A Dynamic Epistemic Logic Analysis of Equality Negation and other Epistemic Covering Tasks[☆,☆☆]

Hans van Ditmarsch[a], Éric Goubault[b], Marijana Lazić[c], Jérémy Ledent[d], Sergio Rajsbaum[e]

[a] *Université de Lorraine, CNRS, LORIA, F-54000 Nancy, France*
[b] *LIX, CNRS, École Polytechnique, Institute Polytechnique de Paris, France*
[c] *Technical University of Munich, Germany*
[d] *Department of Computer and Information Sciences, University of Strathclyde, UK*
[e] *Instituto de Matemáticas, UNAM, Mexico*

Abstract

In this paper we study the solvability of the *equality negation* task in a simple wait-free model where two processes communicate by reading and writing shared variables or exchanging messages. In this task, the two processes start with a private input value in the set $\{0, 1, 2\}$, and after communicating, each one must decide a binary output value, so that the outputs of the processes are the same if and only if the input values of the processes are different. This task is already known to be unsolvable; our goal here is to prove this result using the dynamic epistemic logic (DEL) approach introduced by Goubault, Ledent and Rajsbaum in GandALF 2018. We show that in fact, there is no epistemic logic formula that explains why the task is unsolvable. Furthermore, we observe that this task is a particular case of an *epistemic covering* task. We thus establish a connection between the existing DEL framework and the theory of covering spaces in topology, and prove that the same result holds for any epistemic covering task: no epistemic formula explains the unsolvability.

Keywords: Dynamic Epistemic Logic, distributed computing, equality negation, combinatorial topology, covering spaces.

[☆] A preliminary version of this paper appeared in the Proc. of DaLí workshop <16>.

[☆☆] Eric Goubault and Jérémy Ledent were partially supported by DGA project "Validation of Autonomous Drones and Swarms of Drones" and the academic chair "Complex Systems Engineering" of Ecole Polytechnique-ENSTA-Télécom-Thalès-Dassault-Naval Group-DGA-FX-FDO-Fondation ParisTech.

Sergio Rajsbaum was partially supported by by the UNAM-PAPIIT project IN106520, by the France-Mexico Binational SEP-CONACYT-ANUIES-ECOS grant M12M01, and benefited from the invited professor programme from Ecole Polytechnique.

Marijana Lazić was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 787367 (PaVeS).

Hans van Ditmarsch is also affiliated to IMSc, Chennai, India, as research associate.

Email addresses: hans.van-ditmarsch@loria.fr (Hans van Ditmarsch), Eric.Goubault@polytechnique.edu (Éric Goubault), lazic@in.tum.de (Marijana Lazić), jeremy.ledent@strath.ac.uk (Jérémy Ledent), rajsbaum@im.unam.mx (Sergio Rajsbaum)

1. Introduction

1.1. Background

Functions are the basic objects of study in computability theory. A function is computable if there exists a Turing machine which, given an input of the function domain, returns the corresponding output. If instead of one Turing machine, we have many, and each one gets only one part of the input, and should compute one part of the output, we are in the setting of *distributed computability*, e.g. <1; 28>. The sequential machines are called *processes*,[1] and are allowed to be infinite state machines, to concentrate on the interaction aspects of computability, disregarding sequential computability issues. The notion corresponding to a function is a *task*, roughly, the domain is a set of input vectors, the range is a set of output vectors, and the task specification Δ is an input/output relation between them. An input vector I specifies in its i-th entry the (private) input to the i-th process, and an output vector $O \in \Delta(I)$ states that it is valid for each process i to produce as output the i-th entry of O, whenever the input vector is I. An important example of a task is *consensus*, where each process is given an input from a set of possible input values, and the participating processes have to agree on one of their inputs.

A *distributed computing model* has to specify various details related to how the processes communicate with each other and what type of failures may occur. It turns out that different models may have different power; they can solve different sets of tasks. In this paper we consider the *layered message-passing model* <19>, both because of its relevance to real systems, and because it is the basis to study task computability. This simple, wait-free round-based model where messages can be lost, is described in Section 2.

The theory of distributed computability has been well-developed since the early 1990's <22>, with origins even before <5; 13>, and surveyed in a book <19>. It was discovered that the reason for why a task may or may not be computable is of a topological nature. The input and output sets of vectors are best described as *simplicial complexes*, and a task can be specified by a relation Δ from the input complex $\mathcal{I}$ to the output complex $\mathcal{O}$. The main result is that a task is solvable in the layered message-passing model if and only if there is a certain subdivision of the input complex $\mathcal{I}$ and a certain simplicial map δ to the output complex $\mathcal{O}$ that respects the specification Δ. This is why the layered message-passing model is fundamental; models that can solve more tasks than the layered message-passing model preserve the topology of the input complex less precisely (they introduce "holes").

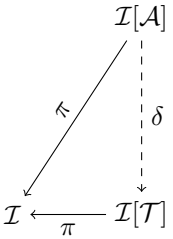
1.2. Motivation

We are interested in understanding distributed computability from the epistemic point of view. What is the knowledge that the processes should gain, to be able to solve a task? This question began to be addressed in <18>, using dynamic epistemic logic (DEL). Here is a brief overview of the approach taken in <18>. A new *simplicial model* for a multi-agent system was introduced, instead of the usual Kripke epistemic $S5$ model based on graphs. Then, the initial knowledge of the processes is represented by a simplicial model, denoted as $\mathcal{I}$, based on the input complex of the task to be solved. The distributed

[1] In different areas processes are called by different names, e.g. agents, processors, threads. In this paper we use indistinctly processes or agents.

computing model is represented by an action model $\mathcal{A}$, and the knowledge at the end of the executions of a protocol is represented by the product update $\mathcal{I}[\mathcal{A}]$, another simplicial model[2]. Remarkably, the task specification is also represented by an action model $\mathcal{T}$, and the product update gives a simplicial model $\mathcal{I}[\mathcal{T}]$ representing the knowledge that should be acquired, by a protocol solving the task. The task $\mathcal{T}$ is *solvable* in $\mathcal{A}$ whenever there exists a morphism $\delta : \mathcal{I}[\mathcal{A}] \to \mathcal{I}[\mathcal{T}]$ such that the diagram of simplicial models below commutes (where π denotes the projection of a product update model onto the initial model, making sure that initial values are preserved).

Thus, to prove that a task is unsolvable, one needs to show that no such δ exists. But one would want to produce a specific formula that concretely represents knowledge that exists in $\mathcal{I}[\mathcal{T}]$, but has not been acquired after running the protocol, i.e. in $\mathcal{I}[\mathcal{A}]$. Indeed, it was shown in <18> that two of the main impossibilities in distributed computability, consensus <13; 27> and approximate agreement <19>, can be expressed by such a formula. However, for other unsolvable tasks (e.g. set agreement), no such formula has been found, despite the fact that no morphism δ exists.



1.3. Contributions

In this paper we show that actually, there are unsolvable tasks for which no such formula exists, namely, the *equality negation* task, defined by Lo and Hadzilacos <26>. This task was introduced as the central idea to prove that the consensus hierarchy <20; 23> is not robust.

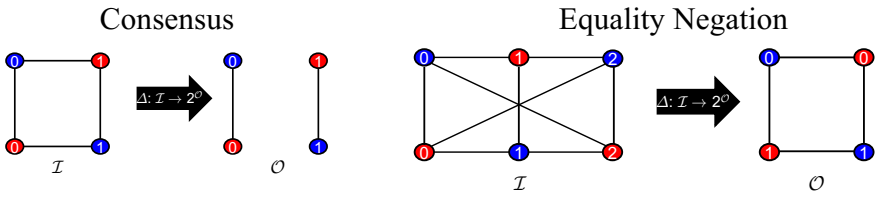
Consider two processes P_0 and P_1, each of which has a private input value, drawn from the set of possible input values $I = \{0, 1, 2\}$. After communicating, each process must irrevocably decide a binary output value, either 0 or 1, so that the outputs of the processes are the same if and only if the input values of the processes are different.

It is interesting to study the solvability of the equality negation task from the epistemic point of view. It is well known that there is no wait-free algorithm for consensus in our model <6; 27>. The same is true for equality negation, as shown in <26; 17>. This is intriguing because there is a formula that shows the impossibility of consensus (essentially reaching common knowledge on input values) <18>, while, as we show here, there is no such formula for equality negation. In more detail, it is well known that consensus is intimately related to connectivity, and hence to common knowledge, while its specification requires deciding values that are in disconnected components of the output complex. The equality negation task is unsolvable for a different reason, since its output complex is connected. Moreover, equality negation is strictly weaker than consensus: consensus can implement equality negation, but not vice versa (the latter is actually a difficult proof in <26>). So it is interesting to understand the difference between the knowledge required to solve each of these tasks.

The binary consensus task and the equality negation task are depicted next. For each one, the input complex $\mathcal{I}$ and output complex $\mathcal{O}$ are represented as graphs (one-dimensional simplicial complexes), with vertices colored red or blue, associated to P_0 and

[2]We use the same symbol $\mathcal{I}$ both for the input complex and the simplicial model, to reduce the amount of notation, because in most of the paper we focus on simplicial models.

P_1 respectively. Each vertex also has a number, an input value in the case of vertices of $\mathcal{I}$ or an output value in the case of vertices of $\mathcal{O}$. An edge $e = \{(P_0, v_0), (P_1, v_1)\} \in \mathcal{I}$ means that a possible initial configuration of the system is when P_0 starts with input value v_0 and P_1 starts with input value v_1. The relation Δ is a carrier map (see Section 4.2). If an edge $e' \in \Delta(e)$, where $e' = \{(P_0, v'_0), (P_1, v'_1)\} \in \mathcal{O}$, it means that according to the task specification, it is valid for P_0 to decide v'_0 and for P_1 to decide v'_1, in any execution starting on input configuration e. For instance, the carrier map Δ specifying consensus sends the input edge $e = 01$ to the set of two output edges $\Delta(e) = \{00, 11\}$, meaning that if the processes start with inputs 0 and 1, they can either both decide 0, or both decide 1 (we write xy as shorthand for the edge $\{(P_0, x), (P_1, y)\}$). Similarly, the carrier map for equality negation sends the edges 00, 11 and 22 to the set $\{01, 10\}$; and the edges 01, 02, 10, 12, 20 and 21 to the set $\{00, 11\}$.



We show that the reason why there is no formula showing the impossibility of equality negation is that the simplicial models $\mathcal{I}$ and $\mathcal{I}[\mathcal{T}]$ are bisimilar. The simplicial model $\mathcal{I}[\mathcal{T}]$ is depicted in Section 5.2, notice that it is different from the equality negation output complex $\mathcal{O}$ above. For this, we work out in detail a bisimulation notion for simplicial models.

Furthermore, we show that the reason why $\mathcal{I}$ and $\mathcal{I}[\mathcal{T}]$ are bisimilar is that the equality negation task is an *epistemic covering* task. The *k-generalized equality negation* task (Section 5.4.1) is similar, but defined in terms of a set of output values of cardinality $2k$. For $k = 1$ it coincides with the equality negation task. For every $k \geq 1$, the task is unsolvable in the wait-free model. However, they are all epistemic covering tasks, and hence the output model is bisimilar to the input model. Thus, no formula can show the impossibility for any of these tasks.

Epistemic covering tasks are a rich family of tasks, with a hierarchical structure. For any input simplicial model, a family of epistemic covering tasks can be defined in terms of covering complexes <14; 30>. None of them are wait-free solvable (except for the trivial ones), yet no formula exists to show that.

Finally, we propose a version of our framework based on an extension of DEL allowing factual change of atomic propositions <8; 4; 7>. In DEL with factual change we can express whether the epistemic covering task is solvable, as this is now represented as reachability of a given goal formula after action model execution. This approach also applies to other tasks involving the setting or resetting of decision variables. Intuitively, the reason why we cannot find a formula witnessing the unsolvability of the task in DEL without factual change is because the update expressivity of DEL without factual change is less than that of DEL with factual change. So, our solution is to enrich the language by adding the facility to change the value of atomic propositions, allowing us to express the required formula in an elegant intuitive way.

1.4. Organization

Section 2 recalls the DEL framework introduced in <18>. We define the layered message-passing model in this context in Section 3. Then, we develop a notion of bisimulation on simplicial models, which is both interesting in its own, but more fundamentally here, will be instrumental in our proof that some tasks cannot be proven unsolvable using DEL. We define the equality negation task in Section 5, and end up this section by generalizing this task to what we call epistemic covering tasks. The important remark is that these tasks are such that they produce product updates in the DEL sense that are similar to topological coverings of the input. In Section 6 we put the pieces together and prove that there is no DEL formula that shows that the equality negation task as well as general covering tasks are not solvable. We end the paper by discussing possible DEL extensions that would give us an explanatory formula for every unsolvable epistemic covering task. Additionally, in Appendix B we prove that the unsolvability of these tasks can be proven in an extension of DEL, with factual change.

2. Preliminaries

2.1. Topological models for epistemic logic

We recap here the new kind of model for epistemic logic based on chromatic simplicial complexes, introduced in <18>. The geometric nature of simplicial complexes allows us to consider higher-dimensional topological properties of our models, and investigate their meaning in terms of knowledge. The idea of using simplicial complexes comes from distributed computability <19; 24>. After describing simplicial models, we explain how to use them in DEL.

Syntax. Let At be a countable set of atomic propositions and Ag a finite set of agents. The language $\mathcal{L}_K$ is generated by the following BNF grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_a\varphi \qquad p \in \mathrm{At},\ a \in \mathrm{Ag}$$

In the following, we work with $n+1$ agents, and write $\mathrm{Ag} = \{a_0, \ldots, a_n\}$.

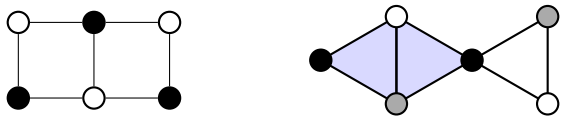
Semantics. The usual semantics for multi-agent epistemic logic is based on Kripke frames. The notion of model that we use here, which is based on simplicial complexes, is merely a reformulation of the usual Kripke models using a different formalism. The benefits of this reformulation is that it makes explicit the topological information of Kripke frames. The precise relationship between the usual Kripke models and our simplicial models is studied thoroughly in <18>.

Definition 1 (Simplicial complex <24>). *A* simplicial complex $\langle V, M\rangle$ *is given by a set* V *of* vertices *and a family* M *of non-empty finite subsets of* V *called* simplices *such that for all* $X \in M$, $Y \subseteq X$ *and* $Y \neq \varnothing$ *implies* $Y \in M$. *We say that* Y *is a* face *of* X.

Usually, the set of vertices is implicit and we simply refer to a simplicial complex as M. We write $\mathcal{V}(M)$ for the set of vertices of M. A vertex $v \in \mathcal{V}(M)$ is identified with the singleton $\{v\} \in M$. Elements of M are called *simplices*, and those which are maximal w.r.t. inclusion are *facets* (or *worlds*), the set of which is denoted by $\mathcal{F}(M)$.

The *dimension* of a simplex $X \in M$ is $|X| - 1$. A simplicial complex M is *pure* if all its facets are of the same dimension n. In this case, we say M is of dimension n. Given a finite set Ag of agents (that we will represent as colors), a *chromatic simplicial complex* $\langle M, \chi \rangle$ consists of a simplicial complex M and a coloring map $\chi : \mathcal{V}(M) \to \mathrm{Ag}$ such that for all $X \in M$, all the vertices of X have distinct colors.

Example 1. The picture below shows two chromatic simplicial complexes. The one on the left is pure of dimension 1: it consists of six vertices and seven edges (i.e., simplices of dimension 1). It is colored using two agents, represented in black and white. The simplicial complex on the right is not pure: it has two facets of dimension 2 (represented as filled triangles), and three facets of dimension 1 (the three rightmost edges forming an empty triangle). It is colored using three agents, black, white and grey.



Simplicial complexes are a prominent tool in algebraic topology because they provide a combinatorial description of various topological spaces. While we introduce them here in full generality, where a model for n agents is of dimension n, the remainder of the paper (from Section 3 onwards) focuses on the two-agent case. Thus, all our models will in fact look like the picture on the left: a (non-directed) graph.

Definition 2 (Simplicial map). *Let C and D be two simplicial complexes. A* simplicial map *$f : C \to D$ maps the vertices of C to vertices of D such that if X is a simplex of C, $f(X)$ is a simplex of D. A* chromatic simplicial map *between two chromatic simplicial complexes is a simplicial map that preserves colors.*

For technical reasons, we restrict to models where all the atomic propositions are saying something about some local value held by one particular agent. All the examples that we are interested in will fit in that framework. Let *Val* be some countable set of values, and $\mathrm{At} = \{p_{a,x} \mid a \in \mathrm{Ag}, x \in \mathit{Val}\}$ be the set of *atomic propositions*. Intuitively, $p_{a,x}$ is true if agent a holds the value x. In all our examples, an agent will always hold exactly one value, but this is not a formal requirement of our framework. We write At_a for the atomic propositions concerning agent a, that is, $\mathrm{At}_a = \{p_{a,x} \mid x \in \mathit{Val}\}$.

Definition 3 (Simplicial model). *A* simplicial model *$\mathcal{M} = \langle C, \chi, \ell \rangle$ consists of a pure chromatic simplicial complex $\langle C, \chi \rangle$ of dimension n, and a labeling $\ell : \mathcal{V}(C) \to \mathscr{P}(\mathrm{At})$ that associates with each vertex $v \in \mathcal{V}(C)$ a set of atomic propositions concerning agent $\chi(v)$, i.e., such that $\ell(v) \subseteq \mathrm{At}_{\chi(v)}$.*

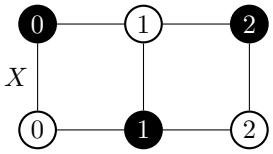
Given a facet $X = \{v_0, \ldots, v_n\} \in C$, we write $\ell(X) = \bigcup_{i=0}^{n} \ell(v_i)$. A *morphism* of simplicial models $f : \mathcal{M} \to \mathcal{M}'$ is a chromatic simplicial map that preserves the labeling: $\ell'(f(v)) = \ell(v)$ (and χ). We denote by $\mathcal{SM}_{\mathrm{Ag,At}}$ the category of simplicial models over the set of agents Ag and atomic propositions At.

Definition 4. *We define the truth of a formula φ in some epistemic state $(\mathcal{M}, X)$ with $\mathcal{M} = \langle C, \chi, \ell \rangle$ a simplicial model, $X \in \mathcal{F}(C)$ a facet of C and $\varphi \in \mathcal{L}_K(\mathrm{Ag}, \mathrm{At})$. The*

satisfaction relation, determining when a formula is true in an epistemic state, is defined as:

$$\begin{array}{lll} \mathcal{M}, X \models p & \textit{if} & p \in \ell(X) \\ \mathcal{M}, X \models \neg\varphi & \textit{if} & \mathcal{M}, X \not\models \varphi \\ \mathcal{M}, X \models \varphi \wedge \psi & \textit{if} & \mathcal{M}, X \models \varphi \textit{ and } \mathcal{M}, X \models \psi \\ \mathcal{M}, X \models K_a\varphi & \textit{if} & \textit{for all } Y \in \mathcal{F}(C), a \in \chi(X \cap Y) \textit{ implies } \mathcal{M}, Y \models \varphi \end{array}$$

Example 2. The picture below shows a simplicial model $\mathcal{M}$ for two agents $\mathrm{Ag} = \{B, W\}$, represented in black and white. The set of values is $Val = \{0, 1, 2\}$, and the labeling ℓ is represented by values inside the vertices. For instance, the value 0 in a white vertex v means that $\ell(v) = \{p_{W,0}\}$, i.e., agent 'W' has value 0 (and nothing else).



One can check that the following statements hold:

$$\mathcal{M}, X \models p_{W,0} \wedge p_{B,0} \qquad \mathcal{M}, X \models K_W(\neg\, p_{B,2}) \qquad \mathcal{M}, X \models K_B(\neg K_W\ p_{B,0})$$

2.2. Equivalence with Kripke models

Simplicial models can be seen as a different formalism to talk about Kripke models. Indeed, as explained in <18>, every simplicial model can be associated with an equivalent Kripke model. Conversely, every Kripke model which is *proper* and *local* can be associated with an equivalent simplicial model. We recall these facts in Theorem 1 below, which is proved in <18>.

A *Kripke frame* $M = \langle S, \sim\rangle$ over a set Ag of agents consists of a set of states S and a family of equivalence relations on S, written $\sim_a$ for every $a \in \mathrm{Ag}$. Two states $u, v \in S$ such that $u \sim_a v$ are said to be *indistinguishable* by a. A Kripke frame is *proper* if any two states can be distinguished by at least one agent. Let $M = \langle S, \sim\rangle$ and $N = \langle T, \sim'\rangle$ be two Kripke frames over the same set of agents Ag. A *morphism* from M to N is a function f from S to T such that for all $u, v \in S$, for all $a \in \mathrm{Ag}$, $u \sim_a v$ implies $f(u) \sim'_a f(v)$.

Definition 5 (Kripke model). *A* Kripke model $M = \langle S, \sim, L\rangle$ *consists of a Kripke frame* $\langle S, \sim\rangle$ *and a function* $L : S \to \mathscr{P}(\mathrm{At})$. *Intuitively,* $L(s)$ *is the set of atomic propositions that are true in the state* s.

A Kripke model is *proper* if the underlying Kripke frame is proper. A Kripke model is *local* if for every agent $a \in \mathrm{Ag}$, $s \sim_a s'$ implies $L(s) \cap \mathrm{At}_a = L(s') \cap \mathrm{At}_a$, i.e., an agent always knows its own values. Let $M = \langle S, \sim, L\rangle$ and $M' = \langle S', \sim', L'\rangle$ be two Kripke models over the same set At. A *morphism of Kripke models* $f : M \to M'$ is a morphism of the underlying Kripke frames such that $L'(f(s)) = L(s)$ for every state s in S. We write $\mathcal{KM}_{\mathrm{Ag,At}}$ for the category of local proper Kripke models.

Theorem 1 (<18>). $\mathcal{SM}_{\mathrm{Ag,At}}$ *and* $\mathcal{KM}_{\mathrm{Ag,At}}$ *are equivalent categories.*

A direct consequence of Theorem 1 is that Definition 4 agrees with the usual definition of truth on Kripke models (see <18>, Prop. 1 for a precise statement of this fact).

2.3. Topological semantics for Dynamic Epistemic Logic

The simplicial models described in Section 2.1 can also be used to model Dynamic Epistemic Logic (DEL). DEL is the study of various epistemic logics extended with model-changing operators <3; 9>. One such logic is the one based on *action models* <2>, which are relational structures that can be used to describe a variety of communication actions.

Syntax. We extend the syntax of epistemic logic with one more construction:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_a\varphi \mid [\alpha]\varphi \qquad p \in \mathrm{At},\ a \in \mathrm{Ag}$$

Intuitively, the formula $[\alpha]\varphi$ means that φ is true after some *action* α has occurred. An action can be thought of as an announcement made by the environment, which is not necessarily public, in the sense that not all agents receive these announcements. The semantics of this new operator should be understood as follows (under assumption that α can be executed in the world X):

$$\mathcal{M}, X \models [\alpha]\varphi \quad \text{if} \quad \mathcal{M}[\alpha], X[\alpha] \models \varphi$$

i.e., the formula $[\alpha]\varphi$ is true in some world X of $\mathcal{M}$ whenever, on condition that the action α can be executed in world X, φ is true in world $X[\alpha]$ of some new model $\mathcal{M}[\alpha]$, where the knowledge of each agent has been modified according to the action α. (In particular, the formula $[\alpha]\varphi$ is vacuously true in all states X that do not satisfy the precondition of α.)

A simplicial complex version of DEL. To define formally what an action is, we first need to introduce the notion of *action model*. An action model describes all the possible actions that might happen, as well as how they affect the different agents.

Definition 6. *An* action model *is a structure $\mathcal{A} = \langle T, \sim, \mathsf{pre}\rangle$, where T is a domain of* actions *such that for each $a \in \mathrm{Ag}$, $\sim_a$ is an equivalence relation on T, and $\mathsf{pre} : T \to \mathcal{L}_{\mathcal{K}}$ is a function that assigns a* precondition *formula $\mathsf{pre}(t)$ to each $t \in T$*

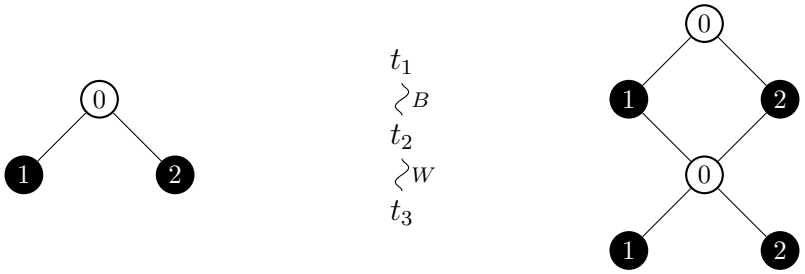
An *action* α is defined as a pair $(\mathcal{A}, t)$ where $t \in T$. Additionally, in order to keep the action language defined above well-defined, action models must have a *finite* domain T. An action model is *proper* if for any two different actions $t, t' \in T$, there is an agent $a \in \mathrm{Ag}$ who can distinguish between them, i.e., $t \not\sim_a t'$.

Definition 7 (Product update model). *Given a simplicial model $\mathcal{M} = \langle C, \chi, \ell\rangle$ and an action model $\mathcal{A} = \langle T, \sim, \mathsf{pre}\rangle$, we define the* product update simplicial model $\mathcal{M}[\mathcal{A}] = \langle C[\mathcal{A}], \chi[\mathcal{A}], \ell[\mathcal{A}]\rangle$ *as follows.*

- *The vertices of $C[\mathcal{A}]$ are pairs (v, E) where $v \in \mathcal{V}(C)$ is a vertex of C; E is an equivalence class of $\sim_{\chi(v)}$; and v belongs to some facet $X \in C$ such that there exists $t \in E$ such that $\mathcal{M}, X \models \mathsf{pre}(t)$. Such a vertex keeps the color and labeling of its first component: $\chi[\mathcal{A}](v, E) = \chi(v)$ and $\ell[\mathcal{A}](v, E) = \ell(v)$.*
- *For each pair $(X, t) \in \mathcal{F}(C) \times T$ such that $\mathcal{M}, X \models \mathsf{pre}(t)$, there is one facet of $C[\mathcal{A}]$ defined by $\{(v, [t]_{\chi(v)}) \mid v \in X\}$, where $[t]_{\chi(v)}$ denotes the equivalence class of t for $\sim_{\chi(v)}$. All the other simplices of $C[\mathcal{A}]$ are the subsets of those facets.*

Intuitively, the facets of $C[\mathcal{A}]$ correspond to pairs (X, t) where $X \in \mathcal{F}(C)$ is a world of $\mathcal{M}$ and $t \in T$ is an action of $\mathcal{A}$, such that $\mathcal{M}, X \models \mathsf{pre}(t)$. Two such facets (X, t) and (Y, t') share the same a-colored vertex whenever $a \in \chi(X \cap Y)$ and $t \sim_a t'$.

Example 3. The pictures below represent a simplicial model $\mathcal{M}$ (left), an action model $\mathcal{A}$ (middle) and their product-update model $\mathcal{M}[\mathcal{A}]$ (right). Here, the action model consists of three actions, $T = \{t_1, t_2, t_3\}$, such that $t_1 \sim_B t_2$ and $t_2 \sim_W t_3$. The precondition relations are all true, i.e. $\mathsf{pre}(t) = \mathit{true}$ for all $t \in T$.



Notice that $\mathcal{M}[\mathcal{A}]$ consists of six worlds, one for each pair (X, t) of a world X of $\mathcal{M}$ together with an action t of $\mathcal{A}$. If we used a similar action model except with non-trivial preconditions, the resulting product-update model would be similar to $\mathcal{M}[\mathcal{A}]$, but we would only keep the edges (X, t) such that $\mathcal{M}, X \models \mathsf{pre}(t)$.

We can finally state the formal semantics of the formula $[\alpha]\varphi$:

$$\mathcal{M}, X \models [\mathcal{A}, t]\varphi \qquad \text{if} \qquad \mathcal{M}, X \models \mathsf{pre}(t) \implies \mathcal{M}[\mathcal{A}], (X, t) \models \varphi$$

3. Layered message-passing

In this section, we describe the computational model that we are interested in. We first explain distributed computing intuition, and then formalize it using action models.

We start with an overview of the *layered message-passing model* for two agents, or *processes* as they are called in distributed computing. More details about this model can be found in <19>. This model is known to be equivalent to the well-studied *read/write wait-free model*, in the sense that it solves the same set of tasks. When there are only two processes involved in the computation, which is what we want to study in this article, the layered message-passing model is easier to understand. Here, we formalize this model as an action model; a more usual presentation can be found in Appendix A.2, along with a proof of equivalence between the two.

3.1. The layered message-passing model.

Let the processes be B, W, to draw them in the pictures with colors black and white. In the *layered message-passing* model, computation is synchronous: B and W take steps at the same time. We will call each such step a *layer*. In each layer, B and W both send a message to each other, where at most one message may fail to arrive, implying that either one or two messages will be received. This is a *full information* model, in the sense that each time a process sends a message, the message consists of its local state (i.e., all

the information currently known to the process), and each time it receives a message, it appends it to its own local state (remembers everything). A protocol is defined by the number N of layers the processes execute. Then, each process should produce an output value based on its state at the end of the last layer. A decision function δ specifies the output value of each process at the end of the last layer.

Given an initial state, an execution can be specified by a sequence of N symbols over the alphabet $\{\bot, B, W\}$, meaning that, if the i-th symbol in the sequence is $\bot$ then in the i-th layer both messages arrived, and if the i-th symbol is B (resp. W) then only B's message failed to arrive (resp. W's) in the i-th layer. As an example, $\bot BW$ corresponds to an execution in which both processes have received each other's message at layer one, then B received the message from W but W did not receive the message from B at layer two, and finally at layer three, W received the message from B but B did not receive the message from W.

For example, there are three 1-layer executions, namely $\bot$, B and W, but from the point of view of process B, there are two distinguished cases: (i) either it did not receive a message, in which case it knows for sure that the execution that occurred was W, or (ii) it did receive a message from W, in which case the execution could have been either B or $\bot$. Thus, for the black process executions B and $\bot$ are indistinguishable.

3.2. The layered message-passing model as an action model.

Consider the situation where the agents $\mathrm{Ag} = \{B, W\}$ each start in an initial global state, defined by input values given to each agent. The values are local, in the sense that each agent knows its own initial value, but not necessarily the values given to other agents. The agents communicate to each other via the layered message-passing model described above. The layered message-passing action model described next is equivalent to the immediate snapshot action model of <18> in the case of two processes.

Let V^{in} be an arbitrary domain of *input values*, and take the set $\mathrm{At} = \{\mathsf{input}_a^x \mid a \in \mathrm{Ag}, x \in V^{in}\}$ of atomic propositions. Consider a simplicial model $\mathcal{I} = \langle I, \chi, \ell \rangle$ called the *input simplicial model*. Moreover, we assume that for each vertex $v \in \mathcal{V}(I)$, corresponding to some agent $a = \chi(v)$, the labeling $\ell(v) \subseteq \mathrm{At}_a$ is a singleton, assigning to the agent a its private input value. Thus, each facet $X \in \mathcal{F}(I)$ represents a possible initial configuration where each agent has been given an input value.

The action model $\mathcal{MP}_N = \langle T, \sim, \mathsf{pre} \rangle$ corresponding to N layers is defined as follows. Let L_N be the set of all sequences of N symbols over the alphabet $\{\bot, B, W\}$ (for $N = 0$, we write $\varepsilon \in L_N$ for the empty sequence). Then, we choose $T = L_N \times \mathcal{F}(I)$. An action (α, X), where $\alpha \in L_N$ and $X \in \mathcal{F}(I)$ represents a possible execution starting in the initial configuration X. We write X_a for the input value assigned to agent a in the input simplex X. Then, $\mathsf{pre} : T \to \mathcal{L}_\mathcal{K}$ assigns to each $(\alpha, X) \in T$ a precondition formula $\mathsf{pre}(\alpha, X)$ which holds exactly in X (formally, we take $\mathsf{pre}(\alpha, X) = \bigwedge_{a \in \mathrm{Ag}} \mathsf{input}_a^{X_a}$). To define the indistinguishability relation $\sim_a$, we proceed by induction on N. For $N = 0$, we define $(\varepsilon, X) \sim_a (\varepsilon, Y)$ when $X_a = Y_a$, since process a only sees its own local state. Now assume that the indistinguishability relation of $\mathcal{MP}_N$ has been defined, we define $\sim_a$ on $\mathcal{MP}_{N+1}$ as follows. Let $\alpha, \beta \in L_N$ and $p, q \in \{\bot, B, W\}$. We define $(\alpha{\cdot}p, X) \sim_B (\beta{\cdot}q, Y)$ if either:

(*i*) $p = q = W$ and $(\alpha, X) \sim_B (\beta, Y)$, or

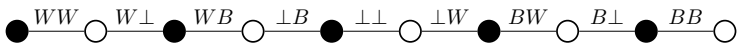
(*ii*) $p, q \in \{\bot, B\}$ and $X = Y$ and $\alpha = \beta$,

and similarly for $\sim_W$, with the role of B and W reversed. Intuitively, either (i) no message was received, and the uncertainty from the previous layers remain; or (ii) a message was received, and the process B can see the whole history, except that it does not know whether the last layer was B or $\bot$.

To see what the effect of this action model is, let us start with an input model $\mathcal{I}$ with only one input configuration X (input values have been omitted).

After one layer of the message passing model, we get the model $\mathcal{I}[\mathcal{MP}_1]$ depicted below. Since $\mathcal{I}$ has only one facet, the action model $\mathcal{MP}_1$ consists of three actions $\{B, W, \bot\}$, and thus $\mathcal{I}[\mathcal{MP}_1]$ has the three corresponding worlds:

After a second layer, we get $\mathcal{I}[\mathcal{MP}_2]$:



The remarkable property of this action model is that it preserves the topology of the input model. This is a well-known fact in distributed computing <19>, reformulated here in terms of DEL.

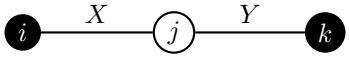
Theorem 2. *Let $\mathcal{I} = \langle I, \chi, \ell \rangle$ be an input model, and $\mathcal{MP}_N = \langle T, \sim, \mathsf{pre} \rangle$ be the N-layer action model. Then, the product update simplicial model $\mathcal{I}[\mathcal{MP}_N]$ is a subdivision of $\mathcal{I}$, where each edge is subdivided into 3^N edges.*

3.3. An alternative presentation of $\mathcal{MP}_N$

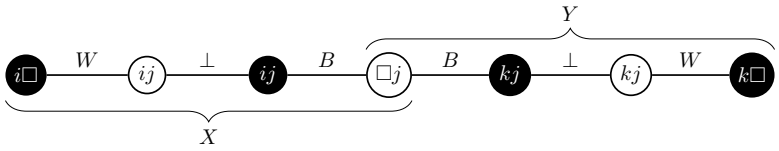
The action model defined in Section 3.2 is unusual in distributed computing. Indeed, it is not standard to think of distributed protocols in terms of whether two executions are *indistinguishable* by some process. Instead, it is more common to think in terms of *local states* and *views* of each process: what information is acquired by the processes when communication occurs? In this section, we reconcile these two points of view.

Recall that, in a given initial state, an execution can be specified by a sequence of N symbols over the alphabet $\{\bot, B, W\}$, meaning that, in the i-th layer, if the i-th symbol is $\bot$ then both messages arrived, and if it is B then only B's message failed to arrive (similarly for W).

To illustrate the notion of *view*, let us take a look at an input model $\mathcal{I}$ depicted below. It consists of two edges, $X = \{(B, i), (W, j)\}$ and $Y = \{(B, k), (W, j)\}$, where i, j, k are input values, or more generally, local states.



After one layer of communication, the situation can be pictured is as follows:



As expected, the edges X and Y have been subdivided into three edges each. The symbols written inside the nodes are called the *views*. Here, each view consists of two symbols st, indicating that the message 's' was received from B, and the message 't' was received from W. The symbol $\square$ indicates that no message was received. For example, starting with inputs i and j, if the execution 'B' happens (meaning that the message sent by process B was lost), then the white process receives no message from B and has the view $\square j$, whereas the black process receives the message from W and has the view ij.

In the N-layer message passing protocol, we iterate the previous construction N times. At each subsequent layer, each process uses its view from the previous layer as its local state for the next layer. For example, starting with input values i and j, as we saw previously, the execution 'B' in the first layer gives the two views $\{(B, ij), (W, \square j)\}$. Then the inputs for the the second layer are $i' := ij$ and $j' := \square j$, and the 2-layer execution 'BB' gives the views $\{(B, i'j'), (W, \square j')\} = \{(B, ij\ \ \square j), (W, \square\ \ \square j)\}$.

We are now ready to define formally the notion of *view* of a process in an N-layer execution, starting from the input edge $X = \{(B, i), (W, j)\}$. We proceed by induction on N. For $N = 0$, we have $\mathsf{view}_B(\varepsilon, X) = i$ and $\mathsf{view}_W(\varepsilon, X) = j$, where ε denotes the empty execution. Let αc be an N-layer execution, where α is an $(N-1)$-layer execution and $c \in \{B, W, \bot\}$. Assume that, after the execution α occurred, the view of B is $V_B := \mathsf{view}_B(\alpha, X)$, and the view of W is $V_W := \mathsf{view}_W(\alpha, X)$. Then:

- if $c = B$, define $\mathsf{view}_B(\alpha c, X) = (V_B, V_W)$ and $\mathsf{view}_W(\alpha c, X) = (\square, V_W)$.
- if $c = W$, define $\mathsf{view}_B(\alpha c, X) = (V_B, \square)$ and $\mathsf{view}_W(\alpha c, X) = (V_B, V_W)$.
- if $c = \bot$, define $\mathsf{view}_B(\alpha c, X) = (V_B, V_W)$ and $\mathsf{view}_W(\alpha c, X) = (V_B, V_W)$.

Using the notion of view, we can give an alternative definition of the layered message-passing action model of Section 3.2. Namely, we keep the same set of actions T (where actions are pairs (α, X) of an execution α and an input configuration X); and we decide that two actions are indistinguishable whenever they produce the same view. The fact that this alternative definition results in the same action model $\mathcal{MP}_N$ as in Section 3.2 is proved in Proposition 1 below.

Before we prove the equivalence between the two definitions, we first show the following simple fact: different choices for the pair (α, X) will necessarily give rise to distinct views for at least one of the processes.

Lemma 3. *For all N, α, β, X, Y, if $\mathsf{view}_B(\alpha, X) = \mathsf{view}_B(\beta, Y)$ and $\mathsf{view}_W(\alpha, X) = \mathsf{view}_W(\beta, Y)$, then $\alpha = \beta$ and $X = Y$.*

Proof. By induction on N. □

Proposition 1. *For any number of layers N, for all $\alpha, \beta \in L_N$ and $X, Y \in \mathcal{F}(\mathcal{I})$,*

$$\mathsf{view}_B(\alpha, X) = \mathsf{view}_B(\beta, Y) \iff (\alpha, X) \sim_B (\beta, Y) \tag{1}$$

and similarly for view_W *and* $\sim_W$*. Here,* $\sim_B$ *is the indistinguishability relation defined in Section 3.2.*

PROOF. We prove it by induction on N.

- For $N = 0$, this holds from the definitions.
- Let αc and βd be executions of length $N+1$, with $c, d \in \{B, W, \bot\}$.

  ($\Leftarrow$): Assume $(\alpha c, X) \sim_B (\beta d, Y)$. By definition, there are two possible cases.

  - $c = d = W$ and $(\alpha, X) \sim_B (\beta, Y)$. We get $\mathsf{view}_B(\alpha, X) = \mathsf{view}_B(\beta, Y)$ by induction hypothesis. Let us write V_B for this view. Then, by definition, we get $\mathsf{view}_B(\alpha c, X) = (V_B, \square)$ and $\mathsf{view}_B(\beta d, Y) = (V_B, \square)$, which are equal.
  - $c, d \in \{\bot, B\}$ and $X = Y$ and $\alpha = \beta$. We write $V_B = \mathsf{view}_B(\alpha, X) = \mathsf{view}_B(\beta, Y)$, and $V_W = \mathsf{view}_W(\alpha, X) = \mathsf{view}_W(\beta, Y)$. Then by definition $\mathsf{view}_B(\alpha c, X) = (V_B, V_W) = \mathsf{view}_B(\beta d, Y)$.

  ($\Rightarrow$): Assume $\mathsf{view}_B(\alpha c, X) = \mathsf{view}_B(\beta d, Y)$. Since a view can never be just '$\square$' (it is either an input value, or a pair of values), we can never have $V_W = \square$. Thus the equality can be true only in one of the two following cases.

  - Either $c = d = W$. In which case, we get $\mathsf{view}_B(\alpha, X) = \mathsf{view}_B(\beta, Y)$ by identifying the first components of the views, and by induction hypothesis, $(\alpha, X) \sim_B (\beta, Y)$. Therefore, $(\alpha c, X) \sim_B (\beta d, Y)$.
  - Or $c, d \in \{\bot, B\}$. By identifying the components of the pair, $\mathsf{view}_B(\alpha, X) = \mathsf{view}_B(\beta, Y)$ and $\mathsf{view}_W(\alpha, X) = \mathsf{view}_W(\beta, Y)$. By Lemma 3 we deduce $\alpha = \beta$ and $X = Y$, and thus we obtain $(\alpha c, X) \sim_B (\beta d, Y)$.

The proof for the correspondence between view_W and $\sim_W$ is similar, with the roles of B and W reversed. □

4. Bisimulation for simplicial models

The notion of bisimulation for simplicial models is fundamental in itself, and indeed will play a crucial role in our study. We start with a definition directly derived from the usual definition for Kripke models. Then we reformulate it into the language of combinatorial topology, using carrier maps.

4.1. Bisimulation between simplicial models

We start with a natural definition of bisimulation between simplicial models, derived from the usual definition for Kripke models. Variants of this definition are discussed in <11>.

Definition 8 (Bisimulation). *Let* $\mathcal{M} = \langle M, \chi, \ell \rangle$ *and* $\mathcal{M}' = \langle M', \chi', \ell' \rangle$ *be two simplicial models. A non-empty relation* $R \subseteq \mathcal{F}(M) \times \mathcal{F}(M')$ *is a* bisimulation *between* $\mathcal{M}$ *and* $\mathcal{M}'$ *if the following conditions hold:*

• **atoms:** *If* $X \; R \; X'$ *then* $\ell(X) = \ell'(X')$.

- **forth:** *For all* $a \in \mathrm{Ag}$*, if* $X\ R\ X'$ *and there exists* $Y \in \mathcal{F}(M)$ *such that* $a \in \chi(X \cap Y)$*, then there exists* $Y' \in \mathcal{F}(M')$ *such that* $Y\ R\ Y'$ *and* $a \in \chi'(X' \cap Y')$*.*
- **back:** *For all* $a \in \mathrm{Ag}$*, if* $X\ R\ X'$ *and there exists* $Y' \in \mathcal{F}(M')$ *such that* $a \in \chi'(X' \cap Y')$*, then there exists* $Y \in \mathcal{F}(M)$ *such that* $Y\ R\ Y'$ *and* $a \in \chi(X \cap Y)$*.*

If there is a bisimulation between $\mathcal{M}$ and $\mathcal{M}'$, then we write $\mathcal{M} \leftrightarrows \mathcal{M}'$. When R is a bisimulation and $X\ R\ X'$, we say that X and X' are *bisimilar*. A *total bisimulation* R is a bisimulation such that for all $X \in \mathcal{F}(M)$ there is a $X' \in \mathcal{F}(M')$ with XRX' and for all $X' \in \mathcal{F}(M')$ there is a $X \in \mathcal{F}(M)$ with XRX' (we also refer similarly to a *total relation*). A relation R is a *simulation* of $\mathcal{M}$ by $\mathcal{M}'$ if it satisfies the **atoms** and the **forth** conditions.

The next lemma states that two bisimilar worlds satisfy exactly the same formulae. This is a well-known fact in the context of Kripke models. The same result holds for bisimulations between simplicial models, since Definition 8 is merely a translation of the usual notion in the equivalent formalism of simplicial models.

Lemma 4. *Let* R *be a bisimulation between* $\mathcal{M}$ *and* $\mathcal{M}'$*. Then for all facets* X, X' *such that* $X\ R\ X'$*, and for every epistemic logic formula* φ*,*

$$\mathcal{M}, X \models \varphi \qquad \textit{iff} \qquad \mathcal{M}', X' \models \varphi. \tag{2}$$

Proof (sketch). This could be proven using the categorical equivalence of Theorem 1 and the fact that bisimulations as we define them correspond, through this equivalence, to classical bisimulations between Kripke models. We do not develop this approach here because formulating it properly would require us to look into the proof of Theorem 1. Instead, a straightforward proof by induction on the structure of the formula φ also yields the result.

4.2. Bisimulation in terms of carrier maps

The definition of bisimulation between simplicial models can be restated and studied through the combinatorial topology perspective. We present a formulation in terms of carrier maps. The concept of carrier map is especially important in the topological perspective of distributed computing surveyed in <19>, and in general in algebraic topology, especially when it is acyclic e.g. <29>. We describe this formulation only for the case of two agents.

Consider $\mathcal{M} = \langle M, \chi, \ell \rangle$ and $\mathcal{M}' = \langle M', \chi', \ell' \rangle$, two simplicial models for our two agents, $\mathrm{Ag} = \{B, W\}$. We will thus be thinking about M, M' as graphs, and use the corresponding terminology, such as vertices, edges, instead of simplexes.

A *carrier map* Φ from $\mathcal{M}$ to $\mathcal{M}'$ sends each vertex of M to a non-empty set of vertices of M', and each edge of M to a non-empty set of edges of M', together with their respective vertices. Thus, for each vertex v of M, $\Phi(v)$ is a subgraph of M' of dimension 0 and and for each edge e, $\Phi(e)$ is a subgraph of M' of dimension 1. Moreover, such a carrier map Φ is usually required to be *monotonic*: for every vertex v and edge e in M, $v \in e$ implies $\Phi(v) \subset \Phi(e)$. Thus in particular, we have $\Phi(e_1 \cap e_2) \subseteq \Phi(e_1) \cap \Phi(e_2)$. For the interpretation of carrier maps as bisimulations, we naturally assume that Φ is chromatic and label-preserving: for every vertex $v' \in \Phi(v)$, $\chi(v) = \chi'(v')$ and $\ell(v) = \ell'(v')$.

Consider an edge $e = \{u, v\}$ in M, with $\chi(u) = B$, $\chi(v) = W$. Recall that $\Phi(e)$ is a subgraph of M', which includes the vertices in $\Phi(u)$ and in $\Phi(v)$. We say that $\Phi(u) \cup \Phi(v)$ is the *boundary* of the graph $\Phi(e)$, and in particular, $\Phi(u)$ is the B-boundary, while $\Phi(v)$ is the W-boundary.

In distributed computing, $\Phi(e)$ may include many edges that do not intersect the boundary of $\Phi(e)$. For instance, a common case is to map an edge e to a path $\Phi(e)$ whose two endpoints consist of the boundary. More generally, carrier maps appear in subdivisions. For instance, in Theorem 2, the product update simplicial model $\mathcal{I}[\mathcal{MP}_N]$ is a subdivision of $\mathcal{I}$, where each edge is subdivided into 3^N edges. There is a carrier map from $\mathcal{I}$ to $\mathcal{I}[\mathcal{MP}_N]$, relating a vertex in $\mathcal{I}$ to a vertex in $\mathcal{I}[\mathcal{MP}_N]$ where the process never receives messages from the other process, and relating an edge $e \in \mathcal{I}$ to the subgraph of all edges in $\mathcal{I}[\mathcal{MP}_N]$, defined by executions starting in e.

A carrier map is a generalization of a simplicial map, and in this sense, a simplicial map is the simplest carrier map. If Φ is a simplicial map, then $\Phi(v)$ consists of only one vertex, for each v, and $\Phi(e)$ consists of only one edge for each e, determined by the map on the vertices.

We have the following simple lemma, which we would like to generalize to carrier maps.

Lemma 5. *If $R \subseteq \mathcal{F}(M) \times \mathcal{F}(M')$ is the relation induced from a chromatic label-preserving simplicial map from $\mathcal{M}$ to $\mathcal{M}'$, then R is a simulation of $\mathcal{M}$ by $\mathcal{M}'$.*

Recall (Theorem 2) that if $\mathcal{I} = \langle I, \chi, \ell \rangle$ is an input model, and $\mathcal{MP}_N = \langle T, \sim, \mathsf{pre} \rangle$ is the N-layer action model, then the product update simplicial model $\mathcal{I}[\mathcal{MP}_N]$ is a subdivision of $\mathcal{I}$. The first projection map $\pi : \mathcal{I}[\mathcal{MP}_N] \to \mathcal{I}$ that sends a vertex v to its carrier (namely, the vertex from which it came from in the subdivision, with the same process name), is a chromatic, label-preserving simplicial map. As a corollary of Lemma 5, we have the following claim.

Lemma 6. *The product-update simplicial model $\mathcal{I}[\mathcal{MP}_N]$ simulates the input model $\mathcal{I}$.*

As we shall see, bisimulations are induced by carrier maps satisfying a very simple *unity* property: every vertex of $\Phi(e)$ must belong to the boundary of $\Phi(e)$. In more detail, if $e' = \{u', v'\} \in \Phi(e)$ for $e = \{u, v\}$, where $\chi(u) = \chi'(u') = B$, $\chi(v) = \chi'(v') = W$, then $v' \in \Phi(v)$ and $u' \in \Phi(u)$.

Consider a total relation $R \subseteq \mathcal{F}(M) \times \mathcal{F}(M')$. In our case, the facets are edges of the corresponding graphs. Let Φ be the unity chromatic carrier map from $\mathcal{M}$ to $\mathcal{M}'$ induced by R by letting $\Phi(e)$ be the subgraph defined by all edges e' such that $e \; R \; e'$, and letting $\Phi(v)$ be the set of vertices in $\Phi(e)$ with the same color, for each e containing v. In the other direction, Φ' from $\mathcal{M}'$ to $\mathcal{M}$ is defined similarly. If Φ from $\mathcal{M}$ to $\mathcal{M}'$ and Φ' from $\mathcal{M}'$ to $\mathcal{M}$ are two carrier maps induced by the same relation R, we say that they are *matching* carrier maps. Notice that if Φ and Φ' are matching carrier maps induced by R, then R must be total (recall that a carrier map sends each vertex and each edge to a non-empty set of vertices and edges, resp.). Conversely, given a carrier map Φ from $\mathcal{M}$ to $\mathcal{M}'$, there is an induced relation $R \subseteq \mathcal{F}(M) \times \mathcal{F}(M')$, obtained by letting $e \; R \; e'$ whenever $e' \in \Phi(e)$.

The following holds for any number of processes, but for simplicity we prove here only the case of two processes, which is what we will need henceforth.

Theorem 7. *A total relation $R \subseteq \mathcal{F}(M) \times \mathcal{F}(M')$ is a bisimulation between $\mathcal{M}$ and $\mathcal{M}'$ if and only if the associated matching unity chromatic carrier maps Φ and Φ' are also monotonic and label-preserving.*

Proof. For any total relation R, the induced carrier maps Φ, Φ', are clearly matching, unity, chromatic. If we assume moreover that R is a bisimulation, then Φ and Φ' are label-preserving thanks to the **atoms** condition. To show that Φ is monotonic, let $v \in e$, for some edge e of M. We need to show that any vertex $v' \in \Phi(v)$ belongs to $\Phi(e)$. By definition of Φ, v' was added to $\Phi(v)$ because there exists an edge e_1 containing v, such that $e_1 \; R \; e_1'$ and e_1' contains v'. By the **forth** requirement of Definition 8, there exists an e' in the edges of M' such that $e \; R \; e'$ and e' contains v'. Thus, by definition of Φ, $v' \in \Phi(e)$. The proof that Φ' is monotonic is similar, but using the **back** requirement Definition 8.

Conversely, let R be the relation inducing Φ, Φ', two monotonic, unity, chromatic and label-preserving carrier maps. We prove that R satisfies the **forth** requirement of Definition 8, using Φ. The proof that it satisfies the **back** requirement of Definition 8 is similar, but using Φ'. Consider $e \; R \; e'$, and any edge e_1, with $e \cap e_1 = v$. Let v' be the vertex of e' with $\chi(v) = \chi'(v')$. Then, $v' \in \Phi(v)$, because Φ is induced by R and Φ is unity. By monotonicity of Φ, $\Phi(v) \subseteq \Phi(e_1)$, which means that v' must be in some edge e_1' of $\Phi(e_1)$. Thus, $e_1 \; R \; e_1'$ with $e_1 \cap e_1' = v'$. □

5. Equality negation and epistemic covering tasks

We first formalize the notion of *task* from distributed computing using the action model formalism in Section 5.1. We define one important example of such a task, the *equality negation* task, in Section 5.2. Then, in Section 5.4 we observe that, from our perspective, it is nothing more than an instance of a large class of epistemic tasks called *covering tasks*. In order to define this family we recall some basic notions about covering spaces in Section 5.3.

5.1. Tasks as action models

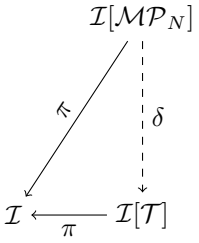
In distributed computing, a task specifies the output values that the agents are allowed to decide, depending on the input state that they started in. Let us fix an input simplicial model $\mathcal{I} = \langle I, \chi, \ell \rangle$, and write Ag for the set of agents.

Definition 9 (Task). *A* task *for $\mathcal{I}$ is an action model $\mathcal{T} = \langle T, \sim, \mathsf{pre} \rangle$, where each action $t \in T$ consists of a function $t : \mathrm{Ag} \to V^{out}$, where V^{out} is an arbitrary set of* output values*, and where the indistinguishability relation is defined as $t \sim_a t'$ when $t(a) = t'(a)$.*

Such an action $t : \mathrm{Ag} \to V^{out}$ is interpreted as an assignment of an output value $t(a)$ for each agent a. Each action t has a precondition that is true in one or more facets of $\mathcal{I}$, interpreted as "if the input configuration is a facet in which $\mathsf{pre}(t)$ holds, and every agent $a \in \mathrm{Ag}$ decides the value $t(a)$, then this is a valid execution of the task". The indistinguishability relation states that an agent is only aware of its own decision.

Definition 10 (Solvability). *The task $\mathcal{T}$ is solvable in the N-layer message-passing model if there exists a morphism $\delta : \mathcal{I}[\mathcal{MP}_N] \to \mathcal{I}[\mathcal{T}]$ such that $\pi \circ \delta = \pi$, i.e., the diagram of simplicial complexes below commutes.*

In the above definition, the two morphisms denoted by $\pi : \mathcal{I}[\mathcal{MP}_N] \to \mathcal{I}$ and $\pi : \mathcal{I}[\mathcal{T}] \to \mathcal{I}$ are simply projections on the first component. The intuition behind this definition is the following. A facet X in $\mathcal{I}[\mathcal{MP}_N]$ corresponds to a pair (i, act), where $i \in \mathcal{F}(\mathcal{I})$ represents input value assignments to all agents, and $act \in \mathcal{MP}_N$ represents an action, codifying the communication exchanges that took place. The morphism δ takes X to a facet $\delta(X) = (i, t)$ of $\mathcal{I}[\mathcal{T}]$, where $t \in \mathcal{T}$ is an assignment of decision values that the agents will choose in the input configuration X.



Moreover, $\mathsf{pre}(t)$ holds in i, meaning that t corresponds to valid decision values for input i. The commutativity of the diagram expresses the fact that both X and $\delta(X)$ correspond to the same input assignment i. Now, consider a single vertex $v \in X$ with $\chi(v) = a \in \mathrm{Ag}$. Then, agent a decides its value solely according to its knowledge in $\mathcal{I}[\mathcal{MP}_N]$: if another facet X' contains v, then $\delta(v) \in \delta(X) \cap \delta(X')$, meaning that a has to decide the same value in both situations.

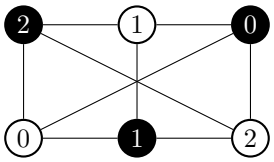
5.2. Equality Negation

The equality negation task was introduced in <26>, and further studied in <17>. In this section, we will be interested only in the case of two processes. Each process starts with an input value in the set $\{0, 1, 2\}$, and has to irrevocably decide on an output value in the set $\{0, 1\}$, such that the decisions of the two processes are the same if and only if their input values are different. In <26>, it has been proved that the equality negation task is unsolvable for two processes in a wait-free model using only read/write registers. (Recall that the layered message-passing model is equivalent to shared read/write registers.) We reproduce this proof, as well as a more topological proof, in Appendix A.3. In Section 6 we will analyze this task using our DEL framework. To this end, let us specify equality negation as an action model.

DEL definition of the task. Let $\mathrm{Ag} = \{B, W\}$ be the two agents (or processes). In the pictures, process B will be associated with black vertices, and process W with white vertices. The atomic propositions are of the form input^i_p, for $p \in \mathrm{Ag}$ and $i \in \{0, 1, 2\}$, meaning that process p has input value i. The input model is $\mathcal{I} = \langle I, \chi, \ell \rangle$ where:

- I is the simplicial complex whose set of vertices is $\mathcal{V}(I) = \mathrm{Ag} \times \{0, 1, 2\}$, and whose facets are of the form $\{(B, i), (W, j)\}$ for all $i, j \in \{0, 1, 2\}$.
- The coloring $\chi : \mathcal{V}(I) \to \mathrm{Ag}$ is the first projection $\chi(p, i) = p$.
- $\ell(p, i) = \{\mathsf{input}^i_p\}$.

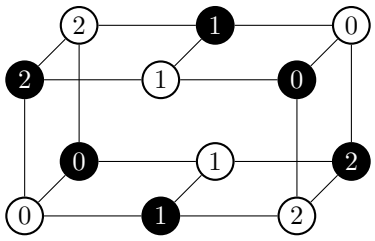
The input model $\mathcal{I}$ is represented below. In the picture, a vertex $(p, i) \in \mathcal{V}(I)$ is represented as a vertex of color p with value i.



We now define the action model $\mathcal{T} = \langle T, \sim, \mathsf{pre}\rangle$ that specifies the task. Since the only possible outputs are 0 and 1, there are four possible actions: $T = \{0,1\}^2$, where by convention the first component is the decision of B, and the second component is the decision of W. Thus, two actions $(d_B, d_W) \sim_B (d'_B, d'_W)$ in T are indistinguishable by B when $d_B = d'_B$, and similarly for W. Finally, the precondition $\mathsf{pre}(d_B, d_W)$ specifies the task as expected: if $d_B = d_W$ then $\mathsf{pre}(d_B, d_W)$ is a formula expressing that the two input values of the processes must be different, and otherwise that they are the same. For instance, we can take $\mathsf{pre}(0,1) = \bigvee_{i\in\{0,1,2\}} \mathsf{input}^i_B \wedge \mathsf{input}^i_W$

The *output model* is obtained as the product update model $\mathcal{O} = \mathcal{I}[\mathcal{T}] = \langle O, \chi_{\mathcal{O}}, \ell_{\mathcal{O}}\rangle$. By definition, the vertices of O are of the form (p, i, E), where $(p, i) \in \mathcal{V}(I)$ is a vertex of $\mathcal{I}$, and E is an equivalence class of $\sim_p$. But note that $\sim_p$ has only two equivalence classes, depending on the decision value (0 or 1) of process p. So a vertex of O can be written as (p, i, d), meaning intuitively that process p started with input i and decided value d. The facets of O are of the form $\{(B, i, d_B), (W, j, d_W)\}$ where either $i = j$ and $d_B \neq d_W$, or $i \neq j$ and $d_B = d_W$. The coloring $\chi_{\mathcal{O}}$ and labeling $\ell_{\mathcal{O}}$ behave the same as in $\mathcal{I}$.

The output model for the equality negation task is depicted below. Decision values do not appear explicitly on the picture, but notice how the vertices are arranged as a rectangular cuboid: the vertices on the front face have decision value 0, and those on the rear have decision value 1.



Each edge of $\mathcal{O}$ corresponds to one possible correct execution of the equality negation task. For instance, the leftmost edge corresponds to the situation where process B started with input value 2, process W started with input value 0, and they both decided on output 0. The only other possible solution for this input configuration is that they both decide 1, which corresponds to the rightmost edge.

5.3. Epistemic coverings

One of the simplest and most important functors of algebraic topology is the *fundamental group,* which creates an algebraic image of a space from the loops in the space (i.e., paths starting and ending at the same point). The fundamental group of a space $\boldsymbol{X}$ is defined so that its elements are loops in $\boldsymbol{X}$ starting and ending at a fixed basepoint $x_0 \in \boldsymbol{X}$, but two such loops are regarded as determining the same element of the fundamental group if one loop can be continuously deformed into the other within the space $\boldsymbol{X}$. There is a very deep connection between the fundamental group and *covering spaces*, and they can be regarded as two ways to view the same notion <30>. This means that algebraic features of the fundamental group can often be translated into the geometric language of covering spaces. Roughly speaking, a space $\boldsymbol{Y}$ is called a covering space of $\boldsymbol{X}$ if $\boldsymbol{Y}$ maps

onto $\boldsymbol{X}$ in a locally homeomorphic way, so that the pre-image of every point in $\boldsymbol{X}$ has the same cardinality. We are considering the combinatorial version of this notion, covering complexes <30>, extended to the epistemic setting.

Let $\mathcal{M} = \langle M, \chi, \ell \rangle$ and $\mathcal{M}' = \langle M', \chi', \ell' \rangle$ be simplicial models for two agents, and $\Phi : \mathcal{M} \to \mathcal{M}'$ and $\Phi' : \mathcal{M}' \to \mathcal{M}$ two matching monotonic, chromatic and label-preserving carrier maps. Let R be the relation induced by Φ, Φ' (recall the definition of matching carrier maps in Section 4.2). We say that $(\mathcal{M}', \Phi')$ is an *epistemic covering* of $\mathcal{M}$, if

- M' is a connected graph
- Φ' is a simplicial map
- for every edge e of M, $\Phi(e)$ is a non-empty set of disjoint edges: if $e', e'' \in \Phi(e)$ with $e' \neq e''$, then $e' \cap e'' = \emptyset$.

Epistemic coverings are coverings in the sense of covering graphs (or more generally, complexes), see e.g. <14; 30>. Observe that M is the image of Φ'; hence, M is also connected, since simplicial maps preserve connectivity. We sometimes denote Φ' by the simplicial map $f : \mathcal{M}' \to \mathcal{M}$, and Φ by f^{-1}. We say that $(\mathcal{M}', \Phi')$ is a *trivial epistemic covering* if Φ $(= f^{-1})$ is also a simplicial map.

Theorem 8. *If $(\mathcal{M}', \Phi')$ is an epistemic covering of $\mathcal{M}$, then the corresponding relation R is a total bisimulation between $\mathcal{M}$ and $\mathcal{M}'$.*

Proof. It is straightforward to check that Φ, Φ' are unity carrier maps, and hence by Theorem 7 the induced relation R is a bisimulation. □

From the theory of covering complexes we get non-trivial properties such as the following one (see Corollary 2.11 in <30>). It says that all sets of bisimilar states have the same cardinality. If s is either a vertex or an edge of M, we call $f^{-1}(s)$ the *sheets* over s, and its cardinality $|f^{-1}(s)|$ is called the *size* of the sheets.

Theorem 9. *If $(\mathcal{M}', f)$ is an epistemic covering of $\mathcal{M}$, then for any two vertices v, v' and for any two edges e, e' in $\mathcal{M}$ we have $|f^{-1}(v)| = |f^{-1}(v')|$ and $|f^{-1}(e)| = |f^{-1}(e')|$.*

5.4. Covering tasks and generalized equality negation

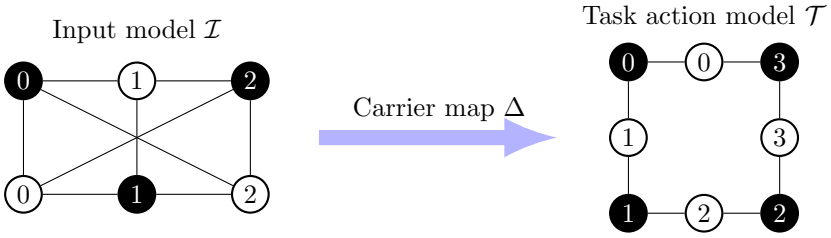
Recall from Section 5.1 that given an input simplicial model, $\mathcal{I} = \langle I, \chi, \ell \rangle$, a task for $\mathcal{I}$ is an action model $\mathcal{T} = \langle T, \sim, \mathsf{pre} \rangle$ for agents Ag, where each action $t \in T$ consists of a function $t : \mathrm{Ag} \to V^{out}$, where V^{out} is an arbitrary domain of output values. If the product update simplicial model $\mathcal{O} = \mathcal{I}[\mathcal{T}]$ is an epistemic covering of $\mathcal{I}$, we call it a *covering task*.

5.4.1. Generalized equality negation

We now come back to the equality negation task for two processes. As it turns out, the output model, $\mathcal{O}$, is a covering of the input model, $\mathcal{I}$, because the projection map $\pi : \mathcal{O} \to \mathcal{I}$ that leaves out decision values is a simplicial map, and its inverse sends each edge to a disjoint set of edges (actually two edges: either with the same output value if the inputs are different, or with distinct output values if the inputs are the same). Thus, there is a bisimulation between the input and output models.

Furthermore, we can define the *k-generalized equality negation* task, for $k \geq 1$, which has $2k$ output values, $V^{out} = \{0, 1, \ldots, 2k-1\}$, and is defined on the same input model $\mathcal{I}$ as follows. Each process starts with an input value in the set $\{0, 1, 2\}$, and has to irrevocably decide on a value in V^{out}, such that the decisions of the two processes are the same if their input values are different. If their input values are the same, then the decisions are such that $d_W = d_B + 1 \pmod{2k}$ (and hence different). For $k = 1$ we have the equality negation task. We can also consider the *∞-generalized equality negation* task, has V^{out} consisting of all integers. If the input values are different, then the outputs must be the same; and if their inputs are the same, then the decisions must be $\{(B, i), (W, i+1)\}$, for any $i \in \mathbb{Z}$.

The next figure illustrates the case of $k = 2$, using a carrier map Δ between two simplicial complexes following the distributed computing formalism (see Appendix A.1). That is, for each input vertex (p, i), $\Delta(p, i)$ is the set of possible decisions, each one an output vertex (p, j), for some output value j. Similarly for input edges, $\Delta(e)$ is the output subgraph of decisions allowed by the task. The right part of the picture corresponds to the task action model $\mathcal{T}$ of the task, viewed as a simplicial model via the equivalence of Theorem 1 (each edge is an action). Notice that $\mathcal{T}$ itself is not a covering of $\mathcal{I}$; the covering is obtained by taking the product update $\mathcal{I}[\mathcal{T}]$.



In the previous figure, there are different edges of $\mathcal{I}$ that are mapped by Δ into the same edge of $\mathcal{T}$, but not so in the product update $\mathcal{I}[\mathcal{T}]$. This is because, for each edge e, $\Delta(e)$ is a disjoint set of edges.

Lemma 10. *Let $\mathcal{I}$ be a connected input model, and consider a task $\mathcal{T}$, with a carrier map Δ, such that for each edge e, $\Delta(e)$ is a disjoint set of edges, and such that $\Delta(\mathcal{I})$ is connected. Then $\mathcal{T}$ is a covering task.*

Proof. The proof is by observing that the inverse carrier map Φ' from $\mathcal{I}[\mathcal{T}]$ to $\mathcal{I}$ is a simplicial map. In more detail, let $\mathcal{I} = \langle I, \chi_{\mathcal{I}}, \ell_{\mathcal{I}} \rangle$, and Φ be the carrier map from $\mathcal{I}$ to $\mathcal{O}$, where we write $\mathcal{O}$ for the product update model $\mathcal{O} = \mathcal{I}[\mathcal{T}] = \langle O, \chi_{\mathcal{O}}, \ell_{\mathcal{O}} \rangle$. Recall that the vertices of O are of the form (p, i, E), $(p, i) \in \mathcal{V}(I)$ is a vertex of $\mathcal{I}$, and E is an equivalence class of $\sim_p$. The following two requirements of an epistemic covering are satisfied: O is a connected graph and for every edge e of I, the edges in $\Phi(e)$ are disjoint. To prove that Φ' is a simplicial map, notice that edges of O that come from the same edge of I are disjoint by assumption, and those that come from different edges of I become disjoint by the product operation.

Lemma 10 together with Theorem 8 imply the following result.

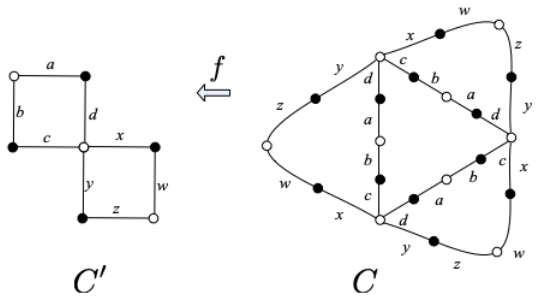
Theorem 11. *Let $\mathcal{I}$ and $\mathcal{O} = \mathcal{I}[\mathcal{T}]$ be the input and output models of the k-generalized equality negation task, for any $k \geq 1$, or for $k = \infty$. This is a covering task, and hence the relation $R = \{(\pi(X), X) \mid X \in \mathcal{F}(\mathcal{O})\} \subseteq \mathcal{I} \times \mathcal{O}$ is a bisimulation between $\mathcal{I}$ and $\mathcal{O}$.*

Notice that the universal covering task for $\mathcal{I}$ is a tree, and hence it is not an "equality negation" task, in the sense that the requirement that processes with different input values should decide the same output value implies there must be cycles in the output model.

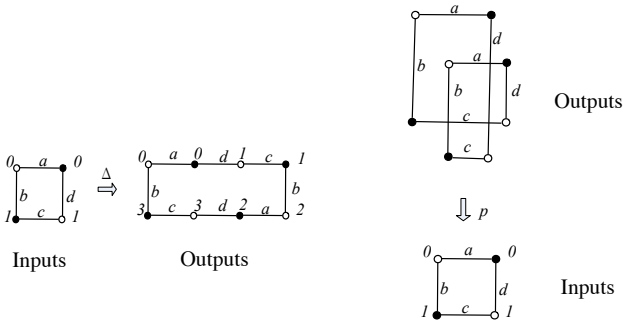
5.4.2. Other covering tasks

In the previous section we assumed that $\mathcal{I}$ is the input model for the equality negation task, but any input model can be used to define a covering task. We show below two examples from <14> (where covering complexes are studied for a different purpose).

Here is an example of a covering task defined by an output model with complex C being a covering of an input model with complex C'. In the figure we assume that a vertex v and $f(v)$ have the same valuations, and an edge in C is sent by f to the edge of the same label in C'.



As another example from <14>, the figure depicts a covering model with sheets of size 2, and how it wraps twice around the input cycle.



For each finite $k \geq 1$, there is a similar example, where the output complex wraps k times around the input cycle. And the infinite line is the universal cover of the cycle (when k is infinity).

Covering tasks can be arranged into a hierarchy, as described in <14>. Let G be the fundamental group of $\mathcal{M}$. For every subgroup H of G there is a connected complex $\mathcal{M}'$ with fundamental group H, and a simplicial map $f : \mathcal{M}' \to \mathcal{M}$ such that $\mathcal{M}'$ is a covering complex of $\mathcal{M}$ (see <30>, Theorem 2.8). In particular, the *universal covering*

is obtained by taking the subgroup that consists of the identity only, which is a covering of any other covering of $\mathcal{M}$. Thus, we have the following theorem (observing that since $\mathcal{I}$ is chromatic, it induces a chromatic covering complex).

Theorem 12. *Let $\mathcal{I}$ be a connected input model for two agents. For every subgroup H of the fundamental group of the underlying complex of $\mathcal{I}$, there exists a task with an output model $\mathcal{O}$ with fundamental group H, such that $\mathcal{O}$ is an epistemic covering of $\mathcal{I}$.*

6. A DEL analysis of Equality Negation and other covering tasks

Our goal is to use the DEL framework of Section 3 to prove the unsolvability of equality negation. In <18>, we devised a proof method to establish such impossibility results, and successfully applied it to two classic distributed-computing examples: *consensus* and *approximate-agreement*. The proof method of <18> relies on an epistemic logic formula φ witnessing the impossibility of solving the task. For the so-called *set-agreement* task, we were not able to find a suitable formula φ.

We will see in this section that this method fails for the equality negation task, and in fact for any epistemic covering task; namely, for any such task, we can prove that no such formula φ exists.

6.1. Outline of impossibility proofs in <18>

We now describe how the set up of <18> is used to prove impossibility results in distributed computing. It is closely related to the usual topological approach to distributed computability <19>, except that the input complex, output complex and protocol complex are now viewed as simplicial models for epistemic logic. By interpreting epistemic logic formulas on those structures, we can understand the epistemic content of the abstract topological arguments for unsolvability. For example, when the usual topological proof would claim that consensus is not solvable because the protocol complex is connected, our DEL framework allows us to say that the reason for impossibility is that the processes did not reach common knowledge of the set of input values. This particular example, among others, is treated in depth in <18>.

To prove impossibility results, our goal is thus to show that the map δ of Definition 10 does not exist. To do so, we rely on the following lemma, which is a reformulation in the simplicial setting of a classic result of modal logics. An epistemic logic formula is called *positive* when it belongs to the following grammar:

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid K_a\varphi \qquad p \in \mathrm{At},\ a \in \mathrm{Ag}$$

Intuitively, a positive formula is not allowed to talk about what an agent does not know.

Lemma 13 (<18>). *Consider simplicial models $\mathcal{M} = \langle C, \chi, \ell\rangle$ and $\mathcal{M}' = \langle C', \chi', \ell'\rangle$, and a morphism $f : \mathcal{M} \to \mathcal{M}'$. Let $X \in \mathcal{F}(C)$ be a facet of $\mathcal{M}$, and let φ be a positive formula. Then, $\mathcal{M}', f(X) \models \varphi$ implies $\mathcal{M}, X \models \varphi$.*

To prove that a task $\mathcal{T}$ is not solvable in $\mathcal{MP}_N$, our usual proof method goes like this. Assume by means of contradiction that the task is solvable, that is, there exists $\delta\colon \mathcal{I}[\mathcal{MP}_N] \to \mathcal{I}[\mathcal{T}]$. The strategy consists of the following four steps:

1. Pick a well-chosen positive epistemic logic formula φ,
2. Show that φ is true in every world of $\mathcal{I}[\mathcal{T}]$,
3. Show that there exists a world X of $\mathcal{I}[\mathcal{MP}_N]$ where φ is false,
4. By Lemma 13, since φ is true in $\delta(X)$ then it must also be true in X, which is a contradiction with the previous point.

This kind of proof is interesting because it explains the reason why the task is not solvable. The formula φ represents some amount of knowledge about the local values that are taken or not taken by other processes, which the processes must acquire in order to solve the task. If φ is given, the difficult part of the proof is usually the third point: finding a world X in the protocol complex where the processes did not manage to obtain the required amount of knowledge. The existence of this world can be proved using theorems of combinatorial topology, such as Sperner's Lemma; see <18> for such examples.

6.2. No formula shows the unsolvability of a covering task

We can finally use Lemma 4 to show that no positive formula φ will allow us to prove the unsolvability of the equality negation task. Actually, the same holds for any *covering task*, namely, one where the output model is a covering model of the input model, under the projection map π.

Theorem 14. *Consider a covering task with input model $\mathcal{I}$ and output model $\mathcal{O}$. Let X be a facet of $\mathcal{I}[\mathcal{MP}_N]$ and let Y be a facet of $\mathcal{O}$ such that $\pi(X) = \pi(Y)$. Then for every positive formula φ we have the following: if $\mathcal{O}, Y \models \varphi$ then $\mathcal{I}[\mathcal{MP}_N], X \models \varphi$.*

Proof. Let φ be a positive formula and assume $\mathcal{O}, Y \models \varphi$. Note that $\pi(Y)$ and Y are bisimilar by Theorem 11. Thus, by Lemma 4, we have $\mathcal{I}, \pi(Y) \models \varphi$. Finally, since $\pi(Y) = \pi(X)$, by Lemma 13 we obtain $\mathcal{I}[\mathcal{MP}_N], X \models \varphi$. □

In the above theorem, the world Y should be thought of as a candidate for $\delta(X)$. The condition $\pi(X) = \pi(Y)$ comes from the commutative diagram of Definition 10. Thus, Theorem 14 says that we will never find a formula φ which is true in $\delta(X)$ but false in X.

Remark 1. Notice that the proof of Theorem 14 does not use any property of the layered message-passing model $\mathcal{MP}_N$. In fact, the only crucial property is that the projection map $\pi : \mathcal{O} \to \mathcal{I}$ is a bisimulation. So, the same Theorem would hold for any other computational model, and even for any other task (non-necessarily a covering one) where the input model $\mathcal{I}$ and the output model $\mathcal{O}$ are bisimilar.

Remark 2. As previously discussed, Theorem 14 does not apply to consensus, since we know that there exists a formula proving its unsolvability. The reason is that in the case of consensus, the relation $R = \{(\pi(X), X) \mid X \in \mathcal{F}(\mathcal{O})\}$ induced by the projection mapping $\pi : \mathcal{O} \to \mathcal{I}$ is not a bisimulation. Indeed, let us show that condition **forth** of Definition 8 does not hold. Namely, if $X = \{(B,0),(W,1)\}$ and $X' = \{(B,0,1),(W,1,1)\}$ and $Y = \{(B,0),(W,0)\}$, then by definition of consensus there cannot exist a facet Y' with $Y\,R\,Y'$ and $B \in \chi'(X' \cap Y')$. Such a facet would have the form $Y' = \{(B,0,1),(W,0,d)\}$, for a $d \in \{0,1\}$, which is not a valid world in the output model of consensus for any decision d.

As we have seen, covering tasks provide a rich source of examples. For each input model $\mathcal{I}$, there are many output models that are covering models of $\mathcal{I}$, and these can be arranged into a hierarchy. Interestingly, the only wait-free solvable covering task is the trivial one. Yet, for every non-trivial covering task, the output and input models are bisimilar, which implies that the unsolvability of the task cannot be shown using a witness formula φ as in Section 6.1.

7. Discussion

The main result of this paper, Theorem 14, shows a limit of the DEL approach for studying distributed tasks. Indeed, we exhibited an example of a task, equality negation, which is known to be unsolvable using traditional topological methods, but for which there is no epistemic logic formula witnessing this impossibility. This is disappointing because, as shown in <18>, the DEL approach has the major benefit that the formula φ used in the impossibility proof pinpoints the amount of knowledge that the processes must be able to reach in order to solve the task.

This "meta-impossibility" result seems to indicate that our logic is *too weak* to express the reason why some tasks are unsolvable. Thus, the way forward seems to be to enrich the language of our logic in order to make it more expressive. Here, "more expressive" may mean that the enriched logic contains formulas that are not equivalent to any formula in the non-enriched logic, but in the setting of dynamic epistemic logic it may also mean that the enriched logic contains dynamic modalities for model transformations that are not equivalent to any transformation in the non-enriched logic; in other words, the enriched logic is more "update-expressive". There are several ways one might achieve this.

Assigning new values to atomic propositions. The epistemic logic formulas that we have only contain atomic variables coming from the input model $\mathcal{I}$; thus, we are only allowed to talk about the processes knowing about each other's *input* values. But many tasks, including equality negation, are specified in terms of the relationship between the inputs and the *outputs* of the processes. In epistemic logic we may also wish to write formulas of the form "Process p knows that process q will decide on the output value 0". Here is how. At the beginning of a computation these decision variables are set to 'false' since we do not know yet which decision will be taken. When a process takes a decision, the action model must be able to change the value of the corresponding variable. This is a feature known as *factual change* in dynamic epistemic logic.

According to distributed computing vocabulary, we use the letter p instead of a to denote a generic agent ("process"), and the atomic variables in the input model $\mathcal{I}$ are written input_p^i ("process p has input i"). If we also allow the use of atomic variables of the form decide_p^d ("process p decides d"), we could write the formula below, specifying the equality negation task:

$$\varphi_{\text{EN}} \;=\; \bigwedge_{p,i,d} \Big(\mathsf{input}_p^i \wedge \mathsf{decide}_p^d \implies (\mathsf{input}_{\bar{p}}^i \wedge \mathsf{decide}_{\bar{p}}^{\bar{d}}) \vee (\mathsf{input}_{\bar{p}}^{\bar{i}} \wedge \mathsf{decide}_{\bar{p}}^{d}) \Big)$$

where $\bar{p}$, $\bar{i}$, $\bar{d}$ denote values different from p, i, d, respectively. Note that $\bar{p}$ and $\bar{d}$ are uniquely defined (since there are only two processes and two decision values), but for $\bar{i}$,

there are two possible inputs different from i. So, for example, $\mathsf{input}_p^{\bar{0}}$ is actually a shortcut for $\mathsf{input}_p^1 \vee \mathsf{input}_p^2$. This formula simply expresses the specification of the task: if process p has input i and decides d, then the other process should either have the same input and decide differently, or have a different input and decide the same.

This kind of approach was developed in <18>, where an ad-hoc construction dubbed "Extended DEL" allows to make sense of these extra atomic propositions decide_p^d and thus produce an impossibility proof. Such an "extension" of DEL is better known in the literature as *DEL with factual change* <4; 7>. A formal treatment of factual change in the setting of simplicial models was carried out in <11>. Appendix B develops this idea and achieves an impossibility proof for equality negation using DEL with factual change.

This approach seems somewhat unsatisfactory from the point of view of epistemic logic, as the formula φ_{EN} above contains no knowledge operator, and simply consists in the specification of the task. However, we should note that a formalization involving factual change has implicit epistemic consequences, namely that the value becomes known (and even common knowledge) to all agents. We omit a detailed explanation as it is out of the scope of this paper. Still, the impossibility proof relying on this formula does not seem to bring any new insights on why the processes cannot solve the task, compared to the usual topological proof of impossibility.

Adding new epistemic operators. Another way to strengthen the language of our logic is to introduce new operators. The only notion of knowledge that we allow in this paper is the standard $K_a\varphi$ operator; but the epistemic logic literature contains many other operators for describing various forms of knowledge.

The most prominent one is *common knowledge*, which allows us to express infinite formulas of the form "everyone knows that everyone knows that everyone knows ...". However, we already know that adding common knowledge will not be sufficient to prove the impossibility of equality negation. Indeed, it was proved in <18> that the analogues of Lemmas 4 and 13 also hold for a logic with common knowledge. Since these are the key ingredients in our "meta-impossibility" proof, there is no hope to find a positive formula witnessing the impossibility of equality negation, even in the presence of common knowledge.

Other epistemic operators based on group knowledge have been investigated in <11>. Such operators can make sense of higher-dimensional simplexes in simplicial models. While these operators might be useful to study other kinds of tasks (such as 2-set agreement), there is little hope that it will yield a new impossibility proof for equality negation, since it is a task between two processes and thus of a 1-dimensional nature.

Bisimulation and other notions of similarity. A benefit of the current paper is that it gives a tangible success criterion for this kind of endeavour: the notion of bisimulation associated with the newly introduced logic should be such that the input and output complexes of equality negation are *not* bisimilar. Along the different enrichments we propose there are again different avenues to pursue. Assuming additional epistemic operators, bisimulation for the enriched logic should extend Definition 8 of bisimulation in the current paper with clauses for these novel epistemic modalities. Although common knowledge does not require such adjustments (but merely enhances expressivity), other group epistemic modalities such as distributed knowledge require stricter notions of bisimulation. Whereas adding factual change opens the avenue to consider notions

of similarity for action models. As known, something akin to bisimulation is too strong to require for similarity of action models, and a weaker notion called *emulation* is then sufficient (if there is an emulation between action models, then the models resulting from updating a given Kripke model with either one or the other action model will be bisimilar) <12>. All these are avenues for further research as well.

8. Conclusion

The equality negation task is known to be unsolvable in the wait-free read/write model. In fact, equality negation it is an instance of a rich family of tasks, that we call epistemic covering tasks, defined in terms of covering spaces, well-known in topology. All covering tasks are unsolvable in the wait-free model.

In this paper, we study these tasks using the simplicial complex semantics of DEL that we proposed in <18>. There are several purposes of doing this. First, the logical formula witnessing the unsolvability of a task usually helps us understand the epistemic content of this task. Unfortunately, as it turns out, there is no such informative formula for the class of epistemic covering tasks. However, this is a nice case study to test the limits of our DEL framework. Indeed, we proved that the basic language of DEL, where formulas are only allowed to talk about input values, is too weak to express the reason why the equality negation, or any other epistemic covering task, is not solvable. While it is possible to fix this issue by introducing extra atomic propositions in our logical language, the formula obtained in this way turns out to be uninteresting.

We hope our work has motivated further interest on the interplay of distributed computing, epistemic logic, and combinatorial topology issues. We used dynamic epistemic logic for this, but it would be interesting to consider other logics.

Many interesting open questions come to mind. We showed in Theorem 14 that no positive formula φ can prove the impossibility of an epistemic covering task. Could the impossibility proof exist for a non-positive formula? Also, it would be interesting to settle the question of the existence of a formula for set agreement, an important task in distributed computing. We stated our results for two processes, to avoid getting into topology technicalities. Some of our results can be easily generalized, such as the bisimulation Theorem 7, and it seems also the epistemic covering task definition. It would be interesting to study in this context the multi-process versions of equality negation discussed in <17>. We presented here (Section 5.4) a family of generalized equality negation tasks for two processes, but we do not know if these are the only equality negation versions on the same input model, and we have not considered multi-process versions in this context. We introduced epistemic covering tasks, as a rich unsolvable family of tasks, which have an output model that is bisimilar to the input model, but perhaps there are other interesting tasks with this property. Finally, maybe there are tasks which do not have this property, but still no formula can show them to be unsolvable? We strongly suspect that the so-called *set-agreement* is an example of such a task.

[1] Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons, Inc., USA, 2004.

[2] A. Baltag, L.S. Moss, and S. Solecki. The logic of common knowledge, public announcements, and private suspicions. In *TARK VII*, pages 43–56, 1998.

[3] A. Baltag and B. Renne. Dynamic epistemic logic. In *The Stanford Encyclopedia of Philosophy, see* `https://plato.stanford.edu/archives/win2016/entries/dynamic-epistemic/`. Metaphysics Research Lab, Stanford University, 2016.

[4] J. van Benthem, V. van Eijck, and B. Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006.
[5] Ofer Biran, Shlomo Moran, and Shmuel Zaks. A combinatorial characterization of the distributed 1-solvable tasks. *J. Algorithms*, 11(3):420–440, 1990.
[6] Benny Chor, Amos Israeli, and Ming Li. On processor coordination using asynchronous hardware. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '87, pages 86–97, New York, NY, USA, 1987. ACM.
[7] H. van Ditmarsch and B. Kooi. Semantic results for ontic and epistemic change. In *Proc. of 7th LOFT*, Texts in Logic and Games 3, pages 87–117. Amsterdam University Press, 2008.
[8] H. van Ditmarsch, W. van der Hoek, and B. Kooi. Dynamic epistemic logic with assignment. In *Proc. of 4th AAMAS*, pages 141–148. ACM, 2005.
[9] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Springer, 2007.
[10] H. van Ditmarsch, W. van der Hoek, B. Kooi, and L.B. Kuijer. Arrow update synthesis. *Information and Computation*, 2020. Online first at `https://doi.org/10.1016/j.ic.2020.104544`.
[11] Hans van Ditmarsch, Éric Goubault, Jérémy Ledent, and Sergio Rajsbaum. Knowledge and simplicial complexes. *CoRR*, abs/2002.08863, 2020. To appear in Synthese-Int. J. for Epistemology, Methodology and Philosophy of Science, Springer.
[12] J. van Eijck, J. Ruan, and T. Sadzik. Action emulation. *Synthese*, 185(1):131–151, 2012.
[13] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.
[14] P. Fraigniaud, S. Rajsbaum, and C. Travers. Locality and checkability in wait-free computing. *Distributed Computing*, 26(4):223–242, 2013.
[15] Eli Gafni and Elias Koutsoupias. Three-processor tasks are undecidable. *SIAM J. Comput.*, 28(3):970–983, 1999.
[16] Éric Goubault, Marijana Lazić, Jérémy Ledent, and Sergio Rajsbaum. A dynamic epistemic logic analysis of the equality negation task. In *Second International Workshop Dynamic Logic- New Trends and Applications (DaLí)*, volume 12005 of *Lecture Notes in Computer Science*, pages 53–70. Springer, 2019.
[17] Éric Goubault, Marijana Lazić, Jérémy Ledent, and Sergio Rajsbaum. Wait-Free Solvability of Equality Negation Tasks. In Jukka Suomela, editor, *33rd International Symposium on Distributed Computing (DISC 2019)*, volume 146 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:16, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
[18] Éric Goubault, Jérémy Ledent, and Sergio Rajsbaum. A simplicial complex model for dynamic epistemic logic to study distributed task computability. *Information and Computation*, June 2020. In print, a preliminary version appeared in Proc. of GandALF 2018.
[19] M. Herlihy, D. Kozlov, and S. Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Elsevier-Morgan Kaufmann, 2013.
[20] Maurice Herlihy. Wait-free synchronization. *ACM Trans. Program. Lang. Syst.*, 13(1):124–149, January 1991.
[21] Maurice Herlihy and Sergio Rajsbaum. The decidability of distributed decision tasks (extended abstract). In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing (STOC), El Paso, Texas, USA, May 4-6, 1997*, pages 589–598, 1997.
[22] Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *J. ACM*, 46(6):858–923, 1999.
[23] Prasad Jayanti. On the robustness of herlihy's hierarchy. In *Proceedings of the Twelfth Annual ACM Symposium on Principles of Distributed Computing*, PODC '93, pages 145–157, New York, NY, USA, 1993. ACM.
[24] D. Kozlov. *Combinatorial Algebraic Topology*. Springer, 2007.
[25] L.B. Kuijer. The expressivity of update logics. *J. Log. Comput.*, 25(3):719–742, 2015.
[26] Wai-Kau Lo and Vassos Hadzilacos. All of us are smarter than any of us: Nondeterministic wait-free hierarchies are not robust. *SIAM J. Comput.*, 30(3):689–728, 2000.
[27] Michael C. Loui and Hosame H. Abu-Amara. Memory requirements for agreement among unreliable asynchronous processes. In *Advances in Computing research*, pages 163–183, Greenwich, CT, 1987. JAI Press.
[28] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.
[29] James R. Munkres. *Elements of Algebraic Topology*. CRC Press, 1984.
[30] J. Rotman. Covering complexes with applications to algebra. *The Rocky Mountain Journal of*

Mathematics, 3(4):641–674, 1973.

Appendix A. Distributed computing through combinatorial topology

In this appendix, we briefly describe the usual topological approach of task computability, focusing on the special case of two processes; additional details appear for example in <19>. The DEL framework that we expose in Section 2 is based on this approach. For instance, the notion of *carrier map* in the context of the topological approach, is replaced by the product update operation in DEL.

Appendix A.1. The topological definition of task solvability

To model a computation for $n+1$ processes, a simplicial complex of dimension n is used; in the case of two processes, a 1-dimensional simplicial complex is simply a graph. Thus, to simplify the formalism, we use here graph-theoretic notions instead of simplicial complexes.

For an (undirected) graph $\mathcal{G}$, we write $\mathcal{V}(\mathcal{G})$ for the set of vertices of $\mathcal{G}$. To mimic the usual vocabulary of the simplicial approach to distributed computability, we use the word *simplex* to refer either to a vertex or an edge of the graph. The edges will usually be identified with two-element sets $\{s, s'\}$, where $s, s' \in \mathcal{V}(\mathcal{G})$ are the two endpoints of the edge; and vertices are identified with singletons $\{s\}$ for $s \in \mathcal{V}(\mathcal{G})$. We say that a graph is *chromatic* if it is equipped with a coloring $\chi : \mathcal{V}(\mathcal{G}) \to C$, where C is a finite set of colors, such that the vertices of every edge have distinct colors. A graph $\mathcal{G}$ is *labeled* if it is equipped with a labeling $\ell : \mathcal{V}(\mathcal{G}) \to L$, where L is a set of labels (without any extra condition). To model a computation between two processes B and W, we use a chromatic labeled graph $\langle \mathcal{G}, \chi, \ell \rangle$. The set of colors $C = \{B, W\}$ is the set of process names. The labeling $\ell(s)$ of s is called the *view* of the corresponding process. Moreover, we require that each vertex in a chromatic labeled graph is uniquely identified by its name $\chi(s)$ and label $\ell(s)$.

Given two graphs $\mathcal{G}$ and $\mathcal{H}$, a map $\mu : \mathcal{V}(\mathcal{G}) \to \mathcal{V}(\mathcal{H})$ from the vertices of $\mathcal{G}$ to the vertices of $\mathcal{H}$ is a *simplicial map* if whenever $\{s_0, s_1\}$ is an edge in $\mathcal{G}$, then $\{\mu(s_0), \mu(s_1)\}$ is either an edge or a vertex of $\mathcal{H}$. If $s_0 \neq s_1$ implies $\mu(s_0) \neq \mu(s_1)$, then the map is said to be *rigid*. (Thus, a rigid simplicial map is a graph homomorphism). When $\mathcal{G}$ and $\mathcal{H}$ are chromatic, we usually assume that the simplicial map μ preserves names: $\chi(s) = \chi(\mu(s))$. Thus, chromatic simplicial maps are rigid. The following is an important property of simplicial maps, easy to check.

Lemma 15. *The image of a connected graph under a simplicial map is connected.*

Given two graphs $\mathcal{G}$ and $\mathcal{H}$, a *carrier map* Φ from $\mathcal{G}$ to $\mathcal{H}$, written $\Phi : \mathcal{G} \to 2^{\mathcal{H}}$, takes each simplex $\sigma \in \mathcal{G}$ to a subgraph $\Phi(\sigma)$ of $\mathcal{H}$, such that Φ satisfies the following *monotonicity* property: for all simplices $\sigma, \tau \in \mathcal{G}$, if $\sigma \subseteq \tau$, then $\Phi(\sigma) \subseteq \Phi(\tau)$. If the set of colors of the subgraph $\Phi(\sigma)$ is equal to the set of colors of σ, then Φ is *name-preserving*.

Let V^{in} be a set of *input values*, and V^{out} a set of *output values*. A *task* for B and W is a triple $(\mathcal{I}, \mathcal{O}, \Delta)$, where

- $\mathcal{I}$ is a pure chromatic *input graph* colored by $\{B, W\}$ and labeled by V^{in};
- $\mathcal{O}$ is a pure chromatic *output graph* colored by $\{B, W\}$ and labeled by V^{out};
- Δ is a name-preserving carrier map from $\mathcal{I}$ to $\mathcal{O}$.

The input graph defines all the possible ways the two processes can start the computation, the output graph defines all the possible ways they can end, and the carrier map defines which input can lead to which outputs. More precisely, each edge $\{(B,i),(W,j)\}$ in $\mathcal{I}$ defines a possible input configuration (initial system state), where the local state of B consists of the input value $i \in V^{in}$ and the local state of W consists of input value $j \in V^{in}$. The processes communicate with one another, and each eventually decides on an output value and halts. If B decides x, and W decides y, then there is an output configuration represented by an edge $\{(B,x),(W,y)\}$ in the output graph. If the following condition is verified,

$$\{(B,x),(W,y)\} \in \Delta(\{(B,i),(W,j)\}),$$

then we say that this run respects the task specification Δ.

We now turn our attention from tasks, to the model of computation in which we want to solve them. Consider a protocol execution in which the processes exchange information through the channels (message-passing, read-write memory, or other) provided by the model. At the end of the execution, each process has its own view (final state). The set of all possible final views themselves form a chromatic graph. Each vertex is a pair (P,p), where P is a process name, and p is the view (final state) of P at the end of some execution. A pair of such vertices $\{(B,p),(W,q)\}$ is an edge if there is some execution where B halts with view p and W halts with view q. This graph is called the *protocol graph.*

There is a carrier map Ξ from $\mathcal{I}$ to $\mathcal{P}$, called the *execution carrier map*, that carries each input simplex to a subgraph of the protocol graph. Ξ carries each input vertex (P,v) to the solo execution in which P finishes the protocol without hearing from the other process. It carries each input edge $\{(B,i),(W,j)\}$ to the subgraph of executions where B starts with input i and W with input j.

The protocol graph is related to the output graph by a *decision map* δ that sends each protocol graph vertex (P,p) to an output graph vertex (P,w), labeled with the same name. Operationally, this map should be understood as follows: if there is a protocol execution in which P finishes with view p and in this view it chooses output w, then (P,p) is a vertex in the protocol graph, (P,w) a vertex in the output graph, and $\delta((P,p)) = (P,w)$. It is easy to see that δ is a simplicial map, carrying edges to edges,because any pair of mutually compatible final views yields a pair of mutually compatible decision values.

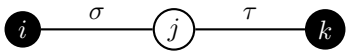
The composition of the decision map $\delta : \mathcal{P} \to \mathcal{O}$ with the carrier map $\Xi : \mathcal{I} \to 2^{\mathcal{P}}$ is a carrier map $\Phi : \mathcal{I} \to 2^{\mathcal{O}}$, which we denote abusively by $\Phi = \delta \circ \Xi$ (one can lift δ to act on powersets in the obvious way). We say that Φ is *carried by* Δ whenever $\Phi(\sigma) \subseteq \Delta(\sigma)$ for every simplex $\sigma \in \mathcal{I}$. We can now define what it means for a protocol to solve a task.

Definition 11. *The protocol* $(\mathcal{I},\mathcal{P},\Xi)$ solves *the task* $(\mathcal{I},\mathcal{O},\Delta)$ *if there exists a simplicial* decision map δ *from* $\mathcal{P}$ *to* $\mathcal{O}$ *such that* $\delta \circ \Xi$ *is carried by* Δ.

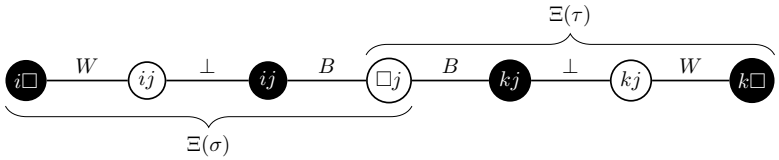
Appendix A.2. The layered message-passing protocol graph

We now define the N-layer protocol of Section 3 using the carrier map formalism described above. We formalize it using the notion of *view* defined in Section 3.3.

Starting with the input graph $\mathcal{I}$ with two input edges $\sigma = \{(B,i),(W,j)\}$ and $\tau = \{(B,k),(W,j)\}$ depicted below,



the protocol graph for the single-layer protocol is as follows:



The carrier map Ξ send each of the two input edges σ and τ to their subdivided image in the protocol graph. Notice how the white vertex $\square j$ in the middle now belongs to both $\Xi(\sigma)$ and $\Xi(\tau)$. Indeed, even if it knows for sure that the execution was 'B', it did not receive the message from B, so it does not know whether the input value of black was i or k. It is remarkable that the single-layer protocol graph in this model is the same as the input graph, except that each input edge is subdivided into three.

In the N-layer message passing protocol, we iterate the previous construction N times. At each subsequent layer, each process uses its view from the previous layer as its input value for the next layer.

It should now be clear that each subsequent layer further subdivides the edges of the previous layer into three. It carries each input edge $\{(B,i),(W,j)\}$ to the subgraph of executions where B starts with input i and W with input j. Formally, the protocol graph $\mathcal{P}_N$ of the N-layer message passing protocol with input graph $\mathcal{I}$ has vertices of the form $(X, \mathsf{view}_X(\alpha,\sigma))$ with $X \in \{B,W\}$, $\alpha \in \{B,W,\bot\}^N$ and $\sigma \in \mathcal{I}$. Its edges are of the form $\{(B, \mathsf{view}_B(\alpha,\sigma)),(W, \mathsf{view}_W(\alpha,\sigma))\}$, with $\alpha \in \{B,W,\bot\}^N$ and $\sigma \in \mathcal{I}$. Thus, a vertex of color X belongs to two such edges whenever $\mathsf{view}_X(\alpha,\sigma) = \mathsf{view}_X(\beta,\tau)$, for some α,β,σ,τ. The carrier map $\Xi : \mathcal{I} \to 2^{\mathcal{P}}$ takes each edge $\sigma \in \mathcal{F}(\mathcal{I})$ to the subgraph of $\mathcal{P}$ containing all the vertices of the form $(X, \mathsf{view}_X(\alpha,\sigma))$, and all edges between them.

Theorem 16. *The protocol graph $\mathcal{P}$ for the N-layer message passing model is a subdivision of the input graph, where each edge is divided into 3^N edges. The execution carrier map Ξ from $\mathcal{I}$ to $\mathcal{P}$, carries each input edge to its subdivision in the protocol graph, and each input vertex (P,v) to the solo execution in which P finishes the protocol without hearing from the other process.*

Lemma 17. *The protocol graph $\mathcal{P}_N$ for the N-layer message-passing protocol is isomorphic to the product update model $\mathcal{I}[\mathcal{MP}_N]$ defined in Section 3.2.*

Proof. This follows almost immediately from Proposition 1, which says that:

$$\mathsf{view}_B(\alpha,\sigma) = \mathsf{view}_B(\beta,\tau) \iff (\alpha,\sigma) \sim_B (\beta,\tau)$$

Once we have this, proving the Lemma is just a matter of unfolding the definitions of the product update model. A vertex of $\mathcal{I}[\mathcal{MP}_N]$ is formally given by a pair (v,E), where v is a vertex of $\mathcal{I}$ (say, of color B) and E is an equivalence class of $\sim_B$. Let $(\alpha,\sigma) \in E$ be an action in E. Then, to the vertex (v,E) of $\mathcal{I}[\mathcal{MP}_N]$ we associate the vertex $(B, \mathsf{view}_B(\alpha,\sigma))$ of $\mathcal{P}_N$; this is well-defined thanks to (1). Surjectivity is obvious.

To show injectivity, assume that (v, E) and (v', E') give the same view. By Proposition 1, we get $E = E'$; and $v = v'$ according to the precondition pre of the action model. Finally, in $\mathcal{I}[\mathcal{MP}_N]$, we get an edge between (v_B, E_B) and (v_W, E_W) whenever there is an action $t = (\alpha, \sigma)$ with $\sigma := \{v_B, v_W\}$ and $t \in E_B$ and $t \in E_W$. Then by definition of our bijection between the vertices, this edge is sent to the pair $\{(B, \mathsf{view}_B(\alpha, \sigma)), (W, \mathsf{view}_W(\alpha, \sigma))\}$, which is also an edge. □

Appendix A.3. Solvability of the equality negation task

The equality negation task has been described in the DEL framework in Section 6. For completeness, and comparison of the two approaches, we study it here using the usual combinatorial topology approach. Also, we present a direct proof of the impossibility of solving it in the layered message-passing model.

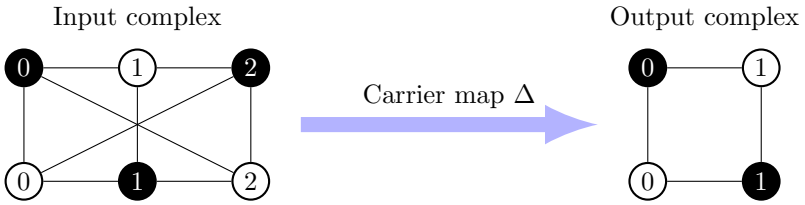
Appendix A.3.1. Definition of equality negation

Recall that in the equality negation task for two processes, $\mathrm{Ag} = \{B, W\}$, each process starts with an input value in the set $\{0, 1, 2\}$, and has to irrevocably decide on a value 0 or 1, such that the decisions of the two processes are the same if and only if their input values are different.

The equality negation task is fomralized as a triple $(\mathcal{I}, \mathcal{O}, \Delta)$. The input complex, $\mathcal{I}$, has vertices $\mathcal{V}(\mathcal{I}) = \mathrm{Ag} \times \{0, 1, 2\}$, and edges of the form $\{(B, i), (W, j)\}$ for all $i, j \in \{0, 1, 2\}$. Thus, each vertex is a pair, whose first component is the name of a process, and whose second component is the input value of the process. Similarly, the output complex, $\mathcal{O}$, has vertices $\mathcal{V}(\mathcal{O}) = \mathrm{Ag} \times \{0, 1\}$, and edges of the form $\{(B, i), (W, j)\}$ for all $i, j \in \{0, 1\}$. The carrier map Δ is as follows

$$\begin{aligned}
\Delta(\{(P, i)\}) &= \{(P, 0), (P, 1)\}, P \in \{B, W\} \\
\Delta(\{(B, i), (W, i)\}) &= \{\{(B, 0), (W, 1)\}, \{(B, 1), (W, 0)\}\} \cup \mathcal{V}(\mathcal{O}) \\
\Delta(\{(B, i), (W, j)\}) &= \{\{(B, 0), (W, 0)\}, \{(B, 1), (W, 1)\}\} \cup \mathcal{V}(\mathcal{O}), i \neq j
\end{aligned}$$

Notice that Δ satisfies the monotonicity property and is name-preserving.



Remark 3. Notice that we do not have an analogue of Lemma 17 here: the output complex is distinct from the product update model $\mathcal{I}[\mathcal{T}]$ of Section 5.2.

Appendix A.3.2. Equality negation impossibility

Direct impossibility. Here we show that there is no solution to the equality negation task in the layered message-passing model, using the usual combinatorial topology approach.

Assume for contradiction, that there is a solution, with some number of layers, N. Then, by Theorem 16, the protocol graph $\mathcal{P}$ for the N-layer message passing model

is a subdivision of the input graph, where each edge is divided into 3^N edges. The execution carrier map Ξ from $\mathcal{I}$ to $\mathcal{P}$, carries each input edge to its subdivision in the protocol graph, and each input vertex (P, v) to the solo execution in which P finishes the protocol without hearing from the other process. Also, Definition 11 states that if $\mathcal{P}$ solves equality negation, then there exists a simplicial decision map δ from $\mathcal{P}$ to $\mathcal{O}$ such that $\delta \circ \Xi$ is carried by Δ.

Consider the subgraph $\mathcal{G}$ of $\mathcal{I}$ induced by all edges that have distinct input values. Notice that $\mathcal{G}$ is connected. Thus, $\Xi(\mathcal{G})$, is connected, by Theorem 16. Recall that Lemma 15 states that the image of a connected graph under a simplicial map is connected. Thus, $\delta(\Xi(\mathcal{G}))$ is a connected subgraph of $\mathcal{O}$.

The specification of the equality negation task states that the decisions should be equal, for all of $\mathcal{G}$. But the subgraph of $\mathcal{O}$ of edges with the same decision is disconnected. Therefore, $\delta(\Xi(\mathcal{G}))$ must be equal to one of the edges in this subgraph, without loss of generality, $\delta(\Xi(\mathcal{G}))$ is equal to the subgraph of $\mathcal{O}$ with the single edge $\{(B,0),(W,0)\}$.

It follows that

$$\begin{aligned} \delta(\Xi(B,0)) &= (B,0) \\ \delta(\Xi(W,0)) &= (W,0). \end{aligned} \tag{A.1}$$

Now consider the input edge $\{(B,0),(W,0)\}$. Notice that $\Xi(\{(B,0),(W,0)\})$ is connected, by Theorem 16. Thus, $\delta(\Xi(\{(B,0),(W,0)\}))$ is equal to one of the connected subgraphs of $\mathcal{O}$ with distinct output values, without loss of generality, the one with edge $\{(B,0),(W,1)\}$. Which implies that $\delta(\Xi(W,0)) = (W,1)$. But this contradicts A.1 above, which states that $\delta(\Xi(W,0)) = (W,0)$.

Impossibility by reduction to consensus. We describe the impossibility of solving equality negation by reduction to consensus, presented in <26>. The setting there is a shared memory system where processes communicate by reading and writing shared registers, which is equivalent to our layered message-passing model.

The proof of <26> is as follows. They call the two processes P_0, P_1. They assume for contradiction, that there is an equality negation algorithm, *Aen*, in the layered message-passing model. Then, they consider a *solo* execution of each of the two processes, P_0, P_1, of *Aen*, where P_k produces an output value without ever having heard of the input value of the other process. Noticing that a solo execution is deterministic, when P_k executes alone with input value v, and the output value of P_k is uniquely defined, denoted Δ_k^v. Then they observe that there are inputs $\alpha_0, \alpha_1 \in \{0,1,2\}$ for P_0, P_1 resp. such that either

(i) $\alpha_0 = \alpha_1$ and $\Delta_0^v = \Delta_1^v$ or

(ii) $\alpha_0 \neq \alpha_1$ and $\Delta_0^v \neq \Delta_1^v$.

Then, they describe the following algorithm that solves consensus, based on *Aen*, for two processes Q_0, Q_1, each one starting with its own input inp_k and deciding a consensus value out_k. Processes execute *Aen*, each one, Q_k, invoking it with input α_k, and they each get back an equality negation value, w_k. Recall that the layered message-passing model is full-information, so when processes execute *Aen*, they also pass to each other their inputs, inp_k. Finally, process Q_k decides a consensus value out_k according to the rule:

1. if $w_k = \Delta_k^{\alpha_k}$ then decide its own input, $out_k := inp_k$,
2. else decide the input value of the other process, $out_k := inp_{(1-k)}$.

The proof is as follows. For validity, we need to check only case 2., and to notice that the execution of *Aen* could not have been a solo execution, and hence indeed Q_k has received the input value of the other process. To prove agreement, consider the values w_0, w_1 in an execution where both processes decide. Consider the execution simulated of *Aen* by Q_0, Q_1: by the negation property, $w_0 = w_1$ iff $\alpha_0 \neq \alpha_1$. But by definition of α_0, α_1, $w_0 = w_1$ iff $\Delta_0^{\alpha_0} \neq \Delta_1^{\alpha_1}$. Thus, if one process Q_k decides according to case 1, the other process has to decide according to case 2, and reciprocally. So, they both decide the same output.

Thus, there is a consensus algorithm in the layered message-passing model, contradicting the classic consensus impossibility of <6; 27>.

Appendix B. A proof of impossibility using factual change

As Section 6.2 shows, the proof method of <18> is not powerful enough to prove the unsolvability of some tasks. This seems to indicate that our logic is too weak: indeed we are only allowed to write formulas about the inputs values of each processes. But the specification of a task is very much about the outputs too!

In our DEL formalization of tasks, the outputs (or decisions) of the processes correspond to the action points of an action model. Thus, it seems natural to introduce new atomic variables whose purpose is to record which actions have happened. This can be done using a classic extension of DEL, called *DEL with factual change* <4; 7>. In this framework, when an action occurs, not only the knowledge of the agents is altered, but the valuation of some atomic variables can also be changed. Thus, one basic use of this idea is to take one atomic variable for each action; set them all to "false" in the initial model; and say that performing an action sets the corresponding variable to "true".

Appendix B.1. Dynamic Epistemic Logic with factual change

We begin by formulating a simplicial complex version of DEL with factual change. This idea has also been explored in <11>, where simplicial action models are considered. Here, we keep the standard notion of action models, but make them act on simplicial models instead of Kripke models.

Definition 12. *An* action model with postconditions *is a structure* $\mathcal{A} = \langle T, \sim, \mathsf{pre}, \mathsf{post}\rangle$, *where* $T, \sim, \mathsf{pre}$ *are the same as in Definition 6, and* $\mathsf{post} : T \times \mathrm{At} \to \mathcal{L}_{\mathcal{K}}$ *is a function that assigns a* postcondition *formula* $\mathsf{post}(t, p)$ *to each* $t \in T$ *and* $p \in \mathrm{At}$.

Intuitively, the semantics of postcondition formulas is the following: when an action t is executed in a world X of a model $\mathcal{M}$, we obtain a world (X, t) of $\mathcal{M}[\mathcal{A}]$ where each atomic proposition p is assigned the truth value of the formula $\mathsf{post}(t, p)$ in $\mathcal{M}, X$. Recall that action models must have *finite* domains T; moreover, we also require that in each action point t only a *finite* set of atomic variables may be assigned a postcondition formula $\mathsf{post}(t, p)$. Formally, this means that we require $\mathsf{post}(t, p) = p$ for all but a finite number of pairs (t, p). Such postconditions are vacuous and will usually be omitted.

Remark 4. In particular, it is possible not to assign any postcondition at all to the actions $t \in T$, in which case, we can simply write $\mathcal{A} = \langle T, \sim, \mathsf{pre}\rangle$. Thus, Definition 6 is a special case of Definition 12.

Given a simplicial model $\mathcal{M} = \langle C, \chi, \ell\rangle$ and an action model $\mathcal{A} = \langle T, \sim, \mathsf{pre}, \mathsf{post}\rangle$, the *factual change product update* $\mathcal{M}[\mathcal{A}] = \langle C[\mathcal{A}], \chi[\mathcal{A}], \ell[\mathcal{A}]\rangle$ is defined similarly to Definition 7: the underlying chromatic simplicial complex $\langle C[\mathcal{A}], \chi[\mathcal{A}]\rangle$ is the same, but the labeling $\ell[\mathcal{A}]$ differs. Recall that the vertices of $C[\mathcal{A}]$ are pairs (v, E) where $v \in \mathcal{V}(C)$ and E is an equivalence class of $\sim_{\chi(v)}$. Such a vertex is colored by the agent $a = \chi(v)$, and gets the labelling prescribed by the postcondition formulas:

$$\ell[\mathcal{A}](v, E) = \{p \in \mathrm{At}_a \mid \text{for all } X \in \mathcal{F}(C) \text{ with } v \in X, \text{ for all } t \in E,\ \mathcal{M}, X \models \mathsf{post}(t, p)\}$$

However, in this contribution, we use postconditions for one specific purpose: to record the actions that occur. In this particular setting, the formula above becomes much more simple. Suppose we have a simplicial model $\mathcal{M}$, on which we apply an action model $\mathcal{A} = \langle T, \sim, \mathsf{pre}\rangle$. What we would like to do is introduce propositional variables p_t for each action $t \in T$, whose intuitive meaning would be "the action t has occurred". But, in general, such variables might not be local, and thus cannot be interpreted on a simplicial model. Instead, we introduce variables p_a^E, where $a \in \mathrm{Ag}$ and E is an equivalence class of $\sim_a$. Intuitively, the meaning of such a formula is that "some action $t \in E$ occurred", and it will always be local for agent a.

So, let $\widehat{\mathrm{At}} = \mathrm{At} \cup \{p_a^E \mid a \in \mathrm{Ag}, E \in T/\sim_a\}$ be our new set of atomic variables. The labelling of the initial model $\mathcal{M}$ stays the same, i.e., all the variables p_a^E are initially set to false. We consider a new action model $\widehat{\mathcal{A}} = \langle T, \sim, \mathsf{pre}, \mathsf{post}\rangle$, where $T, \sim, \mathsf{pre}$ are the same as in $\mathcal{A}$, and the only non-trivial postcondition formulas are $\mathsf{post}(t, p_a^E) = \top$ whenever $t \in E$. (That is, $\mathsf{post}(t, p) = p$ in all other cases.) For this particular use of preconditions, the resulting product update model is $\mathcal{M}[\widehat{\mathcal{A}}] = \langle C[\widehat{\mathcal{A}}], \chi[\widehat{\mathcal{A}}], \ell[\widehat{\mathcal{A}}]\rangle$, where the underlying simplicial complex $\langle C[\widehat{\mathcal{A}}], \chi[\widehat{\mathcal{A}}]\rangle$ is the same as in $\mathcal{M}[\mathcal{A}]$, but the labelling of a vertex (v, E) of color a is $\ell[\mathcal{A}](v, E) = \ell(v) \cup \{p_a^E\}$.

To be even more specific, we will use the above construction with a *task action model* (cf. Definition 9). That is, the actions will be of the form $t : \mathrm{Ag} \to V^{out}$ for some set V^{out} of output values. Moreover, two such actions t and t' are indistinguishable by agent a whenever $t(a) = t'(a)$. Therefore, an equivalence class of $\sim_a$ is determined by the value $t(a)$, and the set $T/\sim_a$ of equivalence classes is simply V^{out}. So, in this context, the new atomic variables p_a^E will be written decide_a^d instead, with $d \in V^{out}$, since the intuitive meaning of this variable is that "the agent a decides on the output d".

Remark 5. In the construction described above, we are targetting one specific action model $\mathcal{A}$ that we want to apply to our initial model $\mathcal{M}$. To build a well-behaved logical theory out of this idea of recording the actions that occur, we should not fix in advance which action model is going to be applied. Fortunately, since all action models are finite, and the finite action models can be enumerated, we can give all equivalence classes for all colors of all action models unique names. Thus we only need a countable number of propositional variables to be able to record the actions from any action model.

Note on the expressive power of DEL with factual change. We should be careful when we claim that DEL with factual change is able to *express* the reason why a task is not

solvable. Of course, factual change allows us to talk about decision values, which will be crucial in the impossibility proof of Section Appendix B.2. But the notion of expressivity that is at stake here is not the standard one for comparing two logical languages.

Two logics have the same *expressivity* if for every formula in the first logic there is an equivalent formula in the second logic, and conversely for every formula in the second logic there is an equivalent formula in the first logic. It is well-known that adding action models and their execution to the language and semantics of epistemic logic does not change the expressive power of the logic. Indeed, one can prove that every formula containing modalities for action models is equivalent to a formula without such dynamic modalities <2; 9>. And conversely the embedding of epistemic logic into action model logic is trivial, as the latter is a language extension.

At the basis of such rewriting techniques eliminating dynamic modalities are axioms such as $[\mathcal{A}, t]p \leftrightarrow (\mathsf{pre}(t) \to p)$. Note that the action model on the left-hand side has 'disappeared' in the right-hand side. More precisely, because the formula $\mathsf{pre}(t)$ on the right-hand side may contain modalities for action models: on the right-hand side there is is one less occurrence of an action model in the formula, and we can continue to rewrite that part as well, eliminating further occurrences.

Similarly, extending action models with factual change does not increase the expressive power of the underlying logic, as again we have rewrite rules to eliminate such dynamic modalities from the language. The crucial axiom now becomes $[\mathcal{A}, t]p \leftrightarrow (\mathsf{pre}(t) \to \mathsf{post}(t, p))$. Again, this eliminates at least one occurrence of modalities for action models in the formula. (Where is it relevant to note that the formula $\mathsf{post}(t, p)$ already counted as such an occurrence in the action model description on the left-hand side.)

Apart from the standard notion of expressivity between languages and logics, another notion of comparison between logics is their so-called *update-expressivity*. The update-expressivity of a dynamic epistemic logic is the relation between pointed Kripke models induced by the execution of epistemic actions such as action models <25; 10>. More precisely, a pair of pointed models is in the relation if executing an action model in the former results in the latter. DEL with factual change has a larger update-expressivity than standard DEL, as we can now relate Kripke models wherein the valuation of atoms in points in the relation is different. It is in that sense that also on the class of simplicial models this novel tool much increases the (update-)expressive power, thus allowing us to formally represent the unsolvability result.

Appendix B.2. Unsolvability of equality negation

We now use the factual change DEL setting to prove the impossibility of solving non-trivial covering tasks. As in the rest of the paper, we illustrate the reasoning using the equality negation task, but it works in general for any covering task.

In this section, since we use distributed computing vocabulary, we use the letter p instead of a to denote a generic agent ("process"). The atomic variables in the input model $\mathcal{I}$ are written input_p^i ("process p has input i"), and the newly introduced atomic variables are written decide_p^d ("process p decides d"). With this logical language, we can write the formula below, specifying the equality negation task:

$$\varphi_{\text{EN}} \;=\; \bigwedge_{p,i,d} \mathsf{input}_p^i \wedge \mathsf{decide}_p^d \implies \Big((\mathsf{input}_{\bar{p}}^i \wedge \mathsf{decide}_{\bar{p}}^{\bar{d}}) \vee (\mathsf{input}_{\bar{p}}^{\bar{i}} \wedge \mathsf{decide}_{\bar{p}}^d) \Big)$$

where $\bar{p}$, $\bar{i}$, $\bar{d}$ denote values different from p, i, d, respectively. Note that $\bar{p}$ and $\bar{d}$ are uniquely defined (since there are only two processes and two decision values), but for $\bar{i}$, there are two possible inputs different from i. So, for example, $\mathsf{input}_p^{\bar{0}}$ is actually a shortcut for $\mathsf{input}_p^1 \vee \mathsf{input}_p^2$. This formula simply expresses the specification of the task: if process p has input i and decides d, then the other process should either have the same input and decide differently, or have a different input and decide the same.

Let $\mathcal{I}$ and $\mathcal{T}$ be the input model and the action model of equality negation task, as described in Section 5.2. Let $\widehat{\mathcal{T}}$ be action model obtained from $\mathcal{T}$ by adding postconditions as described in Section Appendix B.1. The model $\mathcal{I}[\widehat{\mathcal{T}}]$ can be concretely described as follows:

- Its vertices are of the form (p, i, d) with $p \in \mathrm{Ag}$, $i \in \{0, 1, 2\}$ and $d \in \{0, 1\}$. The facets are $\{(B, i, d_B), (W, j, d_W)\}$ where $i = j \iff d_B \neq d_W$.
- The coloring map is $\chi(p, i, d) = p$.
- The atomic variables labeling is $\ell(p, i, d) = \{\mathsf{input}_p^i, \mathsf{decide}_p^d\}$.

Thus, this is almost the same model as the one depicted at the end of Section 5.2, except that we have added some labeling to say where the decide_p^d atomic propositions are true. It is easily checked that the formula φ_{EN} is true in every world of $\mathcal{I}[\widehat{\mathcal{T}}]$.

Now, we would also like to interpret this formula φ_{EN} in the protocol complex $\mathcal{I}[\mathcal{MP}_N]$, but currently this model does not have any information about decision values: since the action model $\mathcal{MP}_N$ does not have postconditions, all the decide_p^d atomic variables are still set to false. But it is precisely the role of the simplicial map $\delta : \mathcal{I}[\mathcal{MP}_N] \to \mathcal{I}[\mathcal{T}]$ to assign decision values to each world of $\mathcal{I}[\mathcal{MP}_N]$. Thus, given such a map δ, we can define a new model $\mathcal{I}[\mathcal{MP}_N]_\delta$ obtained from $\mathcal{I}[\mathcal{MP}_N]$ by setting the decide_p^d atomic variables according to δ.

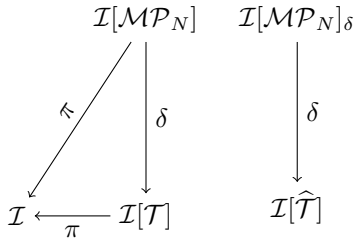
Lemma 18. *Let $\mathcal{M} = \langle M, \chi, \ell\rangle$ be a simplicial model, and $\delta : \mathcal{M} \to \mathcal{I}[\mathcal{T}]$ be a morphism of simplicial models. Then there is a unique model $\mathcal{M}_\delta = \langle M, \chi, \ell_\delta\rangle$ such that $\delta : \mathcal{M}_\delta \to \mathcal{I}[\widehat{\mathcal{T}}]$ is still a morphism of simplicial models.*

Proof. All we have to do is label the worlds of $\mathcal{M}$ with the decide_p^d atomic propositions according to δ. We define $\ell_\delta : \mathcal{V}(M) \to \mathscr{P}(\widehat{\mathrm{At}})$ as $\ell_\delta(m) = \ell(m) \cup \{\mathsf{decide}_p^d\}$, where $\delta(m) = (p, i, d) \in \mathcal{I}[\mathcal{T}]$. Then δ is still a chromatic simplicial map (since we did not change the underlying complexes nor their colors), and moreover we have $\ell_\delta(m) = \ell_{\mathcal{I}[\widehat{\mathcal{T}}]}(\delta(m))$ for all m. The model $\mathcal{M}_\delta$ is unique since any other choice of $\ell_\delta(m)$ would have broken this last condition, so δ would not be a morphism of simplicial models. □

We can finally prove that the equality negation task is not solvable.

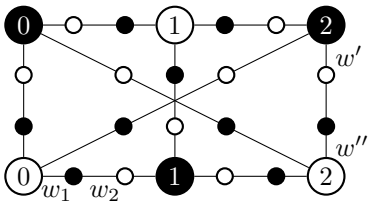
Theorem 19. *The equality negation task for two processes is not solvable in the N-layer message-passing model.*

Proof. Let us assume by contradiction that the task is solvable, i.e., by Definition 10, that there exists a morphism of simplicial models $\delta : \mathcal{I}[\mathcal{MP}_N] \to \mathcal{I}[\mathcal{T}]$ that makes the diagram on the left commute.



By Lemma 18, we can lift δ to a morphism from $\mathcal{I}[\mathcal{MP}_N]_\delta$ to $\mathcal{I}[\widehat{\mathcal{T}}]$ between the extended models. As we remarked earlier, the formula φ_{EN} is true in every world of $\mathcal{I}[\widehat{\mathcal{T}}]$. Therefore, it also has to be true in every world of $\mathcal{I}[\mathcal{MP}_N]_\delta$. Indeed, for any world w, since $\mathcal{I}[\widehat{\mathcal{T}}], \delta(w) \models \varphi$, and δ is a morphism, by Lemma 13, we must have $\mathcal{I}[\mathcal{MP}_N]_\delta, w \models \varphi$. We will now derive a contradiction from this fact.

Recall that the protocol complex $\mathcal{I}[\mathcal{MP}_N]$ is just a subdivision of the input complex $\mathcal{I}$, as depicted below. For simplicity, we depict the input model of the equality negation task, but the same idea holds for any other input model. Also, some input values have been omitted in the vertices on a subdivided edge; it is the same input as the extremity of the edge which has the same color. Furthermore, the picture shows only one subdivision, but our reasoning is unrestricted and it applies to any number of layers N.



Consider some world w_1 on the $(W, 0) - (B, 1)$ edge. In the world w_1, the two processes have different inputs. Since in $\mathcal{I}[\mathcal{MP}_N]_\delta$, the formula φ_{EN} is true in w_1, the decision values have to be the same. Without loss of generality, let us assume that in w_1, both processes decide 0.

We then look at the next world w_2, which shares a black vertex with w_1. Since the inputs are still 0 and 1, and φ is true, and we assumed that process B decides 0, then the white vertex of w_2 also has to decide 0.

We iterate this reasoning along the $(W, 0) - (B, 1)$ edge, then along the $(B, 1) - (W, 2)$ edge, and along the $(W, 0) - (B, 2)$ edge: all the vertices on these edges must have the same decision value 0. Thus, on the picture, the top right $(B, 2)$ corner has to decide 0, as well as the bottom right $(W, 2)$ corner.

Now in the world w', the two input values are equal, so the processes should decide differently. Since the black vertex decides 0, the white vertex must have decision value 1. If we keep going along the rightmost edge, the decision values must alternate: all the black vertices must decide 0, and the white ones decide 1. Finally, we reach the world w'', where both decision values are 0, whereas the inputs are both 2. So the formula φ_{EN} is false in w'', which is a contradiction. □

Remark 6. For simplicity, we have written the proof of Theorem 19 using the equality negation task. Notice however that the exact same proof, using a similar formula φ_{EN_k},

works for all the other generalized equality negation task of Section 5.4.1. More generally, it is known in the context of distributed computing that any non-trivial covering task is unsolvable in the N-layered message-passing model <14>. It should not be difficult to rewrite this proof in a similar way, using an appropriate logical formula φ.

Appendix B.3. Relevance of the factual change approach

It is interesting to compare the epistemic formula φ_{EN} that we used in this paper to prove the unsolvability of equality negation, with the one (let us call it ψ) that was used in <18> to prove the impossibility of solving consensus. In the case of consensus, we did not need the factual change framework. The formula ψ was simply saying that the processes have common knowledge of the input values. This formula is quite informative: it tells us that the main goal of the consensus task is to achieve common knowledge. On the other hand, the formula φ_{EN} is less informative: it is simply stating the specification of the equality negation task. It does not even seem to be talking about knowledge, since there are no K or C operators in the formula. In fact, the epistemic content of φ_{EN} is hidden in the decide_p^d atomic variables. Indeed, their semantics in $\mathcal{I}[\mathcal{MP}_N]_\delta$ is referring to the decision map δ, which assigns a decision value d to each vertex of $\mathcal{I}[\mathcal{MP}_N]$. The fact that we assign decisions to vertices means that each process must decide its output *solely according to its knowledge.*

Despite the fact that it produces less informative formulas, the factual change proof method has two major benefits. First, it seems to be able to prove *any* impossibility result. Indeed, let $\mathcal{T} = \langle T, \sim, \mathsf{pre}\rangle$ be a task action model, on the input model $\mathcal{I}$, and let $\mathcal{P}$ be a protocol action model. Remember that the elements of T are functions $t : \text{Ag} \to V^{out}$ assigning a decision value to each agent. Let φ denote the following formula:

$$\varphi \;=\; \bigwedge_{X\in\mathcal{F}(\mathcal{I})} \left(\bigwedge_{p\in\text{Ag}} \mathsf{input}_p^{X(p)} \implies \bigvee_{\substack{t\in T \\ \mathcal{I},X\models \mathsf{pre}(t)}} \bigwedge_{p\in\text{Ag}} \mathsf{decide}_p^{t(p)} \right) \tag{B.1}$$

where $X(p)$ denotes the input value of process p in the input simplex X. Then we get the following Theorem.

Theorem 20. *The task $\mathcal{T}$ is solvable in the protocol $\mathcal{P}$ if and only if there exists an extension $\overline{\mathcal{I}[\mathcal{P}]}$ of $\mathcal{I}[\mathcal{P}]$ (assigning a single decision value to each vertex of $\mathcal{I}[\mathcal{P}]$) such that φ from (B.1) is true in every world of $\overline{\mathcal{I}[\mathcal{P}]}$.*

Proof. ($\Rightarrow$): We already proved this direction in Section Appendix B.2. Assume that the task is solvable, i.e., by Definition 10, there is a morphism $\delta : \mathcal{I}[\mathcal{P}] \to \mathcal{I}[\mathcal{T}]$ such that $\pi \circ \delta = \pi$. The model $\overline{\mathcal{I}[\mathcal{P}]} := \mathcal{I}[\mathcal{P}]_\delta$ is the one given Lemma 18, where the assignment of decision values is the one given by δ. Then φ is easily seen to be true in every world of $\mathcal{I}[\widehat{\mathcal{T}}]$, and by Lemma 13, it is also true in every world of $\mathcal{I}[\mathcal{P}]_\delta$.

($\Leftarrow$): For the converse, assume that there is a model $\overline{\mathcal{I}[\mathcal{P}]}$ where the formula φ is true in every world. Then we build a map $\delta : \mathcal{I}[\mathcal{P}] \to \mathcal{I}[\mathcal{T}]$ as follows. If a vertex (i, E) of $\overline{\mathcal{I}[\mathcal{P}]}$, colored by agent p, is labeled with the atomic proposition decide_p^d, we send it to the vertex $\delta(i, E) := (p, i, d)$ of $\mathcal{I}[\mathcal{T}]$.

By definition, we have the commutative diagram $\pi(i, E) = i = \pi \circ \delta(i, E)$. We now need to show that δ is a morphism of simplicial models. The coloring and labeling maps are preserved, since by definition they just copy the coloring and labeling of $\mathcal{I}$. We still have to prove that δ sends simplices to simplices.

Let Y be a facet of $\mathcal{I}[\mathcal{P}]$. Let $X = \pi(Y) \in \mathcal{F}(\mathcal{I})$ be the facet of the input complex corresponding to the initial values of the processes in the execution Y. Then we have $\mathcal{I}[\mathcal{P}], Y \models \bigwedge_{p \in \mathrm{Ag}} \mathsf{input}_p^{X(p)}$, and since $\overline{\mathcal{I}[\mathcal{P}]}, Y \models \varphi$, by modus ponens there must be some action $t \in T$, with $\mathcal{I}, X \models \mathsf{pre}(t)$, such that $\overline{\mathcal{I}[\mathcal{P}]}, Y \models \bigwedge_{p \in \mathrm{Ag}} \mathsf{decide}_p^{t(p)}$. Thus, the vertex v of Y which is colored by p must be labeled with the atomic proposition $\mathsf{decide}_p^{t(p)}$, and so the map δ sends it to the vertex $(i, t(p))$ of $\mathcal{I}[\mathcal{T}]$. By definition of the action model, since $\mathcal{I}, X \models \mathsf{pre}(t)$, the set of vertices $\delta(Y) = \{(i, t(p)) \mid i \in X, p = \chi(i)\}$ is a simplex of $\mathcal{I}[\mathcal{T}]$. Therefore, the map δ is a simplicial map. □

This theorem implies that the situation of Section 6.2 cannot happen with the factual change approach: if the task is not solvable, there necessarily exists a world X of $\mathcal{I}[\mathcal{P}]_\delta$ where the formula fails. Of course, finding such a world is usually the hard part of an impossibility proof, but at least we know it exists. In fact, in the particular case of read/write protocols (or, equivalently, layered message-passing), the solvability of tasks is known to be undecidable when there are more than three processes <15; 21>. Thus, according to our Theorem, given a formula φ, the problem of deciding whether there exists a number of layers N and an extension of $\mathcal{I}[\mathcal{MP}_N]$ which validates the formula, is also undecidable.

The second benefit of the factual change framework is that it gives us a way of using epistemic logic as a specification language for tasks. Notice that in Theorem 20, we characterized the solvability of a task without referring to $\mathcal{T}$ itself: the formula φ contains all the information of $\mathcal{T}$. Thus, instead of relying on the commutative diagram of Definition 10, we can specify a task directly as a logical formula. With this in mind, we need not restrict ourselves to formulas such as φ which simply translate the task specification. One could decide to pick a more informative formula, with an interesting epistemic content, and study the solvability of this "task".