

# Back to the Coordinated Attack Problem

Emmanuel Godard and Eloi Perdereau  
(emmanuel.godard|eloi.perdereau)@lis-lab.fr

Laboratoire d'Informatique et Systèmes  
Aix-Marseille Université – CNRS (UMR 7020)

**Abstract.** We consider the well known Coordinated Attack Problem, where two generals have to decide on a common attack, when their messengers can be captured by the enemy. Informally, this problem represents the difficulties to agree in the presence of communication faults. We consider here only omission faults (loss of message), but contrary to previous studies, we do not restrict the way messages can be lost, *i.e.* we make no specific assumption, we use no specific failure metric. In the large subclass of message adversaries where the double simultaneous omission can never happen, we characterize which ones are obstructions for the Coordinated Attack Problem. We give two proofs of this result. One is combinatorial and uses the classical bivalency technique for the necessary condition. The second is topological and uses simplicial complexes to prove the necessary condition. We also present two different Consensus algorithms that are combinatorial (resp. topological) in essence. Finally, we analyze the two proofs and illustrate the relationship between the combinatorial approach and the topological approach in the very general case of message adversaries. We show that the topological characterization gives a clearer explanation of why some message adversaries are obstructions or not. This result is a convincing illustration of the power of topological tools for distributed computability.

**Keywords:** Distributed Systems, Fault-Tolerance, Message-Passing, Synchronous Systems, Coordinated Attack Problem, Two Generals Problem, Two Armies Problem, Consensus, Message adversaries, Omission Faults, Distributed Computability, Simplicial Complexes, Topological methods

## 1 Introduction

### 1.1 Motivations

The Coordinated Attack Problem (also known in the literature as the two generals or the two armies problem) is a long time problem in the area of distributed computing. It is a fictitious situation where two armies have to agree to attack or not on a common enemy that is between them and might capture any of their messengers. Informally, it represents the difficulties to agree in the presence of communication faults. The design of a solution is difficult, sometimes impossible, as it has to address possibly an infinity of lost mutual acknowledgments. It has important applications for the Distributed Databases commit for two processes, see [Gra78]. It was one of the first impossibility results in the area of fault tolerance and distributed computing [AEH75, Gra78].

In the vocabulary of more recent years, this problem can now be stated as the Uniform Consensus Problem for 2 synchronous processes communicating by message passing in the presence of omission faults. It is then a simple instance of a problem that had been very widely studied [AT99, CBGS00, MR98]. See for example [Ray02] for a recent survey about Consensus on synchronous systems with some emphasis on the omissions fault model.

Moreover, given that, if any message can be lost, the impossibility of reaching an agreement is obvious, one can wonder why it has such importance to get a name on its own, and maybe why it has been studied in the first place... The idea is that one has usually to restrict the way the messages are lost in order to keep this problem relevant.

We call an arbitrary pattern of failure by loss of messages a *message adversary*. It will formally describe the fault environment in which the system evolves. For example, the message adversary where any message can be lost at any round except that all messages cannot be lost indefinitely is a special message adversary (any possibility of failure *except one* scenario), for which it is still impossible to solve the Coordinated Attack Problem, but the proof might be less trivial. Given a message adversary, a natural question is whether the Coordinated Attack Problem is solvable against this environment. More generally, the question that arises now is to describe what are exactly the message adversaries for which the Coordinated Attack Problem admit a solution, and for which ones is there no solution. These later message adversaries will be called *obstructions*.

## 1.2 Related Works

The Coordinated Attack Problem is a kind of folklore problem for distributed systems. It seems to appear first in [AEH75] where it is a problem of gangsters plotting for a big job. It is usually attributed to [Gra78], where Jim Gray coined the name “Two Generals Paradox” and put the emphasis on the infinite recursive need for acknowledgments in the impossibility proof.

In textbooks it is often given as an example, however the drastic conditions under which this impossibility result yields are never really discussed, even though for relevancy purpose they are often slightly modified. In [Lyn96], a different problem of Consensus (with a weaker validity condition) is used. In [San06], such a possibility of eternal loss is explicitly ruled out as it would give a trivial impossibility proof otherwise.

This shows that the way the messages may be lost is an important part of the problem definition, hence it is interesting to characterize when the pattern of loss allows to solve the consensus problem or not, *i.e.* whether the fault environment is an obstruction for the Coordinated Attack Problem or not. To our knowledge, this is the first time, this problem is investigated for arbitrary patterns of omission failures, even in the simple case of only two processes. Most notably, it has been addressed for an arbitrary number of processes and for special (quite regular) patterns in [CHLT00,GKP03,Ray02].

A message adversary is oblivious if the set of possible communication pattern is the same at each step. The complete characterization of oblivious message adversaries for which Consensus is solvable has been given in [CGP15]. We consider here also non oblivious adversaries.

Note that while the model has been around for at least tens of years [SW89], the name “message adversary” has been coined only recently by Afek and Gafni in [AG13].

## 1.3 Scope of Application and Contributions

Impossibility results in distributed computing are the more interesting when they are tight, *i.e.* when they give an exact characterization of when a problem is solvable and when it is

not. There are a lot of results regarding distributed tasks when the underlying network is a complete graph and the patterns are simply described by faults (namely in the context of Shared Memory systems), see for example the works in [HS99,SZ00,BG93] where exact topologically-based characterizations is given for the wait-free model.

There are also more recent results when the underlying network can be any arbitrary graph. The results given in [SW07] by Santoro and Widmayer are almost tight. What is worth to note is that the general theorems of [HS99] could not be directly used by this study for the very reason that the failure model for communication networks is not interestingly expressible in the fault model for systems with one to one communication. In the following of [CBS09], we are not interested in the exact cause of a message not being sent/received. We are interested in as general as possible models. See Section 2.3 for a more detailed discussion.

We underline that the omission failures we are studying here encompass networks with *crash failures*, see Example 8. It should also be clear that message adversaries can also be studied in the context of a problem that is not the Consensus Problem. Moreover, as we do not endorse any pattern of failures as being, say, more “realistic”, our technique can be applied for any new patterns of failures.

On the way to a thorough characterization of all obstructions, we address the Consensus problem for a particular but important subclass of failure patterns, namely the ones when no two messages can be lost at the same round. It is long known that the Coordinated Attack Problem is unsolvable if at most one message might be lost at each round [CHLT00,GKP03]. But what happens with strictly weaker patterns was unknown.

Our contribution is the following. In the large subclass of message adversaries where the double simultaneous omission can never happen, we characterize which ones are obstructions for the Coordinated Attack Problem. We give two alternative proofs of this result. One is based on traditional combinatorial techniques, that have been extended to the more involved setting of arbitrary message adversaries. The second one presents an extension of topological techniques suited to arbitrary message adversaries. The combinatorial proof was presented in [FG11]. The topological proof is an improved extract from the Master thesis of one of the authors [Per15] where the notion of terminating subdivision from [GKM14] is applied to the setting of the Coordinated Attack Problem.

More interestingly, the topological characterization gives a nice topological unified explanation of the characterization, which is separated in four different cases in the combinatorial presentation from [FG11]. This result is a convincing illustration of the power of topological tools for distributed computability. Topological tools for distributed computing have sometimes been criticized for being over-mathematically involved. The result presented in this paper shows that distributed computability is inherently linked to topological properties.

But the paper also illustrate some pitfalls of such tools as we could use the given characterization to uncover an error in the main theorem of a paper generalizing the Asynchronous Computability Theorem to arbitrary message adversaries [GKM14]. Mathematically, this error can be traced to the subtle differences between simplicial complexes and abstract simplicial complexes of infinite size. See Remark 29

The outline of the paper is the following. We describe Models and define our Problem in the Section 2. We present numerous examples of application of our terminology and notation in Section 2.5. We then address the characterization of message adversaries without simultaneous faults that are obstructions for the Coordinated Attack Problem in Theorem 19. We give the proofs for necessary condition (impossibility result) in 3.1 and for sufficient condition (explicit algorithm) in 3.2 using the classical bivalency techniques. We then prove the same results in a topological way in section 4. Finally, the two results are compared and we show how the topological explanation gives more intuition about the result.

## 2 Models and Definitions

### 2.1 The Coordinated Attack Problem

**A folklore problem** Two generals have gathered forces on top of two facing hills. In between, in the valley, their common enemy is entrenched. Every day each general sends a messenger to the other through the valley. However this is risky as the enemy may capture them. Now they need to get the last piece of information: are they *both* ready to attack?

This two army problem was originated by [AEH75] and then by Gray [Gra78] when modeling the distributed database commit. It corresponds to the binary consensus with two processes in the omission model. If there is no restriction on the fault environments, then any messenger may be captured. And if any messenger may be captured, then consensus is obviously impossible: the enemy can succeed in capturing all of them, and without communication, no distributed algorithm.

**Possible Environments** Before trying to address what can be, in some sense, the most relevant environments, we will describe different environments in which the enemy cannot be so powerful as to be able to capture any messenger.

This is a list of possible environments, using the same military analogy. The generals name are *White* and *Black*.

- (1) no messenger is captured
- (2) messengers from *General White* may be captured
- (3) messengers from *General Black* may be captured
- (4) messengers from one general are at risk, and if one of them is captured, all the following will also be captured (the enemy got the secret “Code of Operations” for this general from the first captured messenger)
- (5) messengers from one general are at risk (the enemy could manage to infiltrate a spy in one of the armies)
- (6) at most one messenger may be captured each day (the enemy can’t closely watch both armies on the same day)
- (7) any messenger may be captured

Which ones are (trivial) obstructions, and which are not? Nor obstruction, nor trivial? What about more complicated environments?

## 2.2 The Binary Consensus Problem

A set of synchronous processes wish to agree about a binary value. This problem was first identified and formalized by Lamport, Shostak and Pease [PSL80]. Given a set of processes, a consensus protocol must satisfy the following properties for any combination of initial values [Lyn96]:

- *Termination*: every process decides some value.
- *Validity*: if all processes initially propose the same value  $v$ , then every process decides  $v$ .
- *Agreement*: if a process decides  $v$ , then every process decides  $v$ .

Consensus with such a termination and decision requirement for every process is more precisely referred to as the *Uniform Consensus*, see [Ray02] for example for a discussion. Given a fault environment, the natural questions are : is the Consensus solvable, if it is solvable, what is the minimal complexity?

## 2.3 The Communication Model

In this paper, the system we consider is a set  $\Pi$  of only 2 processes named *white* and *black*,  $\Pi = \{\circ, \bullet\}$ . The processes evolve in *synchronized* rounds. In each round  $r$ , every process  $p$  executes the following steps:  $p$  sends a message to the other process, receives a messages  $M$  from the other process, and then updates its state according to the received message.

The messages sent are, or are not, delivered according to the environment in which the distributed computation takes place. This environment is described by a *message adversary*. Such an adversary models exactly when some messages sent by a process to the other process may be lost at some rounds. We will consider arbitrary message adversaries, they will be represented as arbitrary sets of infinite sequences of combinations of communication rounds.

In the following of [CBS09], we are not interested in the exact cause of a message not being sent/received. We only refer to the phenomenon: the content of messages being transmitted or not. The fact that the adversaries are arbitrary means that we do not endorse any metric to count the number of “failures” in the system. There are metrics that count the number of lost messages, that count the number of process that can lose messages (both in send and receive actions). Other metrics count the same parameters but only during a round of the system. The inconvenient of this metric-centric approach is that, even when restricted only to omission faults, it can happen that some results obtained on complete networks are not usable on arbitrary networks. Because in, say, a ring network, you are stating in some sense that every node is faulty. See eg [SW07] for a very similar discussion, where Santoro and Widmayer, trying to solve some generalization of agreement problems in general networks could not use directly the known results in complete networks.

As said in the introduction, the Coordinated Attack Problem is nowadays stated as the Uniform Consensus Problem for 2 synchronous processes communicating by message passing in the presence of omission faults. Nonetheless, we will use Consensus and Uniform Consensus interchangeably in this paper. We emphasize that, because we do not assign omission faults to any process (see previous discussion), there are only correct processes.

## 2.4 Message Adversaries

We introduce and present here our notation.

**Definition 1.** We denote by  $\mathcal{G}_2$  the set of directed graphs with vertices in  $\Pi$ .

$$\mathcal{G}_2 = \{\circ \leftrightarrow \bullet, \circ \leftarrow \bullet, \circ \rightarrow \bullet, \circ \text{---} \bullet\}$$

We denote by  $\Gamma$  the following subset of  $\mathcal{G}_2$

$$\Gamma = \{\circ \leftrightarrow \bullet, \circ \leftarrow \bullet, \circ \rightarrow \bullet\}.$$

At a given round, there are only four combinations of communication. Those elements describe what can happen at a given round with the following straightforward semantics:

- $\circ \leftrightarrow \bullet$ , no process loses messages
- $\circ \leftarrow \bullet$ , the message of process  $\circ$ , if any, is not transmitted
- $\circ \rightarrow \bullet$ , the message of process  $\bullet$ , if any, is not transmitted
- $\circ \text{---} \bullet$ , both messages, if any, are not transmitted<sup>1</sup>.

The terminology *message adversary* has been introduced in [AG13], but the concept is way older.

**Definition 2.** A message adversary over  $\Pi$  is a set of infinite sequences of elements of  $\mathcal{G}_2$ .

We will use the standard following notations in order to describe more easily our message adversaries [PP04]. A (infinite) sequence is seen as a (infinite) word over the alphabet  $\mathcal{G}_2$ . The empty word is denoted by  $\varepsilon$ .

**Definition 3.** Given  $A \subset \mathcal{G}_2$ ,  $A^*$  is the set of all finite sequences of elements of  $A$ ,  $A^\omega$  is the set of all infinite ones and  $A^\infty = A^* \cup A^\omega$ .

An adversary of the form  $A^\omega$  is called an *oblivious adversary*. A word in  $L \subset \mathcal{G}_2^\omega$  is called a *communication scenario* (or *scenario* for short) of message adversary  $L$ . Given a word  $w \in \mathcal{G}_2^*$ , it is called a *partial scenario* and  $\text{len}(w)$  is the length of this word.

Intuitively, the letter at position  $r$  of the word describes whether there will be, or not, transmission of the message, if one is being sent at round  $r$ . A formal definition of an execution under a scenario will be given in Section 2.6.

We recall now the definition of the prefix of words and languages. A word  $u \in \mathcal{G}_2^*$  is a prefix for  $w \in \mathcal{G}_2^*$  (resp.  $w' \in \mathcal{G}_2^\omega$ ) if there exist  $v \in \mathcal{G}_2^*$  (resp.  $v' \in \mathcal{G}_2^\omega$ ) such that  $w = uv$  (resp.  $w' = uv'$ ).

Given  $w \in \mathcal{G}_2^\omega$  and  $r \in \mathbb{N}$ ,  $w|_r$  is the prefix of size  $r$  of  $w$ .

**Definition 4.** Let  $w \in \mathcal{G}_2^*$ , then  $\text{Pref}(w) = \{u \in \mathcal{G}_2^* | u \text{ is a prefix of } w\}$ . Let  $L \subset \mathcal{G}_2^*$ , let  $r \in \mathbb{N}$ ,  $\text{Pref}_r(L) = \{w|_r | w \in L\}$  and  $\text{Pref}(L) = \bigcup_{w \in L} \text{Pref}(w) = \bigcup_{r \in \mathbb{N}} \text{Pref}_r(L)$ .

<sup>1</sup> In order to increase the readability, we note  $\circ \text{---} \bullet$  instead of  $\circ \bullet$ , the double-omission case.

## 2.5 Examples

We do not restrict our study to regular languages, however all message adversaries we are aware of are regular, as can be seen in the following examples, where the rational expressions prove to be very convenient.

We show how standard fault environments are conveniently described in our framework.

*Example 5.* Consider a system where, at each round, up to 2 messages can be lost. The associated message adversary is  $\mathcal{G}_2^\omega$ .

*Example 6.* Consider a system where, at each round, only one message can be lost. The associated message adversary is  $\{\circ\leftrightarrow\bullet, \circ\leftarrow\bullet, \circ\rightarrow\bullet\}^\omega = \Gamma^\omega$ .

*Example 7.* Consider a system where at most one of the processes can lose messages. The associated adversary is the following:

$$S_1 = \{\circ\leftrightarrow\bullet, \circ\leftarrow\bullet\}^\omega \cup \{\circ\leftrightarrow\bullet, \circ\rightarrow\bullet\}^\omega$$

*Example 8.* Consider a system where at most one of the processes can crash, For the phenomena point of view, this is equivalent to the fact that at some point, no message from a particular process will be transmitted. The associated adversary is the following:

$$C_1 = \{\circ\leftrightarrow\bullet\}^\omega \cup \{\circ\leftrightarrow\bullet\}^*(\{\circ\leftarrow\bullet^\omega, \circ\rightarrow\bullet^\omega\})$$

*Example 9.* Finally the seven simple cases exposed by the possible environments described in Section 2.1 are described formally as follows:

$$S_0 = \{\circ\leftrightarrow\bullet\}^\omega \tag{1}$$

$$T_\circ = \{\circ\leftrightarrow\bullet, \circ\leftarrow\bullet\}^\omega \tag{2}$$

$$T_\bullet = \{\circ\leftrightarrow\bullet, \circ\rightarrow\bullet\}^\omega \tag{3}$$

$$C_1 = \{\circ\leftrightarrow\bullet\}^\omega \cup \{\circ\leftrightarrow\bullet\}^*(\{\circ\leftarrow\bullet^\omega, \circ\rightarrow\bullet^\omega\}) \tag{4}$$

$$S_1 = \{\circ\leftrightarrow\bullet, \circ\leftarrow\bullet\}^\omega \cup \{\circ\leftrightarrow\bullet, \circ\rightarrow\bullet\}^\omega = T_\circ \cup T_\bullet \tag{5}$$

$$R_1 = \Gamma^\omega \tag{6}$$

$$S_2 = \mathcal{G}_2^\omega \tag{7}$$

Note that the fourth and first cases correspond respectively to the synchronous crash-prone model [Lyn96] and to the 1-resilient model [Ada03,GKP03]. Even though in our definition, sets of possible scenarios could be arbitrary, it seems that all standard models (including crash-based models) can be described using only regular expressions.

## 2.6 Execution of a Distributed Algorithm

Given a message adversary  $L$ , we define what is a run of a given algorithm  $\mathcal{A}$  subject to  $L$ .

An execution, or run, of an algorithm  $\mathcal{A}$  under scenario  $w \in L$  is the following. At round  $r \in \mathbb{N}$ , messages are sent (or not) by the processes. The fact that the corresponding receive action will be successful depends on  $a$ , the  $r$ -th letter of  $w$ .

- if  $a = \circ \leftrightarrow \bullet$ , then all messages, if any, are correctly delivered,
- if  $a = \circ \leftarrow \bullet$ , then the message of process  $\circ$  is not transmitted (the receive call of  $\bullet$ , if any at this round, returns *null*),
- if  $a = \bullet \leftarrow \circ$ , then the message of process  $\bullet$  is not transmitted (the receive call of  $\circ$ , if any at this round, returns *null*),
- if  $a = \circ \rightarrow \bullet$ , no messages is transmitted.

Then, both processes updates their state according to  $\mathcal{A}$  and the value received. An execution is a (possibly infinite) sequence of such messages exchanges and corresponding local states.

Given  $u \in \text{Pref}(w)$ , we denote by  $s^p(u)$  the state of process  $p$  at the  $\text{len}(u)$ -th round of the algorithm  $\mathcal{A}$  under scenario  $w$ . This means in particular that  $s^p(\varepsilon)$  represents the initial state of  $p$ , where  $\varepsilon$  denotes the empty word.

Finally and classically,

**Definition 10.** *An algorithm  $\mathcal{A}$  solves the Coordinated Attacked Problem for the message adversary  $L$  if for any scenario  $w \in L$ , there exist  $u \in \text{Pref}(w)$  such that the states of the two processes ( $s^\circ(u)$  and  $s^\bullet(u)$ ) satisfy the three conditions of Section 2.2.*

**Definition 11.** *A message adversary  $L$  is said to be solvable if there exist an algorithm that solves the Coordinated Attacked Problem for  $L$ . It is said to be an obstruction otherwise.*

*A message adversary  $L$  is a (inclusion) minimal obstruction if any  $L' \subsetneq L$  is solvable.*

## 2.7 Index of a Scenario

We will use the following integer function of scenarios that will be proved to be a useful encoding of all the important properties of a given message adversary by mapping scenarios in  $\Gamma^r$  to integers in  $[0, 3^r - 1]$ . The intuition to this function will become clearer from the topological point of view in Section 4. By induction, we define the following integer index given  $w \in \Gamma^*$ . First, we define  $\mu$  on  $\Gamma$  by

- $\mu(\circ \rightarrow \bullet) = -1$ ,
- $\mu(\circ \leftrightarrow \bullet) = 0$ .
- $\mu(\bullet \leftarrow \circ) = 1$ ,

**Definition 12.** *Let  $w \in \Gamma^*$ . We define  $\text{ind}(\varepsilon) = 0$ . If  $\text{len}(w) \geq 1$ , then we have  $w = ua$  where  $u \in \Gamma^*$  and  $a \in \Gamma$ . In this case, we define*

$$\text{ind}(w) := 3\text{ind}(u) + (-1)^{\text{ind}(u)}\mu(a) + 1.$$

Let  $n \in \mathbb{N}$ , define  $\text{ind}_n: \Gamma^n \rightarrow [0, 1]$  and  $\overline{\text{ind}}: \Gamma^\omega \rightarrow [0, 1]$  to be respectively the normalization of  $\text{ind}$  and the limit index of the infinite scenarios :

- $\forall w \in \Gamma^n \quad \text{ind}_n(w) = \frac{\text{ind}(w)}{3^n}$
- $\forall w \in \Gamma^\omega \quad \overline{\text{ind}}(w) = \lim_{n \rightarrow +\infty} \text{ind}_n(w|_n)$



The convergence for  $\overline{ind}$  is obvious from the following lemma.

**Lemma 13.** *Let  $r \in \mathbb{N}$ . The application  $ind$  is a bijection from  $\Gamma^r$  to  $\llbracket 0, 3^r - 1 \rrbracket$ .*

*Proof.* The lemma is proved by a simple induction. If  $r = 0$ , then the property holds.

Let  $r > 0$ . Suppose the property is satisfied for  $r - 1$ . Given  $w \in \Gamma^r$ , we have  $w = ua$  with  $u \in \Gamma^{r-1}$  and  $a \in \Gamma$ . From  $ind(w) = 3ind(u) + (-1)^{ind(u)}\mu(a) + 1$  and the induction hypothesis, we get immediately that  $0 \leq ind(w) \leq 3^r - 1$ .

Now, we need only to prove injectivity. Suppose there are  $w, w' \in \Gamma^r$  such that  $ind(w) = ind(w')$ . So there are  $u, u' \in \Gamma^{r-1}$  and  $a, a' \in \Gamma$  such that

$$3ind(u) + (-1)^{ind(u)}\mu(a) + 1 = 3ind(u') + (-1)^{ind(u')}\mu(a') + 1.$$

Then

$$3(ind(u) - ind(u')) = (-1)^{ind(u')}\mu(a') - (-1)^{ind(u)}\mu(a).$$

Remarking that the right hand side of this integer equality has an absolute value that can be at most 2, we finally get

$$\begin{aligned} ind(u) &= ind(u') \\ (-1)^{ind(u)}\mu(a) &= (-1)^{ind(u')}\mu(a') \end{aligned}$$

By induction hypothesis, we get that  $u = u'$  and  $a = a'$ . Hence  $w = w'$ , and  $ind$  is injective, therefore bijective from  $\Gamma^r$  onto  $\llbracket 0, 3^r - 1 \rrbracket$ .

Two easy calculations give

**Proposition 14.** *Let  $r \in \mathbb{N}$ ,  $ind(\circ \rightarrow \bullet^r) = 0$  and  $ind(\circ \leftarrow \bullet^r) = 3^r - 1$ .*

In Figure 1, the indexes for words of length at most 2 are given.

word of length 1	$\circ \rightarrow \bullet$	$\circ \leftrightarrow \bullet$	$\circ \leftarrow \bullet$
index	0	1	2

word of length 2	$\circ \rightarrow \bullet \circ \rightarrow \bullet$	$\circ \rightarrow \bullet \circ \leftrightarrow \bullet$	$\circ \rightarrow \bullet \circ \leftarrow \bullet$
index	0	1	2

word of length 2	$\circ \leftrightarrow \bullet \circ \rightarrow \bullet$	$\circ \leftrightarrow \bullet \circ \leftrightarrow \bullet$	$\circ \leftrightarrow \bullet \circ \leftarrow \bullet$
index	5	4	3

word of length 2	$\circ \leftarrow \bullet \circ \rightarrow \bullet$	$\circ \leftarrow \bullet \circ \leftrightarrow \bullet$	$\circ \leftarrow \bullet \circ \leftarrow \bullet$
index	6	7	8

Fig. 1: Indexes for some short words

We now describe precisely what are the words whose indexes differs by only 1. We have two cases, either they have the same prefix and different last letter or different prefix and same last letter.

**Lemma 15.** *Let  $r \in \mathbb{N}$ , and  $v, v' \in \Gamma^r$ . Then  $\text{ind}(v') = \text{ind}(v) + 1$  if and only if one of the following conditions holds:*

- 15.i  $\text{ind}(v)$  is even and
  - either there exist  $u \in \Gamma^{r-1}$ , and  $v = u \circ \leftarrow \bullet$ ,  $v' = u \circ \rightarrow \bullet$ ,
  - either there exist  $u, u' \in \Gamma^{r-1}$ , and  $v = u \circ \leftarrow \bullet$ ,  $v' = u' \circ \leftarrow \bullet$ , and  $\text{ind}(u') = \text{ind}(u) + 1$ .
- 15.ii  $\text{ind}(v)$  is odd and
  - either there exist  $u \in \Gamma^{r-1}$ , and  $v = u \circ \leftarrow \bullet$ ,  $v' = u \circ \leftarrow \bullet$ ,
  - either there exist  $u, u' \in \Gamma^{r-1}$ , and  $v = u \circ \rightarrow \bullet$ ,  $v' = u' \circ \rightarrow \bullet$ , and  $\text{ind}(u') = \text{ind}(u) + 1$ .

*Proof.* The lemma is proved by a induction. If  $r = 0$ , then the property holds. Let  $r \in \mathbb{N}^*$ . Suppose the property is satisfied for  $r - 1$ .

Suppose there are  $w, w' \in \Gamma^r$  such that  $\text{ind}(w') = \text{ind}(w) + 1$ . So there are  $u, u' \in \Gamma^{r-1}$  and  $a, a' \in \Gamma$  such that

$$3\text{ind}(u) + (-1)^{\text{ind}(u)}\mu(a) + 2 = 3\text{ind}(u') + (-1)^{\text{ind}(u')}\mu(a') + 1.$$

Then

$$3(\text{ind}(u) - \text{ind}(u')) = (-1)^{\text{ind}(u')}\mu(a') - (-1)^{\text{ind}(u)}\mu(a) - 1.$$

Remarking again that this is an integer equality, we then have

- either  $(\text{ind}(u) = \text{ind}(u'))$  and  $(-1)^{\text{ind}(u')}\mu(a') = (-1)^{\text{ind}(u)}\mu(a) + 1$ ,
- either  $(\text{ind}(u') = \text{ind}(u) + 1)$  and  $(-1)^{\text{ind}(u')}\mu(a') = (-1)^{\text{ind}(u)}\mu(a) - 2$ .

This yields the following cases:

1.  $\text{ind}(u) = \text{ind}(u')$  is even,  $a = \circ \leftarrow \bullet$  and  $a' = \circ \leftarrow \bullet$ ,
2.  $\text{ind}(u) = \text{ind}(u')$  is odd,  $a = \circ \rightarrow \bullet$  and  $a' = \circ \leftarrow \bullet$ ,
3.  $\text{ind}(u) = \text{ind}(u')$  is even,  $a = \circ \leftarrow \bullet$  and  $a' = \circ \rightarrow \bullet$ ,
4.  $\text{ind}(u) = \text{ind}(u')$  is odd,  $a = \circ \leftarrow \bullet$  and  $a' = \circ \leftarrow \bullet$ ,
5.  $\text{ind}(u') = \text{ind}(u) + 1$ ,  $\text{ind}(u)$  is even,  $a = \circ \leftarrow \bullet = a'$ ,
6.  $\text{ind}(u') = \text{ind}(u) + 1$ ,  $\text{ind}(u)$  is odd,  $a = \circ \rightarrow \bullet = a'$ ,

Getting all the pieces together, we get the results of the lemma.

Given an algorithm  $\mathcal{A}$ , we have this fundamental corollary, that explicit the uncertainty process can experience between two executions whose indexes differ by only 1: one of the process is in the same state in both cases. Which process it is depends on the parity. In other word, when the first index is even,  $\bullet$  cannot distinguish the two executions, when it is odd, this is  $\circ$  that cannot distinguish the two executions.

**Corollary 16.** *Let  $v, v' \in \Gamma^r$  such that  $\text{ind}(v') = \text{ind}(v) + 1$ . Then,*

- 16.i *if  $\text{ind}(v)$  is even then  $s^\bullet(v) = s^\bullet(v')$ ,*
- 16.ii *if  $\text{ind}(v)$  is odd then  $s^\circ(v) = s^\circ(v')$ .*

*Proof.* We prove the result using Lemma 15 and remarking that either a process receives a message from the other process being in the same state in the preceding configuration  $u$ ; either it receives no message when the state of the other process actually differ.

## 2.8 Characterization

We prove that a message adversary  $L \subset \Gamma^\omega$  is solvable if and only if it does not contain a fair scenario or a special pair of unfair scenarios. We define the following set to help describe the special unfair pairs.

**Definition 17.** A scenario  $w \in \mathcal{G}_2^\omega$  is unfair if  $w \in \mathcal{G}_2^*(\{\circ \dashrightarrow \bullet, \circ \dashleftarrow \bullet\}^\omega \cup \{\circ \dashrightarrow \bullet, \circ \dashrightarrow \bullet\}^\omega)$ . The set of fair scenarios of  $\Gamma^\omega$  is denoted by  $Fair(\Gamma^\omega)$ .

In words, in an *unfair* scenario, there is one or more processes for which the messages are indefinitely lost at some point. And in a *fair* scenario there is an infinity of messages that are received from both processes.

**Definition 18.** We define special pairs as  $SPair(\Gamma^\omega) = \{(w, w') \in \Gamma^\omega \times \Gamma^\omega \mid w \neq w', \forall r \in \mathbb{N} | ind(w|_r) - ind(w'|_r) | \leq 1\}$ .

**Theorem 19.** Let  $L \subset \Gamma^\omega$ , then Consensus is solvable for message adversary  $L$  if and only if  $L \in \{\mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3 \cup \mathcal{F}_4\}$  where

- 19.i  $\mathcal{F}_1 = \{L \subset \Gamma^\omega \mid \exists f \in Fair(\Gamma^\omega) \wedge f \notin L\}$
- 19.ii  $\mathcal{F}_2 = \{L \subset \Gamma^\omega \mid \exists (w, w') \in SPair(\Gamma^\omega) \wedge w, w' \notin L\}$
- 19.iii  $\mathcal{F}_3 = \{L \subset \Gamma^\omega \mid \circ \dashrightarrow \bullet^\omega \notin L\}$
- 19.iv  $\mathcal{F}_4 = \{L \subset \Gamma^\omega \mid \circ \dashleftarrow \bullet^\omega \notin L\}$

We have split the set of solvable scenarios in four families for a better understanding even though it is clear that they largely intersect.  $\mathcal{F}_1$  contains every scenarios for which at least one unfair scenario cannot occur ; in  $\mathcal{F}_2$  both elements of a special pair cannot occur ; finally  $\mathcal{F}_3$  and  $\mathcal{F}_4$  contains every scenarios for which at least one message is received from both processes.

We present two proofs of Theorem 19 in the following sections.

## 2.9 Application to the Coordinated Attack Problem

We consider now our question on the seven examples of Example 9.

The answer to possibility is obvious for the first and last cases. In the first three cases,  $S_0$ ,  $\{\circ \leftrightarrow \bullet, \circ \dashleftarrow \bullet\}^\omega$  or  $\{\circ \leftrightarrow \bullet, \circ \dashrightarrow \bullet\}^\omega$ , Consensus can be reached in one day (by deciding the initial value of  $\circ, \bullet$ , and  $\circ$  respectively). The fourth and fifth cases are a bit more difficult but within reach of our Theorem 19. We remark that the scenario  $\circ \dashleftarrow \bullet \circ \dashrightarrow \bullet \circ \leftrightarrow \bullet^\omega$  is a fair scenario that does not belong to  $C_1$ , nor  $S_1$ . Therefore, those are solvable cases also. In the last case, consensus can't be achieved [Gra78, Lyn96], as said before.

The following observation is also a way to derive lower bounds from computability results.

**Proposition 20.** Let  $O \subset \Gamma^\omega$  an obstruction for the Consensus problem. Let  $L \subset \Gamma^\omega$  and  $r \in \mathbb{N}$  such that  $Pref_r(O) \subset Pref_r(L)$  then, Consensus can not be solved in  $L$  in less than  $r$  steps.

Indeed if every prefixes of length  $r$  of  $O$  in which Consensus is unsolvable are also prefixes of  $L$ , then after  $r$  rounds of any scenario  $w \in L$ , processes solving Consensus in  $L$  would be mean it is also solvable in  $O$ .

Now, using Proposition 20 for the fourth and fifth cases of Example 9 yields the following summary by remarking that their first round are exactly the same as  $\Gamma^\omega$ .

1.  $S_0$  is solvable in 1 round,
2.  $T_\circ$  is solvable in 1 round,
3.  $T_\bullet$  is solvable in 1 round,
4.  $C_1$  is solvable in exactly 2 rounds,
5.  $S_1$  is solvable in exactly 2 rounds.

## 2.10 About Minimal Obstructions

Theorem 19 shows that, even in the simpler subclass where no double omission are permitted, simple inclusion-minimal adversaries may not exist. Indeed, there exists a sequence of unfair scenarios  $(u_i)_{i \in \mathbb{N}}$  such that  $\forall i, j, (u_i, u_j)$  is not a special pair. Therefore  $L_n = \Gamma^\omega \setminus \bigcup_{0 \leq i \leq n} u_i$  defines an infinite decreasing sequence of obstructions for the Coordinated Attack Problem.

Considering the set of words in  $SPair(\Gamma^\omega)$ , it is possible by picking up only one member of a special pair to have an infinite set  $U$  of unfair scenarios such that, by Theorem 19, the adversary  $\Gamma^\omega \setminus U$  is a minimal obstruction. So there is no minimum obstruction.

As a partial conclusion, we shall say that the well known adversary  $\Gamma^\omega$ , even not being formally a minimal obstruction, could be considered, without this being formally defined, as the smallest example of a *simple obstruction*, as it is more straightforward to describe than, say the adversaries  $\Gamma^\omega \setminus U$  above.

This probably explains why the other obstructions we present here have never been investigated before.

## 3 Combinatorial Characterization of Solvable Adversaries

In this part, we consider the adversaries without double omission, that is the adversaries  $L \subset \Gamma^\omega$  and we characterize exactly which one are solvable. When the consensus is achievable, we also give an effective algorithm. In this section, we rely on a classical combinatorial bivalency technique [FLP85].

### 3.1 Necessary Condition: a Bivalency Impossibility Proof

We will use a standard, self-contained, bivalency proof technique. We suppose now on that there is an algorithm  $\mathcal{A}$  to solve Consensus on  $L$ .

We proceed by contradiction. So we suppose that all the conditions of Theorem 19 are not true, *i.e.* that  $Fair(\Gamma^\omega) \cup \{\circ \rightarrow \bullet^\omega, \circ \leftarrow \bullet^\omega\} \subset L$ , and that for all  $(w, w') \in SPair(\Gamma^\omega), w \notin L \implies w' \in L$ .

**Definition 21.** Given an initial configuration, let  $v \in \text{Pref}(L)$  and  $i \in \{0, 1\}$ . The partial scenario  $v$  is said to be  $i$ -valent if, for all scenario  $w \in L$  such that  $v \in \text{Pref}(w)$ ,  $\mathcal{A}$  decides  $i$  at the end. If  $v$  is not 0-valent nor 1-valent, then it is said bivalent.

By hypothesis,  $\circ \leftarrow \bullet^\omega, \circ \rightarrow \bullet^\omega \in L$ . Consider our algorithm  $\mathcal{A}$  with input 0 on both processes running under scenario  $\circ \leftarrow \bullet^\omega$ . The algorithm terminates and has to output 0 by Validity Property. Similarly, both processes output 1 under scenario  $\circ \rightarrow \bullet^\omega$  with input 1 on both processes.

From now on, we have this initial configuration  $I$ : 0 on process  $\circ$  and 1 on process  $\bullet$ . For  $\circ$  there is no difference under scenario  $\circ \rightarrow \bullet^\omega$  for this initial configurations and the previous one. Hence,  $\mathcal{A}$  will output 0 for scenario  $\circ \rightarrow \bullet^\omega$ . Similarly, for  $\bullet$  there is no difference under scenario  $\circ \leftarrow \bullet^\omega$  for both considered initial configurations. Hence,  $\mathcal{A}$  will output 1 for this other scenario. Hence  $\varepsilon$  is bivalent for initial configuration  $I$ . In the following, valency will always be implicitly defined with respect to the initial configuration  $I$ .

**Definition 22.** Let  $v \in \text{Pref}(L)$ ,  $v$  is decisive if

- 22.i  $v$  is bivalent,
- 22.ii For any  $a \in \Gamma$  such that  $va \in \text{Pref}(L)$ ,  $va$  is not bivalent.

**Lemma 23.** There exist a decisive  $v \in \text{Pref}(L)$ .

*Proof.* We suppose this not true. Then, as  $\varepsilon$  is bivalent, it is possible to construct  $w \in \Gamma^\omega$  such that,  $\text{Pref}(w) \subset \text{Pref}(L)$  and for any  $v \in \text{Pref}(w)$ ,  $v$  is bivalent. Bivalency for a given  $v$  means in particular that the algorithm  $\mathcal{A}$  has not stopped yet. Therefore  $w \notin L$ .

This means that  $w$  is unfair, because from initial assumption,  $\text{Fair}(\Gamma^\omega) \subset L$ . Then this means that, w.l.o.g we have  $u \in \Gamma^+$ , such that  $w = u \circ \leftarrow \bullet^\omega$  and  $\text{ind}(u)$  is even. We denote by  $w' = \text{ind}^{-1}(\text{ind}(u) - 1) \circ \leftarrow \bullet^\omega$ . The couple  $(w, w')$  is a special pair. Therefore  $w'$  belongs to  $L$ , so  $\mathcal{A}$  halts at some round  $r_0$  under scenario  $w'$ .

By Corollary 16,  $s_\bullet(w|_{r_0}) = s_\bullet(w'|_{r_0})$  This means that  $w|_{r_0}$  is not bivalent. As  $w|_{r_0} \in \text{Pref}(L)$  this gives a contradiction.

We can now end the impossibility proof.

*Proof.* Consider a decisive  $v \in \text{Pref}(L)$ . By definition, this means that there exist  $a, b \in \Gamma$ , with  $a \neq b$ , such that  $va, vb \in \text{Pref}(L)$  and  $va$  and  $vb$  are not bivalent and are of different valencies. Obviously there is an extension that has a different valency of the one of  $\circ \leftrightarrow \bullet$  and w.l.o.g., we choose  $b$  to be  $\circ \leftrightarrow \bullet$ . Therefore  $a = \circ \leftarrow \bullet$  or  $a = \circ \rightarrow \bullet$ . We terminate by a case by case analysis.

Suppose that  $a = \circ \leftarrow \bullet$ , and  $\text{ind}(v)$  is even. By Corollary 16, this means  $\bullet$  is in the same state after  $va$  and  $vb$ . We consider the scenarios  $va \circ \leftarrow \bullet^\omega$  and  $vb \circ \leftarrow \bullet^\omega$ , they forms a special pair. So one belongs to  $L$  by hypothesis, therefore both processes should halt at some point under this scenario. However, the state of  $\bullet$  is always the same under the two scenarios because it receives no message at all so if it halts and decides some value, we get a contradiction with the different valencies.

Other cases are treated similarly using the other cases of Corollary 16.

### 3.2 A Consensus Algorithm

Given a word  $w$  in  $\Gamma^\omega$ , we define the following algorithm  $\mathcal{A}_w$  (see Algorithm 1). It has messages always of the same type. They have two components, the first one is the initial bit, named *init*. The second is an integer named *ind*. Given a message  $msg$ , we note  $msg.init$  (resp.  $msg.ind$ ) the first (resp. the second) component of the message.

#### Algorithm 1: Consensus Algorithm $\mathcal{A}_w$ for Process $p$ :

```

Data:  $w \in \Gamma^\omega$ 
Input:  $init \in \{0, 1\}$ 
1   $r=0$ ;
2   $initother=null$ ;
3  if  $p = \circ$  then
4     $ind=0$ ;
5  else
6     $ind=1$ ;
7  while  $|ind - ind(w|_r)| \leq 2$  do
8     $msg = (init, ind)$ ;
9     $send(msg)$ ;
10    $msg = receive()$ ;
11   if  $msg == null$  then // message was lost
12      $ind = 3 * ind$ ;
13   else
14      $ind = 2 * msg.ind + ind$ ;
15      $initother = msg.init$ ;
16    $r=r+1$ ;
17 if  $p = \circ$  then
18   if  $ind < ind(w|_r)$  then
19     Output:  $init$ 
20   else
21     Output:  $initother$ 
22 else
23   if  $ind > ind(w|_r)$  then
24     Output:  $init$ 
25   else
26     Output:  $initother$ 

```

We prove that the *ind* values computed by each process in the algorithm differ by only one. Moreover, we show that the actual index is equal to the minimum of the *ind* values.

More precisely, with  $sign(n)$  being  $+1$  (resp.  $-1$ ) when  $n \in \mathbb{Z}$  is positive (resp. negative), we have

**Proposition 24.** *For any round  $r$  of an execution of Algorithm  $\mathcal{A}_w$  under scenario  $v \in \Gamma^r$ , such that no process has already halted,*

$$\begin{cases} |ind_r^\bullet - ind_r^\circ| = 1, \\ sign(ind_r^\bullet - ind_r^\circ) = (-1)^{ind(v)}, \\ ind(v) = \min\{ind_r^\circ, ind_r^\bullet\}. \end{cases}$$

*Proof.* We prove the result by induction over  $r \in \mathbb{N}$ .

For  $r = 0$ , the equations are satisfied.

Suppose the property is true for  $r - 1$ . We consider a round of the algorithm. Let  $u \in \Gamma^{r-1}$  and  $a \in \Gamma$ , and consider an execution under environment  $w = ua$ . There are exactly three cases to consider.

Suppose  $a = \circ \leftrightarrow \bullet$ . Then it means both messages are received and  $ind_r^\circ = 2ind_{r-1}^\bullet + ind_{r-1}^\circ$  and  $ind_r^\bullet = 2ind_{r-1}^\circ + ind_{r-1}^\bullet$ . Hence  $ind_r^\bullet - ind_r^\circ = ind_{r-1}^\circ - ind_{r-1}^\bullet$ . Thus, using the recurrence property, we get  $|ind_r^\bullet - ind_r^\circ| = 1$ .

Moreover, by construction

$$\begin{aligned} ind(v) &= ind(ua) \\ &= 3ind(u) + (-1)^{ind(u)}(\mu(a)) + 1 \\ &= 3ind(u) + 1 \end{aligned}$$

The two indices  $ind(u)$  and  $ind(v)$  are therefore of opposite parity. Hence, by induction property,  $sign(ind_r^\bullet - ind_r^\circ) = (-1)^{ind(v)}$ . And remarking that  $\min\{ind_r^\circ, ind_r^\bullet\} =$

$$\begin{aligned} &\min\{2ind_{r-1}^\bullet + ind_{r-1}^\circ, 2ind_{r-1}^\circ + ind_{r-1}^\bullet\} \\ &= ind_{r-1}^\bullet + ind_{r-1}^\circ + \min\{ind_{r-1}^\circ, ind_{r-1}^\bullet\} \\ &= 2\min\{ind_{r-1}^\circ, ind_{r-1}^\bullet\} + |ind_{r-1}^\bullet - ind_{r-1}^\circ| \\ &\quad + \min\{ind_{r-1}^\circ, ind_{r-1}^\bullet\} \\ &= 3\min\{ind_{r-1}^\circ, ind_{r-1}^\bullet\} + 1 \end{aligned}$$

we get that the third equality is also verified in round  $r$ .

Consider now the case  $a = \circ \rightarrow \bullet$ . Then  $\circ$  gets no message from  $\bullet$  and  $\bullet$  gets a message from  $\circ$ . So we have that  $ind_r^\circ = 3ind_{r-1}^\circ$  and  $ind_r^\bullet = 2ind_{r-1}^\circ + ind_{r-1}^\bullet$ . So we have that  $ind_r^\bullet - ind_r^\circ = ind_{r-1}^\bullet - ind_{r-1}^\circ$ . The first equality is satisfied.

We also have that  $ind(v) = 3ind(u) + (-1)^{ind(u)}\mu(\circ \rightarrow \bullet) + 1 = 3ind(u) + \alpha$ , with  $\alpha = 0$  if  $ind(u)$  is even and  $\alpha = 2$  otherwise. Hence,  $ind(u)$  and  $ind(v)$  are of the same parity, and the second equality is also satisfied.

Finally, we have that  $\min\{ind_r^\circ, ind_r^\bullet\} = \{3ind_{r-1}^\circ, 2ind_{r-1}^\circ + ind_{r-1}^\bullet\}$ . If  $ind(u)$  is even, then  $ind(u) = ind_{r-1}^\circ$  and the equality holds. If  $ind(u)$  is odd, then  $ind(u) = ind_{r-1}^\bullet$  and the equality also holds.

The case  $a = \circ \leftarrow \bullet$  is a symmetric case and is proved similarly.

### 3.3 Correctness of the Algorithm

Given a message adversary  $L$ , we suppose that one of the following holds.

- $\exists f \in Fair(\Gamma^\omega), f \notin L$ ,
- $\exists (u, u') \in SPair(\Gamma^\omega), u, u' \notin L$ ,
- $\circ \rightarrow \bullet^\omega \notin L$ ,
- $\circ \leftarrow \bullet^\omega \notin L$ .

In particular,  $L \subsetneq \Gamma^\omega$  and we denote by  $w$ , a scenario in  $\Gamma^\omega \setminus L$ . If we can choose  $w$  such that it is fair, we choose such a  $w$ . Otherwise  $w$  is unfair, and we assume it is either  $\circ \rightarrow \bullet^\omega$  or  $\circ \leftarrow \bullet^\omega$ , or it belongs to a special pair that is not included in  $L$ .

We consider the algorithm  $\mathcal{A}_w$  with parameter  $w$  as defined above.

**Lemma 25.** *Let  $v \in L$ . There exist  $r \in \mathbb{N}$  such that  $|\text{ind}(v|_r) - \text{ind}(w|_r)| \geq 3$ .*

*Proof.* Given  $w \notin L$ , we have  $v \neq w$  and at some round  $r$ ,  $w|_r \neq v|_r$ . Therefore  $|\text{ind}(v|_r) - \text{ind}(w|_r)| \geq 1$ .

From Lemma 15 and Definition 18, it can be seen that the only way to remain indefinitely at a difference of one is exactly that  $w$  and  $v$  form a special pair. Given the way we have chosen  $w$ , and that  $v \in L$ , this is impossible. So at some round  $r'$ , the difference will be greater than 2 :  $|\text{ind}(v|_{r'}) - \text{ind}(w|_{r'})| \geq 2$ .

Then by definition of the index we have that  $|\text{ind}(v|_{r'+1}) - \text{ind}(w|_{r'+1})| \geq 3$ .

Now we prove the correctness of the algorithm under the message adversary  $L$ .

**Proposition 26.** *The algorithm  $\mathcal{A}_w$  is correct for every  $v \in L$ .*

*Proof.* First we show Termination. This is a corollary of Lemma 25 and Proposition 24.

Consider the execution  $v \in L$ . From Lemma 25, there exists a round  $r \in \mathbb{N}$  such that  $|\text{ind}(v|_r) - \text{ind}(w|_r)| \geq 3$ .

Denote by  $r$  the round when it is first satisfied for one of the process. From the condition of the While loop, it means this process will stop at round  $r$ . If the  $\text{ind}$  value of the other process  $p$  at round  $r$  is also at distance 3 or more from the index of  $w|_r$ , then we are done. Otherwise, from Proposition 24, we have  $|\text{ind}_r^p - \text{ind}(w|_r)| = 2$ . In the following round,  $p$  will receive no message (the other process has halted) and  $|\text{ind}_r^p - \text{ind}(w|_r)| \geq 3$  will hold. Note also, that even though the final comparisons are strict, the output value is well defined at the end since  $\text{ind}_r \neq \text{ind}(w|_r)$  at this stage.

The validity property is also obvious, as the output values are ones of the initial values. Consider the case for  $\bullet$  (the case for  $\circ$  is symmetric). Since the only case where  $\text{initother}^\bullet$  would be *null* is when  $\bullet$  has not received any message from  $\circ$ , i.e. when  $v = \circ \leftarrow \bullet^r$ . But as  $\text{ind}(\circ \leftarrow \bullet^r) = 3^r - 1$  is the maximal possible index for scenario of length  $r$ , and  $\bullet$  outputs  $\text{initother}^\bullet$  only if  $\text{ind}^\bullet < \text{ind}(w|_r)$ , it cannot have to output  $\text{initother}$  when it is *null*. Similarly for  $\circ$ , this proves that *null* can never be output by any process.

We now prove the agreement property. Given that  $|\text{ind}_r^\circ - \text{ind}_r^\bullet| = 1$  by Proposition 24, when the processes halt, from Lemma 25 the  $\text{ind}$  values are on the same side of  $\text{ind}(w|_r)$ . This means that one of the process outputs *init*, the other outputting *initother*. By construction, they output the same value.

## 4 Topological approach

In this Section, we provide a topological characterization of solvable message adversaries for the Consensus Problem. First, we will introduce some basic topological definitions in



Section 4.1, then we will explain the link between topology and distributed computability in Section 4.2 in order to formulate our result in Section 4.3.

We then show in the following Section 5 how this new characterization matches the combinatorial one described by Theorem 19. We also discuss a similar characterization in [GKM14] and we show that our result indicates that there is a flaw in the statement of Theorem 6.1 of [GKM14]. If no restriction are given on the kind of adversary are addressed in this theorem, then, in the case of 2 processes, this would imply that  $\Gamma^\omega \setminus \{w\}$  is solvable for any  $w$ . From our result, this is incorrect when  $w$  belongs to a special pair. This has been confirmed by the authors [Kuz] that the statement has to be corrected by restricting to adversaries  $L$  that are closed for special pairs (if  $\{w, w'\}$  is a special pair, then  $w \in L \Leftrightarrow w' \in L$ ). Moreover, even if the present work is for only 2 processes, the approach that is taken might help correct the general statement of [GKM14].

## 4.1 Definitions

The following definitions are standard definitions from algebraic topology [Mun84]. We fix an integer  $N \in \mathbb{N}$  for this part.

**Definition 27.** Let  $n \in \mathbb{N}$ . A finite set  $\sigma = \{v_0, \dots, v_n\} \subset \mathbb{R}^N$  is called a simplex of dimension  $n$  if the vector space generated by  $\{v_1 - v_0, \dots, v_n - v_0\}$  is of dimension  $n$ . We denote by  $|\sigma|$  the convex hull of  $\sigma$  that we call the geometric realization of  $\sigma$ .

**Definition 28.** A simplicial complex is a collection  $C$  of simplices such that :

- (a) If  $\sigma \in C$  and  $\sigma' \subseteq \sigma$ , then  $\sigma' \in C$ ,
- (b) If  $\sigma, \tau \in C$  and  $|\sigma| \cap |\tau| \neq \emptyset$  then there exists  $\sigma' \in C$  such that
  - $|\sigma| \cap |\tau| = |\sigma'|$ ,
  - $\sigma' \subset \sigma, \sigma' \subset \tau$ .

The simplices of dimension 0 (singleton) of  $C$  are called vertices, we denote  $V(C)$  the set of vertices. The *geometric realization* of  $C$ , denoted  $|C|$ , is the union of the geometric realization of the simplices of  $C$ .

Let  $A$  and  $B$  be simplicial complexes. A map  $f: V(A) \rightarrow V(B)$  is called *simplicial* (in which case we write  $f: A \rightarrow B$ ) if it preserves the simplices, *i.e.* for each simplex  $\sigma$  of  $A$ , the image  $f(\sigma)$  is a simplex of  $B$ .

In this paper, we also work with colored simplicial complexes. These are simplicial complexes  $C$  together with a function  $c: V(C) \rightarrow \Pi$  such that the restriction of  $c$  on any maximal simplex of  $C$  is a bijection. A simplicial map that preserves colors is called *chromatic*.

As a final note, since we only deal with two processes, our simplicial complexes will be of dimension 1. The only simplices are edges (sometimes called *segments*) and vertices, and the latter are colored with  $\Pi = \{\circ, \bullet\}$ .

*Remark 29.* The combinatorial part of simplicial complexes (that is the sets of vertices and the inclusion relationships they have) is usually referred as abstract simplicial complexes.

Abstract simplicial complex can be equivalently defined as a collection  $C$  of sets that are closed by inclusion, that is if  $S \in C$  and if  $S' \subset S$  then  $S' \in C$ .

For finite complexes, the topological and combinatorial notions are equivalent, including regarding geometric realizations. But it should be noted that infinite abstract simplicial complex might not have a unique (even up to homeomorphisms) geometric realization. However, here the infinite complexes we deal with are derived from the subdivision, see below, of a finite initial complex, hence they have a unique realization in that setting. So in the context of this paper, we will talk about “the” realization of such infinite complexes.

The elements of  $|C|$  can be expressed as convex combinations of the vertices of  $C$ , *i.e.*  $\forall x \in |C| \quad x = \sum_{v \in V(C)} \alpha_v v$  such that  $\sum_{v \in V(C)} \alpha_v = 1$  and  $\{v \mid \alpha_v \neq 0\}$  is a simplex of  $C$ .

The geometric realization of a simplicial map  $\delta : A \rightarrow B$ , is  $|\delta| : |A| \rightarrow |B|$  and is obtained by  $f(x) = \sum_{v \in V(C)} \alpha_v f(v)$ .

A *subdivision* of a simplicial complex  $C$  is a simplicial complex  $C'$  such that :

- (1) the vertices of  $C'$  are points of  $|C|$ ,
- (2) for any  $\sigma' \in C'$ , there exists  $\sigma \in C$  such that  $|\sigma'| \subset |\sigma|$ ;
- (3)  $|C| = |C'|$ .

Let  $C$  be a chromatic complex of dimension 1, its *standard chromatic subdivision*  $\text{Chr } C$  is obtained by replacing each simplex  $\sigma \in C$  by its chromatic subdivision. See [HKR13] for the general definition of the chromatic subdivision of a simplex, we present here only the chromatic subdivision of a segment  $[0, 1] \subset \mathbb{R}$  whose vertices are colored  $\circ$  and  $\bullet$  respectively. The subdivision is defined as the chromatic complex consisting of 4 vertices at position  $0, \frac{1}{3}, \frac{2}{3}$  and  $1$ ; and colored  $\circ, \bullet, \circ, \bullet$  respectively. The edges are the 3 segments  $[0, \frac{1}{3}]$ ,  $[\frac{1}{3}, \frac{2}{3}]$  and  $[\frac{2}{3}, 1]$ . The geometric realization of the chromatic subdivision of the segment  $[0, 1]$  is identical to the segment's.

If we iterate this process  $m$  times we obtain the  $m^{\text{th}}$  chromatic subdivision denoted  $\text{Chr}^m C$ . Figure 2 shows  $\text{Chr } [0, 1]$  to  $\text{Chr}^3 [0, 1]$ .

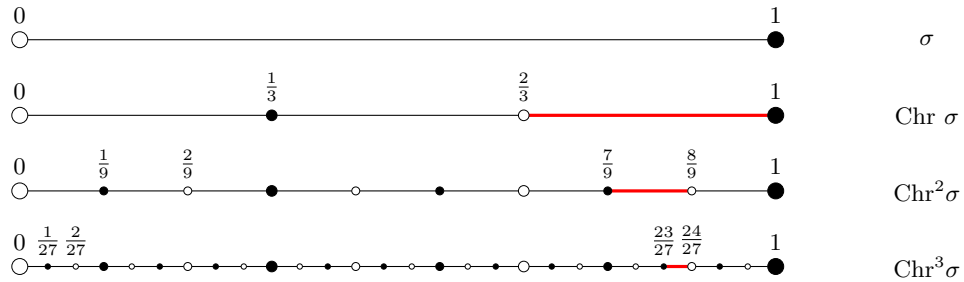


Fig. 2: Chromatic subdivision of the segment.

The correspondence with executions (bolder simplices) is explained at the end of section 4.2.

## 4.2 Topological representation of distributed systems

As shown from the celebrated Asynchronous Computability Theorem [HS99], it is possible to encode the global state of a distributed system in simplicial complexes. First we give the intuition for the corresponding abstract simplicial complex.

We can represent a process in a given state by a vertex composed of a color and a value : the *identity* and the *local state*. A *global configuration* is an edge whose vertices are colored  $\circ$  and  $\bullet$ .

The set of input vectors of a distributed task can thus be represented by a colored simplicial complex  $\mathcal{I}$  (associating all possible global initial configurations).

*Remark 30.* The vertices that belongs to more than one edge illustrate the uncertainty of a process about the global state of a distributed system. In other words, a local state can be common to multiple global configurations, and the process does not know in which configuration it belongs. We have a topological (and geometrical) representation of theses uncertainties.

For example, Figure 3 shows a very simple graph where each colored vertex is associated with a value (0 or 1). The vertex in the middle is common to both edges; it represents the uncertainty of the process  $\bullet$  concerning the value of  $\circ$ , *i.e.*  $\bullet$  doesn't know if it is in the global configuration (0, 0) or (0, 1)

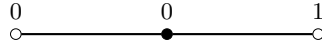


Fig. 3: Example of a simplicial complex with uncertainty

In the same way than  $\mathcal{I}$ , we construct (for a distributed task) the output complex  $\mathcal{O}$  that contains all possible output configurations of the processes. For a given problem, it is possible to construct a relation  $\Delta \subset \mathcal{I} \times \mathcal{O}$  that is chromatic and relates the input edges with the associated possible output edges. So, any task can be topologically represented by a triplet  $(\mathcal{I}, \mathcal{O}, \Delta)$ .

For example, the Binary Consensus task with two processes, noted  $(\mathcal{I}_{2gen}, \mathcal{O}_{2gen}, \Delta_{2gen})$ , is shown in Figure 4. The input complex  $\mathcal{I}_{2gen}$ , on the left-hand side, consists of a square. Indeed, there are only four possible global configurations given that the two process can only be in two different initial states (proposed value 0 or 1). The output complex, on the right-hand side, has only two edges corresponding to the valid configurations for the Consensus (all 0 or all 1). Finally  $\Delta$  maps the input configuration with the possible output ones, according to the *validity* property of the Consensus.

Any protocol can be encoded as a full information protocol <sup>2</sup> any local state is related to what is called the view of the execution. A protocol simplex is a global configuration such

<sup>2</sup> since the full information protocol send all possible information, the computations can be emulated provided it is allowed to send so much information.

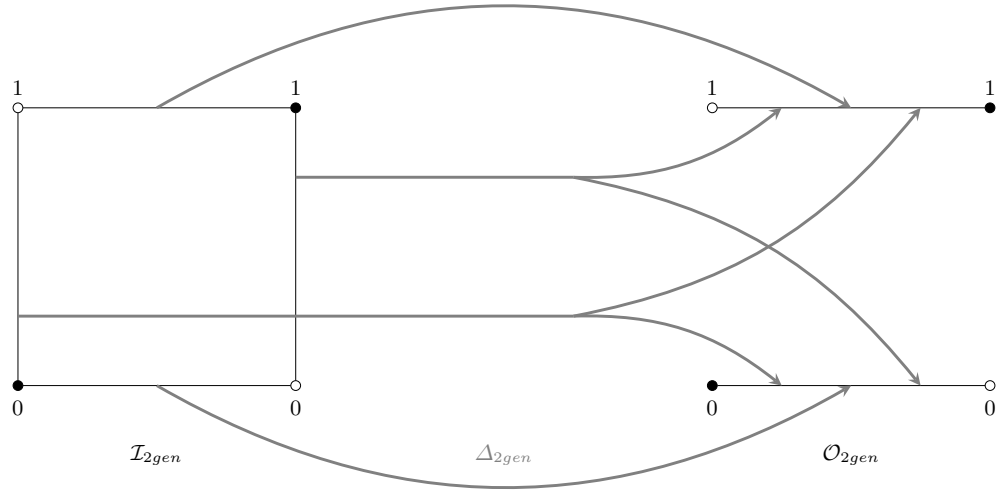


Fig. 4: Representation of the Consensus Task with 2 processes

that there exists an execution of the protocol in which the processes end with these states. The set of all these simplices forms the *protocol complex* associated to an input complex, a set of executions and an algorithm. Given any algorithm, it can be redefined using a full-information protocol, the protocol complex thus depends only on the input complex and the set of executions.

Given an input complex  $\mathcal{I}$ , we construct the *protocol complex at step  $r$*  by applying  $r$  times to each simplex  $\sigma$  the chromatic subdivision shown in Section 4.1 and Figure 2.

The input complex of the Consensus and the first two steps of the associated protocol complex are shown in Figure 5.

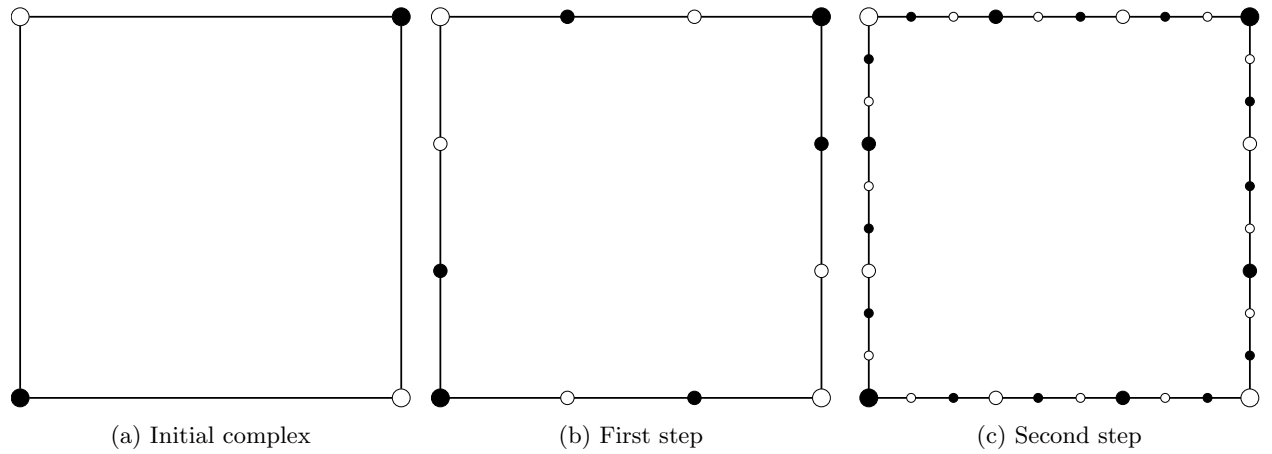


Fig. 5: Protocol complex of the initial, first and second rounds

For example, let  $w_0 = \circ \leftarrow \bullet \circ \leftrightarrow \bullet \circ \rightarrow \bullet^\omega$ .  $w$  corresponds to the following execution :

- first, the message from  $\circ$  is lost;
- $\circ$  and  $\bullet$  receive both message;
- the message from  $\bullet$  is lost;
- ... (this last round is repeated indefinitely)

This can be represented as a sequence of simplices. Each infinite sequence of edges  $(\sigma_0, \sigma_1, \dots)$  with  $\sigma_{i+1} \in \text{Chr}\sigma_i$ , and  $\sigma_0$  the initial configuration, corresponds to a unique scenario and vice versa. Consider Figure 2, at  $[\frac{2}{3}, 1]$  the thick red simplex corresponds to  $\circ \leftarrow \bullet$ . Then, at  $[\frac{7}{9}, \frac{8}{9}]$ , this corresponds to  $\circ \leftarrow \bullet \circ \leftrightarrow \bullet$ . Finally, at  $[\frac{23}{27}, \frac{24}{27}]$  the thick red simplex of  $\text{Chr}^3 \sigma$  corresponds to the execution  $\circ \leftarrow \bullet \circ \leftrightarrow \bullet \circ \rightarrow \bullet$ .

For a given finite execution, the embedding is exactly given by the normalized index  $\text{ind}_n$ . Thus, when the initial simplex is  $[0, 1]$  ( $\circ$  initial value is 0,  $\bullet$  is 1) for an infinite execution, the corresponding sequence of simplices will converge to a point of  $[0, 1]$ . At the limit, the vertex' convergence point is given by  $\overline{\text{ind}}$ . For example,  $w_0$  converges to  $1/9$ .

Without loss of generality, in the rest of the paper, we will describe what happens on the segment  $[0, 1]$  instead of the whole complex. Indeed, the behaviour of subdivision of initial segments is identical (through a straightforward isometry for each segment) as this behaviour does not depend on the initial values. The "gluing" at "corners" is always preserved since it corresponds to state obtained by a process when it receives no message at all.

Given  $x \in [0, 1]$ ,  $i_\circ, i_\bullet \in \{0, 1\}$ , we denote by  $\text{geo}(x, i_\circ, i_\bullet)$  the point in the geometrical realization that corresponds to  $x$  in the simplex  $[(\circ, i_\circ), (\bullet, i_\bullet)]$ , namely  $x(\circ, i_\circ) + (1-x)(\bullet, i_\bullet)$ .

Given  $L \subset I^\omega$ , we denote  $|\mathcal{C}^L| = \{\text{geo}(\overline{\text{ind}}(w), i_\circ, i_\bullet) \mid \exists w \in L, i_\circ, i_\bullet \in \{0, 1\}\}$ . Note that it is possible to define the limit of the geometric realizations this way, but that there are no sensible way to define a geometric realization directly for the corresponding abstract simplicial complex. See Section 5.1.

### 4.3 Topological Characterization

The following definition is inspired by [GKM14].

**Definition 31.** *Let  $C$  be a colored simplicial complex. A terminating subdivision  $\Phi$  of  $C$  is a (possibly infinite) sequence of colored simplicial complexes  $(\Sigma_k)_{k \in \mathbb{N}}$  such that  $\Sigma_0 = \emptyset$ , and for all  $k \geq 1$   $\Sigma_k \subset \text{Chr}^k C$ , and  $\cup_{i \leq k} \Sigma_i$  is a simplicial complex for all  $k$ .*

The intuition for this definition is as follows. It is well known that (non-terminated) runs of length  $r$  in  $I^\omega$  with initial values encoded as  $C$  are represented by a protocol complex that is the chromatic subdivision  $\text{Chr}^k C$ . This definition refines the known correspondence by looking at the actual runs for a given algorithm. When the algorithm stops, the protocol complex should actually be no more refined. For a given algorithm, we end up with protocol complexes that are of a special form : where, possibly, the level of chromatic subdivision is not the same everywhere. We will prove later that those resulting complexes are exactly terminating subdivisions. Or to say it differently, terminating subdivisions are the form of

protocol complexes for non-uniformly terminating algorithms (that is for algorithms that do not stop at the same round for all possible executions).

From the correspondence between words and simplexes of the chromatic subdivision, we can see that the corresponding set of words is an anti-chain for the prefix order relation.

An edge of  $\Sigma_k$  for some  $k$  is called a *stable edge* in the subdivision  $\Phi$ . The union  $\cup_k \Sigma_k$  of stable edges in  $\Phi$  forms a colored simplicial complex, and a stable edge can only intersect another given stable edge at an extremity (a single vertex).

For  $r \in \mathbb{N}$ , we denote by  $K_r(\Phi)$  the complex  $\cup_{k \leq r} \Sigma_k$ . We denote by  $K(\Phi)$  the complex  $\cup_k \Sigma_k$ ; it possibly has infinitely many simplices. Observe that the geometric realization  $|K(\Phi)|$  can be identified with a subset of  $|C|$ . For example, Figure 6 shows a terminating subdivision of  $[0, 1]$  up to round 3.

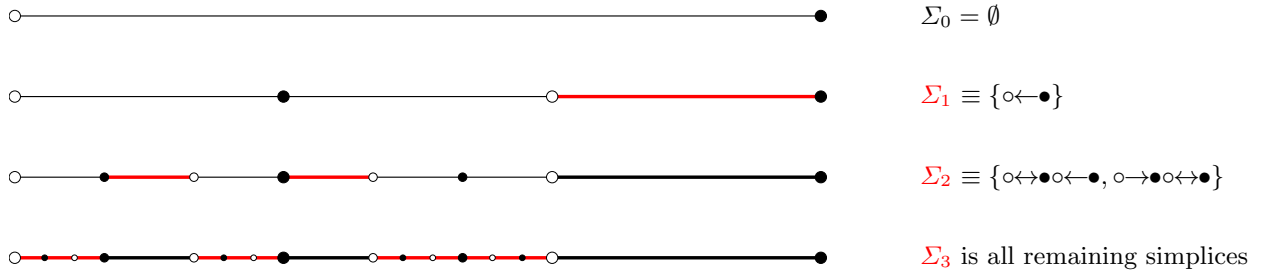


Fig. 6: Example of a terminating subdivision.

The bolder simplices show the stable edges (colored red at first appearance). For the correspondence of simplices with executions, see section 4.2

There is an example in Fig. 6 of a terminating subdivision for  $\Gamma^\omega$ .

The following definition expresses when a terminating subdivision covers all considered scenarios of a given  $L$ . The intuition is that every scenario of  $L$  should eventually land into a simplex of  $\Phi$ . In terms of words, this means that all words of  $L$  have a prefix in the corresponding words of the simplexes of  $\Phi$ .

**Definition 32.** A terminating subdivision  $\Phi$  of  $C$  is admissible for  $L \subseteq \Gamma^\omega$  if for any scenario  $\rho \in L$  the corresponding sequence of edges  $\sigma_0, \sigma_1, \dots$  is such that there exists  $r > 0$ ,  $|\sigma_r|$  is a stable edge in  $K(\Phi)$ .

We can now state and prove our new characterization theorem. First, for simplicity, notice that the output complex Consensus  $O_{2gen}$  has two connected components representing 0 and 1. Thus we can identify it to the set  $\{0, 1\}$  and define the relation  $\Delta'_{2gen} \subset \mathcal{I}_{2gen} \times \{0, 1\}$  analogous to  $\Delta_{2gen}$ .

**Theorem 33.** The task  $T_{2gen} = (\mathcal{I}_{2gen}, \mathcal{O}_{2gen}, \Delta_{2gen})$  is solvable in a sub-model  $L \subseteq \Gamma^\omega$  if and only if there exist a terminating subdivision  $\Phi$  of  $\mathcal{I}_{2gen}$  and a simplicial function  $\delta: K(\Phi) \rightarrow \{0, 1\}$  such that :

33.i  $\Phi$  is admissible for  $L$ ;

- 33.ii For all simplex  $\sigma \in \mathcal{I}_{2gen}$ , if  $\tau \in K(\Phi)$  is such that  $|\tau| \subset |\sigma|$ , then  $\delta(\tau) \in \Delta'_{2gen}(\sigma)$ ;  
 33.iii  $|\delta|$  is continuous.

*Proof. Necessary condition.* Suppose we have an algorithm  $\mathcal{A}$  for solving the Binary Consensus task, we will construct a terminating subdivision  $\Phi$  admissible for  $L$  and a function  $\delta$  that satisfies the conditions of the theorem.

When running  $\mathcal{A}$ , we can establish which nodes of the protocol complex decide a value by considering the associated state of the process. Intuitively,  $\Phi$  and  $\delta$  are built as the following : in each level  $r$ , we consider all runs of length  $r$  in  $L$ . Each run yields a simplex, if the two nodes of the simplex have decided a (unique) value  $v$ , we add this simplex to  $\Sigma_r$  and set  $\delta(\sigma) = v$ .

Formally, let  $\mathcal{C}^L(r)$  be the protocol complex of a full information protocol subject to the language  $L$  at round  $r$  and  $V(\mathcal{C}^L(r))$  its set of vertices. Given a vertex  $x$ , let  $val(x)$  be the value decided by the corresponding process in the execution that leads to state  $x$ . And  $\forall r \geq 0$  define

$$\Sigma_r = \{\{x, y\} \in \mathcal{C}^L(r) \mid x \text{ and } y \text{ have both decided and at least one has just decided in round } r\}$$

$$\delta(x) = val(x) \quad \forall x \in V(\Sigma_r)$$

The function  $\delta$  is well defined since it depends only on the state encoded in  $x$  and in  $\Sigma_r$  all vertex have decided.

For all  $\{x, y\} \in \Sigma_r$ , note that  $val(x) = val(y)$  because  $\mathcal{A}$  satisfies the *agreement* property.

By construction for  $\Phi$ , it is a terminating subdivision. Furthermore,  $\Phi$  is admissible for  $L$ . Since  $\mathcal{A}$  terminates subject to  $L$ , all processes decide a value in a run of  $L$ , so all nodes of the complex protocol restricted to  $L$  will be in  $\Phi$  (by one of their adjacent simplexes).

The condition 33.ii is also satisfied by construction because  $\mathcal{A}$  satisfies the *validity* property, i.e.  $val(x) \in \Delta'_{2gen}(\sigma)$ .

We still have to prove that  $|\delta|$  is continuous, in other words for all  $x \in |K(\Phi)|$ , we must have

$$\forall \varepsilon > 0 \quad \exists \eta_x > 0 \quad \forall y \in |K(\Phi)| \quad |x - y| \leq \eta_x \Rightarrow ||\delta|(x) - |\delta|(y)| < \varepsilon$$

This property for continuity says that when two points  $x, y \in |K(\Phi)|$  are close, their value of  $|\delta|$  is close. When  $\varepsilon$  is small (e.g.  $\varepsilon < 1/2$ ),  $||\delta|(x) - |\delta|(y)| < \varepsilon$  implies that  $|\delta|(x) = |\delta|(y)$ . This is because  $|\delta|$ 's co-domain is discrete. When  $\varepsilon$  is large, the property is always verified and thus is not much of interest. We can thus reformulate it without considering  $\varepsilon$  : we must show that  $\forall x \in |K(\Phi)|$

$$\exists \eta_x > 0 \quad \forall y \in |K(\Phi)| \quad |x - y| \leq \eta_x \Rightarrow |\delta|(x) = |\delta|(y) \tag{8}$$

In defining  $\eta_x$  as followed, we have the continuity condition for all  $x \in V(K(\Phi))$  :

$$\eta_x = \min\left\{\frac{1}{3^{r+1}} \mid \exists r \in \mathbb{N}, \exists y, \{x, y\} \in V(\Sigma_r)\right\}.$$

Since there exists at least one ( $\mathcal{A}$  terminates for any execution in  $L$ ) and at most two such  $y$ , the minimum is well defined for all  $x \in V(K(\Phi))$ . Moreover, we remark that since the geometric realization of simplices of  $\Sigma_r$  are of size  $\frac{1}{3^r}$ , the ball centered in  $x$  and of diameter  $\eta_x$  is included in the stable simplices and the function  $\delta$  is constant on this ball (by agreement property).

Let  $x, y \in V(K(\Phi))$ , let  $z \in [x, y]$ , we define  $\eta_z = \min(d(x, z), d(y, z))$ . By construction, using such function  $\eta$ , for all  $z \in |K(\Phi)|$ , Proposition (8) is satisfied.

**Sufficient condition.** Given a terminating subdivision  $\Phi$  admissible for  $\mathcal{I}_{2gen}$  and a function  $\delta$  that satisfies the conditions of the theorem, we present an algorithm  $\mathcal{A}$  that solves Consensus.

First, we describe how to obtain a function  $\eta$  from the continuity of  $|\delta|$ . For any  $x \in V(K(\Phi))$ , there exists  $\eta(x)$  such that for any  $y \in |K(\Phi)|$ , when  $|x - y| \leq \eta(x)$ , we have  $\delta(y) = \delta(x)$ . Notice we can choose  $\eta$  such that  $\forall x \in |\mathcal{I}| \forall y_1, y_2 \in |K(\Phi)|, |y_1 - x| \leq \eta(y_1) \wedge |y_2 - x| \leq \eta(y_2) \Rightarrow |\delta|(y_1) = |\delta|(y_2)$ . Now, we show how to extend the definition to  $|K(\Phi)|$ . Consider  $w \in |\Phi| \setminus V(K(\Phi))$  and denote  $x, y$  the vertices of  $K(\Phi)$  that defines the segment to which  $w$  belongs. We define  $\eta(w) = \min\{\eta(x), \eta(y)\}$ .

We recall that  $B(z, t)$  (resp.  $\bar{B}(z, t)$ ) is the open (resp. closed) ball of center  $z$  and radius  $t$ . We define a boolean function  $Finished(r, x)$  for  $r \in \mathbb{N}$  and  $x \in |\mathcal{I}|$  that is true when  $\exists y \in V(Chr^r(\mathcal{I})), y \in |K_r(\Phi)|, \bar{B}(x, \frac{1}{3^r}) \subset B(y, \eta(y))$ .

The Consensus algorithm is described in Algorithm 2, using the function  $Finished$  we just defined from  $\eta$ . Notice that  $\eta$  is fully defined on  $|K(\Phi)|$  and that the existential condition at line 10 is over a finite subset of  $|K(\Phi)|$ .

As in Algorithm 1, it has messages always of the same type. They have two components, the first one is the initial bit, named *init*. The second is an integer named *ind*. Given a message *msg*, we note *msg.init* (resp. *msg.ind*) the first (resp. the second) component of the message. The computation of the index is similar. We maintain in an auxiliary variable  $r$  the current round (line 23).

The halting condition is now based upon the position of the geometric realization *geo* of the current round with regards to the terminating subdivision  $K(\Phi)$  and the  $\eta$  function : we wait until the realization is close enough of a vertex in  $|K(\Phi)|$  and the corresponding open ball of radius  $\eta$  contains a neighbourhood of the current simplex. Note that when *initb* or *initw* is still *null*, this means that we are at the corners of the square  $\mathcal{I}$ .

We show that the algorithm solves Consensus. Consider an execution  $w$ , we will prove it terminates. The fact that the output does not depends on  $y$  comes from the choice of  $\eta$ . The properties Agreement and Validity then come immediately from condition 33.ii on  $\delta$  and  $\Delta'_{2gen}$ .

First we prove termination for at least one process. Assume none halts.

The admissibility of  $\Phi$  proves that there is a round  $r_0$  for which the simplex corresponding to the current partial scenario  $w|_{r_0}$  is a simplex  $\sigma$  of  $\Phi$ . Denote  $r_1 \geq r_0$  an integer such that  $\frac{1}{3^{r_1}} < \eta(y)$  with  $y \in |\sigma|$ . Since from round  $r_0$ , all future geometric realizations will remain in  $|\sigma|$ , we have, at round  $r_1$ , that  $Finished$  is true. A contradiction.



**Algorithm 2:** Algorithm  $\mathcal{A}_\eta$  for the binary consensus with two processes for process  $p$  where  $\eta$  is a function  $[0, 1] \rightarrow [0, 1]$ .  $\text{geo}(x, w, b)$  is the embedding function into the geometric realization.

```

Data: function  $\eta$ 
Input:  $\text{init} \in \{0, 1\}$ 
1   $r = 0$ ;
2  if  $p = \bullet$  then
3     $\text{ind} = 1$ ;
4     $\text{initw} = \text{null}$ ;
5     $\text{initb} = \text{init}$ ;
6  else
7     $\text{ind} = 0$ ;
8     $\text{initw} = \text{init}$ ;
9     $\text{initb} = \text{null}$ ;
10 repeat
11    $\text{msg} = (\text{init}, \text{ind})$ ;
12    $\text{send}(\text{msg})$ ;
13    $\text{msg} = \text{receive}()$ ;
14   if  $\text{msg} == \text{null}$  then // message was lost
15      $\text{ind} = 3 * \text{ind}$ ;
16   else
17      $\text{ind} = 2 * \text{msg.ind} + \text{ind}$ ;
18     if  $p = \bullet$  then
19        $\text{initw} = \text{msg.init}$ ;
20     else
21        $\text{initb} = \text{msg.init}$ ;
22    $r = r + 1$ ;
23 until  $\exists y \in V(\text{Chr}^r(\mathcal{I})), y \in |K_r(\Phi)|, \bar{B}(\text{geo}(\text{ind}/3^r, \text{initw}, \text{initb}), \frac{1}{3^r}) \subset B(y, \eta(y))$ ;
Output:  $|\delta|(y)$ 

```

Now, from the moment the first process halts with an index with geometric realization  $x$  because of some  $y$ , if the other one has not halted at the same moment, then this one will receive no other message from the halted process, and its *ind* variable, with geometric realization  $z$ , will remain constant forever. The closed ball centered in  $x$  has a neighbourhood inside the open ball centered in  $y$ , therefore there exists  $r_2$  such that  $\frac{1}{3^{r_2}}$  is small enough and  $Finished(r_2, z)$  is true.

We say that the corner of the squares in Fig. 5 are *corner nodes*. If we compare the two algorithms, we can see that the differences are in the halting condition and the decision value. In Algorithm 1, once a forbidden execution  $w$  has been chosen, the algorithm runs until it is far away enough of a prefix of  $w$  to conclude by selecting the initial value of the corner node which is on the same side as the prefix.

Algorithm 2 is based on the same idea but somehow is more flexible (more general). If there are many holes in the geometric realization, it is possible to have different chosen output values for two different connected components. Of course, the connected components that contains the corner nodes have no liberty in choosing the decision value.

## 5 About the two Characterizations

In this section, we explain how the combinatorial and the topological characterizations match. This is of course convenient but the more important result will be to see that the topological characterization permits to derive a richer intuition regarding the characterization, in particular about the status of the special pairs. We will also illustrate how the Theorem 6.1 of [GKM14] does not handle correctly the special pairs by showing that in this cases, combinatorial and geometric simplicial protocol complexes do differ in precision to handle Consensus computability.

For any adversary  $L \subseteq \Gamma^\omega$ ,  $L$  can be either an obstruction or solvable for Consensus. In other words, the set of sub-models  $\mathcal{P}(\Gamma^\omega)$  is partitioned in two subset : obstructions and solvable languages. Theorem 19 explicitly describes the solvable languages and classes them in four families. In contrast, Theorem 33 gives necessary and sufficient conditions for a sub-model to be either an obstruction or not.

We first give an equivalent version of Theorem 33.

**Theorem 34.** *The task  $T_{2gen}$  is solvable in  $L \subseteq \Gamma^\omega$  if and only if  $|\mathcal{C}^L|$  is not connected.*

*Proof.* ( $\Rightarrow$ ) If  $T_{2gen}$  is solvable in  $L$ , let  $\Phi$  and  $\delta$  as described in Theorem 33. From admissibility, we have that  $|\mathcal{C}^L| \subset |K(\Phi)|$ . Now  $|\delta|$  is a continuous surjective function from  $|\mathcal{C}^L|$  to  $\{0, 1\}$ , this implies that the domain  $|\mathcal{C}^L|$  is not connected since the image of a connected space by a continuous function is always connected.

( $\Leftarrow$ ) With  $|\mathcal{C}^L|$  disconnected, we can associate output value to connected components in such a way that  $\Delta'_{2gen}$  is satisfied. Consider the segment  $[0, 1]$ , there exists  $z \in [0, 1]$  and  $z \notin |\mathcal{C}^L|$ .

We define  $\Phi$  in the following way :

$$\Sigma_k = \{S \mid S = [\overline{ind}(w), \overline{ind}(w) + \frac{1}{3^k}], w \in \Gamma^k, \exists w' \in \Gamma^\omega, ww' \in L \text{ and } \forall i_w, i_b \in \{0, 1\} \text{ } geo(z, i_w, i_b) \notin |S|\}$$

We now denote  $\delta$  the function  $K(\Phi) \rightarrow \{0, 1\}$  such that for  $v \in [0, 1]$ , for  $X = geo(v, i_w, i_b)$ ,  $\delta(X) = i_w$  if  $v < z$  and  $\delta(v) = i_b$  otherwise.

We now check that these satisfies the conditions of Theorem 33. Admissibility comes from the fact that, by construction of  $z$ , there are no runs in  $L$  that converge to  $z$ , therefore for any run  $w \in L$ , at some point,  $\frac{1}{3^k}$  is small enough such that there is an edge in  $\Sigma_k$  that contains  $\overline{ind}_{w|_k}$  and not  $z$ .

The function  $|\delta|$  is continuous and has been defined such that condition 33.ii is also clearly satisfied.

We recall the definitions of *Fair* and *SPair* languages, and the admissible language families defined in Section 2.8.

$$\begin{aligned} Fair(\Gamma^\omega) &= \Gamma^\omega \setminus \{xy \mid x \in \Gamma^* \quad y \in \{\circ \text{---} \bullet, \circ \leftarrow \bullet\}^\omega \cup \{\circ \text{---} \bullet, \circ \rightarrow \bullet\}^\omega\} \\ SPair(\Gamma^\omega) &= \{(w, w') \in \Gamma^\omega \times \Gamma^\omega \mid w \neq w', \quad \forall r \in \mathbb{N} \quad |ind(w|_r) - ind(w'|_r)| \leq 1\} \end{aligned}$$

A message adversary is solvable if it is one of the following (non-exclusive) forms

1.  $\mathcal{F}_1 = \{L \mid \exists f \in Fair(\Gamma^\omega), f \notin L\}$
2.  $\mathcal{F}_2 = \{L \mid \exists (w, w') \in SPair(\Gamma^\omega), w, w' \notin L\}$
3.  $\mathcal{F}_3 = \{L \mid \circ \rightarrow \bullet^\omega \notin L\}$
4.  $\mathcal{F}_4 = \{L \mid \circ \leftarrow \bullet^\omega \notin L\}$

We give now a topological interpretation of this description. We start with a topological characterization of special pairs.

**Lemma 35.** *Let  $w, w' \in \Gamma^\omega$ ,  $w \neq w'$ ,  $\overline{ind}(w) = \overline{ind}(w')$  if and only  $(w, w') \in SPair(\Gamma^\omega)$ .*

*Proof.* This comes from Lemma 25 and the very definition of special pairs.

In others words, special pairs are exactly the runs that have another run that converges to the same geometric realization. So removing only one member of a special pair is not enough to disconnect  $|\mathcal{C}^L|$ . It is also straightforward to see that removing a fair run implies disconnection and that removing  $\circ \rightarrow \bullet^\omega$  or  $\circ \leftarrow \bullet^\omega$  implies disconnecting at the corners.

## 5.1 Counter-Example

Now we look at Theorem 6.1 from [GKM14]. The only difference is that in [GKM14]  $\delta$  is only required to be chromatic. Here we have shown that we need a stronger condition that is the continuity of  $|\delta|$ . We explain why this is strictly stronger, ie how it is possible to have simplicial mapping while not having continuity of the geometric realization.

Consider  $L = \Gamma^\omega \setminus w_0$ , with  $w_0 = \circ \leftrightarrow \bullet \circ \leftarrow \bullet^\omega$ . We define  $\Phi$  as follows.

$\Sigma_1 = \{[0, \frac{1}{3}], [\frac{2}{3}, 1]\}$  For  $r \geq 2$ ,  $\Sigma_r = \{[\frac{2}{3} - \frac{1}{3^{r-1}}, \frac{2}{3} - \frac{2}{3^r}], [\frac{2}{3} - \frac{2}{3^r}, \frac{2}{3} - \frac{1}{3^r}]\}$ .

The terminating subdivision  $\Phi$  is admissible for  $L$  and there exists a simplicial mapping from  $K(\Phi)$  to  $\{0, 1\}$ . We can set  $\delta(x) = 0$  when  $x < \frac{2}{3}$  and  $\delta(x) = 1$  otherwise.

$K(\Phi)$  has two connected components  $[\frac{2}{3}, 1]$  on one side, and all the other segments on the other side. The function  $\delta$  is therefore simplicial since there is no  $[z, \frac{2}{3}]$  interval in  $K(\Phi)$ .

So such a  $L$  satisfies the assumptions for Theorem 6.1 of [GKM14]. To see that it does not satisfy the assumptions of Theorem 33, we remark that  $|K(\Phi)|$  is  $[0, 1]$  and the  $\delta$  function we define above does not have a geometric realization that is continuous and moreover there is no way to define a continuous surjective function from  $[0, 1]$  to  $\{0, 1\}$ . So the statements of the Theorem are not equivalent. To correct Theorem 6.1 of [GKM14], it is needed to add an assumption, that is that  $L$  has to be “closed” for special pairs : either both members belong to  $L$  or none, this has been confirmed by the authors [Kuz].

The simplicial complex  $K(\Phi)$  has two connected components when seen as an abstract simplicial complex, however, it is clear that the geometric realization of  $K(\Phi)$  is the entire interval  $[0, 1]$ , it has one connected component.

## 6 Conclusion

To conclude, we have that the two Theorems 19 and 33 are indeed equivalent, even if their formulation is very different. We emphasize that the topological characterization with Theorem 34 gives a better explanation of the results primarily obtained in [FG11]. The different cases of Theorems 19 are unified when considered topologically. Note also that in the general case, the topological reasoning should be done on the continuous version of simplicial complexes, not on the abstract simplicial complexes. From Section 5.1, we see we have a simplicial complex  $K(\Phi)$  that is disconnected when seen as an abstract simplicial complex, but whose embedding (geometric realization) makes for a connected space.

This study of the solution to the Consensus problem in the general case for two processes is another argument in favor of topological methods in Distributed Computability.

We are aware of [NSW19] that appeared between revisions of this paper. We underline that Theorem 33 cannot be obtained in a straightforward way from [NSW19].

## References

- [Ada03] Giovanni Adagio. Using the topological characterization of synchronous models. *Electr. Notes Theor. Comput. Sci.*, 81, 2003.
- [AEH75] E. A. Akkoyunlu, K. Ekanadham, and R. V. Huber. Some constraints and tradeoffs in the design of network communications. In *Proceedings of the fifth ACM symposium on Operating systems principles*, pages 67–74, Austin, Texas, United States, 1975. ACM.
- [AG13] Yehuda Afek and Eli Gafni. *Asynchrony from Synchrony*, pages 225–239. Number 7730 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013.
- [AT99] Marcos Kawazoe Aguilera and Sam Toueg. A simple bivalency proof that  $t$ -resilient consensus requires  $t + 1$  rounds. *Inf. Process. Lett.*, 71(3-4):155–158, 1999.
- [BG93] Elizabeth Borowsky and Eli Gafni. Generalized flip impossibility result for  $t$ -resilient asynchronous computations. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 91–100, New York, NY, USA, 1993. ACM Press.

- [CBGS00] Bernadette Charron-Bost, Rachid Guerraoui, and André Schiper. Synchronous system and perfect failure detector: Solvability and efficiency issue. In *DSN*, pages 523–532. IEEE Computer Society, 2000.
- [CBS09] Bernadette Charron-Bost and André Schiper. The heard-of model: computing in distributed systems with benign faults. *Distributed Computing*, 22(1):49–71, 2009.
- [CGP15] Étienne Coulouma, Emmanuel Godard, and Joseph G. Peters. A characterization of oblivious message adversaries for which consensus is solvable. *Theor. Comput. Sci.*, 584:80–90, 2015.
- [CHLT00] Soma Chaudhuri, Maurice Herlihy, Nancy A. Lynch, and Mark R. Tuttle. Tight bounds for k-set agreement. *J. ACM*, 47(5):912–943, 2000.
- [FG11] Tristan Fevat and Emmanuel Godard. Minimal obstructions for the coordinated attack problem and beyond. In *Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pages 1001–1011, May 2011.
- [FLP85] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.
- [GKM14] Eli Gafni, Petr Kuznetsov, and Ciprian Manolescu. A generalized asynchronous computability theorem. In Magnús M. Halldórsson and Shlomi Dolev, editors, *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 222–231. ACM, 2014.
- [GKP03] Rachid Guerraoui, Petr Kouznetsov, and Bastian Pochon. A note on set agreement with omission failures. *Electr. Notes Theor. Comput. Sci.*, 81, 2003.
- [Gra78] Jim Gray. Notes on data base operating systems. In *Operating Systems, An Advanced Course*, pages 393–481, London, UK, 1978. Springer-Verlag.
- [HKR13] Maurice Herlihy, Dmitry N. Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, 2013.
- [HS99] Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *J. ACM*, 46(6):858–923, 1999.
- [Kuz] Petr Kuznetsov. Personal Communication.
- [Lyn96] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.
- [MR98] Yoram Moses and Sergio Rajsbaum. The unified structure of consensus: A *ayered analysis* approach. In *PODC*, pages 123–132, 1998.
- [Mun84] James R. Munkres. *Elements Of Algebraic Topology*. Addison Wesley Publishing Company, 1984.
- [NSW19] Thomas Nowak, Ulrich Schmid, and Kyrill Winkler. Topological characterization of consensus under general message adversaries. In *PODC*, pages 218–227. ACM, 2019.
- [Per15] Eloi Perdereau. Caractérisation topologique du problème des deux généraux. Master’s thesis, Université Aix-Marseille, 2015.
- [PP04] J.E. Pin and D. Perrin. *Infinite Words*, volume 141 of *Pure and Applied Mathematics*. Elsevier, 2004.
- [PSL80] L. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.
- [Ray02] Michel Raynal. Consensus in synchronous systems: a concise guided tour. *Pacific Rim International Symposium on Dependable Computing, IEEE*, 0:221, 2002.
- [San06] N. Santoro. *Design and Analysis of Distributed Algorithms*. Wiley, 2006.
- [SW89] Nicola Santoro and Peter Widmayer. Time is not a healer. In *STACS 89*, volume 349 of *Lecture Notes in Computer Science*, pages 304–313–313. Springer Berlin / Heidelberg, 1989.
- [SW07] Nicola Santoro and Peter Widmayer. Agreement in synchronous networks with ubiquitous faults. *Theor. Comput. Sci.*, 384(2-3):232–249, 2007.
- [SZ00] M. Saks and F. Zaharoglou. "wait-free k-set agreement is impossible: The topology of public knowledge. *SIAM J. on Computing*, 29:1449–1483, 2000.