Alternative proof for consensus impossibility

Luis Carlos Soldevilla Estrada

June 2019

1 Abstract

In distributed computing *k-approximate agreement* is a task where n processes propose a value and later they have to decide on one of them, so that in total k distinct values are decided. A special case of this set of tasks is the 0-approximate agreement, also known as consensus, where the same value (or 0 distinct values) is decided among all processes. In past papers, it has been proved that consensus does not have a solution in a synchronous system and topological tools have been used to prove the impossibility of consensus.

While connectivity of graphs has been used to prove consensus's impossibility, this paper aims to analyze this problem from a slightly different perspective. Through the disconnectivity of simplicial complexes derived from the interaction of processes, we intend to give an alternative proof of consensus's impossibility. Our proof uses new graphs that give us different tools to prove consensus's impossibility

2 Introduction

In this paper, our aim is to analyze consensus from two slightly different perspectives. First, we will explain the impossibility of this task through a world problem and a couple of connected simplicial complexes (which will be defined later). The second way to analyze consensus's impossibility is by presenting a new approach on how we relate processes in a system and,

unlike the first perspective, how disconnectivity of simplicial complexes give us information about consensus's impossibility. We find the latter approach quite interesting because it can give us new elements, in this case disconnectivity, to consider when analyzing other impossibility results or tasks. It is important to note that the second approach only applies to systems of 2 and 3 processes and that the general cases may be subject of a new research stream.

3 Basic definitions

3.1 What is distributed computing?

Distributed computing is a field of computer science that studies tasks that can be solved using partial information of the reality. As an example to understand better this definition and how this computer science field works, consider two friends, Bob and Alice, playing a card game. At the beginning and for the rest of the game, each player has two cards and no information about the other's cards. In every round each player can decide whether to drop a card, so that both players can see it, and draw a new card from the deck, or stay still. The aim of this game is to form a staircase, as in poker, with the cards that Alice and Bob have. To achieve this task, each individual must form a staircase on their own and coordinate (or communicate) with the other participant, by dropping cards in each round, to create the desired staircase.

3.2 Topology, distributed computing and graph theory

Taking the example of 3.1 as the starting point, we will define essential concepts that are crucial when studying distributed computing. Theoretically speaking, Alice and Bob can be considered as computers, processors or any other computing entity, and are generally denoted as processes p_i. Just as the rules of the game presented above, processes start with their own input (the cards with which Alice and Bob started) and after every *round* of communication they receive new messages (the card that one of the participants drops). The set of messages that p_i receives and its initial input are part of what p_i sees about the reality, denoted as $view_i$ of p_i. Putting together

n sequential processes: $p_1, p_2, ..., p_n$, we get a *system*. Besides defining the essential properties of processes, we need to establish what kind of communication system we are going to use throughout the article. This paper deals with a synchronous system of communication: messages between processes are sent at the same time, processes send a message to each other while they receive information that comes from other entities.

After describing the essential properties of processes in distributed computing, we need to define how topology relates to the study of processes and their interaction. *Topology* is a field of mathematics that studies how objects' properties are preserved after deformations, twistings and stretchings. However, the area of topology that is exploited the most is *combinatorial topology* which studies properties of geometric figures by analyzing them through elementary geometric figures [1]. The building blocks of these figures are called *vertices* and we denote them as the pair $(p_i,\ x_i)$, where x_i represents the initial input of p_i in every round of communication. Consider the participation of processes $p_1, p_2, ..., p_n$, the *state* of the system after r rounds is denoted as the set:

$$s = \{(p_1, x_1), (p_2, x_2), ..., (p_n, x_n)\}$$

This set is called a *simplex* of dimension $n - 1$. If we put together all possible simplexes of a system, we get a set denoted as *simplicial complex (complex)* K. All the elements of K satisfy a couple of conditions: if $s \in K$ and $\hat{s} \subseteq s$, then $\hat{s} \in K$ (closure under inclusion).

In addition to topology, we need the essentials of graph theory to analyze a 2-process system. We start with the definition of a graph. (All the definitions presented in the rest of the chapter are taken from (5))

Definition 1 A *graph* is a finite S together with a collection G of subset of S, such that

1. If $X \in G$, then $|X| \leq 2$
2. For all $s \in S$, we have $\{s\} \in G$
3. If $X \in G$ and $Y \subseteq X$, then $Y \in G$

[1] "Combinatorial topology", Merriam-Webster dictionary, https://www.merriam-webster.com/dictionary/combinatorial%20topology

Complementing property 1 of definition 1, the reader must know that if $X = \{(p_i, x_i), (p_j, x_j)\}$, then X is an edge; and if $X = (p_i, x_i)$, then it is a vertex.

Now, consider two graphs G and H, we define a *vertex map* $\mu : V(G) \to V(H)$ as a function that carries vertex of G to a vertex of H, where $V(G)$ represents the set of vertices of G. The vertex map μ is a *simplicial map* if it also carries simplices to simplices: that is, if $\{s_0, s_1\}$ is a simplex in G, then $\{\mu(s_0), \mu(s_1)\}$ is a simplex in H.

Besides simplicial maps and vertex maps, another object that will be useful for this paper are the *carrier maps*. They are defined as follows:

Definition 2 Given two graphs G and H, a *carrier map* $\phi : G \to 2^H$ takes each simplex $\sigma \in G$ to a subgraph $\phi(\sigma)$ of H such that ϕ satisfies the following *monotonicity* property: For all $\sigma, \tau \in G$, if $\sigma \subseteq \tau$, then $\phi(\sigma) \subseteq \phi(\tau)$

The map of definition 2 is essential to define what a *task* is. First, consider processes A, B, V^{in} as a domain of *input values*, and V^{out} as a domain of *output values*. A *task* is defined as a triple (I, O, Δ) that has the following properties,

- I is an *input graph* whose vertices' values come from V^{in}
- O is an *output graph* whose vertices' values come from V^{out}
- Δ is a carrier map from I to O

After defining a task, we have to understand how to represent it graphically. Consider a round of communication between process and the new *views* that each process gets at the end of the round. One way to represent all the possible views or final states of the processes is through a graph whose vertices are (p_i, x_i), just as we defined it previously. Such graph is usually called *protocol graph* P and is strongly related to a map Ξ called the *execution carrier map*, that carries each simplex to a subgraph of the protocol graph. More specifically, Ξ carries each input vertex (p_i, x_i) to the solo execution in which p_i finishes the protocol without haring from other processes. Additionally, it carries each edge $\{(p_i, x_i), (p_k, x_k)\}$ to the subgraph of executions where p_i starts with x_i and p_k with x_k.

To finalize the set of definitions needed for future sections, we are going

to present two useful concepts taken from (5) to determine the solvability of a task.

Definition 3 The decision map δ is *carried by* the carrier map Δ if

- For each input vertex s, $\delta\big(\Xi(s)\big) \subseteq \Delta(s)$
- For each input edge σ, $\delta\big(\Xi(\sigma)\big) \subseteq \Delta(\sigma)$

Definition 4 The protocol $\big(I, P, \Xi\big)$ *solves* the task $\big(I, O, \Delta\big)$ if there is a simplicial *decision map* δ from P to O such that $\delta \circ \Xi$ is carried by Δ

4 Consensus problem

4.1 Communication after one round

To study the impossibility of consensus in a 2-process system we consider the following problem:

Consider a 2-process system, ***A*** *with input* u *and* ***B*** *with* v*. Processes communicate through messages in a synchronous way: at the same time, both processes send a message to each other, while each computing entity receives a message that was sent to it. Now consider the possibility that whenever a process send a message, the latter can get lost. Determine if the consensus task can be solved after a round of communication.*

The best way to approach this problem is through the drawing of simplicial complexes (or graphs) to represent all possible communications between **A** and **B**. First, we start by representing the initial state of the system:

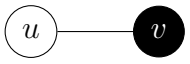


Figure 1: Initial state of the system

From now on process **A** is represented as the white node and **B** as the black node. Although the initial state of this system does not give us too much information, it is important to understand it. The connection between nodes or vertices means that the processes can communicate and exchange

their views (the content of each vertex) with each other. So, considering the first round of communication among processes we get:

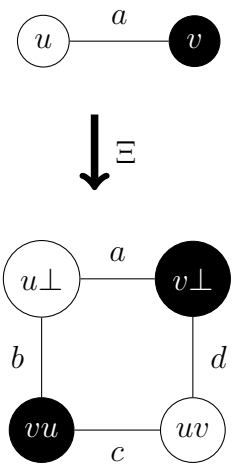


Figure 2: State of the system after one round of communication

In figure 2 we present all the possible views that the processes can acquire after communicating for the first time. Each edge represents a possible final state of processes A and B. Moreover, the arrow represents the map that we apply to the initial configuration to get to the graph desired, which is P.

To understand figure 2, recall the constraint of the problem presented at the beginning of the section: whenever messages are sent between processes they can get lost. The first possible scenario is when the messages sent by A and B are lost and both processes stay still (edge a). ($\perp$ stands for the message that did not get to a process) Another possibility that must be considered is when only one message did not arrive to the other process: b describes when input of process A arrives to B but not viceversa, and d describes the opposite situation. Finally, we get the scenario where the messages arrive to A and B without any problem and the processes have each others' view (edge c).

The vertices of the simplicial complex presented in figure 2 represent the set of values (view) that each process can decide from. To determine whether consensus can be solved in this specific system we need to make sure that regardless of the possible decisions the processes can make, the final decisions will always be the same. Therefore, to prove the impossibility of consensus we only need to find a scenario where processes A and B decide on different values. Consider the following possibility:

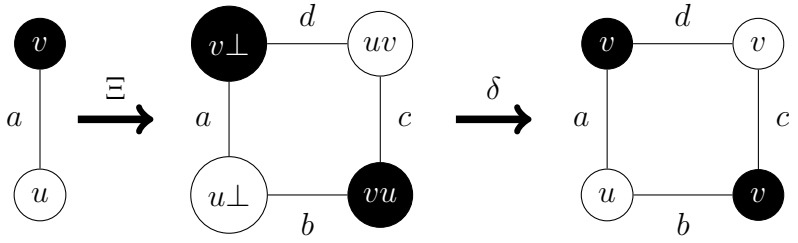
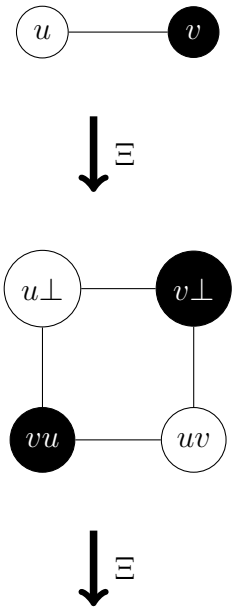


Figure 3: Decision complex after on round of communication

The complex representing the possible decisions of the processes is obtained by applying a decision map δ to the simplex produced by Ξ. In figure 3 we only represent one possible set of decisions, but this is enough to prove consensus's impossibility. Edges d and c do not have any problem but if we analyze a and b we find that there is a situation in which A and B decide different values. This sole example suffices to prove that consensus in this system is impossible to solve only for 1 round.

4.2 Analysis of 2 rounds of communication

Now that we have explained how we obtain the complex representing the first round of communication, we can present directly the complex of 2 communication rounds.



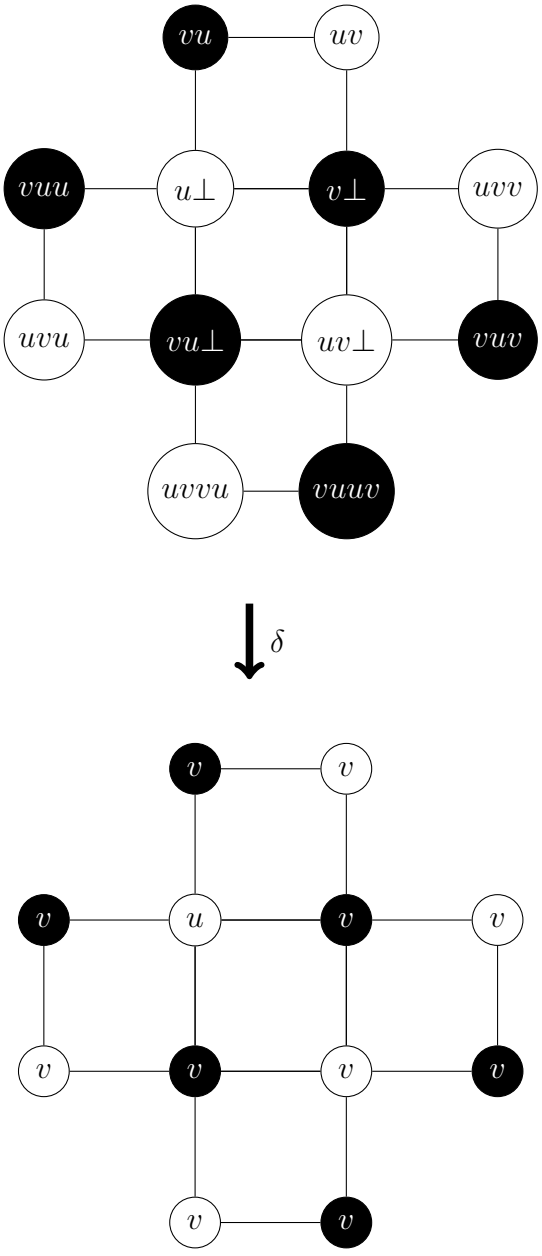


Figure 4: P after two rounds of communication

The analysis to determine the solvability of consensus goes the same way as in section 4.1. Considering the complex produced by δ, we can clearly see that we can find a decision complex where there is at least one edge joining two different outputs. Since we only need to find a counter example to prove impossibility, we can take figure 4 as our proof and present the same argument as in figure 3.

To prove that consensus is still impossible to solve for n rounds, we need to take into account what Ξ and δ do to our graphs, and recall what **Definition 4** states. Let consensus be described by the task $\big(I, O, \Delta\big)$, and, thus, the carrier map Δ would send I to a graph where all possible decisions of A and B are the same. To illustrate, consider the product of applying Δ:

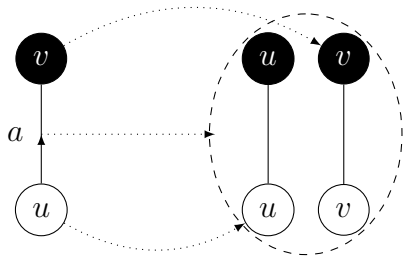


Figure 5: Δ applied to vertices and edges of the initial graph

Considering what the carrier map Δ should do if consensus were solvable in this system (above figure), we need to prove that for all simplicial decision maps δ, $\delta \circ \Xi$ is not carried by Δ. To prove the latter we only need to find an edge from I, the input graph, such that $\delta\big(\Xi\big(\sigma\big)\big) \not\subseteq \Delta\big(\sigma\big)$.

First, we have to express mathematically what Δ does to the edge of I in the figure above:

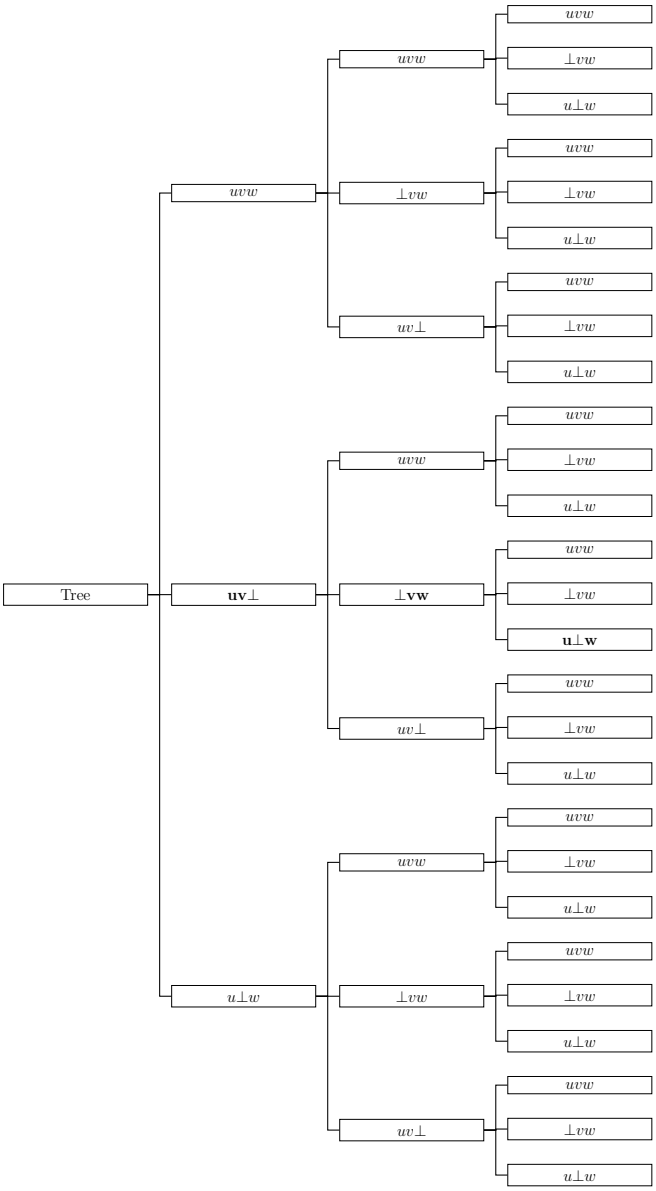
$$\Delta\big(\{\big(A, u\big), \big(B, v\big)\}\big) =$$
$$\{\{\big(A, u\big), \big(B, u\big)\}, \{\big(A, v\big), \big(B, v\big)\}, \big(A, u\big), \big(A, v\big), \big(B, u\big), \big(B, v\big)\}$$

Let's take the only edge in I: $\sigma = \{\big(A, u\big), \big(B, v\big)\}$, and let δ be any simplicial decision map. From figure 2, definition 3, and applying Ξ to σ we know that $\{\big(A, u\bot\big), \big(B, v\bot\big)\} \in \Xi\big(\sigma\big)$, and, since δ is a simplicial map, $\{\big(A, u\big), \big(B, v\big)\} \in \delta\big(\Xi\big(\sigma\big)\big)$ (we avoided writing the whole output of $\Xi\big(\sigma\big)$ because we only care about a specific edge of $\Xi\big(\sigma\big)$). Finally, we clearly have that $\{\big(A, u\big), \big(B, v\big)\} \notin \Delta\big(\sigma\big)$, and $\delta\big(\Xi\big(\sigma\big)\big) \not\subseteq \Delta\big(\sigma\big)$. Therefore, δ is not carried by Δ, for any arbitrary δ, and the task $\big(I, O, \Delta\big)$ (consensus) cannot be solved.

4.3 Analysis in a 3-process system

As the number of processes increases, the graph needed to analyze the consensus task becomes more complicated. This is why we are analyzing the

same problem presented in the last section for a 3-process system but with a *tree*. In this analysis consider 3 processes A, B, and C with initial inputs u, v and w, respectively. The following tree represents the different initial states.



Each node of the tree represents a vertex in the protocol graph and every triple formed by joining connected nodes between levels, just as the bold one, represents a face of such graph. Figure above only represents the first round of communications and allows us to give a proof about the impossibility of

consensus. Taking the first bolded triple we have the face:

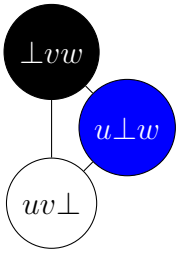


Figure 6: graph representation of bolded sequence of leaves in Figure 6

where blue stands for C, black B and white A. After applying any δ to this subgraph, we have to analyze three different scenarios. For consensus to be accomplished in this subgraph, we can have all the processes decide u, but B cannot because he has not even received it. Another option could be to decide v, but C can't for the same reason of the previous case. Lastly, all processes can decide w, however A has not received that view and, thus, can't decide on it. The dismissal of scenarios where the processes can decide on the same value leads to the existence of a subgraph in the output graph where consensus is not achieved. Consequently, we can confirm the impossibility of consensus for the first round. After proving our goal only for 1 round, the generalization's proof for n rounds is similar to the one presented for a 2-process system.

5 Alternative proof for consensus's impossibility

5.1 Analysis in a 2-process system

Although in this section we present a slightly different proof for consensus's impossibility, some new concepts and definitions are closely related to protocol graphs presented earlier. Let's start with the building blocks of our graphs: vertices in this environment do not represent inputs, views or outputs, instead they only represent processes. Additionally, two processes are said to have a *perfect communication* if both processes receive each other's messages. Using this new definition, connectivity in this new environment is defined as follows:

Definition 5 Two vertices are connected if there is a *perfect communication* between the processes they represent.

To understand better this definition consider how the protocol graph, denoted as $\mathcal{P}$, in our new environment would look like:

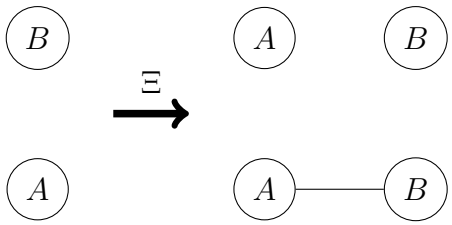


Figure 7: Protocol graph $\mathcal{P}$

Before moving on with the explanation of $\mathcal{P}$, we need to understand the configuration of the initial state. The disconnectivity of the latter is due to the fact that before any kind of communication A and B are just standing still. The reader can depict them just as two entities that do not have any interaction at all.

To clarify our graph $\mathcal{P}$, we have to consider the graph P presented in figure 2:

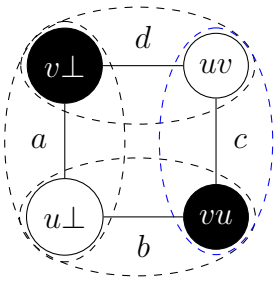


Figure 8: Graph of all possible views after 1 communication round

If we observe carefully our protocol graph P, presented above, we can see that regarding our new communication's definition we have two different scenarios. The first one (surrounded by a blue ellipse) describes a perfect communication between processes and the other situation (all the other ellipses) describes imperfect communication. Although the latter contains several scenarios (like edge d or b), the only representation we can give it in $\mathcal{P}$ is a disconnected pair of vertices. Regarding the first situation, we translate it as

a connected subgraph of $\mathcal{P}$ because the blue ellipse encompasses the scenario where A and B receive each other's messages (perfect communication).

Although in $\mathcal{P}$ we cannot differentiate between scenarios like edge d or c, it gives us a simpler way to prove consensus's impossibility. Recall that disconnectivity between A and B represents all possible imperfect communication situations and, thus, it encompasses the escenario of edge a in P. Consequently, the sole existence of disconnectivity ensures the existence of such scenario and the impossibility of consensus. The argument presented above is just for the reader to get an intuitive idea about how the formal proof (presented below) goes.

To generalize our proof of consensus's impossibility we need to analyze what type of communications we have when applying P n times. Every round that passes, processes' views can get to each other or at least one may fail because of the characteristics of the problem presented at the beginning of section 4.1. This leads to the existence of perfect and imperfect communication and, thus, to many connected and disconnected subgraphs just as those from $\mathcal{P}$. To avoid repetition of such graphs, we represent all the final states of processes after n rounds with $\mathcal{P}$.

To conclude and formalize the arguments made above, we need to prove the following:

Claim 1 Let the sequence $p_1, p_2, \ldots, p_n$ be the sequence of processes in a system, whose views are j for each p_j. If $1 \leq i \leq n-1$ messages fail to get to their destination, then consensus cannot be solved.

Proof. One way to start the proof is through a tree, just as we did in section 4.4, and find a sequence of connected leaves that forces the processes to decide on different outputs. Since drawing such a structure for n processes wouldn't be so intuitive, we are only going to use the idea behind it.

Every leaf in a level of a tree represents all possible views of process p_j where i messages have not arrived, that is, we have i knots ($\perp$) in the view of p_j. Furthermore, each view (or leaf) of p_j, will be connected with all possible views of p_{j+1} and so on. Following this logic let's consider one of the possible views of p_1, namely $\{\underbrace{1, \ldots, n-i}_{n-i}, \underbrace{\perp, \ldots, \perp}_{i}\}$. We say that all the values missing in p_1's view are *covered* by $\perp$ and we put those in a set $\mathcal{S}$. Namely, the values that we have so far in $\mathcal{S}$ are $n-i+1, \ldots, n$. (The

reader should note that although all views are ordered in ascending way, it does not affect the proof) For p_2 we are cosidering the view that looks like $\{\underbrace{2,\ldots,n-i-1}_{n-i-1},\underbrace{\perp,\ldots,\perp}_{i},n\}$ and we add all the values covered by $\perp$: $n-i,\ldots,n-1$. For the subsequent processes we consider the following views:

$$views_{p_1} = \{1,\ldots,n-i,\perp,\ldots,\perp\}$$

$$views_{p_2} = \{2,\ldots,n-i-1,\perp,\ldots,\perp,n\}$$

$$views_{p_3} = \{3,\ldots,n-i-2,\perp,\ldots,\perp,n-1,n\}$$

$$\vdots$$

$$views_{p_j} = \{j,\perp,\perp,\ldots,\perp,n-\big(i+j\big),\ldots,n-1,n\}$$

For each process p_j we add all the values covered by $\perp$ to $\mathcal{S}$. Since we have a finite amount of processes, the action of adding covered values to $\mathcal{S}$ will terminate. If $j < n$, we stop adding values to $\mathcal{S}$ and all the views of processes p_k, for $j < k \leq n-1$, are chosen randomly. Otherwise, we just stop at p_n. At the end, $\mathcal{S}$ will have all values that in some view of some p_j were not present, and by the construction of $\mathcal{S}$, it contains all values $1,2,3,\ldots,n$. This last property of $\mathcal{S}$ is the key to prove consensus's impossibility. Assume that p_j decides on k, where k belongs to its view and $0 < k < n$. Since $S = \{1,\ldots,n\}$, $k \in S$ and thus p_j would be deciding on a value that in some other view of other process is not present. This last argument is for an arbitrary k and j, and we conclude that any choice that the processes make, it will be in $\mathcal{S}$.

□

The relation between this claim and our new definitions of perfect communication and graph $\mathcal{P}$ can be deduced inmediately from the views we chose in our claim's proof. Observing these sets carefully, we notice that there is imperfect communication between several processes. For example, between p_n and p_1 there is not a perfect communication scenario because the latter did not receive any message from p_n. Just as this case we can find much more

if we analyze which messages were not received by each process. Recall that imperfect communication translates to disconnectivity between processes in our graph $\mathcal{P}$ and, thus, to the fact that at least i messages (where $1 < i$) are not received by processes. Applying claim 1 and using disconnectivity of vertices in $\mathcal{P}$, we get that consensus is not solvable.

5.2 Disconnectivity in a 3-process system

The protocol complex (graph) that represents the possible communications between processes A, B and C is taken from the tree presented in section 4.3. In this case we represent the protocol complex as separate faces because the complete figure would be 3D and the reader would not be able to see what is happenining in detail.

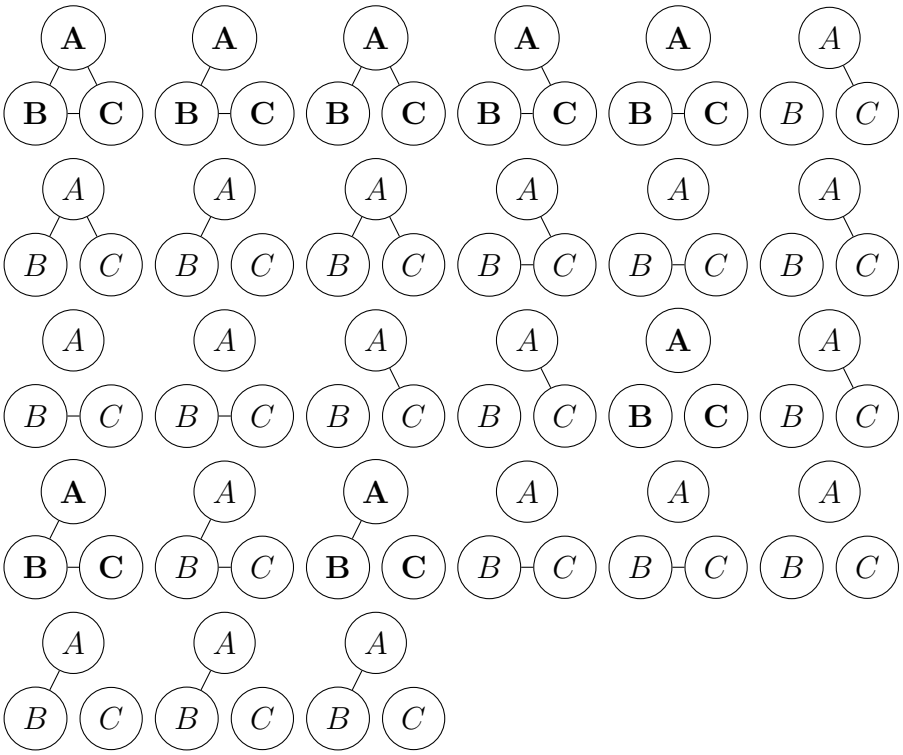


Figure 9: all faces of graph $\mathcal{P}$ for three processes

Each figure is a representation of connected leaves in the tree of section 4.3. However, instead of showing just their inputs we focus on their communication. For example, assume we have a sequence of views: $view_A =$

$\big(u, \perp, w\big), view_B = \big(u, v, \perp\big), view_C = \big(u, \perp, w\big)$. The graph that would represent this, in terms of communication, would be a triangle where A is disconnected to B, because v did not get to A; A would be connected to C since their views contain both u and w; and B would not be connected to C since neither of them has the message of the other. As the reader may have realized, many of the triangles are repeated, but all of those can be represented just by the figures that have bolded vertices. Thus, to represent all the possible communications between A, B, and C we only need 9 triangles. Even though these graphs encompass less information than P, they represent a more compact way of analyzing consensus solvability.

Finally, the proof that consensus is not solvable in a 3-process system is very similar to the one presented for a 2-process system. Since the proof presented in section 5.1 works for n processes , using claim 1 and presenting the same arguments as in the last paragraph of section 5.1 suffices to prove consensus impossibility.

6 Conclusion

This report presented an alternative way to prove consensus impossibility in a 2 and 3 process system. While consensus's impossibility has already been proven in previous papers, the main focus of this report has been the usage of a special characteristic of simplicial maps, namely disconnectivity, that led us to analyze consensus from a different perspective.

The main point of this paper has been the introduction of an intuitively analysis method that gave the readers a better understanding on how consensus and its impossibility work, and, therefore, motivate them to investigate more on new ways to prove these results. The report started with the presentation of the task through a world problem and all the basic topological and graph theory definitions needed to understand the mathematical processes presented in this work.

The core part of the report has been organized in two different sections. The first presents a proof of consensus impossibility using the definitions presented at the beginning and the principles in (5). Secondly, the readers are presented with new definitions and theorems that are essential for the alternative proof of the impossibility of consensus. This way the reader could see the difference between the two approaches and realize how disconnectivity

of graphs gave a different approach while analyzing consensus.

References

[1] A. Dan. *The renaming problem: recent developments and open questions.* Microsoft Research, TU Berlin (2014)

[2] A. Hagit, C. Armando. *A non-topological proof for the impossibility of k-set agreement.* Theoretical Computer Science (2013)

[3] A. Hagit , A. Paz. *Counting-based impossibility proofs for set agreement and renaming.* Journal of parallel and distributed computing (2016)

[4] C. Armando, R. Michel, and R. Sergio. *The renaming problem in shared memory systems: an introduction.* Computer Science Review (2011)

[5] H. Maurice, K. Dmitry, and R. Sergio. *Distributed computing through combinatorial topology.* Morgan Kaufmann, Massachusetts (2014)