y]$ if and only if $Y \in GX$. We think of $[X]$ as an r-simplex whose faces must be identified with (i.e., "glued" to) cells of lower dimension. The rigidity condition ensures that the identification of the faces is well-defined. The reason this construction may not yield a simplicial complex is that two distinct cells may share the same boundary, or a single cell may have several of its boundary faces glued together. With this construction there is a cellular projection map

$$\pi\colon \mathcal{K} \to \mathcal{K}/G$$

defined by $\pi(X) = [X]$.

Notice that if $f\colon \mathcal{K} \to \mathcal{L}$ is a G-map of rigid G-complexes, then f induces a cellular map

$$[f]\colon \mathcal{K}/G \to \mathcal{L}/G$$

defined by $[f]([X]) = [f(X)]$, and the following diagram commutes:

$$\begin{array}{ccc} \mathcal{K} & \xrightarrow{f} & \mathcal{L} \\ \downarrow{\scriptstyle \pi} & & \downarrow{\scriptstyle \pi} \\ \mathcal{K}/G & \xrightarrow{[f]} & \mathcal{L}/G \end{array}$$

If $F\colon |\mathcal{K}| \times [0,1] \to |\mathcal{L}|$ is a G-equivariant homotopy, then F induces a homotopy

$$[F]\colon \mathcal{K}/G \times [0,1] \to \mathcal{L}/G\ .$$

If $\mathcal{K}$ is rigid, if $\sigma(\mathcal{K})$ is a rigid G-subdivision, and if f is as in Lemma 4.1.2, then

$$[f]\colon \sigma(\mathcal{K})/G \to \mathcal{K}/G$$

is homotopic to the identity (when we identify $\sigma(\mathcal{K})/G$ with $\mathcal{K}/G$).

[9] To be precise, we should probably write $|\mathcal{K}|/G$, although we will always understand the notations $\mathcal{K}/G$ and $|\mathcal{K}|/G$ to mean the same space. Quotient spaces of this sort can be defined even if $\mathcal{K}$ is not rigid, but they may not be so well behaved.

4.2 Anonymous protocols.

We now consider group actions on complexes of configurations that arise by permuting processes. Let Σ be the symmetric group of all permutations of the $n+1$ processes. An element $g \in \Sigma$ acts on a configuration (input, output, protocol, or task) by permuting process identifiers. More precisely, if $x = (p, v)$ is a pair associating process and value, we define

$$gx = (gp, v) \ .$$

In other words, $\mathrm{id}(gx) = g\,\mathrm{id}(x)$ and $\mathrm{val}(gx) = \mathrm{val}(x)$. The same definition applies when val represents an inval, outval, or view. The definition also applies to triples: if $t = (p, v, w)$, then $gt = (gp, v, w)$. If (X, Y) is a pair of configurations satisfying $\mathrm{ids}(Y) \subseteq \mathrm{ids}(X)$, then $\mathrm{ids}(gY) \subseteq \mathrm{ids}(gX)$, so that (gX, gY) is also such a pair. We write $g(X, Y) = (gX, gY)$. This definition is consistent with the identification of a matched pair with a configuration of triples.

Let $\mathcal{K}$ be a complex of configurations and let Γ be a subgroup of Σ. We say that $\mathcal{K}$ is a *Γ-symmetric complex*, or, briefly, a *Γ-complex*, if the action of Γ on $\mathcal{K}$ is simplicial in the sense defined in the preceding section. In a similar fashion, we say that the input-output relation Δ is a Γ-relation if $g(X, Y) \in \Delta$ whenever $g \in \Gamma$ and $(X, Y) \in \Delta$.

It is straightforward to check that if $(\mathcal{I}, \mathcal{O}, \Delta)$ is a well-posed task and Δ is a Γ-relation, then $\mathcal{I}$, $\mathcal{O}$, and $\mathcal{T}$ are Γ-complexes. Furthermore, the maps α and ω are Γ-maps.

Now consider a protocol that is wait-free executable on the Γ-complex $\mathcal{K}$ of input configurations. We say that the protocol is *Γ-anonymous on $\mathcal{K}$* if it satisfies the following property of executions (compare with [HR1]).

(E6): *Let e be an execution with X_e a simplex of $\mathcal{K}$ and let $g \in \Gamma$. Then there is an execution ge such that* (1) *the full input configuration of ge is obtained from the full input configuration of e by the action of g, and* (2) *the event $(ge)_i$ is the same as e_i except that all processes are permuted by g.*

Note that (E6)(2) means not only that the process associated with the event is permuted, but, for example, that the permutation is also applied to the view returned from a scan. From (E6) it follows that $F_{ge} = gF_e$, $NF_{ge} = gNF_e$, $X_{ge} = gX_e$, and $Y_{ge} = gY_e$. It is not difficult to see that if the protocol is wait-free executable and Γ-anonymous on $\mathcal{K}$, then $\mathcal{P}(\mathcal{K})$ is a Γ-complex and $\Delta_{\mathrm{ex}}(\mathcal{K})$ is a Γ-relation, and hence $\mathcal{O}_{\mathrm{ex}}(\mathcal{K})$ and $\mathcal{T}_{\mathrm{ex}}(\mathcal{K})$ are Γ-complexes. Furthermore, ψ, δ, and ϑ are Γ-maps.

Let $\mathcal{K}$ be a Γ-complex of inputs and suppose that a protocol is wait-free executable and Γ-anonymous on $\mathcal{K}$. In order to complete the diagram 3.2.1 in a Γ-symmetric fashion, it remains to arrange a chromatic Γ-subdivision $\chi(\mathcal{K})$ and a span φ that is a chromatic Γ-map. We shall not pursue here the general conditions under which Γ-equivariant spans can be found, but, rather, remain content to show that they can be found when $\mathcal{K}$ is a rigid Γ-complex. A simple condition on $\mathcal{K}$ that ensures rigidity of the action of Γ is the following.

(R): *For any simplex $X = \{x_0, \ldots, x_r\}$ of $\mathcal{K}$, $\mathrm{val}(x_0), \ldots, \mathrm{val}(x_r)$ are distinct.*

Suppose that $\mathcal{K}$ is a rigid Γ-complex of inputs on which a protocol is wait-free executable and Γ-anonymous. Since chromatic maps preserve the dimensions of

simplices, applying Lemma 4.1.1 to the Γ-maps $\psi\colon \mathcal{P}(\mathcal{K}) \to \mathcal{K}$ and $\alpha\colon \mathcal{T}_{\mathrm{ex}}(\mathcal{K}) \to \mathcal{K}$ shows that each of $\mathcal{P}(\mathcal{K})$ and $\mathcal{T}_{\mathrm{ex}}(\mathcal{K})$ is a rigid Γ-complex. Similarly, if $(\mathcal{I}, \mathcal{O}, \Delta)$ is a well-posed task with Δ a Γ-relation, and if the action of Γ on $\mathcal{I}$ is rigid, then from $\alpha\colon \mathcal{T} \to \mathcal{I}$ we infer that the action of Γ on $\mathcal{T}$ is rigid.

We can now give the following generalization of the Theorem on Spans.

Theorem 4.2.1: *Let $\mathcal{K}$ be a rigid Γ-complex of inputs, and consider a protocol that is wait-free executable and Γ-anonymous on $\mathcal{K}$. Then there is a chromatic Γ-subdivision $\chi(\mathcal{K})$ that is a rigid Γ-complex and a Γ-equivariant span $\varphi\colon \chi(\mathcal{K}) \to \mathcal{P}(\mathcal{K})$.*

Proof: The reader is referred to the proof of the Theorem on Spans in [HS3, pp. 35–36]. The structure of the proof is to construct the subdivision $\chi(\mathcal{K})$ and the span φ inductively on the skeleta of $\mathcal{K}$. The topology of $\mathcal{P}(\mathcal{K})$ guarantees that once the subdivision and span have been defined on the boundary of a simplex X, they can be extended over X itself. We need only modify the argument, as in the discussion of G-subdivisions of a rigid G-complex and the proof of Lemma 4.1.2 in the preceding section, so that whenever we extend the subdivision and span to X, we also extend Γ-equivariantly to all simplices in the orbit ΓX. $\square$

Under the hypotheses of the theorem, we obtain the commutative diagram

$$\begin{array}{ccccc}
\chi(\mathcal{K}) & \xrightarrow{\varphi} & \mathcal{P}(\mathcal{K}) & \xrightarrow{\vartheta} & \mathcal{T}_{\mathrm{ex}}(\mathcal{K}) \\
\downarrow{\scriptstyle f_\chi} & & \downarrow{\scriptstyle \psi} & & \downarrow{\scriptstyle \alpha} \\
\mathcal{K} & \xrightarrow{=} & \mathcal{K} & \xrightarrow{=} & \mathcal{K}
\end{array}$$

of rigid Γ-complexes and Γ-maps. Passing to quotients, we obtain

$$\begin{array}{ccccc}
\chi(\mathcal{K})/\Gamma & \xrightarrow{[\varphi]} & \mathcal{P}(\mathcal{K})/\Gamma & \xrightarrow{[\vartheta]} & \mathcal{T}_{\mathrm{ex}}(\mathcal{K})/\Gamma \\
\downarrow{\scriptstyle [f_\chi]} & & \downarrow{\scriptstyle [\psi]} & & \downarrow{\scriptstyle [\alpha]} \\
\mathcal{K}/\Gamma & \xrightarrow{=} & \mathcal{K}/\Gamma & \xrightarrow{=} & \mathcal{K}/\Gamma
\end{array}$$

where $[f_\chi]$ is homotopic to the identity. If $(\mathcal{I}, \mathcal{O}, \Delta)$ is a well-posed task with Δ a Γ-relation, if $\mathcal{I}$ is a rigid Γ-complex, and if the protocol is a wait-free and Γ-anonymous solution, then we obtain the commutative diagram

$$\begin{array}{ccccc}
\chi(\mathcal{I}) & \xrightarrow{\varphi} & \mathcal{P}(\mathcal{I}) & \xrightarrow{\vartheta} & \mathcal{T} \\
\downarrow{\scriptstyle f_\chi} & & \downarrow{\scriptstyle \psi} & & \downarrow{\scriptstyle \alpha} \\
\mathcal{I} & \xrightarrow{=} & \mathcal{I} & \xrightarrow{=} & \mathcal{I}
\end{array}$$

of rigid Γ-complexes and Γ-maps. Passing to quotients gives

$$\begin{array}{ccccc}
\chi(\mathcal{I})/\Gamma & \xrightarrow{[\varphi]} & \mathcal{P}(\mathcal{I})/\Gamma & \xrightarrow{[\vartheta]} & \mathcal{T}/\Gamma \\
\downarrow{\scriptstyle [f_\chi]} & & \downarrow{\scriptstyle [\psi]} & & \downarrow{\scriptstyle [\alpha]} \\
\mathcal{I}/\Gamma & \xrightarrow{=} & \mathcal{I}/\Gamma & \xrightarrow{=} & \mathcal{I}/\Gamma
\end{array}$$

The following generalization of Theorem 3.2.2 is now immediate.

Theorem 4.2.2: *Let $(\mathcal{I},\mathcal{O},\Delta)$ be a well-posed decision task with Δ a Γ-relation, $\mathcal{I}$ a rigid Γ-complex, and $\mathcal{T}$ the associated task complex. Suppose also that the task admits a wait-free and Γ-anonymous solution.*

(1) *If $\mathfrak{F}_*$ is a covariant functor on the category of finite cellular complexes and cellular maps and if $\mathfrak{F}_*$ transforms homotopy equivalences into surjections, then the morphism $\mathfrak{F}_*([\alpha])\colon \mathfrak{F}_*(\mathcal{T}/\Gamma) \to \mathfrak{F}_*(\mathcal{I}/\Gamma)$ is a surjection.*
(2) *If $\mathfrak{F}^*$ is a contravariant functor on the category of finite cellular complexes and cellular maps and if $\mathfrak{F}^*$ transforms homotopy equivalences into injections, then the morphism $\mathfrak{F}^*([\alpha])\colon \mathfrak{F}^*(\mathcal{I}/\Gamma) \to \mathfrak{F}^*(\mathcal{T}/\Gamma)$ is an injection.*

As in Subsection 3.2, we can associate obstructions to the wait-free and Γ-anonymous computability of a task with Γ-symmetric input-output relation and rigid Γ-complex of inputs by choosing an appropriate functor $\mathfrak{F}_*$ or $\mathfrak{F}^*$.

5. Applications

In this section we give some non-trivial examples of well-posed decision tasks that have associated to them non-zero obstructions and therefore admit no wait-free solution. All of these tasks have been considered before in the literature, and none of the impossibility results below is new. Nor is any claim made that the proofs using obstructions are simpler than proofs given previously. The goal is, rather, to show that the obstruction method is powerful enough to yield interesting impossibility results.

For each of the examples below, obstructions will be obtained using homology with coefficients in the $\mathbb{F}$ of integers mod-2.

5.1 Consensus.

In the consensus task, each of the $n+1$ processes begins with an input value from the set $V_I = \{0,\dots,m\}$, where $m > 0$. All input configurations are allowed. All processes that finish the task are required to reach consensus in the sense that they all have the same output value. It is also required that the common output value of the finishing processes be the input value of at least one effectively participating process. The input-output relation Δ therefore consists of all pairs (X,Y) of input and output configurations such that $\mathrm{ids}(Y) \subseteq \mathrm{ids}(X)$, $\mathrm{vals}(X) \subseteq V_I$, and $\mathrm{vals}(Y) = \{v\}$ for some $v \in \mathrm{vals}(X)$.

We first explore the topology of the input complex. Fix $v \in V_I$ and let X_v denote the unique full input configuration in which every process has input value v. Let $\mathcal{S}_v$ be the closed star of X_v in $\mathcal{I}$. In other words, $\mathcal{S}_v$ is the subcomplex of $\mathcal{I}$ consisting of all configurations that have or can be extended to have v as a value. Also, let $\mathcal{L}_v$ denote the subcomplex of $\mathcal{I}$ consisting of all simplices that do *not* meet X_v. The simplices of $\mathcal{L}_v$ are precisely those configurations X for which $v \notin \mathrm{vals}(X)$. Notice that $\mathcal{L}_v$ has m^{n+1} full simplices. Also, notice that

$$\mathcal{S}_v \cap \mathcal{L}_v = \mathcal{L}_v^{(n-1)} ,$$

where $\mathcal{L}_v^{(n-1)}$ denotes the $(n-1)$-skeleton of $\mathcal{L}_v$.

The realization $|\mathcal{I}|$ can be deformed to a homotopy equivalent space $\mathfrak{I}$ by contracting $|\mathcal{S}_v|$ to a point. The deformation can be defined in two stages. In the first stage, all configurations that have *only* the value v are contracted to a single base point, b. This amounts to contracting $|X_v|$ to the point b, which we can think of as the barycenter of $|X_v|$. Simplices of $\mathcal{L}_v$ are unaffected by the first stage of the deformation. The realization of a simplex X in $\mathcal{S}_v$ that is not a face of X_v is deformed as follows. X can be written uniquely as a join

$$X = X_0 * X_1 \ ,$$

where X_0 is a face of X_v and X_1 is a simplex of $\mathcal{L}_v^{(n-1)}$. Since $|X_0|$ is deformed to the point b, $|X|$ is deformed to

$$|\{b\} * X_1| \ .$$

The result of the first stage of deformation is again the realization of a simplicial complex, $\mathcal{I}'$, and the structure of $\mathcal{I}'$ is

$$\mathcal{I}' = \{b\} * \mathcal{L}_v^{(n-1)} \bigcup_{\mathcal{L}_v^{(n-1)}} \mathcal{L}_v \ .$$

In other words, $\mathcal{I}'$ is obtained from $\mathcal{L}_v$ by joining the $(n-1)$-skeleton $\mathcal{L}_v^{(n-1)}$ to the point b. In the second stage of deformation, we simply contract $|\{b\} * \mathcal{L}_v^{(n-1)}|$ to b. The result is the space

$$\mathfrak{I} = |\mathcal{I}| \Big/ |\mathcal{S}_v| \sim b = |\mathcal{L}_v| \Big/ |\mathcal{L}_v^{(n-1)}| \sim b \ .$$

But the last space, formed by collapsing the $(n-1)$-skeleton of an n-dimensional complex to a point, is a wedge of n-spheres, one for each n-simplex of the original complex. Thus,

$$\mathfrak{I} = \bigvee_1^N S^n \ ,$$

where we have written $N = m^{n+1}$. Each full input configuration from $\mathcal{L}_v$ gives rise to one of the n-spheres by collapsing its boundary to b.

The reduced homology of a wedge of spheres has one generator in dimension k for each k-sphere. Thus, we have

$$H_n(\mathcal{I}) \cong H_n(\mathfrak{I}) \cong \mathbb{F}^N \ .$$

It is worth spending a moment to describe homology generators in terms of the original complex $\mathcal{I}$. Let X be a full input configuration from $\mathcal{L}_v$, so that $v \notin \mathrm{vals}(X)$. X gives an n-sphere in $\mathfrak{I}$ by collapsing its boundary to b. In order to obtain the corresponding n-sphere in $\mathcal{I}$, which we denote by S_X, we need to "uncollapse" the boundary. Notice that each proper face X_1 of X determines a unique proper face X_0 of X_v such that $\mathrm{ids}(X_0) \cap \mathrm{ids}(X_1)$ is empty and $\mathrm{ids}(X_0) \cup \mathrm{ids}(X_1)$ is the set of all process identifiers. Then the join $X_0 * X_1$ is an n-simplex of $\mathcal{S}_v$. The union of all such n-simplices, together with X and X_v, gives the desired sphere S_X. The number of choices for X_1 is

$$\sum_{k=1}^{n} \binom{n+1}{k} = 2^{n+1} - 2 \ .$$

Thus, when we add X and X_v, we have 2^{n+1} n-simplices in S_X.

Next, we turn to the task complex. For each $v \in V_I$, let $\mathcal{T}_v$ denote the subcomplex of $\mathcal{T}$ consisting of matched pairs (X, Y) for which $\text{vals}(Y) = \{v\}$. Notice that $|\mathcal{T}|$ separates into connected components $|\mathcal{T}_v|$. Once v is understood, the configuration Y contributes no information to a matched pair (X, Y) of $\mathcal{T}_v$, and we see that $\mathcal{T}_v$ is isomorphic as a complex to $\mathcal{S}_v$. Since $|\mathcal{S}_v|$ can be contracted to a point, as in the construction of $\mathfrak{I}$, each component $|\mathcal{T}_v|$ can be contracted to a point, and so $|\mathcal{T}|$ is homotopy equivalent to a space consisting of $m+1$ points. In particular,

$$H_n(\mathcal{T}) = 0 \ .$$

This means that α_* must be the zero map in dimension n, and so all of the non-zero homology classes in $H_n(\mathcal{I})$ are obstructions to wait-free solution of the consensus task.

5.2 Set consensus.

The set consensus task of Chaudhuri [Ch] is a generalization of the consensus task. In the k-set consensus task, each of the $n+1$ processes begins with an input value from the set $V_I = \{0, \ldots, m\}$, where $m \geq n$. All input configurations are allowed. The finishing processes are required to choose output values from among the input values of the effectively participating processes, and the set of all output values of finishing processes is required to have no more than k elements. The input-output relation Δ therefore consists of all pairs (X, Y) of input and output configurations such that $\text{ids}(Y) \subseteq \text{ids}(X)$, $\text{vals}(Y) \subseteq \text{vals}(X) \subseteq V_I$, and $|\text{vals}(Y)| \leq k$.

The task just described is also referred to as $(m+1, k)$-consensus (see [HR1]). The task admits trivial solution for $k \geq n+1$, so we henceforth assume that $k \leq n$. It is easy to see that any solution to $(m+1, k)$-consensus for $m \geq n$ will implement a solution of $(n+1, k)$-consensus. Also, any solution to $(n+1, k)$-consensus for $k \leq n$ will implement a solution of $(n+1, n)$-consensus. Thus, we consider only the $(n+1, n)$-consensus task below. The case $n = 1$ reduces to consensus as discussed in the preceding subsection, so we assume $n \geq 2$.

It is known (see [SZ, BG2, HS1]) that the $(n+1, n)$-consensus task does not admit wait-free solution. At present, we do not have a proof that the impossibility of wait-free solution of $(n+1, n)$-consensus is detected by the obstruction method. Nevertheless, there seems to be good evidence that obstructions exist using homology with coefficients in $\mathbb{F}$. In this subsection, we show one way to study the homology map α_* and report results of automated computations that show the existence of obstructions to the wait-free solution of the $(3, 2)$-, $(4, 3)$-, and $(5, 4)$-consensus tasks.

The input complex for $(n+1, n)$-consensus is the same as the input complex for consensus described in the preceding subsection with $m = n$. Thus,

$$|\mathcal{I}| \simeq \mathfrak{I} = \bigvee_1^N S^n \ ,$$

where $N = n^{n+1}$. Obstructions can therefore exist only in dimension n, and we have

$$H_n(\mathcal{I}) \cong \mathbb{F}^N \ .$$

The output complex $\mathcal{O}$ can be viewed as the subcomplex of the input complex consisting of simplices X for which $|\text{vals}(X)| \leq n$. Notice that

$$\mathcal{I}^{(n-1)} \subset \mathcal{O} \ ,$$

and the simplices of $\mathcal{I} - \mathcal{O}$ are precisely the $(n+1)!$ full configurations in which the input values are distinct. Let us fix the process identifiers as $0, \ldots, n$. We can use the symmetric group Σ on $\{0, \ldots, n\}$ to index the full input configurations with distinct input values by writing

$$X^\sigma = \{(0, \sigma(0)), \ldots, (n, \sigma(n))\}$$

for $\sigma \in \Sigma$. It follows that

$$H_n(\mathcal{I}, \mathcal{O}) = C_n(\mathcal{I}, \mathcal{O}) \cong \mathbb{F}^{(n+1)!} \ ,$$

with basis the set $\{X^\sigma : \sigma \in \Sigma\}$.

Lemma 5.2.1: *The image of the projection map*

$$p : H_n(\mathcal{I}) \to H_n(\mathcal{I}, \mathcal{O})$$

is spanned by the set of all sums of the form $X^\sigma + X^\tau$, where $\sigma, \tau \in \Sigma$. $H_n(\mathcal{O})$ is of rank $n^{n+1} - (n+1)! + 1$; for any $\sigma \in \Sigma$, $\{\partial X^\sigma\}$ is a basis for $H_{n-1}(\mathcal{O})$, which is of rank one.

Proof: Since $H_{n-1}(\mathcal{I}) = 0$, the inclusion of $\mathcal{O}$ in $\mathcal{I}$ gives rise to a long exact sequence

$$0 \to H_n(\mathcal{O}) \to H_n(\mathcal{I}) \xrightarrow{p} H_n(\mathcal{I}, \mathcal{O}) \xrightarrow{\partial} H_{n-1}(\mathcal{O}) \to 0 \ . \qquad (*)$$

We first show that every sum of the form $X^\sigma + X^\tau$ is in the image of p. Since Σ is generated by transpositions, it suffices to prove the case when τ is obtained from σ by a transposition. To ease notation, we assume, without loss of generality, that $\sigma(i) = \tau(i)$ for $i \geq 2$, and we write

$$\sigma(0) = v = \tau(1) \quad \text{and} \quad \sigma(1) = w = \tau(0) \ ,$$

where $v \neq w$. Let

$$F = X^\sigma \cap X^\tau = \{(2, \sigma(2)), \ldots, (n, \sigma(n))\} \ ,$$

and let

$$X = \{(0, w), (1, w)\} * F \ .$$

Recall the notation from the discussion of consensus in the preceding subsection. We have $X \in \mathcal{L}_v$, and we can build an n-sphere S_X from X by joining faces of X to faces of X_v. Recall that S_X represents the homology class in $H_n(\mathcal{I})$ that is identified via isomorphism with the class in $H_n(\mathcal{J})$ obtained by collapsing the boundary of X to a point. Notice that each of X^σ and X^τ is a simplex in S_X, obtained as

$$X^\sigma = \{(0, v)\} * \big(\{(1, w)\} * F\big) \quad \text{and} \quad X^\tau = \{(1, v)\} * \big(\{(0, w)\} * F\big) \ .$$

Furthermore, every simplex of S_X apart from X^σ and X^τ has either the value v repeated or the value w repeated. Thus,

$$p : S_X \mapsto X^\sigma + X^\tau \ .$$

Now we show that the image of p is generated by sums of the form $X^\sigma + X^\tau$. Fix v and recall the notations from the discussion of consensus. A basis for $H_n(\mathcal{I})$ is given by the set of homology classes represented by spheres S_X, where X ranges over the full configurations from $\mathcal{L}_v$. If $|\mathrm{vals}(X)| < n$, then no simplex of S_X has distinct values. In this case, $S_X \subset \mathcal{O}$, and we have $p(S_X) = 0$. Otherwise, $\mathrm{vals}(X) = \{0, \ldots, n\} - \{v\}$, and X has one repeated value w. Then S_X is as in the preceding paragraph, and so $p(S_X) = X^\sigma + X^\tau$ for appropriately chosen σ and τ.

The statements about $H_n(\mathcal{O})$ and $H_{n-1}(\mathcal{O})$ are now immediate from the exact sequence $(*)$. $\square$

The task complex $\mathcal{T}$ consists of all matched pairs (X, Y) such that $\mathrm{vals}(Y) \subseteq \mathrm{vals}(X)$ and $|\mathrm{vals}(Y)| \leq n$. The topology of $\mathcal{T}$ is more complicated than in the case of consensus, and the dimension of $H_n(\mathcal{T})$ is large compared to that of $H_n(\mathcal{I})$. Many of the classes in $H_n(\mathcal{T})$ map to $H_n(\mathcal{O})$, though. In order to make this precise, we introduce a subcomplex $\mathcal{S} \subset \mathcal{T}$. A matched pair (X, Y) is in $\mathcal{S}$ if and only if it can be extended to a full matched pair $(X', Y') \in \Delta$ with $X' \in \mathcal{O}$. In this case, $\mathrm{vals}(X) \cup \mathrm{vals}(Y) \subseteq \mathrm{vals}(X')$ and $|\mathrm{vals}(X')| \leq n$, so a necessary condition for (X, Y) to be in $\mathcal{S}$ is

$$|\mathrm{vals}(X) \cup \mathrm{vals}(Y)| \leq n \ . \tag{A}$$

If, in addition to (A), we have

$$|\mathrm{vals}(Y) - \mathrm{vals}(X)| \leq n + 1 - |\mathrm{ids}(X)| \ , \tag{B}$$

then we can find a full extension X' of X satisfying $\mathrm{vals}(X') = \mathrm{vals}(X) \cup \mathrm{vals}(Y)$, and it follows that $(X, Y) \in \mathcal{S}$.

We let α' denote the restriction of α to $\mathcal{S}$, and we let α'' denote the map of pairs $(\mathcal{T}, \mathcal{S}) \to (\mathcal{I}, \mathcal{O})$ arising from α. Then we have the following commutative diagram with exact rows.

$$\begin{array}{ccccccc}
0 & \longrightarrow & H_n(\mathcal{S}) & \longrightarrow & H_n(\mathcal{T}) & \longrightarrow & H_n(\mathcal{T}, \mathcal{S}) \\
 & & \downarrow{\scriptstyle \alpha'_*} & & \downarrow{\scriptstyle \alpha_*} & & \downarrow{\scriptstyle \alpha''_*} \\
0 & \longrightarrow & H_n(\mathcal{O}) & \longrightarrow & H_n(\mathcal{I}) & \xrightarrow{p} & H_n(\mathcal{I}, \mathcal{O})
\end{array} \tag{5.2.2}$$

The map α'_* is actually a surjection. This follows because we can form an embedding

$$\beta : \mathcal{O} \to \mathcal{S}$$

by $\beta(Y) = (Y, Y)$. Thus, $\alpha' \circ \beta$ is the identity map on $\mathcal{O}$, and surjectivity of α'_* follows. From the diagram (5.2.2) we then have that the image of α_* contains $H_n(\mathcal{O})$ and, further, that α_* surjects only if the image of α''_* contains the image of p.

In fact, the behavior of α''_* is rather restricted, as we shall see in Proposition 5.2.3 below. We first make some observations about the relative homology $H_n(\mathcal{T}, \mathcal{S})$. The relative chain group $C_n(\mathcal{T}, \mathcal{S})$ has as basis the set of matched pairs of full configurations from $\mathcal{T}$ that can be written in the form (X^σ, Y). Since $\mathrm{vals}(X^\sigma)$ contains all input values, Y can be any configuration with $|\mathrm{vals}(Y)| \leq n$. It is not, however, true that $\partial(X^\sigma, Y)$ is a chain from $\mathcal{S}$, and so (X^σ, Y) itself is not a relative cycle.

In order to understand which faces of (X^σ, Y) are in $\mathcal{S}$ and which are not, we use the following face notation: for the configuration $X = \{(0, v_0), \ldots, (n, v_n)\}$ and for $0 \leq i \leq n$, we write

$$F_i X = X - \{(i, v_i)\} .$$

For a matched pair, $F_i(X, Y) = (F_i X, F_i Y)$. With this notation,

$$\partial(X^\sigma, Y) = (F_0 X^\sigma, F_0 Y) + \cdots + (F_n X^\sigma, F_n Y) .$$

Since $|\mathrm{vals}(F_i X^\sigma)| = n$, it follows that $(F_i X^\sigma, F_i Y)$ satisfies (B), and thus $(F_i X^\sigma, F_i Y)$ is in $\mathcal{S}$ if and only if

$$|\mathrm{vals}(F_i X^\sigma) \cup \mathrm{vals}(F_i Y)| \leq n . \qquad (\mathrm{A}')$$

Since $\mathrm{vals}(F_i X^\sigma)$ contains all values apart from $\sigma(i)$, (A′) holds if and only if

$$\sigma(i) \notin \mathrm{vals}(F_i Y) . \qquad (\mathrm{C})$$

Since $|\mathrm{vals}(Y)| \leq n$, (C) must hold for at least one i, $0 \leq i \leq n$, and so (X^σ, Y) has at least one face in $\mathcal{S}$. On the other hand, Y must have at least one repeated value, so (C) must fail for at least one i, $0 \leq i \leq n$. Thus, (X^σ, Y) has between 1 and n of its faces in $\mathcal{S}$. Notice also that if $\sigma \neq \tau$, then matched pairs of the form (X^σ, Y) and (X^τ, Y') cannot share a face of dimension $(n-1)$.

Consider a chain $T = T_1 + \cdots + T_r$ in $C_n(\mathcal{T}, \mathcal{S})$, where $T_j = (X^{\sigma_j}, Y_j)$. T is a relative cycle, and hence a homology class in $H_n(\mathcal{T}, \mathcal{S})$, if and only if the faces of the various T_j that do *not* lie in $\mathcal{S}$ cancel in pairs over the field $\mathbb{F}$. Suppose that T is a relative cycle. Since T_i and T_j can share a face of dimension $n-1$ only if they have the same input configuration, the chain

$$T^\sigma = \sum_{\sigma_j = \sigma} T_j$$

must itself be a relative cycle. We can therefore decompose T as

$$T = \sum_{\sigma \in \Sigma} T^\sigma ,$$

where each T^σ is a relative cycle.

Proposition 5.2.3: *The map α''_* in dimension n is either the zero map or a surjection on $H_n(\mathcal{I}, \mathcal{O})$. If α''_* is the zero map in dimension n, then α_* has image $H_n(\mathcal{O})$ in dimension n and therefore does not surject on $H_n(\mathcal{I})$.*

Proof: Suppose that α''_* is not the zero map. Then $\alpha''_*(T)$ is non-zero for some relative cycle $T \in H_n(\mathcal{T}, \mathcal{S})$. Decompose T as

$$T = \sum_{\sigma \in \Sigma} T^\sigma .$$

Then $\alpha''_*(T^\sigma)$ must be non-zero for some $\sigma \in \Sigma$. This means that X^σ is in the image of α''_*. But then by symmetry it follows that α''_* surjects on $H_n(\mathcal{I}, \mathcal{O})$.

The statement concerning α_* follows from the diagram (5.2.2). □

From the proof of the proposition and the preceding discussion, we see that in order to show that α''_* is zero in dimension n, it is enough to check that for a fixed choice of σ, any relative cycle T^σ as above has an even number of simplices. Such a computation is straightforward to automate, which we have done for $n = 2, 3$, and

4. In each case, we have found that α''_* is the zero map in dimension n, and therefore the non-zero classes in $H_n(\mathcal{I})/H_n(\mathcal{O})$ are obstructions to wait-free solution of the $(n+1,n)$-consensus task for these values of n. It seems reasonable to conjecture that α''_* is always the zero map in dimension n.

5.3 Renaming.

The renaming task was posed in [At+]. We follow the description of the task in [HR1]. The $n+1$ processes begin with distinct names from $V_I = \{0,\dots,N\}$, where $N > n$. The processes are required to choose distinct output names from $V_O = \{0,\dots,K\}$, where $n \le K < N$. The input-output relation Δ therefore consists of all pairs (X,Y) of input and output configurations such that $\mathrm{ids}(Y) \subseteq \mathrm{ids}(X)$, $\mathrm{vals}(X) \subset V_I$, $\mathrm{vals}(Y) \subseteq V_O$, and

$$|\mathrm{vals}(X)| = |\mathrm{ids}(X)| \quad \text{and} \quad |\mathrm{vals}(Y)| = |\mathrm{ids}(Y)| \ .$$

(The last equalities impose the distinctness requirements on the names.)

The task just described can be referred to more precisely as the (N,K) renaming task. In order to avoid trivial protocols, some condition of anonymity or comparison base with regard to process identifiers is appropriate. In [At+], the authors show that a wait-free comparison-based solution exists in the message-passing model when $K \ge 2n+1$, but not for $K \le n+2$. On the other hand, [HS1] shows that there is no wait-free comparison-based solution in the read-write model when $K \le 2n$. Anonymous solutions for renaming are considered in [HR1], while [HS3] treats comparison-based solutions.

In this subsection, we consider Σ-anonymous wait-free solutions for the $(n+1,n)$ renaming task with $n \ge 1$. Σ always denotes the symmetric group on the process identifiers. The choices $N = n+1$ and $K = n$ are special and have been made because they simplify the topology considerably. The distinctness requirements on the names ensure that condition (R) is satisfied by the input and output complexes. As pointed out in Section 4, it follows that the action of Σ on each of the complexes $\mathcal{I}$, $\mathcal{O}$, and $\mathcal{T}$ is rigid.

To simplify notation, we assume that the process identifiers are $0,\dots,n$. The output complex $\mathcal{O}$ has $(n+1)!$ full configurations, obtained by permuting the output values from $V_O = \{0,\dots,n\}$ among the processes. As in the preceding subsection, we use the elements of Σ to index these configurations. For $\sigma \in \Sigma$, we write

$$Y^\sigma = \{(0,\sigma(0)),\dots,(n,\sigma(n))\} \ .$$

For $\sigma \ne \tau$, the dimension of the intersection $Y^\sigma \cap Y^\tau$ is at most $n-2$, with equality if and only if τ is obtained from σ by a transposition. Notice that if Y is an output configuration, the orbit ΣY consists of all configurations Y' such that $\mathrm{vals}(Y') = \mathrm{vals}(Y)$. In particular, ΣY^σ contains all the full output configurations. It follows that the quotient space $\mathcal{O}/\Sigma$ is a realization of the simplicial complex of non-empty subsets of V_O, which is the complex of faces of a single n-simplex.

Next, we consider the input complex. We show first that $\mathcal{I}$ is a connected pseudo n-manifold [Mu, p. 261]. An $(n-1)$-dimensional input configuration X_1 determines the unique process $i \notin \mathrm{ids}(X_1)$ and the two values v,w of $V_I - \mathrm{vals}(X_1)$. It follows that X_1 can be extended to exactly two full input configurations, namely

$$\{(i,v)\} * X_1 \quad \text{and} \quad \{(i,w)\} * X_1 \ .$$

Now suppose that $X = \{(0, v_0), \ldots, (n, v_n)\}$ is a full input configuration and pick two processes, i and j. Let

$$F = X - \{(i, v_i), (j, v_j)\} ,$$

let w be the unique value in $V_I - \mathrm{vals}(X)$, and define

$$\Xi = \{(i, w), (j, v_j)\} * F , \quad \Xi' = \{(i, w), (j, v_i)\} * F , \quad \Xi'' = \{(i, v_j), (j, v_i)\} * F .$$

Then $X \cap \Xi$, $\Xi \cap \Xi'$, and $\Xi' \cap \Xi''$ are all $(n-1)$-simplices. Ξ'' results from X by applying the transposition from Σ that interchanges i and j. By repeating this procedure, it follows that if X' is in the orbit ΣX, then X can be connected to X' by a sequence of full input configurations such that successive configurations intersect in an $(n-1)$-simplex. In order to connect X similarly to a full configuration X' from a different orbit, we first connect X to some full configuration in $\Sigma X'$ by changing a single value in X, as by the sequence X, Ξ above. (Notice that $\Xi \in \Sigma X'$ if and only if $\mathrm{vals}(\Xi) = \mathrm{vals}(X')$, and this can be arranged by choosing (i, w) appropriately.) Then the preceding procedure can be used to find a sequence from Ξ to X'. This proves that $\mathcal{I}$ is a connected pseudo n-manifold. It follows [Mu, p. 262] that

$$H_n(\mathcal{I}) \cong \mathbb{F} ,$$

with generator the cycle that is the sum of all full input configurations.

Since the orbit of an input configuration X consists of all configurations X' such that $\mathrm{vals}(X') = \mathrm{vals}(X)$, it follows that the quotient space $\mathcal{I}/\Sigma$ is a realization of the simplicial complex of subsets $S \subset V_I$ satisfying $1 \leq |S| \leq n+1$. The non-empty subsets of V_I form the complex of faces of an $(n+1)$-simplex, and so $\mathcal{I}/\Sigma$ realizes the subcomplex of proper faces. $\mathcal{I}/\Sigma$ is therefore homeomorphic to an n-sphere. It follows that

$$H_n(\mathcal{I}/\Sigma) \cong \mathbb{F} ,$$

and the map π_*, induced by the projection map

$$\pi : \mathcal{I} \to \mathcal{I}/\Sigma ,$$

sends the generator of $H_n(\mathcal{I})$ to $(n+1)!$ times the generator of $H_n(\mathcal{I}/\Sigma)$. Since $(n+1)!$ is even, π_* is the zero map in dimension n.

The task complex $\mathcal{T}$ consists of all matched pairs (X, Y) such that $\mathrm{vals}(X) \subset V_I$, $\mathrm{vals}(Y) \subseteq V_O$, and $|\mathrm{vals}(X)| = |\mathrm{ids}(X)| = |\mathrm{ids}(Y)| = |\mathrm{vals}(Y)|$. Fix a full output configuration Y_0, and let

$$\mathfrak{Y} = \omega^{-1}(Y_0) ,$$

the subcomplex of $\mathcal{T}$ consisting of pairs whose output configuration is a face of Y_0. We define a simplicial map

$$f_\Sigma : \mathcal{T} \to \mathfrak{Y}$$

as follows. For any matched pair (X', Y') in $\mathcal{T}$, there is a unique matched pair (X, Y) with $Y \subseteq Y_0$ such that (X', Y') is in the orbit $\Sigma(X, Y)$. By rigidity, there is a unique simplicial map $(X', Y') \to (X, Y)$ induced by elements of Σ, and this map is the definition of f_Σ on (X', Y'). The uniqueness ensured by rigidity guarantees that the definition of f_Σ is consistent. Let π' denote the restriction of the projection $\pi : \mathcal{T} \to \mathcal{T}/\Sigma$ to $\mathfrak{Y}$. Since $\mathfrak{Y}$ contains exactly one simplex in each orbit of Σ, it follows that $\mathcal{T}/\Sigma$ is a a realization of a simplicial complex isomorphic to $\mathfrak{Y}$, and

π' determines this isomorphism. Furthermore, any pair (X, Y) in $\mathfrak{Y}$ is determined by its input configuration X. Writing α' for the restriction of α to $\mathfrak{Y}$, α' gives an isomorphism of complexes $\mathfrak{Y} \cong \mathcal{I}$. Summarizing, the following diagram commutes:

$$\begin{array}{ccccc} \mathcal{T} & \xrightarrow{f_\Sigma} & \mathfrak{Y} & \xrightarrow{\alpha'} & \mathcal{I} \\ \downarrow{\scriptstyle \pi} & & \downarrow{\scriptstyle \pi'} & & \downarrow{\scriptstyle \pi} \\ \mathcal{T}/\Sigma & \xrightarrow{=} & \mathcal{T}/\Sigma & \xrightarrow{[\alpha]} & \mathcal{I}/\Sigma \end{array}$$

We have shown already that $\pi{:}\mathcal{I} \to \mathcal{I}/\Sigma$ induces the zero map in n-dimensional homology. Since α' and π' are isomorphisms of complexes, it follows that $[\alpha]$ also induces the zero map in n-dimensional homology. Thus, the nonzero class in $H_n(\mathcal{I}/\Sigma)$ is an obstruction to Σ-anonymous wait-free solution of the $(n+1, n)$ renaming task.

Appendix: Topological Background

In this appendix, we attempt to give a quick introduction to the material from algebraic topology that has been used to study questions of computability in models of fault-tolerant distributed systems. We also highlight some facts that do not seem to have been cited before in this line of research, but which are needed in the present paper.

A.1 Topological spaces.

A *topological space* is a set X together with a specification of the family of subsets of X that are *open*. The family of open subsets of X is required to include X, the empty set, and to be closed under arbitrary union and finite intersection. Such a family of open sets is called a *topology* for X. Intuitively, the topology determines the way in which X is put together internally, what parts of X are "connected," a notion of "closeness" of points of X, and so on. A common way of generating a topology for X is through a distance function (or metric) on X. A distance function on X is a symmetric function d from $X \times X$ to the non-negative real numbers that assigns zero distance from a point to itself (i.e., $d(x, x) = 0$), assigns positive distance to a pair of distinct points, and satisfies the triangle rule:

$$d(x, z) \leq d(x, y) + d(y, z) \ .$$

From a distance function d, one can define the open ε-neighborhood of a point $x \in X$ to be

$$B_\varepsilon(x, X) = \{x' \in X{:}\, d(x, x') < \varepsilon\} \ ,$$

a generalization of open neighborhoods from calculus. One then defines a subset $Y \subseteq X$ to be open if and only if for every $y \in Y$ there is an $\varepsilon > 0$ such that

$$B_\varepsilon(y, X) \subseteq Y \ .$$

The topology formed in this way is called the *metric topology* for X arising from d. For example, any subset X of Euclidean space $\mathbb{R}^n$ can be topologized by restricting the usual Euclidean distance function to X. In this way, one can obtain topologies

for all the familiar hypersurfaces of multi-variable calculus. Common examples are the unit n-sphere (the set of unit vectors from $\mathbb{R}^{n+1}$), tori of various dimensions, projective spaces, and so on. Every topological space considered in this paper can be realized as a subset of an appropriate Euclidean space together with the metric topology just described.

A.2 Continuity, homeomorphism, and homotopy.

If X and Y are spaces with metric topologies, then continuity of a function $f\colon X \to Y$ is defined as in calculus: f is continuous if and only if

$$\lim_{x \to x_0} f(x) = f(x_0)$$

for every $x_0 \in X$. Continuity is usually the minimum requirement for a function between topological spaces to be of topological interest. A generalization of the Intermediate Value Theorem from calculus shows that continuous images of connected spaces are again connected. Similarly, a generalization of the Extremum Principle from calculus shows that continuous images of compact spaces are again compact. Continuous functions between topological spaces are often referred to as *continuous maps*, or simply *maps*. Compositions of continuous maps are again continuous, and the identity function from a space X to itself is continuous. A map $f\colon X \to Y$ is called a *homeomorphism* if it is bijective and the inverse $f^{-1}\colon Y \to X$ is also continuous. In this case X and Y are said to be *homeomorphic*. Homeomorphic spaces are regarded as topologically equivalent. For example, the surfaces of a tetrahedron, a cube, and a solid circular cone of finite height are all homeomorphic to the unit 2-sphere.

A weaker notion of equivalence of topological spaces can be obtained using continuous deformations, or *homotopies*. Two maps $f, g\colon X \to Y$ are *homotopic*, written $f \simeq g$, if there is a map

$$F\colon X \times [0,1] \to Y$$

satisfying $F(-,0) = f$ and $F(-,1) = g$. Such F is called a *homotopy* from f to g. Here $[0,1]$ denotes the unit interval of the real line, and the cartesian product $X \times [0,1]$ can be thought of as a cylinder of unit height and with cross section X. For each fixed $t \in [0,1]$, $F(-,t)\colon X \to Y$ is a map, and as t moves from 0 to 1, $F(-,t)$ deforms continuously from f to g. Notice that by taking $F(x,t) = f(x)$, it follows that f is homotopic to itself.

Example A.2.1. Let X be all of $\mathbb{R}^n$, let $f\colon X \to X$ by $f(x) = 0$, and let g be the identity map of X. Then $f \simeq g$ via the homotopy

$$F(x,t) = tx \ .$$

Example A.2.2. Let X be the space of non-zero vectors of $\mathbb{R}^n$, let $f\colon X \to X$ by $f(x) = x/|x|$, the map sending a vector x to the unit vector in the same direction as x, and let g be the identity map of X. Then $f \simeq g$ via the homotopy

$$F(x,t) = tx + (1-t)\,\frac{x}{|x|} \ .$$

(The reader should check that $F(x,t) \neq 0$ for $x \in X$ and $t \in [0,1]$.)

If $f\colon X \to Y$ and $g\colon Y \to X$ are maps such that $f \circ g$ is homotopic to the identity map of Y, then g is called a *right homotopy inverse* for f, and if $g \circ f$ is homotopic

to the identity map of X, then g is called a *left homotopy inverse* for f. If g is both a left and right homotopy inverse for f, then the same is true of f for g. In this case, f and g are called *homotopy inverses* of one another and the spaces X and Y are said to be *homotopy equivalent*, which is written $X \simeq Y$. Notice that if $f: X \to Y$ is a homeomorphism, then f and f^{-1} are homotopy inverses, and thus homotopy equivalence is indeed a weaker notion than homeomorphism.

EXAMPLE A.2.3. Let $X = \mathbb{R}^n - \{0\}$ and let $Y = \{x \in \mathbb{R}^n : |x| = 1\}$, the unit $(n-1)$-sphere. Define $f: X \to Y$ by $f(x) = x/|x|$, and define $g: Y \to X$ by $g(y) = y$. Then the homotopy of Example A.2.2 shows that $g \circ f$ is homotopic to the identity map of X, and the composition $f \circ g$ is the identity map of Y. Thus, f and g are homotopy inverses, and $X \simeq Y$.

A.3 Categories and functors.

The goal of algebraic topology is to study topological spaces by systematically associating to them algebraic objects that are topological invariants. By "topological invariant" we mean an object that is the same, in an appropriate sense, for homeomorphic spaces. The difference of the algebraic invariants associated to two spaces then provides a measure of the topological difference of the spaces. The most useful invariants are defined not only on spaces, but on maps as well. The formal way to describe such invariants is with the language of categories and functors.

For concreteness, let us fix a field $\mathbb{F}$ and assume that the algebraic objects will be taken from among the family of all $\mathbb{F}$ vector spaces. The algebraic functions between such vector spaces are the $\mathbb{F}$-linear maps. The composition of two $\mathbb{F}$-linear maps is $\mathbb{F}$-linear, and the identity function on any $\mathbb{F}$ vector space is $\mathbb{F}$-linear. Formally, one then says that the $\mathbb{F}$ vector spaces and $\mathbb{F}$-linear maps form a *category*. Similarly, the family of topological spaces and continuous maps form a category. A systematic association of invariants to topological spaces and maps can then be expressed by a *functor* $\mathfrak{F}$ from the topological category to the category of $\mathbb{F}$ vector spaces. For each topological space X, $\mathfrak{F}(X)$ is an $\mathbb{F}$ vector space, and for each continuous map $f: X \to Y$, $\mathfrak{F}(f)$ is an $\mathbb{F}$-linear map from the vector space $\mathfrak{F}(X)$ to the vector space $\mathfrak{F}(Y)$. The functor is required to send the identity map of X to the identity $\mathbb{F}$-linear map of $\mathfrak{F}(X)$, and it is also required to respect compositions in the sense that

$$\mathfrak{F}(f \circ g) = \mathfrak{F}(f) \circ \mathfrak{F}(g)$$

whenever $f \circ g$ is defined. Such a functor is called *covariant* because it preserves the order of compositions of maps. An example of such a functor is singular homology with coefficients in the field $\mathbb{F}$. One can also consider functors that always reverse the order of compositions, and such functors are called *contravariant*. An example is singular cohomology with coefficients in $\mathbb{F}$. Contravariant functors typically arise when a duality principle is applied, in which case the roles of injections and surjections tend to be interchanged.

There is usually a trade off between how finely a functor detects topological differences and how easily values of the functor can be calculated. Common functors, such as singular homology, singular cohomology, and the homotopy group functor, do not distinguish spaces that are homotopy equivalent. More precisely, if $\mathfrak{F}$ is one of these functors and if $f, g: X \to Y$ are homotopic maps of topological spaces, then $\mathfrak{F}(f)$ and $\mathfrak{F}(g)$ are equal as maps from $\mathfrak{F}(X)$ to $\mathfrak{F}(Y)$. It follows that if $X \simeq Y$, then $\mathfrak{F}(X)$ and $\mathfrak{F}(Y)$ are isomorphic as algebraic objects.

A.4 Simplicial complexes and maps.

The topological spaces that arise in the study of computability in fault-tolerant distributed systems are given by discrete data, such as configurations of input values for the processes, and these spaces can be described as *simplicial complexes*. All simplicial complexes considered in this paper are finite. A finite simplicial complex is a finite family $\mathcal{K}$ of non-empty finite sets that satisfies the following hereditary property: if $S \in \mathcal{K}$ and S' is a non-empty subset of S, then $S' \in \mathcal{K}$. An element $S \in \mathcal{K}$ is called a *simplex*, and the points of S are its *vertices*. The dimension of a simplex is one less than the number of its vertices, and a simplex of dimension k is called a k-simplex. If S' is a non-empty (proper) subset of S, then S' is called a (*proper*) *face* of S. The complex $\mathcal{K}$ is said to be a *pure complex* of dimension n if for each simplex $S \in \mathcal{K}$ there is an n-dimensional simplex $S_1 \in \mathcal{K}$ such that S is a face of S_1. In this case, an n-simplex of $\mathcal{K}$ is said to be a *full* simplex. The union of all simplices of a complex gives the vertex set V of the complex. It is customary to identify the vertex v with the 0-simplex $\{v\}$. More generally, the collection of all faces of a simplex S forms a simplicial complex $\mathcal{S}$. The distinction between S and $\mathcal{S}$ is often blurred.

If $\mathcal{K}$ has no simplex of dimension greater than one, then $(V, \mathcal{K})$ is an undirected graph. By "drawing" a graph as a subset of $\mathbb{R}^n$ in such a way that edges intersect only at vertices, the graph is realized as a topological space. In a similar fashion, any simplicial complex $\mathcal{K}$ can be realized suitably as a subset $|\mathcal{K}|$ of $\mathbb{R}^n$ for large enough n. We will refer to such $|\mathcal{K}|$ as a *realization* of $\mathcal{K}$.[10] A simple way to obtain $|\mathcal{K}|$ is the following. Take n to be the number of elements of V and to choose for each vertex v a vector $|v| \in \mathbb{R}^n$ such that the set $\{|v| : v \in V\}$ is linearly independent. To a simplex $S \in \mathcal{K}$, one associates the *convex hull* $|S|$ of $\{|v| : v \in S\}$. If $v_0, \ldots, v_k$ are the vertices of S, then $|S|$ is the set of points $x \in \mathbb{R}^n$ that can be written in the form

$$x = \sum_{i=0}^{k} t_i |v_i| \ ,$$

where the t_i are non-negative real numbers satisfying $\sum_{i=0}^{k} t_i = 1$. For any such x, the corresponding tuple $(t_0, \ldots, t_k)$ is unique, and these numbers are called the *barycentric coordinates* of x. The "central" point of $|S|$ is obtained when each $t_i = 1/(k+1)$, and this point is called the *barycenter* of $|S|$. The topological space $|\mathcal{K}|$ is obtained as the union of the sets $|S|$ for all simplices $S \in \mathcal{K}$. Notice that the linear independence of $\{|v| : v \in V\}$ ensures that

$$|S| \cap |S'| = |S \cap S'| \tag{A.4.1}$$

for any simplices $S, S' \in \mathcal{K}$. This property generalizes the condition that in drawing a graph, edges must intersect only at vertices. In general, any way of picking the points $|v|$ in $\mathbb{R}^n$ so that the convex hulls associated to simplices satisfy (A.4.1) gives an admissible realization. Any two such realizations of $\mathcal{K}$ are homeomorphic topological spaces. We abuse language by referring to *the* realization $|\mathcal{K}|$, when what is meant is any one of the possible homeomorphic choices for $|\mathcal{K}|$.

If S is a 1-simplex, then $|S|$ is a line segment; if S is a 2-simplex, then $|S|$ is a triangular face; if S is a 3-simplex, then $|S|$ is a solid tetrahedron; and so forth.

[10] The notation $|\mathcal{K}|$ for a realization is standard, although it conflicts with the common notation for cardinality of a set. Context should always make clear the meaning of $|\cdot|$.

Thus, the convex hulls of the various simplices of $\mathcal{K}$ give a generalized triangular decomposition, or triangulation, of $|\mathcal{K}|$, which is the analog of the familiar notion from geometry of triangular decomposition of a planar polygonal region.

Let $\mathcal{K}$ and $\mathcal{L}$ be simplicial complexes with vertex sets V and W, respectively. A *simplicial map* from $\mathcal{K}$ to $\mathcal{L}$ is a function $f\colon V \to W$ such that $f(S)$ is a simplex of $\mathcal{L}$ whenever S is a simplex of $\mathcal{K}$. Notice that $f(S)$ has dimension no greater than that of S. The simplicial map f determines a function $\mathcal{K} \to \mathcal{L}$ by $S \mapsto f(S)$, although not every function $\mathcal{K} \to \mathcal{L}$ arises in this way. When we write $f\colon \mathcal{K} \to \mathcal{L}$, we mean that f is a simplicial map. If a simplicial map f is bijective as a map of vertex sets, then we say that f is an isomorphism of simplicial complexes. The identity function $V \to V$ determines the identity simplicial map $\mathcal{K} \to \mathcal{K}$, which is an isomorphism. The composition of two simplicial maps is simplicial. Thus, the finite simplicial complexes and simplicial maps form a category.

A simplicial map $f\colon \mathcal{K} \to \mathcal{L}$ gives rise to a continuous map $|f|\colon |\mathcal{K}| \to |\mathcal{L}|$ in the following way. First define $|f|(|v|) = |f(v)|$ for each vertex $v \in V$. Then define $|f|$ on the convex hull $|S|$ of a simplex $S \in \mathcal{K}$ by extending piecewise-linearly. More precisely, if $v_0, \ldots, v_k$ are the vertices of $|S|$, then for a point $x \in |S|$ with barycentric coordinates $(t_0, \ldots, t_k)$, define

$$|f|(x) = \sum_{i=0}^{k} t_i |f(v_i)| \ .$$

It is straightforward to check that the map $|f|$ defined in this way is continuous. If f is an isomorphism of simplicial complexes, then $|f|$ is a homeomorphism.

Notice that if f is the identity simplicial map of $\mathcal{K}$, then $|f|$ is the identity map of $|\mathcal{K}|$. It is not difficult to check that if f and g are simplicial maps whose composition $f \circ g$ is defined, then $|f \circ g| = |f| \circ |g|$. Thus, $|\cdot|$ can be interpreted as a covariant functor from the category of simplicial complexes and simplicial maps to the category of topological spaces and continuous maps.[11] The distinction between $\mathcal{K}$ and $|\mathcal{K}|$ is often blurred, and when we speak of topological properties of $\mathcal{K}$, we mean those of $|\mathcal{K}|$.

A simplicial complex $\mathcal{K}$ is a *subcomplex* of the complex $\mathcal{L}$ provided $\mathcal{K} \subseteq \mathcal{L}$. In this case, the inclusion ι of the vertex set of $\mathcal{K}$ in the vertex set of $\mathcal{L}$ is a simplicial map, and any realization $|\mathcal{L}|$ determines a unique realization $|\mathcal{K}| \subseteq |\mathcal{L}|$ such that this last inclusion is $|\iota|$. Particularly important examples of subcomplexes are the skeleta of a complex. For a simplicial complex $\mathcal{K}$ and a non-negative integer n, the *n-skeleton* of $\mathcal{K}$, denoted $\mathcal{K}^{(n)}$, is the subcomplex of $\mathcal{K}$ consisting of all simplices of dimension at most n. $\mathcal{K}^{(0)}$ is identified with the vertex set of $\mathcal{K}$, while $\mathcal{K}^{(1)}$ is an undirected graph. If $\mathcal{S}$ the complex of faces of an $(n+1)$-simplex, then $|\mathcal{S}^{(n)}|$ is homeomorphic to an n-sphere.

If S and T are disjoint simplices, then their union is given the special notation $S * T$ and called the *join* of S and T. $S * T$ is a simplex of dimension one more than the sum of the dimensions of S and T. If $\mathcal{K}$ and $\mathcal{L}$ are disjoint complexes, then the join $\mathcal{K} * \mathcal{L}$ is the complex of all simplices of the form S, T, or $S * T$, where S is a simplex of $\mathcal{K}$ and T is a simplex of $\mathcal{L}$. In the special case where $\mathcal{L}$ consists

[11] Strictly speaking, we must make a choice of realization $|\mathcal{K}|$ for each finite simplicial complex $\mathcal{K}$, or we must understand $|\mathcal{K}|$ to be an appropriate equivalence class.

of a single vertex v, the join $\mathcal{K} * \{v\}$ is called a *cone*. The realization $|\mathcal{K} * \{v\}|$ has the structure of a cone with vertex $|v|$ and cross sections homeomorphic to $|\mathcal{K}|$.

A.5 Chromatic complexes and maps.

The simplicial complexes that arise in the study of computability in fault-tolerant distributed systems have additional data associated to the vertices. The vertices give pairings of processes with associated values of an appropriate type, such as input values or output values. Thus, each vertex can be thought of as having a *color*, which is the identifier of the associated process. The vertex is also labelled by a value, but we reserve the term "color" to mean the process identifier. A simplex in such a complex represents a consistent and mutually compatible arrangement of values among processes. The assignment of distinct values to a single process is not consistent, and so for these complexes it is required that distinct vertices in a simplex have distinct colors.

To make this requirement somewhat more precise, let us fix a simplex S, to be thought of as the set of process identifiers, and let $\mathcal{S}$ be the complex of faces of S. An *S-chromatic complex* is a simplicial complex $\mathcal{K}$ together with a simplicial map $c_{\mathcal{K}}: \mathcal{K} \to \mathcal{S}$ such that $c_{\mathcal{K}}(X)$ is a k-simplex of $\mathcal{S}$ whenever X is a k-simplex of $\mathcal{K}$. The map $c_{\mathcal{K}}$ is the *coloring* of the complex. If $\mathcal{L}$ is another S-chromatic complex, then a simplicial map $f: \mathcal{K} \to \mathcal{L}$ is called *S-chromatic* if $c_{\mathcal{L}} \circ f = c_{\mathcal{K}}$. In this case, $f(X)$ must be a k-simplex of $\mathcal{L}$ whenever X is a k-simplex of $\mathcal{K}$. When the simplex S of colors is understood, it is dropped from the notation.

A.6 Subdivisions and simplicial approximation.

A simplicial map $f: \mathcal{K} \to \mathcal{L}$ gives rise to a continuous map $|f|: |\mathcal{K}| \to |\mathcal{L}|$, but a continuous map $g: |\mathcal{K}| \to |\mathcal{L}|$ need not be of the form $|f|$ for any simplicial f. In fact, g need not even be homotopic to any $|f|$. The reason is that $|f|$ is severely restricted by the condition of piecewise-linearity arising from the fixed simplicial structure of $\mathcal{K}$. A similar situation arises in calculus, when a smooth curve $\gamma: [0,1] \to \mathbb{R}^n$ is approximated by a polygonal curve obtained by connecting the points $\gamma(x_i)$ for a partition $\{x_i\}$ of $[0,1]$. The strategy for improving the polygonal approximation is to refine the partition of $[0,1]$. The analogous refinement for $\mathcal{K}$ produces a finer triangulation of $|\mathcal{K}|$ and is called a *subdivision*.

To be precise, let $\mathcal{K}$ be a simplicial complex with realization $|\mathcal{K}|$. A subdivision of $\mathcal{K}$ is a simplicial complex $\sigma(\mathcal{K})$ for which there is a realization $|\sigma(\mathcal{K})|$ that is equal to $|\mathcal{K}|$ and which satisfies the following properties.

(1) For every simplex S of $\sigma(\mathcal{K})$ there is a simplex X of $\mathcal{K}$ such that $|S| \subseteq |X|$. The minimal such X is called the *carrier* of S, written carrier(S).
(2) For every simplex X of $\mathcal{K}$, there is a finite collection $S_1, \ldots, S_r$ of simplices of $\sigma(\mathcal{K})$ such that $|X| = |S_1| \cup \cdots \cup |S_r|$.

The convex hulls $|S_1|, \ldots, |S_r|$ of condition (2) give the refined triangulation, or subdivision, of $|X|$.

The *barycentric subdivision* of $\mathcal{K}$ is a uniform subdivision that ensures that every simplex of $\mathcal{K}$ is divided non-trivially. This subdivision can be described inductively on the skeleta. For each simplex X of $\mathcal{K}$, we add the barycenter b of $|X|$, which is a new vertex if X is of dimension greater than zero. If X is of dimension greater than zero, then the proper faces of X are assumed already to have been subdivided, and

each join of $\{b\}$ to a simplex in the subdivision of a proper face of X is added as a new simplex.

By iterating barycentric subdivision, one can obtain arbitrarily fine subdivisions of $\mathcal{K}$, which are analogous to partitions of arbitrarily fine mesh in calculus. It is a theorem that if $g\colon|\mathcal{K}| \to |\mathcal{L}|$ is a continuous map, then for a sufficiently fine subdivision $\sigma(\mathcal{K})$ of $\mathcal{K}$ there exists a simplicial map $f\colon\sigma(\mathcal{K}) \to \mathcal{L}$ that approximates g in the sense that $|f| \simeq g$. (See [Mu], §§14–16.)

Now fix a subdivision $\sigma(\mathcal{K})$ of $\mathcal{K}$. We can identify the realizations $|\sigma(\mathcal{K})| = |\mathcal{K}|$. The identity map $|\sigma(\mathcal{K})| \to |\mathcal{K}|$ is continuous and therefore can be approximated simplicially, perhaps after further subdividing $\sigma(\mathcal{K})$. It turns out that no further subdivision is necessary, according to the following

Lemma A.6.1: *Let $\mathcal{K}$ be a finite simplicial complex, and let $\sigma(\mathcal{K})$ be a subdivision. There exists a simplicial map $f\colon\sigma(\mathcal{K}) \to \mathcal{K}$ such that $f(S) \subseteq \text{carrier}(S)$ for every simplex S in $\sigma(\mathcal{K})$. For any such f, the map $|f|\colon|\sigma(\mathcal{K})| \to |\mathcal{K}|$ on the realizations is homotopic to the identity map $|\sigma(\mathcal{K})| \to |\mathcal{K}|$ relative to the vertices of $\mathcal{K}$.*

See, for example, Lemma 3.4.2 of [S]. A generalization of this Lemma is proved in Section 4.

If $\mathcal{K}$ is a chromatic complex, then a *chromatic subdivision* of $\mathcal{K}$ is a subdivision $\chi(\mathcal{K})$ of $\mathcal{K}$ that is itself a chromatic complex and for which the colors of the vertices of any simplex S of $\chi(\mathcal{K})$ form a subset of the colors of the vertices of carrier(S). *Standard chromatic subdivision* is a construction similar to barycentric subdivision that gives uniform chromatic subdivisions. When subdividing X, one adds a cluster of vertices bearing all the colors appearing in X in an appropriate arrangement about the barycenter and then makes the joins that are allowed by the chromatic condition. See [HS2] for details.

Suppose $\mathcal{K}$ is a chromatic complex and $\chi(\mathcal{K})$ is a chromatic subdivision. For each simplex S in $\chi(\mathcal{K})$, there is a unique face of carrier(S) whose vertices have the same colors as S, and there is a unique color-preserving vertex map from S to that face. It follows that there is a unique chromatic simplicial map $f_\chi\colon\chi(\mathcal{K}) \to \mathcal{K}$ satisfying $f_\chi(S) \subseteq \text{carrier}(S)$ for every simplex S in $\chi(\mathcal{K})$. From Lemma A.6.1, it follows that $|f_\chi|$ is homotopic to the identity map $|\chi(\mathcal{K})| \to |\mathcal{K}|$.

A.7 Simplicial homology.

There are many different homology functors, defined on various categories and tailored to detect various nuances of spaces. We describe one of the simplest of these, the simplicial homology functor. We will use only coefficients in the two-element field $\mathbb{F}$ of integers mod-2. This restriction simplifies the presentation by avoiding orientations. See [Mu] for a more general account.

Let $\mathcal{K}$ be a simplicial complex. An *n-chain* of $\mathcal{K}$ is a sum $S_1 + \cdots + S_k$, where each S_i is an n-simplex of $\mathcal{K}$. Repeated occurrences of the same simplex in the sum are understood to cancel in pairs. Such a sum can be thought of as a linear combination with coefficients from $\mathbb{F}$, where the simplices themselves are vectors. The set of n-chains of $\mathcal{K}$ forms an $\mathbb{F}$ vector space, denoted $C_n(\mathcal{K})$, and has as a basis the set of n-simplices of $\mathcal{K}$. If $\mathcal{K}$ has no simplex of dimension n, then $C_n(\mathcal{K}) = 0$. The sequence of all $C_n(\mathcal{K})$, $n \geq 0$, is called the *simplicial chain complex* of $\mathcal{K}$ and is denoted $C_\sharp(\mathcal{K})$. There is a boundary operator ∂ on $C_\sharp(\mathcal{K})$ defined as follows. For $n \geq 1$, an n-simplex $S \in \mathcal{K}$ has $n+1$ faces of dimension $n-1$, and $\partial_n(S)$ is

defined to be the chain that is the sum of these faces. For a vertex v, we understand $\partial_0(v) = 0$. This definition of ∂_n on a basis of $C_n(\mathcal{K})$ extends uniquely to an $\mathbb{F}$-linear map $\partial_n : C_n(\mathcal{K}) \to C_{n-1}(\mathcal{K})$, where we understand $C_{-1}(\mathcal{K}) = 0$. A key property of the boundary operator is that

$$\partial_n \circ \partial_{n+1} = 0 \tag{A.7.1}$$

for $n \geq 0$.

An n-chain whose boundary is zero is called an *n-cycle*. The kernel of ∂_n is the subspace of n-cycles, which is written as $Z_n(\mathcal{K})$. From (A.7.1), it follows that the image of ∂_{n+1} is a vector subspace of $Z_n(\mathcal{K})$, and the quotient vector space

$$H_n(\mathcal{K}) = Z_n(\mathcal{K}) / \operatorname{im} \partial_{n+1}$$

is defined to be the *n-dimensional homology* of $\mathcal{K}$. If $T \in Z_n(\mathcal{K})$, we write $[T]$ for the coset in $H_n(\mathcal{K})$ represented by T. It is customary to write $H_*(\mathcal{K})$ for the sequence of all $H_n(\mathcal{K})$, $n \geq 0$. Some useful examples of homology are given in Subsection A.9, after we describe the relationship between simplicial and singular homologies.

Notice that the definition of simplicial homology relies critically on simplicial structure. It is not obvious how the particular simplicial structure of a complex $\mathcal{K}$, and thus the corresponding triangulation of a realization $|\mathcal{K}|$, affects $H_*(\mathcal{K})$. In fact, two simplicial complexes with homeomorphic realizations have isomorphic simplicial homology, and so simplicial homology is a topological invariant. One strategy for the proof of this fact uses subdivision and simplicial approximation (see Chapter 2 of [Mu]).

An attractive feature of the simplicial homology $H_*(\mathcal{K})$ is that it can be computed effectively from $\mathcal{K}$. Once $\mathcal{K}$ has been represented, the boundary operators are easily calculated and can be represented by matrices. The homology $H_*(\mathcal{K})$ is then computable by using Gaussian elimination. See [Mu, §11] for more details. Notice that since $\mathcal{K}$ is finite, the dimensions of the simplices of $\mathcal{K}$ are bounded. If N is the maximum dimension of any simplex of $\mathcal{K}$, then $C_n(\mathcal{K}) = 0$ for $n > N$. It follows that $H_n(\mathcal{K}) = 0$ for $n > N$, and

$$H_N(\mathcal{K}) = Z_N(\mathcal{K}) \ .$$

If $f : \mathcal{K} \to \mathcal{L}$ is a simplicial map and if S is an n-simplex of $\mathcal{K}$, then we define $f_n(S) = f(S)$ if $f(S)$ is an n-simplex, and $f_n(S) = 0$ if $f(S)$ is of dimension less than n. This definition on the basis of $C_n(\mathcal{K})$ gives a unique extension to a linear map $f_n : C_n(\mathcal{K}) \to C_n(\mathcal{L})$, which is referred to as the map of n-chains *induced* by f. The sequence of all such f_n, $n \geq 0$, is denoted $f_\sharp : C_\sharp(\mathcal{K}) \to C_\sharp(\mathcal{L})$ and is called the *chain map* induced by f. One checks that for $n \geq 0$,

$$\partial_{n+1} \circ f_{n+1} = f_n \circ \partial_{n+1} \ ,$$

where the boundary operator on the left is from $C_\sharp(\mathcal{L})$ and the boundary operator on the right is from $C_\sharp(\mathcal{K})$. It follows that there is a well-defined linear map

$$H_n(\mathcal{K}) \to H_n(\mathcal{L})$$

in homology given by

$$[T] \mapsto [f_n(T)]$$

for $T \in Z_n(\mathcal{K})$. The sequence of these homology maps for $n \geq 0$ is called the map *induced* by f in homology and is denoted $f_* : H_*(\mathcal{K}) \to H_*(\mathcal{L})$. The identity

simplicial map $\mathcal{K} \to \mathcal{K}$ clearly induces the identity map on $H_*(\mathcal{K})$. It is not difficult to show that if $f \circ g$ is a composition of simplicial maps, then $(f \circ g)_* = f_* \circ g_*$. Thus, simplicial homology with coefficients in $\mathbb{F}$ is a covariant functor from the category of finite simplicial complexes and simplicial maps to the category of finite-dimensional $\mathbb{F}$ vector spaces and $\mathbb{F}$-linear maps.

If $\mathcal{K}$ is a subcomplex of $\mathcal{L}$, then the inclusion $\iota\colon \mathcal{K} \to \mathcal{L}$ gives an injective chain map $\iota_\sharp\colon C_\sharp(\mathcal{K}) \to C_\sharp(\mathcal{L})$. The quotient $C_\sharp(\mathcal{L})/C_\sharp(\mathcal{K})$ is denoted $C_\sharp(\mathcal{L},\mathcal{K})$ and is called the *relative simplicial chain complex* associated to the pair $(\mathcal{L},\mathcal{K})$. From this definition, there is a short exact sequence[12]

$$0 \to C_\sharp(\mathcal{K}) \xrightarrow{\iota_\sharp} C_\sharp(\mathcal{L}) \xrightarrow{p} C_\sharp(\mathcal{L},\mathcal{K}) \to 0\ , \tag{A.7.2}$$

where p is projection on the quotient. The boundary operator ∂ from $C_\sharp(\mathcal{L})$ gives rise to a well-defined boundary operator on $C_\sharp(\mathcal{L},\mathcal{K})$, which we also denote by ∂. The kernel of ∂ on $C_n(\mathcal{L},\mathcal{K})$ is the subspace of relative n-cycles, denoted $Z_n(\mathcal{L},\mathcal{K})$. Notice that a chain $T \in C_n(\mathcal{L})$ represents a relative n-cycle if and only if $\partial T \in C_{n-1}(\mathcal{K})$. The relative homology $H_n(\mathcal{L},\mathcal{K})$ is defined as above:

$$H_n(\mathcal{L},\mathcal{K}) = Z_n(\mathcal{L},\mathcal{K})/\mathrm{im}\,\partial\ .$$

From the short exact sequence (A.7.2), one derives a long exact sequence in homology:

$$\cdots \xrightarrow{\partial} H_n(\mathcal{K}) \xrightarrow{\iota_*} H_n(\mathcal{L}) \xrightarrow{p_*} H_n(\mathcal{L},\mathcal{K}) \xrightarrow{\partial} H_{n-1}(\mathcal{K}) \xrightarrow{\iota_*} \cdots \tag{A.7.3}$$

The connecting map $\partial\colon H_n(\mathcal{L},\mathcal{K}) \to H_{n-1}(\mathcal{K})$ is defined as follows. For an element $\tau \in H_n(\mathcal{L},\mathcal{K})$, let $T \in C_n(\mathcal{L})$ be a chain representing a relative n-cycle in the coset τ. Then $\partial T \in Z_{n-1}(\mathcal{K})$. The coset of ∂T in $H_{n-1}(\mathcal{K})$ gives the value of $\partial\tau$. Of course, there is some checking to be done to ensure that this map is well-defined.

A.8 Singular homology.

An alternative homology is obtained from the *singular homology functor*, which is defined on the category of topological spaces and continuous maps. We assume that singular homology is also computed with coefficients in $\mathbb{F}$. For a definition of singular homology, see Chapter 4 of [Mu]. Suffice it to say here that the definition uses notions of chains and boundaries analogous to those introduced above, and the singular homology functor has the exact sequence property (A.7.3).

A basic theorem in homology theory is that for any simplicial complex $\mathcal{K}$ and realization $|\mathcal{K}|$, there is an isomorphism

$$j_{\mathcal{K}}\colon H_*(\mathcal{K}) \to H_*(|\mathcal{K}|)\ . \tag{A.8.1}$$

Here $H_*(|\mathcal{K}|)$ denotes the singular homology of the realization $|\mathcal{K}|$, and the isomorphism is understood to be a sequence of isomorphisms, one for each dimension. Furthermore, if $f\colon \mathcal{K} \to \mathcal{L}$ is a simplicial map, then there is a commutative diagram

$$\begin{array}{ccc} H_*(\mathcal{K}) & \xrightarrow{j_{\mathcal{K}}} & H_*(|\mathcal{K}|) \\ \downarrow{\scriptstyle f_*} & & \downarrow{\scriptstyle |f|_*} \\ H_*(\mathcal{L}) & \xrightarrow{j_{\mathcal{L}}} & H_*(|\mathcal{L}|) \end{array} \tag{A.8.2}$$

[12] A sequence $\cdots \xrightarrow{f} V \xrightarrow{g} \cdots$ of linear maps is *exact* if the kernel of g is equal to the image of f. For a longer sequence, exactness means exactness at every junction.

$|f|_*$ denotes the map induced in singular homology by $|f|$. (See [Mu, §34].)

A property of singular homology is that if $\alpha, \beta: X \to Y$ are homotopic maps, then the induced maps $\alpha_*, \beta_*: H_*(X) \to H_*(Y)$ are equal. In particular, maps that are homotopy inverses of one another induce singular homology maps that are inverses of one another, and thus $X \simeq Y$ implies $H_*(X)$ is isomorphic to $H_*(Y)$. From the diagram (A.8.2) it follows that if $f, g: \mathcal{K} \to \mathcal{L}$ are simplicial maps such that $|f| \simeq |g|$, then the maps f_* and g_* induced in simplicial homology are equal. Similarly, if $|f|$ has a left/right/two-sided homotopy inverse, then f_* must be an injection/surjection/isomorphism.

As a simple application of these ideas, we revisit the situation of a subdivision $\sigma(\mathcal{K})$.

Lemma A.8.3: *Let $\sigma(\mathcal{K})$ be a subdivision of $\mathcal{K}$ and let $f: \sigma(\mathcal{K}) \to \mathcal{K}$ be as in Lemma* A.6.1. *Then $f_*: H_*(\sigma(\mathcal{K})) \to H_*(\mathcal{K})$ is an isomorphism.*

Proof: We have the commutative diagram

$$\begin{array}{ccc} H_*(\sigma(\mathcal{K})) & \xrightarrow{j_{\sigma(\mathcal{K})}} & H_*(|\sigma(\mathcal{K})|) \\ \downarrow{\scriptstyle f_*} & & \downarrow{\scriptstyle |f|_*} \\ H_*(\mathcal{K}) & \xrightarrow{j_{\mathcal{K}}} & H_*(|\mathcal{K}|) \end{array}$$

from (A.8.2). According to Lemma A.6.1, $|f|$ is homotopic to the identity map $|\sigma(\mathcal{K})| \to |\mathcal{K}|$, and it follows from the property of singular homology cited above that $|f|_*$ is the identity map. Since the j maps are isomorphisms, f_* is an isomorphism. □

A.9 Homology examples.

We give here a brief collection of homology examples that give ample background for the applications in Section 5. All homology is with coefficients in $\mathbb{F}$, and for simplicity we consider only topological spaces that are homotopy equivalent to realizations of finite simplicial complexes. When speaking of $H_n(X)$, we refer to n as the *dimension* of the homology and to $\dim_{\mathbb{F}} H_n(X)$ as the *rank* of the homology.

The rank of $H_0(X)$ is equal to the number of path components of X. Thus, X is path-connected if and only if $H_0(X) \cong \mathbb{F}$. From the definition of simplicial homology, it is easy to see that for a one-point space $\{x\}$, $H_0(\{x\}) \cong \mathbb{F}$ and $H_i(\{x\}) = 0$ for $i \neq 0$. The *reduced homology* of a space X, denoted $\tilde{H}_*(X)$, is the relative homology $H_*(X, \{x\})$, where $x \in X$. From the exact sequence (A.7.3), $H_0(X) \cong \tilde{H}_0(X) \oplus \mathbb{F}$, while $H_n(X) \cong \tilde{H}_n(X)$ for $n > 0$, and thus either of $H_*(X)$ and $\tilde{H}_*(X)$ can be obtained from the other. We use reduced homology when it is convenient.

Let X and Y be disjoint spaces. If we write $X \sqcup Y$ for the space that is the disjoint union of X and Y, then

$$H_*(X \sqcup Y) \cong H_*(X) \oplus H_*(Y) \ ,$$

where we mean that there is an isomorphism in each dimension. Similarly, we can form a product space $X \times Y$, appropriately topologized, and the simplest form of the Künneth Theorem gives

$$H_*(X \times Y) \cong H_*(X) \otimes H_*(Y) \ .$$

By $X \vee Y$, we mean the one-point union, or *wedge*, of X and Y. This space is obtained by identifying one point $x_0 \in X$ with one point $y_0 \in Y$. Then

$$\tilde{H}_*(X \vee Y) \cong \tilde{H}_*(X) \oplus \tilde{H}_*(Y) \ .$$

The *smash product* $X \wedge Y$ is obtained from $X \times Y$ by identifying the subspace $X \times \{y_0\} \cup \{x_0\} \times Y$, which is homeomorphic to $X \vee Y$, to a point. One has

$$\tilde{H}_*(X \wedge Y) \cong \tilde{H}_*(X) \otimes \tilde{H}_*(Y) \ .$$

Each of these constructions is associative, up to homeomorphism, and can be iterated. Furthermore, $\times$ distributes over $\sqcup$ and $\wedge$ distributes over $\vee$, up to homeomorphism.

If S^n is an n-sphere, then $\tilde{H}_*(S^n)$ is non-zero only in dimension n, where the homology has rank one: $\tilde{H}_n(S^n) \cong \mathbb{F}$. The constructions of the preceding paragraph can be applied to build spaces from spheres whose homology is easily computed. For example, a 2-torus T^2 is homeomorphic to the product $S^1 \times S^1$ of 1-spheres, and thus

$$H_*(T^2) \cong H_*(S^1) \otimes H_*(S^1) \ .$$

Explicitly, $H_0(T^2)$ and $H_2(T^2)$ are of rank one, while $H_1(T^2)$ is of rank two. As another example, let

$$X = \bigvee_1^m S^n \ ,$$

the m-fold wedge of n-spheres. Then $\tilde{H}_n(X)$ is of rank m, while $\tilde{H}_i(X) = 0$ for $i \neq n$.

References

[Af+] Y. Afek, H. Attiya, D. Dolev, E. Gafni, M. Merritt, and N. Shavit, *Atomic snapshots of shared memory*, Journal of the ACM **40** (1993), no. 4, 873–890.

[At+] H. Attiya, A. Bar-Noy, D. Dolev, D. Peleg, and R. Reischuk, *Renaming in an asynchronous environment*, Journal of the ACM (1990).

[BMZ] O. Biran, S. Moran, and S. Zaks, *A combinatorial characterization of the distributed 1-solvable tasks*, Journal of Algorithms **11** (1990), 420–440.

[B] E. Borowsky, *Capturing the power of resiliency and set consensus in distributed systems*, Ph.D. Thesis, UCLA, 1995.

[BG1] E. Borowsky and E. Gafni, *Immediate atomic snapshots and fast renaming*, Proceedings of the 12th ACM Symposium on Principles of Distributed Computing, 1993, pp. 41–51.

[BG2] E. Borowsky and E. Gafni, *Generalized FLP impossibility result for t-resilient asynchronous computations*, Proceedings of the 25th ACM Symposium on the Theory of Computing, 1993, pp. 91–100.

[BG3] E. Borowsky and E. Gafni, *A simple algorithmically reasoned characterization of wait-free computations*, manuscript.

[Ch] S. Chaudhuri, *More* choices *allow more* faults: *set consensus problems in totally asynchronous systems*, Information and Computation **105** (1993), 132–158.

[FLP] M. Fischer, N. Lynch, and M. Paterson, *Impossibility of distributed consensus with one faulty process*, Journal of the ACM **32** (1985), no. 2, 374–382.

[GK] E. Gafni and E. Koutsoupias, *Three-processor tasks are undecidable*, manuscript.

[HR1] M. Herlihy and S. Rajsbaum, *Algebraic spans*, Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Systems, 1995.

[HR2] M. Herlihy and S. Rajsbaum, *A primer on algebraic topology and distributed computing*, Springer Lecture Notes in Computer Science, vol. 1000.

[HR3] M. Herlihy and S. Rajsbaum, *The decidability of distributed decision tasks*, manuscript.

[HS1] M. Herlihy and N. Shavit, *The asynchronous computability theorem for t-resilient tasks*, Proceedings of the 25th ACM Symposium on the Theory of Computing, 1993, pp. 111–120.

[HS2] M. Herlihy and N. Shavit, *A simple constructive computability theorem for wait-free computation*, Proceedings of the 26th ACM Symposium on the Theory of Computing, 1994, pp. 101–110.

[HS3] M. Herlihy and N. Shavit, *The topological structure of asynchronous computability*, Brown TR CS-96-03, 1996.

[Mu] J. Munkres, *Elements of Algebraic Topology*, Addison-Wesley, 1984.

[SZ] M. Saks and F. Zaharoglou, *Wait-free k-set agreement is impossible: The topology of public knowledge*, Proceedings of the 25th ACM Symposium on the Theory of Computing, 1993, pp. 101–110.

[S] E. Spanier, *Algebraic Topology*, Springer, 1966.

Department of Computer Sciences, The University of Texas at Austin, Austin, Texas 78712

E-mail address: havlicek@cs.utexas.edu