

# Using the Topological Characterization of Synchronous Models <sup>★</sup>

Giovanni Adagio <sup>1</sup>

*Distributed Programming Laboratory  
School of Computer Science and Communication Systems  
Swiss Federal Institute of Technology in Lausanne (EPFL)  
Institute of Mathematics  
School of Basic Sciences  
Swiss Federal Institute of Technology in Lausanne (EPFL)*

---

## Abstract

This paper contributes to the characterization of synchronous models of distributed computing using topological techniques. We consider a generic synchronous model with send-omission failures and use a topological structure corresponding to a bounded number of rounds of the model. We observe some nice properties of the structure and derive from these properties necessary and sufficient conditions to solve consensus in this model.

---

## 1 Introduction

### Motivations

Several distributed computing models have proliferated in the last decades. Results that have been proven in these models are difficult to compare, essentially because relationships between these models are not clear. Quite recently, some preliminary steps have been taken towards providing a mathematical framework to unify these models: basically, the observation that connectivity is at the heart of many distributed computing lower bounds has led to some topological (or graph-based) characterizations of distributed models [4,7]. To our knowledge, however, the only complete characterization has been given so far for the asynchronous model with process crash failures [7]: the idea is to

---

<sup>★</sup> This work is partially supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation (under grant number 5005-67322).

<sup>1</sup> G. Adagio is a short name for *Gruppo di Algoritmi Distributi e AlGebra a l'InstitutO Politecnico di Losanna*. The group gathers, in alphabetical order, S. Blanc, R. Guerraoui, K. Hess, P. Kouznetsov, P.-E. Parent, B. Pochon, and O. Sauvageot.

represent a set of global states of the system as a mathematical structure called a *simplicial complex* [9]. Interestingly, the iterative model based on the *immediate snapshot memory* [2] has a very regular structure, corresponding to a subdivision of a simplicial complex, and is used to completely characterize the asynchronous model [7]. Very few problems are solvable in the asynchronous model however, and it is very tempting to seek characterizing models with some synchrony assumptions. Some topological constructs characterizing particular executions of a synchronous model have indeed been proposed [6], and were used to derive nice and succinct proofs of various lower bounds results. These were however only partial characterizations (i.e., considering particular executions only) and concerned a completely synchronous system.<sup>2</sup> An open and rather challenging question is how to completely characterize a generic model, parametrized with some synchrony assumptions, in a comprehensible and easy-to-use way.

## Contributions

Our approach is based on the iterative round-by-round failure detector model [5], where we consider send-omissions as the only source of failures.

In this preliminary attempt, we illustrate our characterization with a notion of graph sequences, and we (1) give a proof of the lower bound of  $f + 1$  rounds for consensus in the  $f$ -resilient omission model and (2) derive from our characterization an algorithm that matches the lower bound. We believe that the main contribution of this note is the way we derive the algorithm: we use two observations from our characterization about connected components containing omission-free executions. The first observation is a sufficient condition for a connected component to contain a omission-free execution, and the second observation relates, for a connected component with a certain number of rounds, the number of faulty processes with the existence of an omission-free execution in that component.

Our graph-based characterization has a major limitation, however: it is based on the indistinguishability of two global states for one process (it is, in a sense, customized for consensus), and its extension (e.g., to set agreement) is not trivial. We show that such a characterization allows us to reason about executions of an omission model with  $f$  possible faulty processes.

## Roadmap

Section 2 introduces our system model. Section 3 presents the characterization. Section 4 presents an application of our characterization to consensus by showing that  $f + 1$  rounds are necessary and sufficient to solve consensus in the  $f$ -resilient model.

---

<sup>2</sup> A semi-synchronous model is also considered in [6], but the difference with a synchronous model is rather small.

## 2 Model

We consider a distributed system of  $n$  processes  $\Pi = \{p_1, \dots, p_n\}$ . Processes communicate by message-passing, and each pair of processes is connected by a reliable channel. Processes may however fail by send-omissions. We assume that, in any execution of the system, at most  $f$  processes may lose messages, and we call such processes *faulty* processes. We call such a model the  $f$ -resilient omission model [5]. An execution of the  $f$ -resilient omission model is *omission-free*, if and only if all messages are sent and received in every round.

We represent executions in our model as in the *communication graphs* approach of [4,8]. We assume that processes execute a *full-information* protocol [7]: in each round, every process sends its entire local state to every other process. One round of any such execution can be described by a directed graph, the vertexes of which are labeled by the process ids and their local state in that round [4,8]. There is a directed edge in this graph from process  $p_i$  to process  $p_j$ , whenever  $p_j$  receives the message from  $p_i$ . A lack of an edge between processes means that the message is lost. We always assume that a process receives its own message, and we omit the corresponding edge. Therefore, a multi-round execution of such a system is represented by a sequence of graphs, one graph per round.

We call an  $r$ -round *execution* an execution in which all processes execute a full-information protocol for  $r$  rounds and do not execute any step for any round  $r' > r$ . An  $r$ -*sequence* is the sequence of graphs corresponding to an  $r$ -round execution.

## 3 Topological characterization

Before presenting our characterization, we first recall some basic concepts borrowed from algebraic topology (formally defined, for instance, in [9]). These concepts have been recently used in distributed computing, for instance, in [3,4,6,7].

### 3.1 Background

We represent a global state of our system of  $n$  processes by a  $(n-1)$ -dimensional simplex  $S^{n-1} = \{s_1, \dots, s_n\}$  of  $n$  vertexes, where each vertex  $s_i = \langle p_i, v_i \rangle$  corresponds to a process  $p_i$  and its local state  $v_i$  [7]. A non-empty simplex  $T$  is a *face* of a simplex  $S$  if and only if all vertexes of  $T$  are vertexes of  $S$ . A *simplicial complex*  $C$  is a set of simplexes, closed under containment, such that any face of any simplex of  $C$  is also part of  $C$ .

In our model, we consider an initial configuration (i.e., an initial global state) where processes have generic input values. The state of the system at the end of an execution in which processes started with generic values is represented by a simplex.

### 3.2 Moves

A *move* from an execution  $e$  consists in adding or removing a single arrow to  $e$ . For an execution  $e$  and a move  $s$ , we denote by  $s(e)$  the execution resulting from applying  $s$  to  $e$ .

An *elementary* move  $s$  from an  $r$ -round execution  $e$  to an  $r$ -round execution  $e'$  is a move such that  $e' = s(e)$ , and there exists at least one process  $p_i$  such that  $p_i$ 's local state after round  $r$  in  $e$  or  $e'$  is identical when starting from the same initial state in both executions.

A process  $p_i$  can *change its mind* at the beginning of a  $r$ -round execution  $e$  if and only if there exists a process  $p_j$  such that  $p_j$ 's local state after  $r$  rounds does not depend on  $p_i$ 's initial state.

We define a notion of path between two executions as in [4] (called a similarity chain in [4]). For a given execution  $e$ , a  $k$ -*path*  $P$  is a sequence  $\{s_i\}_{i=1}^k$  of elementary moves, such that  $s_1$  is an elementary move for  $e$ ,  $s_2$  is an elementary move for  $s_1(e)$ , etc. We denote by  $length(P)$  the length of the sequence and we simply say a *path* when the length of the sequence is not relevant.

For any execution  $e$ , we denote by  $P(e)$  the execution resulting from successively applying the elementary moves of  $P$  to  $e$  (that is, applying  $s_1$  to  $e$ ,  $s_2$  to  $s_1(e)$ , etc.). For two executions  $e$  and  $e'$ , we say that  $e'$  is *reachable* from  $e$ , and we write  $e \sim e'$ , if and only if there exists a path  $P$  such that  $P(e) = e'$ .

### 3.3 Characterization and Connectivity

The characterization takes into account all possible executions of a full-information protocol running in the  $f$ -resilient omission model, and corresponds to a generalization of [1]. The generalization is threefold: (i) we consider an arbitrary number of failures  $f$  ([1] considers only one failure), (ii) we generalize the failures to send-omissions (which allows for some synchrony assumptions [5]) and (iii) we have a notion of *degree* of similarity between two global states (the degree of similarity corresponds to the dimension of the intersection of the two corresponding simplexes) [6].

We established a correspondence between a set of  $r$ -sequences of graphs and a simplicial complex, by identifying an  $r$ -sequence of graphs in our model with a simplex of dimension  $n - 1$ . In our characterization, we thus consider the simplicial complex that represents all possible  $r$ -executions of our system.

For any given  $r$ -execution  $e$  and elementary move  $s$ , consider the execution  $e' = s(e)$ . Executions  $e$  and  $e'$  both correspond to simplexes in a simplicial complex. By definition of  $s$ , there exists at least one process  $p_j$  which does not distinguish between  $e$  and  $e'$ , so we can *glue* the simplexes corresponding to  $e$  and  $e'$  on the vertex corresponding to  $p_j$  (or on the simplex determined by processes that do not distinguish the two global states).

Figure 1 gives an example of how simplexes are glued together. A tetrahedron represents an execution of a 4-process protocol. The different ways of

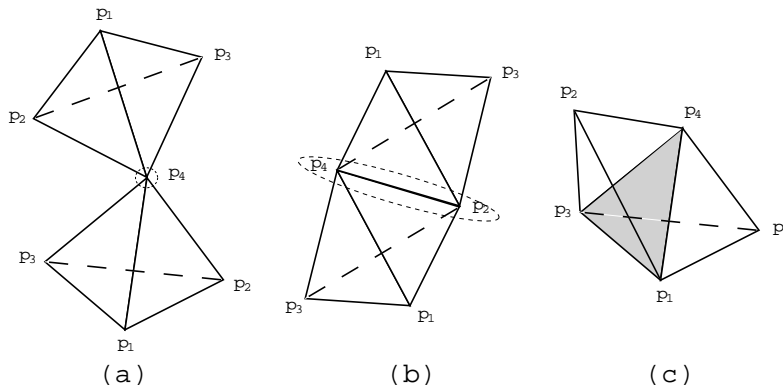


Fig. 1. Gluing simplexes

gluing two simplexes are given by the following three situations. In case **(a)**, process  $p_4$  is the only process that cannot distinguish between the two executions. In case **(b)**, processes  $p_4$  and  $p_2$  cannot distinguish between the two executions. In case **(c)**, processes  $p_4$ ,  $p_3$  and  $p_1$  cannot distinguish between the two executions.

Two simplexes  $S_i$  and  $S_j$  are said to be *adjacent* in a simplicial complex if their corresponding executions  $e_i$  and  $e_j$  differ by at least one elementary move. Two adjacent simplexes  $S_i$  and  $S_j$  always have a face, corresponding to  $S_i \cap S_j$ , in common.

A  $k$ -path for an execution  $e$  implies a sequence of  $k+1$  simplexes  $S_0, \dots, S_k$  in a simplicial complex, of dimension  $(n-1)$  each, such that  $S_0$  corresponds to execution  $e$ , for every  $i \in [1, k-1]$ ,  $S_i$  and  $S_{i+1}$  are adjacent.

Two simplexes of a simplicial complex are *congruent* if and only if there exists a path in the simplicial complex that connect both simplexes. Congruency is a reflexive, symmetric and transitive relation, and we can therefore partition a simplicial complex into congruency classes, which form the *connected components* of the complex. In the rest of the paper, we interchangeably use the notion of execution, simplex, or sequence of graphs.

### 3.3.1 On the sufficiency for omission-freedom

The following lemma gives a sufficient condition for a connected component to contain the omission-free execution. Roughly speaking, it says that if a component includes a path in which every process can change its mind, then the component also includes the omission-free execution.

**Lemma 3.1** *If a connected component contains executions  $e_i$  ( $1 \leq i \leq n$ ), such that process  $p_i$  can change its mind in  $e_i$ , then this component contains the omission-free execution  $e_0$ .*

**Proof.** Let  $C$  be any connected component satisfying the condition of the lemma, i.e., there exists an execution  $e \in C$  and a path  $P = \{s_i\}$  of elementary moves, defined on  $e$ , that passes through executions  $e_1, \dots, e_n$  where, respectively, processes  $p_1, \dots, p_n$  change their mind. We want to show that

there exists a sequence of elementary moves that connects  $e$  with the omission-free execution  $e_0$ .

Consider an elementary move  $s$ , defined on an execution  $e$ , that modifies (adds or removes an arrow in) the  $r$ -th round of  $e$ . Let  $e'$  be an execution that is identical to  $e$  in rounds  $r' \geq r$ . Obviously,  $s$  applied to  $e'$  is also elementary, as the same process cannot distinguish  $e'$  and the result of applying  $s$  to  $e'$ .

Now it is easy to see that if we drop from  $P$  all elementary moves which remove arrows in the first round of executions, then we still obtain a path  $P'$  of elementary moves. The corresponding execution  $e'_1$  differs from  $e_1$  in the first round only. Thus  $p_1$  can change its mind at the end of the first round of  $e'_1$ , and some process does not see it at the end of the execution (otherwise  $p_1$  could not change its mind in  $e_1$ ). So we can let  $p_1$  receive all messages in the first round of  $e'_1$ , i.e., adding arrows from all processes to  $p_1$  in the first round is an elementary move. We then continue applying the moves of  $P'$  until we reach  $e'_2$  and so on.

As a result, we obtain a path that connects  $e$  to an execution in which every message is received in the first round. Moreover, the sequence passes through executions  $e_1^1, \dots, e_n^1$  where, respectively, processes  $p_1, \dots, p_n$  can change their states at the end of the first round.

Inductively applying the argument to the second round etc., we finally obtain a path that connects  $e$  to the omission-free execution. Note that we modify the elementary moves only by adding arrows, i.e., we do not introduce more failures. As a result, we cannot violate the limit of at most  $f$  faulty processes in every execution of our model.  $\square$

### 3.3.2 On the necessity of omissions

We observe another property of connected components. Roughly speaking, this property gives the least number of faulty processes in some execution of a connected component which contains at the same time (i) the omission-free execution and, (ii) an execution in which the initial local state of several processes is never received by some process.

**Lemma 3.2** *Any path  $P$  that connects the  $r$ -round omission-free execution  $e_0$  with an execution  $e^i$  in which  $i$  processes can simultaneously change their minds ( $i < n$ ), passes by an execution in which at least  $r + i - 1$  processes are faulty.*

**Proof.** We proceed by induction on  $r$ . The case  $r = 1$  is trivial: a process can change its mind in a 1-round execution only if it is faulty. Now assume that the claim holds for  $r$ -round executions, for any  $i$ . Consider a path  $P$  that satisfies the condition of the lemma, namely,  $P$  connects the  $(r + 1)$ -round omission-free execution  $e_0$  with an execution  $e^i$  in which  $i$  processes can simultaneously change their minds.

- (i) Assume that a set  $X$  of  $i$  processes can change their minds in  $e^i$ . It is not difficult to see that we can remove all the links departing from the set  $X$

in the first round in an elementary way: for any process  $p \in X$ , there is a process  $q$  that cannot see the change of  $p$ 's initial value. Thus,  $q$  cannot see that a link from  $p$  is removed. We define by  $\tilde{P}$  an extension of path  $P$  by adding the moves in which all the links from  $X$  are removed. The last execution of  $\tilde{P}$  is  $\tilde{e}^i$  in which no message from  $X$  is ever received.

- (ii) It is important to notice that the set of processes that are faulty in all executions between  $e^i$  and  $\tilde{e}^i$  are also faulty in  $e^i$ . Indeed all processes of  $X$  are already faulty in  $e^i$  and no more faulty processes can be obtained by removing links from them.
- (iii) Consider the last move of  $\tilde{P}$  in which the last link from  $X$  to some process  $p \in \Pi \setminus X$  is removed. We denote the resulting execution by  $e_{r+1}^i$  and the last  $r$  rounds of it by  $e_r^i$ .
- (iv) Since the removal of the link to  $p$  in the first round is an elementary move, then  $p$  can change its initial value in the beginning of the second round. As a result,  $p$  can change its mind in  $e_r^i$ . Since all processes from  $X$  can change their minds in  $e_{r+1}^i$ , they can change their minds also in  $e_r^i$ . As a result, path  $P$  restricted to the last  $r$  rounds leads to an execution  $e_r^i$  in which  $i + 1$  processes from the set  $X \cup \{p\}$  can change their minds.
- (v) By the induction hypothesis,  $\tilde{P}$  passes through an execution  $e$ , in which at least  $r + i$  processes are faulty. The executions obtained by the moves  $\tilde{P} \setminus P$  do not contain more faulty processes than  $e^i$ . Thus,  $P$  also passes through an execution in which at least  $r + i = (r + 1) + i - 1$  processes are faulty.

□

## 4 Application to Consensus

Informally, in the consensus problem, each process proposes a value, and all processes must then agree (or decide) on a single value among the proposed ones. More precisely, we require that (*validity*) every decided value is a proposed value, (*agreement*) no two process decide differently, and (*termination*) every process eventually decides.<sup>3</sup> In particular, if all processes propose the same value, then by the validity requirement this value must be decided. As an immediate consequence we obtain the following proposition.

**Proposition 4.1** *Consensus cannot be solved if and only if there exists a connected component  $C$  containing, for every  $i \in [1, n]$ , an  $r$ -round execution  $e_i$ , such that process  $p_i$  can change its mind in  $e_i$ .*

**Proof.** “If” direction ( $\Leftarrow$ ):

Consider a connected component  $C$  that satisfies the condition of the proposition, namely,  $C$  contains, for every  $i \in [1, n]$ , execution  $e_i$  in which process

<sup>3</sup> Note that a process is not allowed to halt in our model, and thus, every process must eventually decide.

$p_i$  can change its mind. Consider the initial state of the system in which all processes propose the value 0. By validity, all processes decide 0, in any execution, and in particular in  $e_1$ . Consider executions  $\{e_i\}$  for increasing values of  $i$ , starting from  $e_1$ , and ending at  $e_n$ . Considering execution  $e_i$ , one can change the initial value of process  $p_i$  to a different value, say 1. After considering execution  $e_n$ , all processes now propose 1. By validity, they all decide 1 in  $e_n$ . There are two cases to consider: (i) there exists two adjacent executions  $e'$ , where processes decide 0, and  $e''$ , where processes decide 1, or (ii) there exists an execution  $e_j$  after which processes decide on 0 if  $p_j$  proposes 0 and decide 1 if  $p_j$  proposes 1. For case (i) and because of adjacency, there exists at least one process  $p_j$  which has the same local state at the end of executions  $e'$  and  $e''$  and which decide 0 in  $e'$  and 1 in  $e''$  – a contradiction. For case (ii), by definition there exists a process  $p_{j'}$  whose local state at the end of execution  $e_i$  does not contain  $p_i$ 's local state, thus  $p_{j'}$  decides the same value in  $e_i$  independently of  $p_i$ 's value – a contradiction.

“Only if” direction ( $\Rightarrow$ ):

We prove the contrapositive of the claim, namely that if there is no connected component which contains executions  $e_i$  ( $1 \leq i \leq n$ ), such that process  $p_i$  can change its mind in  $e_i$ , then consensus is solvable.

For any connected component  $C$ , we denote by  $K$  the set of processes such that for any execution  $e \in C$ , processes in  $K$  cannot change their minds. For any connected component  $C$  satisfying the condition of the proposition,  $|K| \geq 1$  and every process is aware of the initial values of processes in  $K$ , in any execution  $e \in C$ . We can thus define a deterministic decision function  $\delta_C$  of the initial values of processes in  $K$ . Function  $\delta_C$  outputs the decision value associated with the component  $C$ . By construction,  $\delta_C$  outputs just a single decision value within  $C$ , and thus ensures agreement. Validity follows from the fact that  $\delta_C$  only takes as arguments initial values of processes. Termination follows from the fact the each process executes a bounded number of rounds before deciding (each execution in  $C$  is composed of  $r$  rounds).  $\square$

#### 4.1 Application to Consensus: Necessity

In this section, we observe another specific property of a connected component. Basically, in any connected component containing the omission-free  $r$ -round ( $1 \leq r \leq f$ ) execution  $e_0$ , there exists a path from  $e_0$  to an  $r$ -execution where a process can change its mind. This allows us to deduce a lower bound of  $f + 1$  rounds for solving consensus in the  $f$ -resilient omission model.

**Theorem 4.2** *If a connected component  $C$  contains the omission-free  $r$ -round ( $1 \leq r \leq f$ ) execution  $e_0$ , there exists an execution  $e \in C$ , such that a process  $p$  can change its mind in  $e$ .*

**Proof.** First note that it suffices to prove the proposition for  $r = f$ , as this proof immediately leads to the case where  $r < f$ . By symmetry, as  $C$  contains



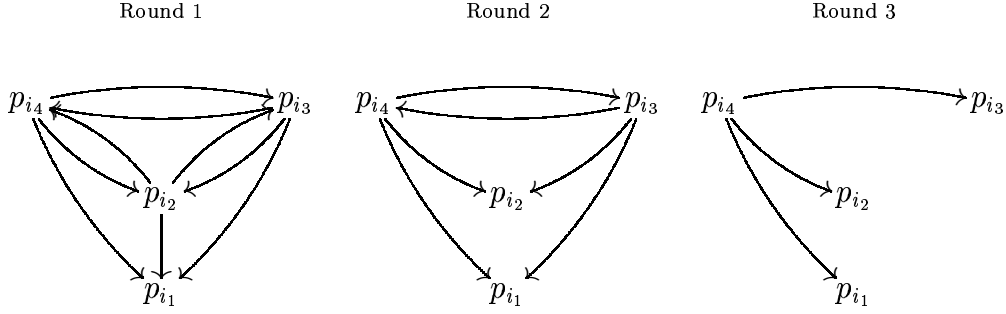


Fig. 2. Execution  $e$  with  $n = 4$  and  $f = 3$

the failure-free execution  $e_0$ , the theorem implies that any process can change its mind in some execution in  $\mathcal{C}$ .

Consider now the  $f$ -round execution  $e$  where, in each round  $k$ , exactly  $k$  processes fail by omission. (An execution  $e$  is given in Fig. 2 for the case where  $n = 4$  and  $f = 3$ .)

**Round 1:** Exactly one process  $p_{i_1}$  fails by omitting all its messages.

**Round 2:** Exactly two processes  $p_{i_1}, p_{i_2}$  fail by omitting all their messages.

...

**Round  $f$ :** Exactly  $f$  processes  $p_{i_1}, \dots, p_{i_f}$  fail by omitting all their messages.

In execution  $e$ , as  $p_{i_1}$  fails by omitting all its messages, it can clearly change its mind. We show how one can construct a path  $P$  from the omission-free execution  $e_0$  to execution  $e$ . In round  $f$ , it is easy to see that we can remove any arrow of any execution so that some process cannot notice it. As a result, we can construct a sequence of elementary moves that connect  $e_0$  to an execution  $e_1$  in which no arrows depart from processes  $p_{i_1}, \dots, p_{i_f}$ .

Suppose by induction that, for any  $k+1$  processes  $p_{i_1}, \dots, p_{i_{k+1}}$ , there exists an execution  $e_{f-k}$  ( $e_{f-k} \sim e_0$ ) in which  $p_{i_1}, \dots, p_{i_{k+1}}$  fail by omitting all their messages in round  $k' \geq k+1$ . We exhibit a sequence of elementary moves to reach execution  $e_{f-k+1}$ , where  $k$  processes  $p_{i_1}, \dots, p_{i_k}$  fail by omitting all their messages in round  $k$ .

Denote by  $e_{f-k+1}^0$  the execution identical to  $e_{f-k}$  except that, in round  $k$ , no message is exchanged among processes  $p_{i_1}, \dots, p_{i_k}$ . Indeed, all these processes are silent starting from the next round. Thus,  $e_{f-k} \sim e_{f-k+1}^0$ . We also denote by  $m_1, \dots, m_j, \dots, m_l$  the messages sent in round  $k$ , from processes  $p_{i_1}, \dots, p_{i_k}$  to the remaining processes. We consider the sequence of elementary moves that successively remove messages  $m_1, \dots, m_j, \dots, m_l$ , and we denote by  $e_{f-k+1}^1, \dots, e_{f-k+1}^l$  the corresponding executions (in  $e_{f-k+1}^j$ , messages  $m_1, \dots, m_j$  are lost). We show, by induction on  $j$ , that  $e_0 \sim e_{f-k+1}^j$ .

Initially, we have seen that execution  $e_{f-k} \sim e_{f-k+1}^0$ . By the induction hypothesis, there is an execution  $e_{f-k+1}^{j-1} \sim e_{f-k+1}^0$ . Consider execution  $e_{f-k+1}^j$

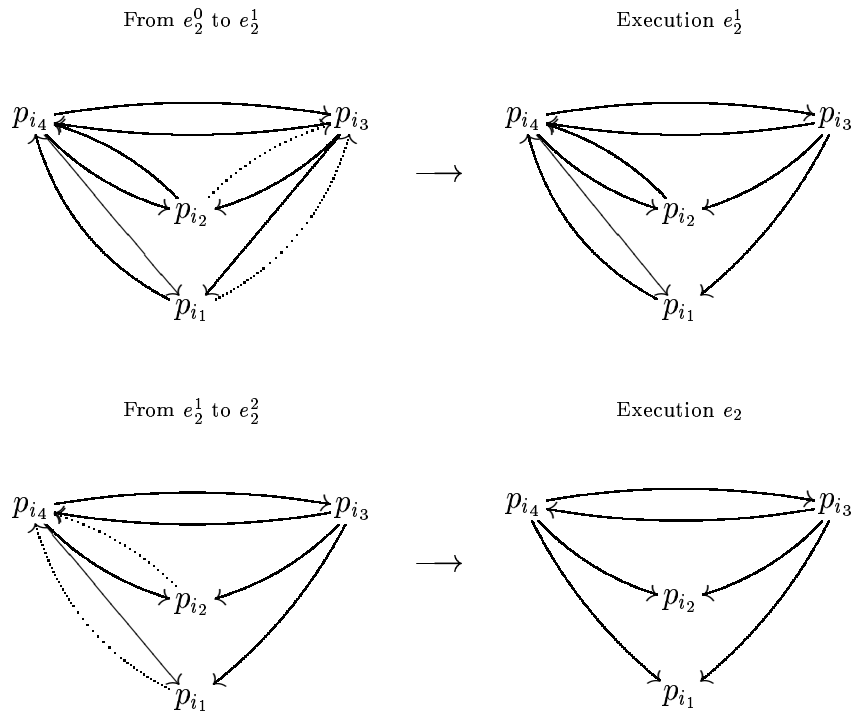


Fig. 3. Induction step for round 2 with  $n = 4$  and  $f = 3$

where messages  $m_1, \dots, m_j$  are lost. Denote by  $p_{i_{k+1}}$  the process that receives  $m_j$  in  $e_{f-k+1}^{j-1}$ . As  $p_{i_{k+1}}$  does not send any message starting from round  $k+1$ , by construction, then  $e_{f-k+1}^j$  is indistinguishable from  $e_{f-k+1}^{j-1}$ . By the induction hypothesis,  $e_{f-k+1}^j \sim e_{f-k+1}^{j-1} \sim e_{f-k+1}^0$ , which implies in turn that  $e_{f-k+1}^j \sim e_{f-k+1}^0$ . For  $j = l$ , we have  $e_{f-k} \sim e_{f-k+1}$ . We conclude by considering execution  $e_f = e$ .

The example presented in Fig. 3 considers round 2 of executions  $e_1$  and  $e_2$  for the case where  $n = 4$  and  $f = 3$ . Figure 3 illustrates the inductive step to connect  $e_1$  to  $e_2$ . The two graphs at the top illustrate the connection from  $e_2^0$  to  $e_2^1$ , by considering that process  $p_{i_3}$  fail by omitting all its messages in round 3. The two graphs at the bottom illustrate the connection from  $e_2^1$  to  $e_2^2$ , by considering that process  $p_{i_4}$  fail by omitting all its messages in round 3. The resulting execution corresponds to  $e_2$ . □

From Proposition 4.1 we immediately obtain the following corollary.

**Corollary 4.3** *In an  $f$ -resilient omission model, no protocol can solve consensus in  $f$  rounds.*

## 4.2 Application to Consensus: Sufficiency

In this section, we show how one can reason about the solvability of consensus by using the properties on connected components previously observed. We establish a contradiction with the base assumption on our  $f$ -resilient omission model, which allows us to conclude that consensus is solvable in the  $f$ -resilient model in exactly  $f + 1$  rounds.

Precisely, Lemma 3.1 gives a sufficient condition for a connected component to contain the omission-free execution, whereas Lemma 3.2 gives a characteristic of the connected component containing the omission-free execution. We deduce an upper bound on the number of rounds to solve consensus by using the two lemmas in a complementary way.

**Theorem 4.4** *There is a protocol that solves consensus in an  $f$ -resilient omission model processes in  $f + 1$  rounds.*

**Proof.** Assume no protocol solves consensus in  $f + 1$  rounds. By Proposition 4.1, there exists a component that, for every process  $p_i$ , contains an execution in which  $p_i$  can change its mind. By Lemma 3.1, the component also contains the omission-free execution. Thus there is a path  $P$  that links the omission-free execution with an execution in which one process can change its initial value. By Lemma 3.2,  $P$  passes through an execution in which  $f + 1$  processes are faulty. This contradicts with the assumption that at most  $f$  processes can omit messages in any execution of our model.  $\square$

## 5 Acknowledgments

Eli Gafni, Maurice Herlihy and Sergio Rajsbaum introduced us to the application of topology to distributed computing.

## References

- [1] O. Biran, S. Moran, and S. Zaks. A combinatorial characterization of the distributed 1-solvable tasks. *Journal of Algorithms*, 11(3):420–440, September 1990.
- [2] E. Borowsky and E. Gafni. Immediate atomic snapshots and fast renaming. In *Proceedings of the 12<sup>th</sup> ACM Symposium on Principles of Distributed Computing*, pages 41–51, 1993.
- [3] E. Borowsky and E. Gafni. A simple algorithmically reasoned characterization of wait-free computations (extended abstract). In *Proceedings of the 16<sup>th</sup> ACM Symposium on Principles of Distributed Computing (PODC)*, pages 189–198, 1997.
- [4] S. Chaudhuri, M. Herlihy, N. A. Lynch, and M. R. Tuttle. Tight bounds for  $k$ -set agreement. *Journal of the ACM (JACM)*, 47(5):912–943, 2000.

- [5] E. Gafni. Round-by-round fault detector—unifying synchrony and asynchrony. In *Proceedings of the 17<sup>th</sup> ACM Symposium on Principles of Distributed Computing*, 1998.
- [6] M. Herlihy, S. Rajsbaum, and M. Tuttle. Unifying synchronous and asynchronous message-passing models. In *Proceedings of the 17<sup>th</sup> ACM Symposium on Principles of Distributed Computing*, pages 133–142, 1998.
- [7] M. Herlihy and N. Shavit. The topological structure of asynchronous computability. *Journal of the ACM*, 46(6):858–923, 1999.
- [8] Y. Moses and M. R. Tuttle. Programming simultaneous actions using common knowledge. *Algorithmica*, 3(1):121–169, 1988.
- [9] J. R. Munkres. *Elements of Algebraic Topology*. Addison-Wesley, Reading MA, 1984.